CASCADE: An Asset-driven Approach to Build Security Assurance Cases for Automotive Systems

MAZEN MOHAMAD and RODI JOLAK, Chalmers and University of Gothenburg, Sweden
ÖRJAN ASKERDAL, Volvo Trucks, Sweden
JAN-PHILIPP STEGHÖFER, Chalmers and University of Gothenburg, Sweden
RICCARDO SCANDARIATO, Hamburg University of Technology, Germany

Security Assurance Cases (SAC) are structured arguments and evidence bodies used to reason about the security of a certain system. SACs are gaining focus in the automotive industry, as the needs for security assurance are growing in this domain. However, the state-of-the-arts lack a mature approach able to suit the needs of the automotive industry. In this article, we present CASCADE, an asset-driven approach for creating SAC, which is inspired by the upcoming security standard ISO/SAE-21434 as well as the internal needs of automotive Original Equipment Manufacturers (OEMs). CASCADE also differentiates itself from the state-of-the-art by incorporating a way to reason about the quality of the constructed security assurance case. We created the approach by conducting an iterative design science research study. We illustrate the results using the example case of the road vehicle's headlamp provided in the ISO standard. We also illustrate how our approach aligns well with the structure and content of the ISO/SAE-21434 standard, hence demonstrating the practical applicability of CASCADE in an industrial context.

CCS Concepts: • **Software and its engineering**; • **Security and privacy** → *Systems security*;

Additional Key Words and Phrases: Security, assurance cases, automotive systems

ACM Reference format:
Mazen Mohamad, Rodi Jolak, Örjan Askerdal, Jan-Philipp Steghöfer, and Riccardo Scandariato. 2023. CASCADE: An Asset-driven Approach to Build Security Assurance Cases for Automotive Systems. *ACM Trans. Cyber-Phys. Syst.* 7, 1, Article 3 (February 2023), 26 pages.
https://doi.org/10.1145/3569459

1 INTRODUCTION

Assurance cases are structured bodies of arguments and evidence used to reason about a certain property of a system. **Security Assurance Cases (SACs)** are a type of assurance case for the field of cyber-security. In this article, we turn our attention to the creation of a SAC, with a particular focus on the domain of automotive applications. As vehicles become more advanced and connected,

This work is partially supported by the CASUS research project funded by VINNOVA, a Swedish funding agency.
Authors' addresses: M. Mohamad, R. Jolak, and J.-P. Steghöfer, Chalmers and University of Gothenburg, Chalmersplatsen 4, 412 96 Gothenburg, Sweden; emails: mazen.mohamad@gu.se, rodi.jolak@cse.gu.se, jan-philipp.steghofer@gu.se; Ö. Askerdal, Volvo Trucks, Herkulesgatan 75 SE-405 08, Gothenburg, Sweden; email: orjan.askerdal.3@volvo.se; R. Scandariato, Hamburg University of Technology, Am Schwarzenberg-Campus 1, 21073 Hamburg, Germany; email: riccardo.scandariato@tuhh.de.

2378-962X/2023/02-ART3
https://doi.org/10.1145/3569459

security scrutiny has increased in this domain. Furthermore, new standards and regulations push towards assuring security for vehicular systems by using SAC. Similar to safety cases, which are required in safety standards, e.g., ISO-26262 [18], SACs are explicitly *required* in ISO/SAE-21434 [19]. Additionally, SACs are required for all systems in production. In literature, some studies suggest the creation of SAC based on requirements derived from security standards [6, 8]. However, there is no approach that helps to achieve conformance with the upcoming ISO/SAE-21434 standard. Additionally, since the requirements for SAC are new, there is no evidence in the literature that the knowledge base in industry is mature enough to achieve conformity to these requirements. Moreover, quality assurance of the SACs is missing in the reported approaches in literature, even though it is a very important aspect. For different stakeholders to use a SAC, it is essential to trust that the SAC's argument is built with a sufficient level of completeness and that the evidence provides a sufficient level of confidence to justify the targeted claims. Finally, we identified that the lack of industry involvement is a significant issue in current approaches. This results in gaps between research and industry.

To bridge these gaps, we have worked together with Volvo Trucks and Volvo Cars, two international automotive OEMs located in Sweden, to develop CASCADE, the asset-driven approach for SAC creation presented in this article. CASCADE is inspired by the structure of requirements and work products of ISO/SAE-21434. It is asset-driven, i.e., the resulted SACs have assets as drivers of the structure of the security arguments. Therefore, it allows creating security assurance based on what is valuable in the system.

Additionally, we integrated quality assurance in SACs created with CASCADE by distinguishing between product-related claims and case quality-claims, as well as building arguments for both. CASCADE also governs where in the argument these case quality-claims need to be introduced.

From a methodological standpoint, we created and validated our approach in a three-cycle design science research study as follows: First, we created a high-level structure of an asset-driven SAC, which included the identification of the assets, the tracing of such assets to system elements (e.g., processing, communication, and storage operations), and the identification of the relevant security assets for each asset. Second, we improved the structure of CASCADE to better align with the development activities at automotive companies and better cover their needs for SAC, including the need to confront with ISO/SAE-21434. We then illustrated the approach using the exemplary case study mentioned in ISO/SAE-21434. Finally, we presented the resulting approach to the security experts from an industrial automotive OEM and gathered their feedback. Last, we further evaluated CASCADE by mapping the requirements and work products of ISO/SAE-21434 standard to the corresponding CASCADE elements. We analyzed the results and created a guideline for practitioners and researchers to enable the replication of the mapping activity on the same or a different standard.

The results are presented in Sections 4–7, after discussing the related work in Section 2 and the methodology in Section 3.

This work is an extension of a previously published study [22]. The overall contributions of this work are the following:

- CASCADE, an asset-driven approach for the creation of the security assurance cases, which is inspired by the upcoming security standard in the automotive domain. CASCADE also differentiates itself from the state-of-the-art by incorporating a way to reason about the quality of the constructed security assurance case.
- A validation of the CASCADE approach by (i) involving an automotive OEM located in Sweden and (ii) illustrating how CASCADE aligns with the structure and content of the ISO/SAE-21434 standard, hence demonstrating the practical applicability of CASCADE in an industrial context.

- A guideline to map items of security standards to the elements of a CASCADE security case. This guideline helps practitioners to fulfill their compliance requirements.

2 BACKGROUND AND RELATED WORK

In this section, we provide background information about SAC, security assets in automotive, and their corresponding security threats and attacks. We also review related work of asset-based approaches in the literature.

2.1 Security Assurance Cases

Assurance cases are bodies of evidence organized in structured arguments, used to justify that certain claims about a system's property hold [10]. They are defined by the GSN standard [11] as *"A reasoned and compelling argument, supported by a body of evidence, that a system, service or organisation will operate as intended for a defined application in a defined environment."*

Security Assurance Cases (SACs) are a special type of assurance cases where the argument consists of claims about security for the system in question, and the evidence justifies these security-related claims. SACs consist of the following primary components [5, 20]:

- Security claims: These claims vary in terms of abstraction level. At the beginning of the argument, they are high level, but then they get more fine-grained as the argument evolves. The claims are usually derived from requirement specifications of the system in question.
- Context: This refers to the context in which the claims should hold. In other words, it sets the scope of the argument. For example, a context could be a document that defines the required level of security desired by the organization creating the SAC.
- Strategy: Building the argument means that the claims are decomposed into sub-claims in an iterative manner until they reach a point where evidence can be assigned to support them. During this process, decisions have to be made on how to decompose a certain claim, e.g., by considering the different security properties or different possible threats. These decisions are called strategies in SAC.
- Evidence: A body of evidence is provided to prove the claims created in the argument. An example of a piece of evidence is a test report showing that the code base of a certain ECU passes all the tests. Another example is a report created by a security expert that proves that a realization of a vulnerability is not possible.

SAC can be expressed in a textual or graphical format [5]. The most common graphical formats are the **Goal Structure Notation** (**GSN** [26]), and the **Claims, Arguments, and Evidence** notation (**CAE** [1]).

2.2 Automotive Assets and Related Security Threats

According to Reference [25], there are four categories of assets in automotive systems that are targeted by security threats and attacks. These assets are hardware, software, network and communication, and data storage.

- *Hardware*: This asset category includes sensors, actuators, and the hardware part of the **Electronic Control Units (ECUs)**. These assets are often threatened by disruption or direct interventions that influence their availability and integrity. Examples of attacks on these assets include fault injection through the installation of malicious hardware leading to compromising availability.
- *Software*: This category includes external libraries, **Operating Systems (OS)**, applications, virtualization, and the software part of the ECUs. Security threats and attacks on software

assets include the manipulation of software, such as tampering attacks that often target software availability and integrity.

- *Network/Communication*: Refers to internal or external communication. Internal communication assets are busses such as CAN, FlexRay, LIN, MOST, and automotive Ethernet. External communication assets are WiFi, Bluetooth, and **Vehicle to Everything (V2X)** communication. Examples of attacks on these assets include fabrication or jamming attacks, spoofing, message collision, eavesdropping, hijacking, and **denial of service (DoS)**. These attacks might compromise the confidentiality, integrity, and availability of the these assets.
- *Data storage*: Sensitive data including user data, backups, cryptographic keys, forensics logs, and system information and reports. These assets are targeted by unauthorized access and malicious manipulation that often influence the confidentiality, integrity, and availability of the data.

2.3 Asset-based Approaches

Researchers have been exploring several asset-based approaches for creating the argument part of SAC. Biao et al. [28] suggest dividing the argument into different layers and using different patterns (one per layer) to create the part of the argument that corresponds to each layer. Assets are considered as one of these layers, and the pattern used to create it includes claims that the assets are "under protection," and strategies to break down critical assets. Biao et al. [28], however, do not consider the quality of the cases and only focus on creating arguments without touching upon the evidence part. Luburic et al. [21] also present an asset-centric approach for security assurance. The info used in their approach is taken from: *(i)* asset inventories; *(ii)* **Data Flow Diagrams (DFD)** of particular assets and the components that manipulate them; and *(iii)* the security policy that defines protective mechanisms for the components from the previous point. They propose a domain model where assets are the centerpieces. The assets are linked to security goals. The argument considers the protection of the assets throughout their life-cycles by arguing about protecting the components that store, process, and transmit those assets. The SAC they provide is very high level and includes two strategies: "reasonable protection for all sensitive assets" and arguing over the dataflow of each related component. The authors state that the main limitations of their approach are asset and dataflow granularity. In our study, we consider the assets to be the driver of our approach, but we extend the argument to reach the level of concrete security requirements. We also derive our strategies from an industrial standard and validate our approach in collaboration with an OEM. Furthermore, we extend our approach to include case-quality aspects.

2.4 Standard-based Approaches

Multiple researchers have explored extracting requirements from security standards for the creation of the arguments of SAC. None of these studies have, however, targeted the upcoming standard ISO/SAE-21434 for cybersecurity in automotive. Finnegan et al. [4, 9] present a framework for the creation of security cases for medical devices. Their framework incorporates multiple security documents, e.g., standards and best practices, to develop a security argumentation pattern. The pattern provides a "comprehensive matrix showing the link between the security risks, associated causes, the mitigating security controls and evidence of those controls being implemented to establish the security capability."

Ankrum et al. [6] studied the mapping of security and safety standards in safety-critical domains to assurance cases. They used the **Goal Structuring Notation (GSN)** and **ASCAD (Claims – Arguments – Evidence)**, which are the most common notations for documenting assurance cases. The security standard used in the study was the Common Criteria for Information Technology

Security Evaluation, ISO/IEC 15408:1999 [16]. Ankrum et al. describe challenges they encountered while conducting the mapping as well as lessons learned.

