Asset-driven Security Assurance Cases with Built-in Quality Assurance

Mazen Mohamad
Chalmers and University of Gothenburg
Gothenburg, Sweden
mazen.mohamad@gu.se

Örjan Askerdal
Volvo Trucks
Gothenburg, Sweden
orjan.askerdal.3@volvo.se

Rodi Jolak
Chalmers and University of Gothenburg
Gothenburg, Sweden
rodi.jolak@cse.gu.se

Jan-Philipp Steghöfer
Chalmers and University of Gothenburg
Gothenburg, Sweden
jan-philipp.steghofer@gu.se

Riccardo Scandariato
Hamburg University of Technology
Hamburg, Germany
riccardo.scandariato@tuhh.de

***Abstract*—Security Assurance Cases (SAC) are structured arguments and evidence bodies used to reason about security of a certain system. SACs are gaining focus in the automotive domain as the needs for security assurance are growing. In this study, we present an approach for creating SAC. The approach is inspired by the upcoming security standards ISO/SAE-21434 as well as the internal needs of automotive Original Equipment Manufacturers (OEMs). We created the approach by extracting relevant requirements from ISO/SAE-21434 and illustrated it using an example case of the head lamp items provided in the standard. We found that the approach is applicable and helps to satisfy the requirements for security assurance in the standard as well as the internal compliance needs in an automotive OEM.**

***Index Terms*—security, assurance cases, automotive systems**

I. Introduction

Assurance cases are structured bodies of arguments and evidence used to reason about a certain property of a system. Security Assurance Cases (SAC) are a type of assurance case for the field of cyber-security. In this paper, we turn our attention to the creation of a SAC, with particular focus on the domain of automotive applications. As vehicles become more advanced and connected, security scrutiny has increased in this domain. Furthermore new standards and regulations push towards assuring security for vehicular systems by using SAC. Similarly to safety cases, which are required in safety standards, e.g., ISO-26262 [1], SACs are explicitly *required* in ISO/SAE-21434 [2]. Additionally, SACs are required for all systems in production.

In literature, there are some studies that suggest the creation of SAC based on requirements derived from security standards [3], [4]. However, there is no approach which helps achieving 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. In order for different stakeholders to use an 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 actually 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, an international automotive OEM, to develop CASCADE, the asset-driven approach for SAC creation presented in this paper. CASCADE is based on the 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 quality claims, as well as building arguments for both.

From a methodological standpoint, we created and validated our approach 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 analyzed the ISO/SAE-21434 standard and extracted the requirements and work products that are relevant to SAC. Accordingly, we mapped the extracted items to the elements of our asset-driven SAC. 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. The results are presented in Sections III–V, after discussing the related work in Section II.

This work is partially supported by the CASUS research project funded by VINNOVA, a Swedish funding agency.

II. 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 literature.

A. Security Assurance Cases

Assurance cases are bodies of evidence organized in structured arguments, used to justify that certain claims about a systems property hold [5]. The argumentation in a Security Assurance Case (SAC) consists of claims about security for the system in question, and the evidence justifies these security-related claims. SAC consist of the following primary components: (i) security claims, (ii) the context in which the claims should hold, (iii) an argument about the security claim, (iv) the strategy used to build the argument, and (v) a body of evidence to prove the claims [6], [7]. SAC can be expressed in a textual or graphical format [7]. The most common graphical formats are the Goal Structure Notation (GSN, [8]), and the Claims, Arguments, and Evidence notation (CAE, [9]).

B. Automotive Assets and Related Security Threats

According to [10], 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 and information leakage.
- *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 which 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 includes fabrication or jamming attacks, spoofing, message collision, eavesdropping, hijacking, and denial of service (DoS). These attacks target the confidentiality, integrity, availability, and privacy 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, availability, and privacy of the data.

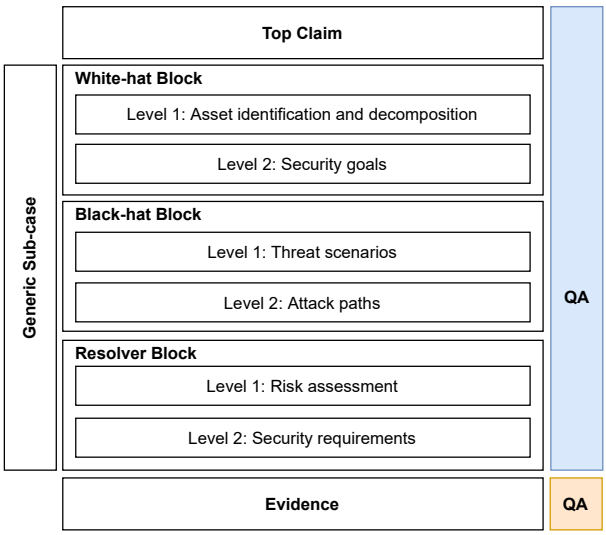

Fig. 1. The CASCADE approach for creating security assurance cases

C. Asset based approaches

Researchers have been exploring several asset-based approaches for creating the argument part of SAC. Biao et al. [11] 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. [11], however, do not consider the quality of the cases and only focus on creating arguments without touching upon the evidence part.

Luburic et al. [12] 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 center pieces. 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 data-flow of each related component. The authors illustrate the approach with a conference management system example. They state that the main limitations of their are asset and data flow granularity. In our study, we consider the 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.

III. CASCADE

In general terms, assets are artifacts of interest to a certain entity. In computer security, these artifacts can be hardware, software, network and communication, or data [10]. 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 cyber security 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 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.

A. Elements of an SAC in CASCADE

We use GSN [8] 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 generic claims, strategies, contexts, assumptions, and evidence that are not bound to a specific artifact, but instead are applicable to a wider range of artifacts in the context of a product, program, or organization.

B. Building blocks of the CASCADE approach

The asset-based approach consists of building blocks, as shown in Figure 1. Each block contains a sub-set of the case. In the following sub-sections, we explain the blocks and their contents.

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., 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, 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 which 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.

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 which corresponds to their artefact. On a higher level, these SAC can be combined together, and generic arguments that are applicable to the sub-SAC can be provided.

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.

a) 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 which 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.

b) Decomposition of assets: To decompose assets, we look into the types of the identified assets. This gives an indicator 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.

[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.

c) 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 which has relevant integrity and availability properties.

d) 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) 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.

a) 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".

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

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.

a) 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.

b) Requirements: At this point, requirements of risk treatments identified in the previous level are to be expressed as claims. This level may contain multiple decomposition 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, this might require a fine grained requirement decomposition which 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, 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.

6) Evidence: The evidence is a crucial part of an 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 as justified. If the creators of the SAC cannot assign evidence to claims, 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 which is not covered by any test report, then there might be a need to create test cases to cover that claim.

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. The completeness also relates to the context of the argumentation strategy. The context provides the information needed to determine if the completeness claim is fulfilled or not.

The second aspect is *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 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.

IV. 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.

A. Top Claim

We start by constructing the Top Claim block consisting of:

- *C:1* the top security claim for the headlamp item.
- *Cnxt:1.1* a context node setting the scope of the claim.
- *Assmp:1.1* an assumption node, stating that the item is physically protected.

[2]A security goal is preserving a security concern (CIA) for an asset [14]

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.

B. White-hat Block

The White-hat block is presented in Figure 2. 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 artifacts 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 quality claim *QC: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*. Lastly, strategy *S:