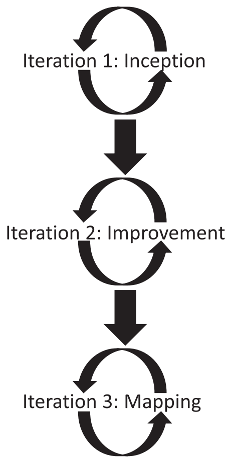
In this work, we are using the upcoming ISO/SAE 21434 standard to structure an approach for creating SAC, while also considering the industrial needs from the automotive domain. The reason for using this particular standard is that it includes a requirement that explicitly requires a security case to be created. Hence, if a company wants to conform with the standard, then it needs to create security cases for its products. However, conformance with the standard is not the only need of automotive OEMs when it comes to assurance cases. We have identified multiple usage scenarios in our previous work [23]. To create SAC that are suitable for these scenarios, the quality aspect of the cases needs to be considered, which we cover in our approach.

Birch et al. [7] suggest a layered model for structuring arguments of automotive safety assurance cases. The model is built to help satisfying the requirement to build safety cases in ISO 26262 [18]. It consists of a core argument layer representing the rationale behind safety requirements, as well as three other layers of arguments representing the relationship of the requirements to the corresponding artefacts, the means used in their development, and the environment in which safety activities are undertaken. The core argument proposed by Birch et al. serves a similar purpose to our case quality-claims in terms of completeness, but only on the requirements level. In our work, we require such claims to be added every time a decomposition takes place. This difference is due to the different natures of functional safety and security. In functional safety assurance cases, the arguments are driven by the requirements, while in our approach, we drive it by assets and only reach the requirements after going through multiple levels of arguments. Moreover, the layered approach includes a layer for the environment and another for the means used in developing. In our approach, we regard these as being claims that potentially apply to different cases and would be reused, and hence document them in a special argumentation block we call the generic-sub case. Additionally, we consider another type of case quality-claims, which is the confidence claims associated with the evidence.

3 METHODOLOGY

Design science research is a problem-solving methodology that aims at developing artefacts to extend existing boundaries in a given context [14]. We conducted three research iterations, following the design science guidelines proposed by Hevner et al. [14]. In each iteration, we followed the five-step process proposed by Vaishnavi and Kuechler [27], which consists of *awareness of the problem*, *suggestion*, *development*, *evaluation*, and *conclusion* steps. The three-iteration process is depicted in Figure 1. The figure also shows a brief summary of each design science research step for each cycle.

In the first iteration, *Inception*, our aim was to address the needs for security assurance cases that were identified in a previous study [23]. Specifically, we investigated how an asset-based approach for the creation of security assurance cases can assist automotive companies to fulfill their needs to conform with the upcoming ISO/SAE-21434 standard. We suggested an initial asset-based approach and used an open-source system specification of a supermarket management system [3] to illustrate it. The approach and the outcome of the illustration were discussed and evaluated with security experts at two large automotive OEMs. The main input from the experts was focused on the need to align the structure of the approach with the internal way of working at the companies, which is also one of the needs identified in our previous work [23]. Another aspect of improvement that emerged from the evaluation of the inception iteration is the need for a mechanism to assure the quality of the approach's outcome. We concluded that there is a need to further consider the internal needs of companies when designing an approach for creating SAC.



	Inception	Improvement	Mapping
Awareness of the problem	Need for an approach to conform with ISO/SAE 21434	Need to further consider internal needs	Need for evaluation against upcoming standards
Suggestion	Asset-based approach	Integrate quality assurance	Mapping to ISO/SAE-21434
Development	Initial approach developed	CASCADE	Mapping table and guidelines
Evaluation	Discussion with industrial partners	Illustration on ISO/SAE 21434 use case + evaluation at OEM	Lessons learned from the mapping activity
Conclusion	Need to further consider internal needs	Need for evaluation against upcoming standards	Potential enhancement and future work

Fig. 1. Three-iteration design science research.

In the second iteration *Improvement*, we aimed at incorporating the experience and feedback gathered in the first iteration to improve the asset-based approach. The suggestion of this iteration was to integrate quality assurance aspects within the approach. Hence, in the development phase, we created CASCADE, an asset-based approach for SAC creation with built-in quality assurance. The structure of CASCADE is inspired by the structure of requirements and work products of ISO/SAE-21434 and incorporates the need to assure the quality of the SAC. It is based on the requirements that emerged from applying the initial approach as well as on the way of working at the automotive companies we consulted in the first iteration. To evaluate CASCADE, we applied it on an example case of a vehicle's headlamp item from ISO/SAE-21434 and presented the outcome to the security experts at two OEMs we also consulted in the first iteration. As a conclusion of the second iteration, we identified areas for enhancement of CASCADE to fulfill a wider range of the internal needs of the company, as well as the need to assess whether CASCADE has the capacity to include the items of the security standard ISO/SAE-21434.

In the third iteration *Mapping*, we aimed at further validating CASCADE by analyzing the mapping of the requirements and work products from ISO/SAE 21434 to CASCADE elements. The mapping allows us to evaluate CASCADE's coverage of the items of the standard (requirements and work products) and to point out potential improvements of CASCADE. For each requirement and work product in ISO/SAE 21434 (168 in total), we identified the corresponding SAC element, CASCADE element, CASCADE block, and CASCADE level it belongs to.

Two researchers contributed to the mapping activity to avoid assessment bias. First 17 (10%) of the requirements and work products were selected using the stratified random sampling method where each clause in the standard was treated as one strata. The two researchers then independently mapped the requirements and work products to CASCADE and reached an agreement of 71%. For the remaining 29%, there was a calibration exercise where the disagreement was discussed and an agreement on the mapping was achieved for all 17 requirements and work products. The researchers then each worked on half of the requirements and work products and reviewed the other half.

The mapping results are available in the table in Appendix A. As an evaluation step, we studied the lessons learned from the process and results of the mapping activity and came up with potential enhancements on CASCADE. The result of this step is available in Section 7.

Based on the experience gained during the mapping, we constructed a guideline to enable the replication of the mapping activity we conducted in this study or to test it on other security standards in automotive or other safety critical domains. As conclusion, we identified areas for future work to further improve CASCADE and its applicability in the automotive domain.

4 CASCADE

In general terms, assets are artefacts of interest to a certain entity. In computer security, these artefacts can be hardware, software, network and communication, or data [25]. The importance of assets makes them the target of attackers.

The CASCADE approach for creating security assurance cases takes the importance of assets to organizations into consideration. Hence, it builds the argumentation by putting assets in focus, with the goal to show that these assets are secure from cybersecurity attacks. Our aim is to prove that a given artefact is secure by arguing that its assets are secure.

An important design principle in the CASCADE approach is the integration of quality assurance of the cases in terms of argumentation completeness and evidence confidence. Each level of argumentation (i.e., strategy) is associated with at least one claim about the completeness, and each level of evidence is associated with at least one claim about confidence. A similar concept is used by Hawkins et al. [13] to argue about the confidence of safety cases.

We use the same security terminologies and concepts used in ISO/SAE 21434. The only exception is *Security Goal* concept in the standard, which we split up in CASCADE in two different levels (security goals and threat scenarios). Appendix B includes a table with the core security terms and concepts in CASCADE, their definitions, and the corresponding ISO/SAE 21434 concepts.

4.1 Elements of a SAC in CASCADE

We use GSN [26] to create SAC using the CASCADE approach. The elements of the notation are: *(i)* Claim:[1] a security claim about the artefact in question; *(ii)* Strategy: a method used to decompose a claim into sub-claims; *(iii)* Evidence/Solution: a justification of a Claim/set of claims; *(iv)* Context: used to set a scope of a given claim; and *(v)* Assumption: used to document the assumptions made for a certain claim.

In addition to these, we have created additional types of elements to be used in our approach: *(vi)* **Case Quality-claims (CQ-claims)**: represent claims about the quality of the created case itself; *(vii)* **Case Quality-evidence (CQ-evidence)**: represent evidence used to justify CQ-claims; and *(viii)* Generic sub-case: consist of claims, strategies, contexts, assumptions, and evidence that are not bound to a specific artefact, but instead are applicable to a wider range of artefacts in the context of a product, program, or organization.

4.2 Building Blocks of the CASCADE Approach

The asset-based approach consists of building blocks, as shown in Figure 2. Each block contains a sub-set of the case. In the following subsections, we explain the blocks and their contents.

4.2.1 *Top Claim.*

This block consists of the top security claim of the artefact in question. It also includes the context of the claim and assumptions made to set the scope of the claim. If we are considering a software system, e.g., then we might make an assumption that the hardware is secure. The top claim differs between different organizations and drives the granularity of the SAC. For example, a service provider might consider the security of a service to be the top claim,

[1]In GSN, the terms goal and subgoal are used to refer to high and low abstraction levels of argumentation claims, respectively. To avoid confusion, we refer to these as claims.

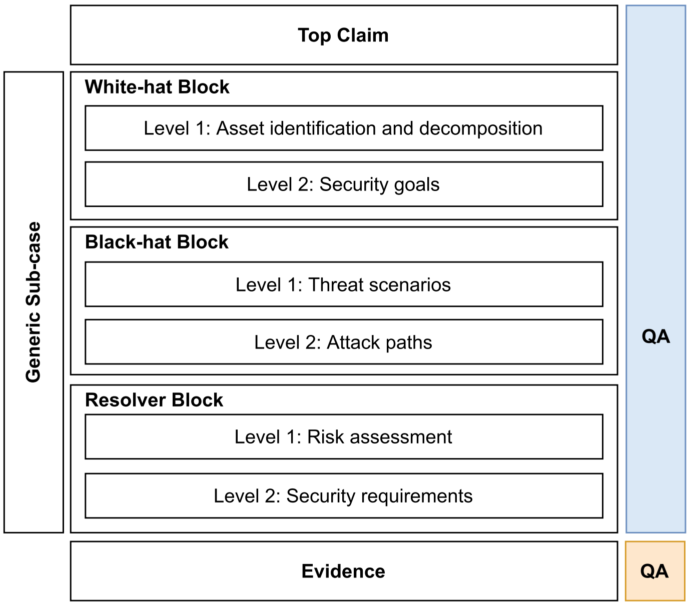


Fig. 2. The CASCADE approach for creating security assurance cases.

but an automotive OEM might need to consider the whole vehicle's security, which requires the incorporation of different services or user functions. Similarly, depending on the intended usage of the SAC, the top claim might include back-end systems or only on-board systems. For example, to assure the security of a complete vehicle, it is important to make sure that not only the vehicle's components are secure, but also the back-end systems that communicate with the vehicle. In contrast, to ensure that a certain end-user function in the vehicle is secure, it might be enough to only consider the corresponding sub-systems in the vehicle itself.

As CASCADE does not specify an abstraction level for the top claims of the SACs, hence, it is possible to create SACs for complete products or for various components or functionalities of a system. However, in the latter case, there is a need for compositional claims combining two or more components, especially when there are inter-dependencies among the components. This might cause the emergence of new assets, security goals, threats, and so on. CASCADE does not specifically handle these dependencies, but rather reflects the applied security measures and controls. However, it is important to document the scope of the claims in context nodes, as well as all the assumptions made when creating the argument.

4.2.2 Generic Sub-case. This block contains a sub-case that is applicable not only to the artefact for which the SAC is being created but instead to a larger context. For example, if a company defines a cybersecurity policy, enforced by cybersecurity rules and processes, then the policy can be used in security claims for all its products. These claims can be re-used when creating SAC for individual artefacts. Another example is when certain claims can be made on a product level. Then these claims can be reused for all SAC of individual components of that product. Our aim with this block is to make the approach scalable in larger organizations with complex products and multiple teams. Each team can work on a part of the SAC that corresponds to their artefact. On a higher level, these SACs can be combined together, and generic arguments that are applicable to the sub-SAC can be provided.

4.2.3 White-hat Block. This block starts with the identification of assets, which is the driver of our approach. Asset identification is done by conducting an analysis to find the artefacts of the system that are likely to be subject to an attack.

When the assets are identified, they can be further decomposed during the different phases of the development life-cycle. For example in an OEM, a high-level asset analysis is done at the concept phase, and later a low-level analysis is conducted during implementation, where more information about the assets and their usage is known.

Linking assets to higher-level claims. To link the assets to the main claim, we identify which assets exist and which components use or have access to these assets. For example, in a vehicle, the driver's information can be considered an asset that is accessible by the infotainment system of the vehicle. Hence, we link this asset to the claims of the security of the infotainment system. To make this more concrete, we look at the traceability of the asset. For example, we consider the assets *(i)* "at rest," which refers to where the assets are stored; *(ii)* "on the move," when the asset is in transition between two entities, e.g., when sensor data is being transferred from the sensor to an ECU; and *(iii)* "in use," which is when the asset is being used, e.g., when some diagnostics data is being processed by a back-end system.

Decomposition of assets. To decompose assets, we look into the types of the identified assets. This gives an indicator of whether the asset would have implications on the local part of the vehicle (one electronic control unit/ECU) or on a bigger part of the vehicle (multiple ECUs). We also look into the relations among assets, e.g., dependability.

Linking assets to the lower level. To link the asset to the lower level in the approach, i.e., the security goals, we identify the relevant security properties for the assets. Specifically, we look into the **Confidentiality, Integrity, and Availability (CIA)** triad. For example, the vehicle engine's start functionality is an asset that has relevant integrity and availability properties.

Identification of security goals. When we have identified the relevant security properties for each asset, we create claims representing the security goals.[2] Following our example of the engine start request, a claim about the achievement of a security goal would be that the availability of the request is preserved. One combination of asset/security property might lead to several goals; for example, that the engine start is available using a connected mobile app and a web portal. To make sure the relevant properties are covered when identifying security goals, we consider damage scenarios that lead to compromising the security goals, e.g., that the engine start request is unavailable or an unintended start of the engine occurs, which would damage the integrity of the asset.

4.2.4 Black-hat Block. In this block, we aim to identify the scenarios that might lead to not fulfilling the identified security goals and hence cause harm to our identified assets.

Identification of threat scenarios. When we have identified the claims about the achievement of security goals, we proceed by identifying the threat scenarios and creating claims for negating the possibility of these scenarios. We connect these claims to the corresponding claims about achieving security goals. For example, a claim handling a threat scenario connected to the claim "Unintended request for engine start is not possible" might be identified by considering a threat model, e.g., STRIDE [15]. Hence, a claim might look like: "Spoofing a request for engine start is not possible."

Identification of possible attack paths. In this step, we identify possible attack paths that can lead to the realization of a threat scenario. Each threat scenario might be associated with multiple

[2]A security goal is preserving a security concern (CIA) for an asset [12].

attack paths. We then claim the opposite of these attack paths. An example of an attack path is "An attacker compromises the cellular interface and sends a request to start the engine," and the claim would be to negate the possibility for that.

4.2.5 Resolver Block. This block is the last one in the argumentation part of the CASCADE approach. It links the claims derived from the attack paths to the evidence.

Risk assessment. In this level, we assess the risk of the identified attack paths. Based on the risk level, the creators of the SAC create claims to treat the risk by, e.g., accepting, mitigating, or transferring it.

Requirements. At this point, requirements of risk treatments identified in the previous level are to be expressed as claims. This level may contain multiple decompositions of claims, based on the level of detail the creators of the SAC wish to achieve, which is driven by the potential usage of the SAC. For instance, if the SAC is to be used by a development team to assess the security level, then this might require a fine-grained requirement decomposition that might go all the way to the code level. In contrast, if the SAC is to be used to communicate security issues with outside parties, then a higher level of granularity might be chosen. In either case, it is important to reach an *"actionable"* level, meaning that the claims should reach a point where evidence can be assigned to justify them.

4.2.6 Evidence. The evidence is a crucial part of a SAC. The quality of the argument does not matter if it cannot be justified by evidence. In our approach, evidence can be provided at any block of the argumentation. For example, if it can be proven in the black-hat block that a certain asset is not subject to any threat scenario, then evidence can be provided, and the corresponding claims can be considered justified. If the creators of the SAC cannot assign evidence to claims, then this is an indicator that either the argument did not reach an actionable point or that there is a need to go back and make development changes to satisfy the claims. For example, if we reach a claim that is not covered by any test report, then there might be a need to create test cases to cover that claim.

4.2.7 Case Quality Assurance. We consider two main aspects of quality assurance for SAC in CASCADE.

The first aspect is *completeness*, which refers to the level of coverage of the claims in each argumentation level of the SAC. Each level in CASCADE includes at least one strategy. For each strategy, we add at least one completeness claim that refines it. The role of this claim is to make sure that the strategy covers all and only the relevant claims on the argumentation level.

These completeness claims in CASCADE are used to gain confidence in the provided arguments. We have identified the strategy elements in the arguments to be attention points where these claims need to be made. The content of the completeness claims depends on the way of working in a company, i.e., the documented activities and procedures that are expected to be formed. This dependency should be documented in a context node that sets the scope for the completeness claim. For example, if a company uses pre-defined catalogues of known attack patterns, e.g., the **Common Attack Pattern Enumeration and Classification (CAPEC)** catalogue [2], then the context node should refer to the catalogue, and the claim would hence take the form: "All attack paths have been considered in accordance to document X," where the document refers to the catalogue. Another example is when a company defines a **Definition of Done (DoD)** for their activities, then the context nodes should refer to the corresponding DoD.

Additionally, working with cybersecurity always involves uncertainties. This is due to multiple factors, e.g., the limited knowledge of the capabilities of an attacker or unforeseen vulnerabilities. This again emphasizes the importance to set a scope for the completeness arguments. In addition to the context nodes, assumption nodes also need to be specified. These together provide the information needed to determine if the completeness claim is fulfilled or not within the scope of knowledge the company has, given the assumptions made by the adopting organization. It is also important to phrase the claims in a way that reflects its scope and limitations.

The second aspect is *evidence-confidence*, which indicates the level of certainty that a claim is fulfilled based on the provided evidence. This is used in each level of a security assurance case where at least one claim is justified by evidence. The evidence-confidence aspect is expressed as a claim, which takes the form: "The evidence provided for claim X achieves an acceptable level of confidence." What makes an acceptable level of confidence is defined in the context of the strategy. The confidence claim itself must be justified by evidence.

5 EXAMPLE CASE

To validate our approach, we apply CASCADE on the headlamp item use case from ISO/SAE-21434, which includes the headlamp system, navigation ECU, and gateway ECU. Figures 3, 4, and 5 present the resulted part of the SAC that correspond to each block of CASCADE. The file shapes in the top-right corners of each level in Figures 3, 4, and 5 indicate the requirement in ISO/SAE-21434, which mainly drives the argument in the corresponding level.

The strategies used to form the argument of the headlamp example case are created to demonstrate the structure of a security assurance case built with CASCADE. We believe that it would be beneficial to have an initial set of strategies for each block as a starting point. However, this requires an industrial application of CASCADE resulting multiple SACs, from which patterns can be extracted, and hence a set of strategies. This is a future work that we intend to do.

5.1 Top Claim

We start by constructing the Top Claim block consisting of:

- *C:1* the top security claim for the headlamp item, which is that the headlamp item is adequately secure.
- *Cnxt:1.1* a context node setting the scope of the claim. This scope is set by the headlamp item boundary description, which is available in the example case in ISO/SAE-21434.
- *Assmp:1.1* an assumption node, stating that the item is physically protected. Hence, we only consider the security of the software part of the headlamp item in our argument.

The context node refers to an external document, which is the item boundary and preliminary architecture of the headlamp item, as identified in ISO/SAE-21434.

5.2 White-hat Block

The White-hat block is presented in Figure 3. We first apply a strategy *S:1.1* to decompose our main claim based on the identified assets of the headlamp item. In our example, the main assets are the `CAN Frame`, which holds transmitted messages, and the `Firmware`, which includes control functions of the artefacts inside the headlamp system, e.g., the power switch. We create two claims *C:1.1.1* and *C:1.1.2* indicating that the two assets are acceptably secure. The strategy *S:1.1* is associated with a case quality-claim *CQ:1.1.1*, to ensure the completeness of the decomposition associated with it, and hence the completeness of the case in general.

The two identified assets are further decomposed into sub-assets. This decomposition is based on the components and functions the asset belongs to. For example, based on claim *C:1.1.2*, we apply strategy *S:1.2.2* and decompose the `CAN Frame` asset into a number of sub-assets. Moreover, we create security claims for the identified sub-assets: *C:1.2.2.1*, *C:1.2.2.2*, *C:1.2.2.3*, and *C:1.2.2.4*. Last, strategy *S:1.2.2* is associated with case quality-claim *CQ:1.2.2.1*.

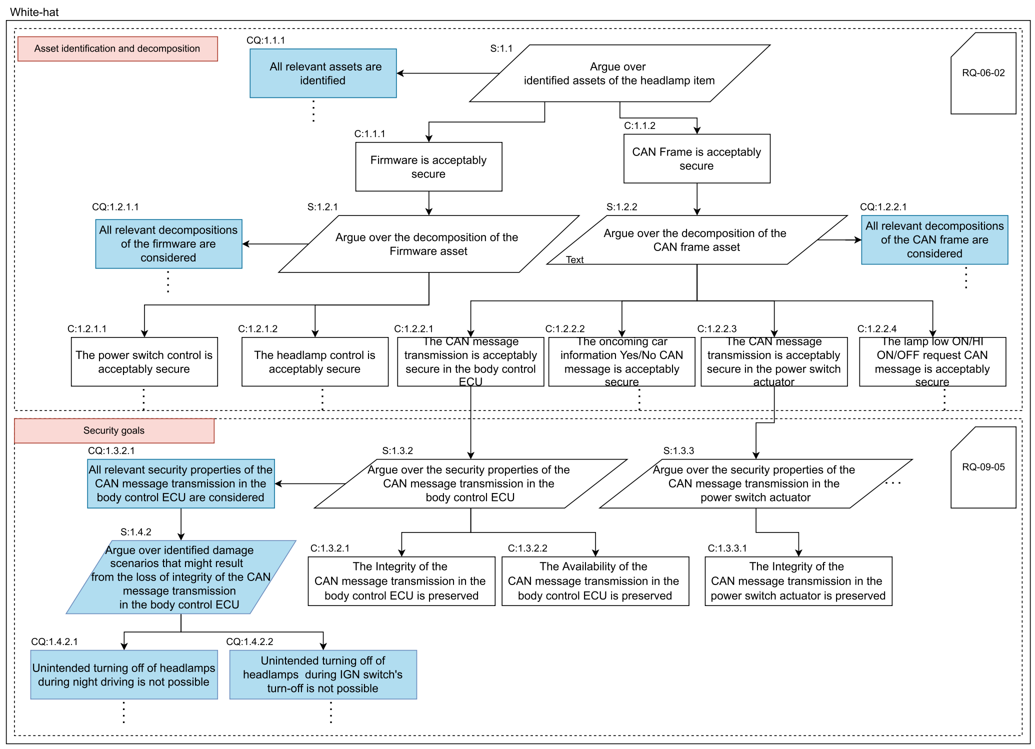

White-hat
Asset identification and decomposition
CQ:1.1.1
All relevant assets are identified
S:1.1
Argue over identified assets of the headlamp item
RQ-06-02
C:1.1.1
Firmware is acceptably secure
C:1.1.2
CAN Frame is acceptably secure
CQ:1.2.1.1
All relevant decompositions of the firmware are considered
S:1.2.1
Argue over the decomposition of the Firmware asset
S:1.2.2
Argue over the decomposition of the CAN frame asset
Text
CQ:1.2.2.1
All relevant decompositions of the CAN frame are considered
C:1.2.1.1
The power switch control is acceptably secure
C:1.2.1.2
The headlamp control is acceptably secure
C:1.2.2.1
The CAN message transmission is acceptably secure in the body control ECU
C:1.2.2.2
The oncoming car information Yes/No CAN message is acceptably secure
C:1.2.2.3
The CAN message transmission is acceptably secure in the power switch actuator
C:1.2.2.4
The lamp low ON/HI ON/OFF request CAN message is acceptably secure
Security goals
CQ:1.3.2.1
All relevant security properties of the CAN message transmission in the body control ECU are considered
S:1.3.2
Argue over the security properties of the CAN message transmission in the body control ECU
S:1.3.3
Argue over the security properties of the CAN message transmission in the power switch actuator
RQ-09-05
S:1.4.2
Argue over identified damage scenarios that might result from the loss of integrity of the CAN message transmission in the body control ECU
C:1.3.2.1
The Integrity of the CAN message transmission in the body control ECU is preserved
C:1.3.2.2
The Availability of the CAN message transmission in the body control ECU is preserved
C:1.3.3.1
The Integrity of the CAN message transmission in the power switch actuator is preserved
CQ:1.4.2.1
Unintended turning off of headlamps during night driving is not possible
CQ:1.4.2.2
Unintended turning off of headlamps during IGN switch's turn-off is not possible

Fig. 3. White-hat block of the headlamp use case. The file-shaped box in the top-right corner of each level indicates the requirement in ISO/SAE-21434 that drives the argument in that level.

At this point, we link the assets to the security goals (i.e., second level). To do so, we apply an argumentation strategy (e.g., *S:1.3.2*) to decompose the security claims of the sub-assets based on the CIA triad attributes. As a result, we create claims about the achievement of security goals such as *C:1.3.2.1*: "*The integrity of CAN message transmission in the body control ECU is preserved.*" To make sure that we cover the relevant properties, we create a case quality-claim *CQ:1.3.2.1* and argue (*S:1.4.2*) about possible damage scenarios that could invalidate the claims. Accordingly, we create case quality-claims that make sure that these damage scenarios do not happen. An example of these claims is *CQ:1.4.2.1*: "*Unintended turning off of headlamps during night driving is not possible.*" At this point, the claim is fine-grained enough and counts as a security goal. Next, we create the black-hat block.

5.3 Black-hat Block

Here, we argue over the threat scenarios that could lead to compromising a security goal.

Figure 4 shows a part of the black-hat block of the headlamp use case. This part is associated with the claim about achieving a security goal *C:1.4.2.1*, which is shown in Figure 3. We start by creating strategy *S:1.5.1* to argue over the used threat model. If, e.g., STRIDE is used as a threat model, then the strategy would be to create a claim for each STRIDE category. In our example case, we create claim *C:1.5.1.1*: "*Spoofing of a signal leading to loss of integrity of the CAN message of* Lamp Request *signal of power switch actuator ECU is not possible.*" To ensure the completeness of the case, we further associate the strategy *S:1.5.1* with a case quality-claim (*CQ:1.5.1.1*).

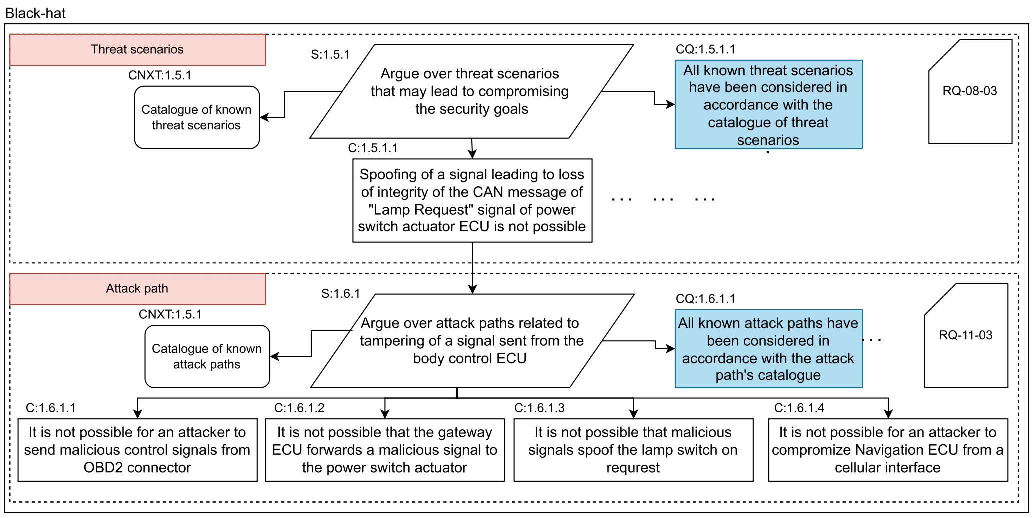

Black-hat
Threat scenarios
S:1.5.1
Argue over threat scenarios that may lead to compromising the security goals
CNXT:1.5.1
Catalogue of known threat scenarios
CQ:1.5.1.1
All known threat scenarios have been considered in accordance with the catalogue of threat scenarios
RQ-08-03
C:1.5.1.1
Spoofing of a signal leading to loss of integrity of the CAN message of "Lamp Request" signal of power switch actuator ECU is not possible
Attack path
S:1.6.1
Argue over attack paths related to tampering of a signal sent from the body control ECU
CNXT:1.5.1
Catalogue of known attack paths
CQ:1.6.1.1
All known attack paths have been considered in accordance with the attack path's catalogue
RQ-11-03
C:1.6.1.1
It is not possible for an attacker to send malicious control signals from OBD2 connector
C:1.6.1.2
It is not possible that the gateway ECU forwards a malicious signal to the power switch actuator
C:1.6.1.3
It is not possible that malicious signals spoof the lamp switch on requrest
C:1.6.1.4
It is not possible for an attacker to compromize Navigation ECU from a cellular interface

Fig. 4. Black-hat block of the headlamp use case. The file-shaped box in the top-right corner of each level indicates the requirement in ISO/SAE-21434 that drives the argument in that level.

At this point, our claims become more concrete as we have a specific item, asset, container component, security property, damage scenario, and threat scenario. We use the analysis of attack paths to further decompose and populate the example case. We apply strategy *S:1.6.1* to argue over the attacks and create attack path claims. The resulting claims negate the possibility for an attack path to take place, e.g., *C:1.6.1.4* "*It is not possible for an attacker to compromise the Navigation ECU from a cellular interface.*" As for all strategies in CASCADE, we associate the strategy used in the attack path with a case quality-claim (*CQ:1.6.1.1*) to ensure the completeness of the case. To set a scope for the case quality-claims, we add the context nodes (*CNTX:1.5.1*) and (*CNTX:1.6.1*). These nodes refer to pre-defined catalogues of known threat scenarios and attack paths and are used to make sure that they have been considered, and hence, gain confidence in the completeness of the SAC.

5.4 Resolver and Evidence Blocks

In this stage, we create the resolver block by investigating ways to resolve the attack paths based on a risk assessment and creating requirements for the intended risk treatments.

Figure 5 shows a part of the resolver block for our example case associated with the attack path *C:1.6.1.4*. The outcome of the risk assessment would be to accept, mitigate, transfer, or solve the risk. When a risk is accepted, then there is no need to further decompose the claim. In the other cases, a strategy (*S:1.7.1*) to decompose the risk of an attack path has to be created. In our example, we create claim *C:1.7.1.1* to mitigate the risk as follows: "*The risk of an attacker compromising the Navigation ECU from a cellular interface is reduced.*"

This leads to the stage where we argue on the requirements to specify how the risk has to be reduced or mitigated. An example of a requirement claim is *C:1.8.1.1*: "*The received data is verified if it is sent from a valid entity.*"

Figure 5 also shows the evidence block that provides examples of evidence to justify the requirement's claims. The evidence (e.g., *E:1.1*) is supported by case quality-evidence (e.g., *CQE:1.1*), which, in turn, is complemented with case quality-claims (e.g., *CQ:1.8.1.1*) to confidently justify the associated requirement claims.

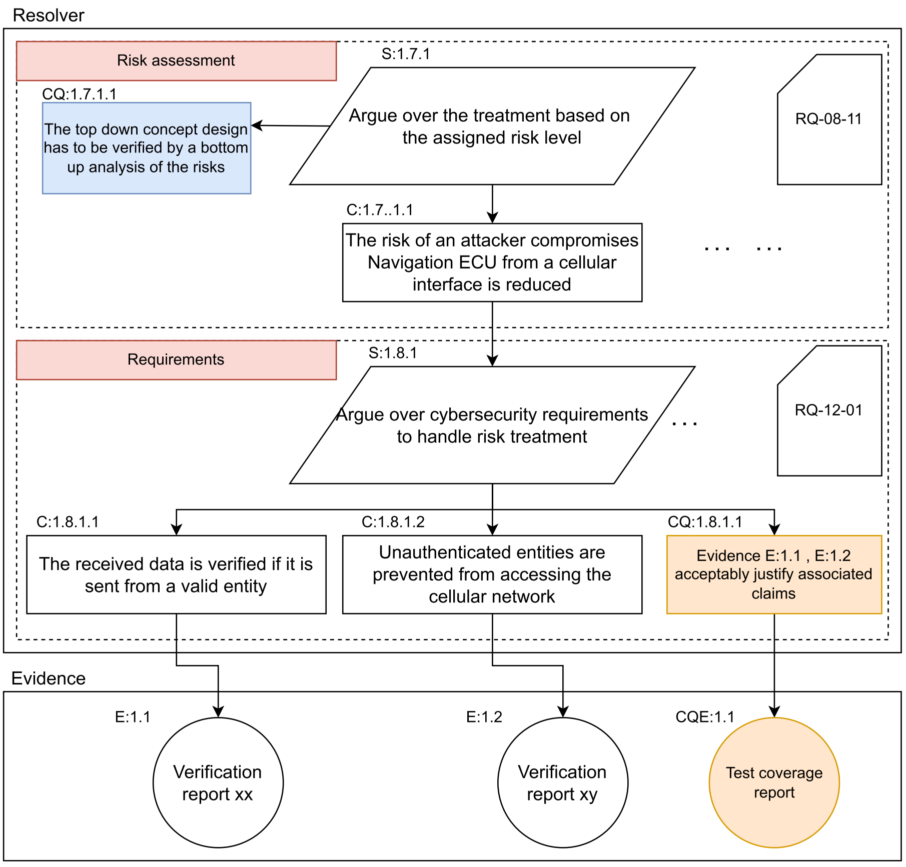

Resolver
Risk assessment
S:1.7.1
Argue over the treatment based on the assigned risk level
CQ:1.7.1.1
The top down concept design has to be verified by a bottom up analysis of the risks
RQ-08-11
C:1.7..1.1
The risk of an attacker compromises Navigation ECU from a cellular interface is reduced
. . . . . .
Requirements
S:1.8.1
Argue over cybersecurity requirements to handle risk treatment
. . .
RQ-12-01
C:1.8.1.1
The received data is verified if it is sent from a valid entity
C:1.8.1.2
Unauthenticated entities are prevented from accessing the cellular network
CQ:1.8.1.1
Evidence E:1.1 , E:1.2 acceptably justify associated claims
Evidence
E:1.1
Verification report xx
E:1.2
Verification report xy
CQE:1.1
Test coverage report

Fig. 5. Resolver and evidence blocks of the headlamp use case. The file-shaped box in the top-right corner of each level indicates the requirement in ISO/SAE-21434 that drives the argument in that level.

5.5 Generic Sub-case Block

Figure 6 shows the last block in our example; the generic sub-case. This block includes claims that are relevant to the example case but are not specific to it. For example, claim *C:G2* states that "*The company has a security-aware culture*," which is supported by two evidence statements: *E:G2.1* and *E:G2.2* to prove that the employees of the company were given a security training. The evidence provided to support claim *C:G2* in this example are used to demonstrate the type of evidence that can be used in the generic sub-case block. In a real-world scenario, these evidence might not suffice and additional contexts, assumptions, and evidence might need to be provided to justify the security culture of the company, e.g., in terms of tradeoffs that have to be made between spending resources for security or other activities. When all the evidence are provided, it is important to assess the confidence they provide and create a confidence CQ-claim and support it with relevant evidence. Similar to other blocks, the generic sub-case block might include strategies (e.g., *S:G1*) to break down claims. Moreover, these strategies are associated with case quality-claims (e.g., *QC:G.1.1*), as shown in Figure 6.

6 MAPPING TO ISO/SAE-21434

This section provides a set of guidelines that are extracted during the mapping exercise from the requirements and work products of ISO/SAE-21434 to elements of security assurance cases

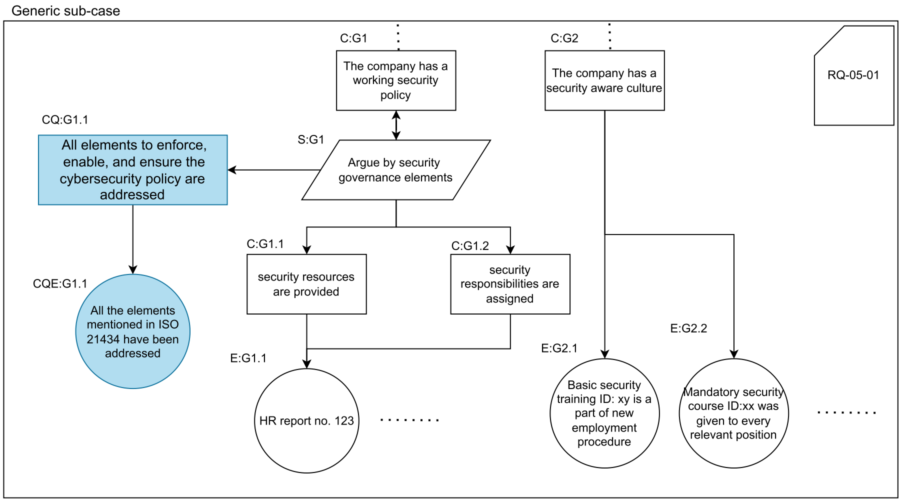

Generic sub-case
C:G1
The company has a working security policy
C:G2
The company has a security aware culture
RQ-05-01
CQ:G1.1
All elements to enforce, enable, and ensure the cybersecurity policy are addressed
S:G1
Argue by security governance elements
CQE:G1.1
All the elements mentioned in ISO 21434 have been addressed
C:G1.1
security resources are provided
C:G1.2
security responsibilities are assigned
E:G1.1
HR report no. 123
E:G2.1
Basic security training ID: xy is a part of new employment procedure
E:G2.2
Mandatory security course ID:xx was given to every relevant position

Fig. 6. Generic sub-case block of the headlamp use case. The file-shaped box in the top-right corner indicates the requirement in ISO/SAE-21434 that drives the argument.

according to the GSN notation. The mapping exercise is based on the **FDIS (Final Draft International Standard)** version of ISO/SAE-21434.

Figure 7 shows a mapping workflow that can be used to map each ISO/SAE-21434 item to a CASCADE element. Given an ISO/SAE-21434 item, it is first classified into either a *Requirement* or *Work Product*:

- Requirement: As a general rule of thumb, requirements form the argument part of the SAC. A requirement is first checked whether or not it includes an implication. If an implication (A implies B) is included, then it means that there are two possible outcomes: "not A or B." Here, the corresponding SAC element is either an assumption of the negation of A or a requirement B. For example, let us consider the following requirement: "If the cybersecurity activity X is applied, then a rationale R shall be provided." If the cybersecurity activity X is not applied, then this requirement is mapped to an *Assumption*. If otherwise X is in fact applied (i.e., that a rationale R is provided), then the requirement is checked whether it is a process- or product-oriented.
  If the requirement is process-oriented (e.g., refers to the overall cybersecurity management or culture in the organization), then it is considered as a *Claim within the Generic sub-case*. Otherwise, the requirement is considered as product-oriented. At this point, the product-oriented requirement is checked whether it targets the quality of the item in question.
  If the requirement is targeting the quality, then it is mapped to a ***Case Quality-Claim (CQ Claim)***. Otherwise, it is mapped to a *Claim*. For instance "An assessment shall judge whether the available evidence provides confidence in the cybersecurity of the item" would be mapped into a confidence claim. In contrast, a requirement such as "The validation of the item shall confirm that all the risks identified during the phase X are handled" would be a case quality-claim for completeness.
- Work Product: Work products form the evidence part of the resulting SAC. The work product can be either a (i) document that sets a scope to the cybersecurity work, which in this case is considered as *Context* or (ii) document that includes sources for, e.g., cybersecurity monitoring, which in this case is considered as an *Evidence*.

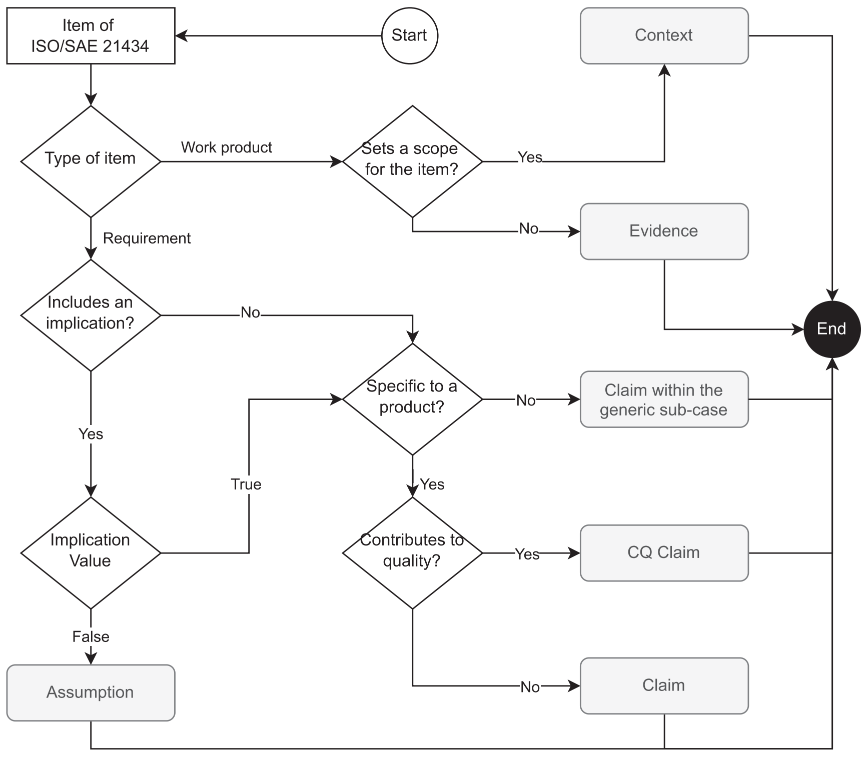


Fig. 7. The workflow of mapping ISO/SAE-21434 items to elements of CASCADE (in grey).

The mapping of the ISO/SAE-21434 requirements into different blocks of CASCADE is rather a straightforward task. The used rules are the following:

- requirements about security goals and asset identification fall into the white-hat block,
- requirements about risk assessment, e.g., identification and analysis of attack paths, attack feasibility, or impact rating, would fall into the black-hat block. Similarly, requirements about vulnerability analysis also fall into the black-hat block, and
- requirements about risk treatment fall into the resolver block.

Table 1 shows the requirements and work products in ISO/SAE-21434 that drive the arguments in each of CASCADE block and level.

More detailed results of the conducted mapping between CASCADE and ISO/SAE-21434 items are reported in Appendix A. The first two columns in the table of Appendix A refer to item ID and type from the ISO/SAE-21434 standard. The third column shows the corresponding SAC element, which is then specified for CASCADE in column 4. Columns 5 and 6 include the corresponding block and level for the item in CASCADE's structure. Finally, the last column shows the work products that support the elements that emerge from the corresponding requirement items, as per the specification of ISO/SAE-21434.

7 VALIDATION

To evaluate our approach, we reached out to a security expert from the cybersecurity team at Volvo Trucks, which is a leading OEM that manufactures trucks in Sweden. We conducted several sessions during the development of CASCADE where we discussed the approach, its limitations, and

Table 1. Requirements and Work Products from ISO/SAE 21434 that Are Relevant to the Arguments

CASCADE Block	CASCADE Level	Requirements and Work products
Top Claim		RQ-11-01, RQ-15-05, RQ-15-06
Generic sub-case		RQ-05-01:RQ-05-15, RQ-06-01, RQ-06-04, RQ-06-23, RQ-06-24, RQ-07-01, RQ-08-01, RQ-08-04:RQ-08-07, RQ-09-01:RQ-09-04, RQ-10-19:RQ-10-21, RQ-13-06, RQ-15-01:RQ-15-03, WP-05-01:WP-05-05, WP-07-01, WP-10-07, WP-10-08, WP-13-03, WP-15-01
White hat	Asset ID	RQ-06-02, RQ-07-03, RQ-08-02
	Security Goals	RQ-06-03:RQ-06-17, RQ-06-19, RQ-06-20, RQ-06-22, RQ-06-25:RQ-06-27, RQ-07-02, RQ-07-04, RQ-09-05, RQ-09-07, RQ-09-09, RQ-11-01, RQ-11-02, RQ-14-01
Black hat	Threat Scenarios	RQ-08-03, RQ-08-08, RQ-11-03, RQ-13-03, WP-13-02
	Attack Paths	RQ-08-09, RQ-08-10
Resolver	Risk Assessment	RQ-07-05:RQ-07-07, RQ-08-11, RQ-08-12, RQ-09-06, RQ-09-08, RQ-10-09:RQ-10-11, RQ-10-13, RQ-10-16:RQ-10-18, RQ-11-04,
	Security Requirements	RQ-06-29, RQ-07-08, RQ-09-10:RQ-09-12, RQ-10-01:RQ-10-08, RQ-10-12, RQ-10-14, RQ-10-15, RQ-12-01, RQ-13-01:RQ-13-05, RQ-15-04, RQ-15-07

In each CASCADE block colons indicate ranges of requirements and work products.

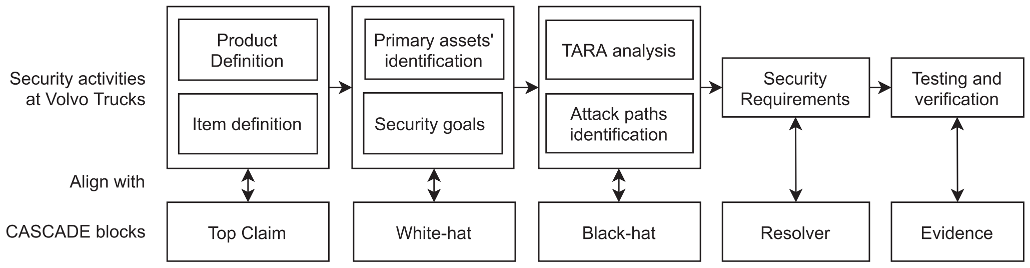


Fig. 8. Mapping of the company's security activities to CASCADE blocks.

possible enhancements. When the approach was fully developed, we conducted a final evaluation session with the expert. We first discussed the way of working of the company when it comes to security activities and security assurance. We used the headlamp example from ISO/SAE-21434 as a context for this discussion. We then presented our approach and the example case for the headlamp item. The expert evaluated the approach by discussing how the overall structure of a SAC should look from the company's perspective to satisfy the requirement for security cases in ISO/SAE-21434 and mapping the different elements of the example case to the internal way of working. The expert also provided insights on how to further enhance the approach.

Figure 8 shows the different security activities at the company along with the corresponding CASCADE block. A link between an activity and a block indicates that the outcomes of the activity are used to create the SAC elements in the corresponding block.

Software products at Volvo Trucks contain both on-board and off-board parts. The off-board parts establish the communication between the vehicles and the back-end systems. For example, the diagnostics services receive data from the vehicle's ECUs and store and use it in a back-end system. The on-board parts are software components installed in the ECUs of the vehicle, e.g., the engine control and the head-up display unit. These parts are divided into items to facilitate the security-related analysis. The items of the off-board systems can be seen as individual services that communicate with the vehicles, whereas the on-board items are end-user functionalities, e.g., external lighting and automated parking assistance. To argue about the security of a complete product, both off-board and on-board items have to be considered. Hence, if the company wants to adopt CASCADE to create a SAC for a complete product, then the top claim block would contain claims for the individual items of that product. The Generic sub-case block of CASCADE helps to remove the redundancy of arguments and evidence applicable to different items.

The assets of a product are identified by considering damage scenarios on the items. In general, these assets can be generalized into the following categories:

- Vehicle's functionality (the attackers want to use the vehicle or tamper with the vehicle's functionality for their own purpose or impede the rightful user from utilizing the vehicle functionality);
- Information (the attackers want to gain access to sensitive information); and
- Brand (the attackers want to discredit the brand and/or credit themselves).

The identified assets are further categorized into primary and secondary assets in accordance with the definitions in ISO-27005 [17]. Considering the headlamp example case, two possible damage scenarios would be "Losing the headlamp will drastically reduce the driver's sight and the vehicle's visibility, which may result in a severe accident" and "Applying the headlamp at incorrect times could dazzle other vehicles, which may increase the risk of an accident." These lead to considering the headlamp functionality as a primary asset.

Then, relevant security attributes for the primary asset are identified and security goals are derived:

- The integrity of the headlamp control functionality shall be preserved.
- The availability of the headlamp control functionality shall be preserved.

The identification of assets and security goals corresponds to the white-hat block of CASCADE, as shown in Figure 8. These goals only take relevant security properties into consideration, i.e., integrity and availability. Hence, other properties such as confidentiality and authenticity are not considered. During the concept design phase, a **Threat Assessment and Remediation Analysis (TARA)** is performed on the item's primary assets using STRIDE, which will result in cybersecurity requirements on certain components that are considered as supporting assets. These requirements are converted to claims in the black-hat block of CASCADE, as shown in Figure 8.

After that, attack path analyses are performed bottom-up using an attack library, including but not limited to:

- Intended over-the-air connection (e.g., 2.5G, 3G, 4G, or 5G, Wi-Fi, WPAN, Bluetooth, IrDA, Wireless USB, and UBW);
- Intended physical connection points (e.g., OBD, USB, and CD-Rom);
- Unintended over-the-air disturbances (e.g., Radar, Laser, electro-magnetic, microwaves, infra-waves, ultrasound, and infra-sound);
- Unintended physical connection points (e.g., ECU, network, sensors, and actuators).

These attack paths are also expressed as claims in the Black-hat block. Then the component design will be started considering all requirements, including cybersecurity. The components and systems are described in "product descriptions." These are verified against the requirements, including cybersecurity. The cybersecurity requirements in the product description correspond to the requirements of the Resolver block in CASCADE. The components and systems are then tested against the product descriptions, and the test results are considered as evidence in CASCADE.

Other SAC requirements also emerged from the discussion with the security expert. For instance, they emphasized the need to validate that production, operation, service, and decommissioning are all adequately handled. We believe that this would be covered by QA claims in the resolver block. Another requirement is that the product along with the SAC is maintained throughout the life-cycle. This is not covered by CASCADE, and we consider it to be an important complementary aspect for future work. In particular, we will be looking into methods to ensure traceability between the elements of SACs and the corresponding development artefacts. This traceability allows impact analysis for maintaining SACs. Last, the expert stressed that it is important to argue that the performed product work is adequate with respect to cybersecurity policies and practices adopted by the company. We believe that to cover this, both product-related arguments should be in place, as well as claims about the adequacy of the applied security policy, which is covered by the generic sub-case block of CASCADE.

To get a deeper insight of how applicable CASCADE would be in an industrial context and what the strengths and weaknesses of it are, we extended our evaluation and included another large OEM in Sweden, Volvo cars.

We did an open discussion session with a security expert at the company where we presented CASCADE and the example case. Here, we present the main discussion points and the expert's input:

- The work of work products is done in an iterative manner at Volvo Cars. In the design phase, for instance, there sometimes could be uncertainties, which would be cleared in the next phases, hence there is a need to go back and rework on work packages. This might also be the case within a phase in different development iterations. The interviewee stated that CASCADE allows the creation of SAC *on the run*, referring to the structure of CASCADE being aligned with the security activities and the iterative way of working applied in the company. Hence, CASCADE does not require complete artefacts and work products to be in place to start building the case, but can rather be built and compiled progressively during the product development phases, e.g., concept, design, and implementation. During these phases the SAC provides an overview of the security of the ongoing work, which supports:
  - Early identification of potential issues and decrease the cost of fixing these issues.
  - Simplifying the change impact analysis to make decisions regarding the integration of new development in terms of: *(i)* what needs to be addressed in regards to updated artefacts/work products, e.g., TARA, concept, V&V needs; *(ii)* what needs to be updated in the assurance case; *(iii)* the needed reviews and re-assessments needed to account for the change.
  - Gaining a faster understanding of the security impact and posture of a change to support decisions on deployment
- Quality of security artefacts, is done at the company as reviews. There are reviews to make sure that the analysis is complete and the scope is covered (boundaries). The expert agrees that the case should reflect that, which is done through the case quality assurance part of CASCADE. SACs have the potential to be used in multiple usage scenarios [23]. Hence, there are many stakeholders to SACs. These stakeholders require different abstraction levels of the SAC. For some stakeholders the detailed case quality arguments and evidence might not be required, but rather an abstracted view where the quality of the case is indicated and visualized using a metric. The complete case quality arguments must, however, be in place to measure the completeness of the SAC's argument. Distinguishing the case quality-claims in CASCADE is good, as it enables the possibility for the abstraction. However, there should be a property on the case level that directly indicates the quality of the case rather than having to go through all the corresponding elements. This has been implemented in some of the tools used for creating and managing assurance cases, e.g., Reference [24], and is considered a state of practice.

- There is no clear distinction between what is specific to CASCADE and what is a one-to-one mapping to the structure of ISO/SAE-21434. Moreover, it is unclear whether the approach would sufficiently cover the requirements of the standard. A more in-depth analysis between the standard and CASCADE is needed, according to the expert.

Areas of Improvement

The mapping of ISO/SAE-21434 has enabled us to pinpoint areas of improvement as well as strong points in CASCADE. In this section, we discuss our main findings of the evaluation cycle.

- Re-usability: The separation of item-related and generic arguments in CASCADE enabled us to consider the re-usability of the claims already while creating them, i.e., in an early stage. The standard does not explicitly distinguish between *(i)* product-related requirements and work products; and *(ii)* process-related requirements and work products. However, by deriving claims from the requirement and determining the CASCADE corresponding block, this could easily be achieved. Re-using claims is very important to reduce the overhead of creating SAC for different products.
  Re-use of process-related claims is, however, not the only type of re-usability possible in creating SAC. The other type is the re-use of product-related claims that are applicable to multiple items. For example, if there are multiple items in a vehicle that are connected to the outside world through a gateway, then a security argument about that gateway would be applicable to all the connected items and should be re-used in each of their corresponding security arguments. This type of re-usability is only possible to achieve when the architecture of the system is known and the relations and dependencies of different items are known.
  To improve CASCADE, we should restructure the generic sub-case block to be able to cover both types of re-usability. Additionally, we need to establish the techniques and mechanisms needed to shift parts of arguments between different SAC.
- Uncertainty: Throughout the mapping activity, we encountered several occasions where there were uncertainties. This is the case when a requirement includes a condition in it, e.g., in the form of an if statement. As discussed in Section 6, we consider this type of requirement to either lead to an assumption or to a claim (we have mapped them to assumptions in Appendix A).
  This leads us to the need for capturing these uncertainties in a SAC at an early stage of its creation. As SAC should follow the development life-cycle, it is important to make sure to tag the parts of a given SAC that need further development. Uncertainties certainly lead to incomplete arguments in the early stages of SAC creation. To resolve this issue, we could introduce a mechanism to tag a scope of a given argument as incomplete, as well as provide alternative solutions based on a certain condition. In our case, that would be the condition in the requirement. This would be similar to how behavioral views (sequence diagrams, in particular) in UML handle alternatives.
  This change would affect the process of creating a SAC. However, the end result would be the same once the development of the item in question is finalized and the SAC is fully created. It might, however, also be interesting to keep a history log of the choices that have been made in the process. This log can be reflected in the case to show arguments that were subject to decision points during the development process.
- Adequacy: The items in the standard are not enough to cover the case quality claims and evidence. In CASCADE, we have a rule that every level of argumentation (when a claim is broken down into sub-claims) must be examined for completeness, and every piece of evidence must be examined for confidence. However, the items of the standard do not suffice to cover these quality needs.

Additionally, the creation of a SAC requires introducing contexts and assumptions that are not necessarily captured by the standard.
Moreover, the items of the standard might not even be enough to claim the required level of security according to an organization's standard. Hence, conforming with the standard and reflecting the work in a SAC does not mean that the SAC is complete. There is a need for a definition of done for the SAC work, which would have to be considered as a context for the top claim.

As a summary of the validation of this work, we showed that CASCADE is well aligned with the internal way of working at automotive OEM's, which makes it suitable for the creation of SAC on the run and during the development life-cycle. We also found that the generic sub-case and quality blocks help to serve the abstraction and completeness requirements of the evaluating companies. We have identified points of improvement in the approach, especially when we attempted to map the requirements and work products of ISO/SAE-21434 to elements and structure of CASCADE. These include the need for a concrete mechanism for reusing arguments, handling the uncertainty in the requirements, and the need to have a definition of done for the outcome. These will be the basis for future work.

8 CONCLUSION AND FUTURE WORK

In this work, we presented CASCADE, an asset-driven approach for the creation of security assurance cases with built-in quality assurance. CASCADE is geared towards automotive companies that have the need to conform with the upcoming ISO/SAE-21434 security standard.

We illustrated CASCADE using an example case from ISO/SAE-21434 and validated it at two industrial OEMs. We also mapped the requirements and work products of the standard to elements of the approach and evaluated its capability to include these items. Additionally, we synthesized the knowledge gained during the mapping activity and created a guideline for practitioners who want to conduct a similar activity.

As a future work, we plan to further enhance the approach to take into consideration the maintenance of SAC. We will also work on defining the mechanisms and techniques required for the re-usability and the quality assurance of the outcome of CASCADE. Additionally, we plan on incorporating additional requirements sources to better cover the needs of the automotive industry.

APPENDICES

A ISO/SAE-21434–CASCADE MAPPING

Item ID	Item type	Corresponding SAC element	Corresponding CASCADE element	Corresponding CASCADE block	Corresponding CASCADE level	Supporting WP
RQ-05-01	Requirement	Claim	Claim	Generic Sub-case		WP-05-01
RQ-05-02	Requirement	Claim	Claim	Generic Sub-case		WP-05-01
RQ-05-03	Requirement	Claim	Claim	Generic Sub-case		WP-05-01
RQ-05-04	Requirement	Claim	Claim	Generic Sub-case		WP-05-01
RQ-05-05	Requirement	Claim	Claim	Generic Sub-case		WP-05-01
RQ-05-06	Requirement	Claim	Claim	Generic Sub-case		WP-05-01
RQ-05-07	Requirement	Claim	Claim	Generic Sub-case		WP-05-02
RQ-05-08	Requirement	Claim	Claim	Generic Sub-case		WP-05-02
RQ-05-09	Requirement	Claim	Claim	Generic Sub-case		WP-05-02
RQ-05-10	Requirement	Claim	Claim	Generic Sub-case		WP-05-01
RQ-05-11	Requirement	Claim	Claim	Generic Sub-case		WP-05-03
RQ-05-12	Requirement	Claim	Claim	Generic Sub-case		WP-05-01
RQ-05-13	Requirement	Claim	Claim	Generic Sub-case		WP-05-04
RQ-05-14	Requirement	Claim	Claim	Generic Sub-case		WP-05-04
RQ-05-15	Requirement	Claim	Claim	Generic Sub-case		WP-05-05
WP-05-01	Work product	Evidence	Evidence	Generic Sub-case		
WP-05-02	Work product	Evidence	Evidence	Generic Sub-case		
WP-05-03	Work product	Evidence	Evidence	Generic Sub-case		
WP-05-04	Work product	Evidence	Evidence	Generic Sub-case		
WP-05-05	Work product	Evidence	Evidence	Generic Sub-case		
RQ-06-01	Requirement	Claim	CQ claim	Generic Sub-case		WP-06-01
RQ-06-02	Requirement	Claim	CQ claim	White hat	Asset ID	WP-06-01
RQ-06-03	Requirement	Claim	CQ claim	White hat	Security goals	WP-06-01
RQ-06-04	Requirement	Claim	CQ claim	Generic Sub-case	Security goals	WP-06-01
RQ-06-05	Requirement	Claim	Claim	White hat	Security goals	WP-06-01
RQ-06-06	Requirement	Claim	CQ claim	White hat	Security goals	WP-06-01
RQ-06-07	Requirement	Claim	CQ claim	White hat	Security goals	WP-06-01
RQ-06-08	Requirement	Claim	CQ claim	White hat	Security goals	WP-06-01
RQ-06-09	Requirement	Claim	CQ claim	White hat	Security goals	WP-06-01
RQ-06-10	Requirement	Claim	CQ claim	White hat	Security goals	WP-06-01
RQ-06-11	Requirement	Context	Context	White hat	Security goals	WP-06-01
RQ-06-12	Requirement	Context	Context	White hat	Security goals	WP-06-01
RQ-06-13	Requirement	Context	Context	White hat	Security goals	WP-06-01
RQ-06-14	Requirement	Claim	CQ claim	White hat	Security goals	WP-06-01
RQ-06-15	Requirement	Claim	CQ claim	White hat	Security goals	WP-06-01
RQ-06-16	Requirement	Claim	Claim	White hat	Security goals	WP-06-01
RQ-06-17	Requirement	Claim	CQ claim	White hat	Security goals	WP-06-01
RQ-06-18[3]	Requirement					WP-06-02
RQ-06-19	Requirement	Claim	CQ claim	White hat	Security goals	WP-06-03
RQ-06-20	Requirement	Claim	CQ claim	White hat	Security goals	WP-06-03
RQ-06-21	Requirement	Claim	CQ claim	Evidence		WP-06-03
RQ-06-22	Requirement	Context	Context	White hat	Security goals	WP-06-03
RQ-06-23	Requirement	Claim	Claim	Generic sub-case		WP-06-03
RQ-06-24	Requirement	Claim	Claim	Generic sub-case		WP-06-03
RQ-06-25	Requirement	Context	Context	White hat	Security goals	WP-06-03
RQ-06-26	Requirement	Claim	CQ claim	White hat	Security goals	WP-06-03
RQ-06-27	Requirement	Claim	Claim	White hat	Security goals	WP-06-03
RQ-06-28	Requirement	Assumption	Assumption	Top claim		WP-06-04
RQ-06-29	Requirement	Claim	CQ claim	Resolver	Security requirements	WP-06-04
WP-06-01	Work Product	Evidence	Evidence			
WP-06-02[4]	Work Product					
WP-06-03	Work Product	Evidence	Evidence			
WP-06-04	Work Product	Evidence	Evidence			
RQ-07-01	Requirement	Claim	Claim	Generic sub-case		WP-07-01 WP-07-05

(Continued)

[3]RQ-06-18 is the requirement to create the cybersecurity case.
[4]WP-06-02 is the work-product referring to the cybersecurity case.

Continued

Item ID	Item type	Corresponding SAC element	Corresponding CASCADE element	Corresponding CASCADE block	Corresponding CASCADE level	Supporting WP
RQ-07-02	Requirement	Claim	CQ claim	White hat	Security goals	WP-07-02
WP-07-01	Work Product	Context	Context	Generic sub-case		
WP-07-02	Work Product	Evidence	Evidence			
RQ-07-03	Requirement	Claim	Claim	White hat	Asset ID	
WP-07-03	Work Product	Evidence	Evidence			
RQ-07-04	Requirement	Claim	Claim	White hat	Security goals	
WP-07-04	Work Product	Evidence	Evidence			
RQ-07-05	Requirement	Claim	CQ claim	Resolver	Risk assessment	
RQ-07-06	Requirement	Claim	CQ claim	Resolver	Risk assessment	
RQ-07-07	Requirement	Claim	Claim	Resolver	Risk assessment	
RQ-07-08	Requirement	Assumption	Assumption	Resolver	Security requirements	
WP-07-05	Work Product	Evidence	Evidence			
RQ-08-01	Requirement	Claim	Claim	Generic sub-case		WP-08-01
RQ-08-02	Requirement	Assumption	Assumption	White hat	Asset ID	WP-08-02
WP-08-01	Work Product	Evidence	Evidence			
WP-08-02	Work Product	Evidence	Evidence			
RQ-08-03	Requirement	Claim	Claim	Black hat	Threat scenarios	WP-08-03
WP-08-03	Work Product	Evidence	Evidence			
RQ-08-04	Requirement	Claim	CQ claim	Generic sub-case		WP-08-04
RQ-08-05	Requirement	Assumption	Assumption	Generic sub-case		WP-08-04
RQ-08-06	Requirement	Claim	CQ claim	Generic sub-case		WP-08-04
RQ-08-07	Requirement	Claim	CQ claim	Generic sub-case		WP-08-04
WP-08-04	Work Product	Evidence	Evidence			
RQ-08-08	Requirement	Claim	Claim	Black hat	Threat scenarios	WP-08-05
RQ-08-09	Requirement	Claim	Claim	Black hat	Attack paths	
WP-08-05	Work Product	Evidence	Evidence			
RQ-08-10	Requirement	Claim	CQ claim	Black hat	Attack paths	WP-08-06
WP-08-06	Work Product	Evidence	Evidence			
RQ-08-11	Requirement	Claim	CQ claim	Resolver	Risk assessment	WP-08-07
WP-08-07	Work Product	Evidence	Evidence			
RQ-08-12	Requirement	Claim	CQ claim	Resolver	Risk assessment	WP-08-08
WP-08-08	Work Product	Evidence	Evidence			
RQ-09-01	Requirement	Claim	Claim	Generic sub-case		WP-09-01
RQ-09-02	Requirement	Claim	Claim	Generic sub-case		WP-09-01
RQ-09-03	Requirement	Claim	Claim	Generic sub-case		WP-09-01
RQ-09-04	Requirement	Claim	Claim	Generic sub-case		WP-09-01
WP-09-01	Work Product	Evidence	Evidence			
RQ-09-05	Requirement	Claim	CQ claim	White hat	Security goals	WP-09-02
RQ-09-06	Requirement	Claim	CQ claim	Resolver	Risk assessment	WP-09-03
RQ-09-07	Requirement	Assumption	Assumption	White hat	Security goals	WP-09-04
RQ-09-08	Requirement	Context	Context	Resolver	Risk assessment	WP-09-05
RQ-09-09	Requirement	Claim	CQ claim	White hat	Security goals	WP-09-06
WP-09-02	Work Product	Evidence	Evidence			
WP-09-03	Work Product	Evidence	Evidence			
WP-09-04	Work Product	Evidence	Evidence			
WP-09-05	Work Product	Evidence	Evidence			
WP-09-06	Work Product	Evidence	Evidence			
RQ-09-10	Requirement	Claim	Claim	Resolver	Security requirements	WP-09-07
RQ-09-11	Requirement	Context	Context	Resolver	Security requirements	WP-09-07
RQ-09-12	Requirement	Claim	CQ claim	Resolver	Security requirements	WP-09-08
WP-09-07	Work Product	Evidence	Evidence			
WP-09-08	Work Product	Evidence	Evidence			
RQ-10-01	Requirement	Claim	CQ claim	Resolver	Security requirements	WP-10-01
RQ-10-02	Requirement	Claim	CQ claim	Resolver	Security requirements	WP-10-01
RQ-10-03	Requirement	Claim	CQ claim	Resolver	Security requirements	WP-10-01
RQ-10-04	Requirement	Claim	Claim	Resolver	Security requirements	WP-10-01
RQ-10-05	Requirement	Claim	Claim	Resolver	Security requirements	WP-10-02
RQ-10-06	Requirement	Assumption	Assumption	Resolver	Security requirements	WP-10-02
RQ-10-07	Requirement	Claim	CQ claim	Resolver	Security requirements	WP-10-03
RQ-10-08	Requirement	Claim	CQ claim	Resolver	Security requirements	WP-10-03
RQ-10-09	Requirement	Claim	Claim	Resolver	Risk assessment	WP-10-04
RQ-10-10	Requirement	Claim	CQ claim	Resolver	Risk assessment	WP-10-04

(Continued)

Continued

Item ID	Item type	Corresponding SAC element	Corresponding CASCADE element	Corresponding CASCADE block	Corresponding CASCADE level	Supporting WP
RQ-10-11	Requirement	Claim	CQ claim	Resolver	Risk assessment	WP-10-04
RQ-10-12	Requirement	Claim	CQ claim	Resolver	Security requirements	WP-10-06
RQ-10-13	Requirement	Claim	Claim	Resolver	Risk assessment	WP-10-05
RQ-10-14	Requirement	Assumption	Assumption	Resolver	Security requirements	WP-10-06
RQ-10-15	Requirement	Claim	Claim	Resolver	Security requirements	WP-10-06
RQ-10-16	Requirement	Claim	CQ claim	Resolver	Risk assessment	WP-10-06
RQ-10-17	Requirement	Assumption	Assumption	Resolver	Risk assessment	WP-10-06
RQ-10-18	Requirement	Assumption	Assumption	Resolver	Risk assessment	WP-10-06
RQ-10-19	Requirement	Claim	Claim	Generic sub-case		WP-10-07
RQ-10-20	Requirement	Claim	CQ claim	Generic sub-case		WP-10-07
RQ-10-21	Requirement	Claim	CQ claim	Generic sub-case		
WP-10-01	Work Product	Evidence	Evidence			
WP-10-02	Work Product	Evidence	Evidence			
WP-10-03	Work Product	Evidence	Evidence			
WP-10-04	Work Product	Evidence	Evidence			
WP-10-05	Work Product	Evidence	Evidence			
WP-10-06	Work Product	Evidence	Evidence			
WP-10-07	Work Product	Evidence	Evidence	Generic sub-case		
WP-10-08	Work Product	Evidence	Evidence	Generic sub-case		
RQ-11-01	Requirement	Claim	CQ-Claim	White hat	Security goals	WP-11-02
RQ-11-02	Requirement	Context	Context	White hat	Security goals	WP-11-01
RQ-11-03	Requirement	Claim	Claim	Black hat	Attack paths	WP-11-02
RQ-11-04	Requirement	Claim	CQ-Claim	Resolver	Risk assessment	WP-11-02
WP-11-01	Work Product	Context	Context	Top claim		
WP-11-02	Work Product	Evidence	Evidence			
RQ-12-01	Requirement	Claim	CQ-Claim	Resolver	Security requirements	WP-12-01
WP-12-01	Work Product	Evidence	Evidence			
RQ-13-01	Requirement	Claim	CQ-Claim	Resolver	Security requirements	WP-13-01
RQ-13-02	Requirement	Claim	Claim	Resolver	Security requirements	
RQ-13-03	Requirement	Context	Context	Black hat	Threat-scenarios	WP-13-02
WP-13-01	Work Product	Evidence	Evidence			
WP-13-02	Work Product	Context	Context	Black hat	Threat-scenarios	
RQ-13-04	Requirement	Claim	CQ-Claim	Resolver	Security requirements	
RQ-13-05	Requirement	Claim	CQ-Claim	Resolver	Security requirements	
RQ-13-06	Requirement	Claim	Claim	Generic sub-case		WP-13-03
WP-13-03	Work Product	Context	Context	Generic sub-case		
RQ-14-01	Requirement	Claim	CQ-Claim	White hat	Security goals	
RQ-15-01	Requirement	Claim	Claim	Generic sub-case		
RC-15-01	Recommendation	Evidence	Evidence	Generic sub-case		
RQ-15-02	Requirement	Claim	Claim	Generic sub-case		
RQ-15-03	Requirement	Claim	Claim	Generic sub-case		WP-15-01
RQ-15-04	Requirement	Assumption	Assumption	Resolver	Security requirements	WP-15-01
RQ-15-05	Requirement	Assumption	Assumption	Top Claim		WP-15-01
RQ-15-06	Requirement	Assumption	Assumption	Top Claim		WP-15-01
RQ-15-07	Requirement	Assumption	Assumption	Resolver	Security requirements	WP-15-01
WP-15-01	Work Product	Context	Context	Generic sub-case		

B CASCADE CONCEPTS

Definition of CASCADE Core Security Concepts and Correspondence to ISO/SAE 21434 Terminology

CASCADE Concept	Definition	Corresponding ISO/SAE 21434 concept
Security Assurance Case	A structured body of arguments and evidence used to reason about the security of a certain item.	Cybersecurity case
Asset	Any piece of tangible or intangible artefact that has a value to an organization. Compromising an asset might lead to damage scenarios.	Asset
Security Property	An attribute of an asset including the CIA triad.	Cybersecurity Property
Security Goal	A requirement to preserve a security property of a certain asset.	Cybersecurity Goal[3]
Threat Scenario	An action that would lead to a compromising a security goal.	Threat Scenario
Attack Path	A set of actions that might lead to the realization of a threat scenario.	Attack Path
Risk	A combination of probability and impact of an uncertainty to the security of an asset.	Risk

ACKNOWLEDGMENT

We sincerely thank four anonymous reviewers whose comments and suggestions helped improve and clarify this manuscript.

REFERENCES

[1] Adelard. 2022. *Claims, Arguments and Evidence (CAE).* https://www.adelard.com/asce/cae/. Accessed July 12, 2022.
[2] The MITRE Corporation (MITRE). 2022. *Common Attack Pattern Enumeration and Classification (CAPEC).* http://capec.mitre.org/. Accessed July 2, 2022.
[3] Sebastian Herold, Holger Klus, Yannick Welsch, Constanze Deiters, Andreas Rausch, Ralf Reussner, Klaus Krogmann, Heiko Koziolek, Raffaela Mirandola, Benjamin Hummel, et al. 2008. CoCoME-the common component modeling example. In *Proceeding of the Common Component Modeling Example*, Springer, 16–53.
[4] A. Finnegan and F. McCaffery. 2014. Towards an international security case framework for networked medical devices. In *International Conference on Computer Safety, Reliability, and Security.* Springer, 197–209.
[5] Rob Alexander, Richard Hawkins, and Tim Kelly. 2011. Security assurance cases: Motivation and the state of the art. *High Integrity Systems Engineering Department of Computer Science University of York Deramore Lane York YO10 5GH.* https://www-users.cs.york.ac.uk/~rhawkins/papers/York%20CESG%20security%20case%20report.pdf.
[6] T. S. Ankrum and A. H. Kromholz. 2005. Structured assurance cases: Three common standards. In *9th IEEE International Symposium on High-Assurance Systems Engineering (HASE'05).* 99–108. DOI: https://doi.org/10.1109/HASE.2005.20
[7] John Birch, Roger Rivett, Ibrahim Habli, Ben Bradshaw, John Botham, Dave Higham, Helen Monkhouse, and Robert Palin. 2014. A layered model for structuring automotive safety arguments (short paper). In *10th European Dependable Computing Conference.* 178–181. DOI: https://doi.org/10.1109/EDCC.2014.24
[8] Lukasz Cyra and Janusz Gorski. 2007. Supporting compliance with security standards by trust case templates. In *2nd International Conference on Dependability of Computer Systems (DepCoS-RELCOMEX'07).* IEEE, 91–98.
[9] Anita Finnegan and Fergal McCaffery. 2014. A security argument pattern for medical device assurance cases. In *IEEE International Symposium on Software Reliability Engineering Workshops.* IEEE, 220–225.
[10] Charles B. Weinstock, Howard F. Lipson, and John Goodenough. 2007. Arguing security-creating security assurance cases. Technical Report. Carnegie Mellon University. https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=293629. Accessed July 12, 2022.
[11] GSN Community Standard Working Group. 2011. GSN community standard. Retrieved from www.goalstructuringnotation.info/.

[3]The Cybersecurity Goal in ISO/SAE 21434 associates the cybersecurity requirement with one or more threat scenarios. In CASCADE, we capture the same concept in two different levels (Security Goals and Threat Scenarios) to form the argumentation part of the SAC.

[12] C. Haley, R. Laney, J. Moffett, and B. Nuseibeh. 2008. Security requirements engineering: A framework for representation and analysis. *IEEE Trans. Softw. Eng.* 34, 1 (2008), 133–153. https://doi.org/10.1109/TSE.2007.70754

[13] Richard Hawkins, Tim Kelly, John Knight, and Patrick Graydon. 2011. A new approach to creating clear safety arguments. In *Advances in Systems Safety*. Springer, 3–23.

[14] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. 2004. Design science in information systems research. *MIS Q.* 28, 1 (Mar. 2004), 75–105.

[15] Michael Howard and Steve Lipner. 2006. *The Security Development Lifecycle*, Vol. 8. Microsoft Press, Redmond, WA.

[16] International Organization for Standardization. 1999. Information technology — Security techniques – Evaluation criteria for IT security – Part 1: Introduction and general model. https://www.iso.org/standard/27632.html. Accessed July 12, 2022.

[17] International Organization for Standardization. 2018. ISO 27005 Information technology — Security techniques — Information security risk management. https://www.iso.org/standard/75281.html. Accessed July 7, 2022.

[18] International Organization for Standardization. 2018. ISO 26262 Road vehicles – Functional safety, 2nd ed. https://www.iso.org/standard/68383.html. Accessed July 12, 2022.

[19] International Organization for Standardization and Society of Automotive Engineers. 2018. ISO / SAE 21434 Road vehicles – Cybersecurity Engineering, CD Draft. https://www.iso.org/standard/70918.html. Accessed July 12, 2022.

[20] J. Knight. 2015. The importance of security cases: Proof is good, but not enough. *IEEE Secur. Privac.* 13, 4 (July 2015), 73–75. https://doi.org/10.1109/MSP.2015.68

[21] Nikola Luburić, Goran Sladić, Branko Milosavljević, and Aleksandar Kaplar. 2018. Demonstrating enterprise system security using an asset-centric security assurance framework. In *8th International Conference on Information Society and Technology*.

[22] Mazen Mohamad, Örjan Askerdal, Rodi Jolak, Jan-Philipp Steghöfer, and Riccardo Scandariato. 2021. Asset-driven security assurance cases with built-in quality assurance. In *IEEE/ACM 43rd International Conference on Software Engineering Workshops*. Retrieved from https://www.rodijolak.com/asset-driven/pre-print.pdf.

[23] Mazen Mohamad, Alexander Åström, Örjan Askerdal, Jörgen Borg, and Riccardo Scandariato. 2020. Security assurance cases for road vehicles: An industry perspective. In *15th International Conference on Availability, Reliability and Security(ARES'20)*. Association for Computing Machinery, New York, NY. https://doi.org/10.1145/3407023.3407033

[24] Gdansk University of Technology. 2010–2019. NOR-STA. Retrieved from https://www.nor-sta.eu/en/.

[25] Thomas Rosenstatter, Kim Strandberg, Rodi Jolak, Riccardo Scandariato, and Tomas Olovsson. 2020. REMIND: A framework for the resilient design of automotive systems. In *IEEE Secure Development (SecDev)*. IEEE, 81–95.

[26] John Spriggs. 2012. *GSN-The Goal Structuring Notation: A Structured Approach to Presenting Arguments*. Springer Science & Business Media.

[27] Bill Kuechler and Vijay Vaishnavi. 2008. On theory development in design science research: anatomy of a research project. *European Journal of Information Systems* 17, 5 (2008), 489–504.

[28] B. Xu, M. Lu, and D. Zhang. 2017. A layered argument strategy for software security case development. In *IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. 331–338. https://doi.org/10.1109/ISSREW.2017.52

Received 1 July 2021; revised 7 March 2022; accepted 14 September 2022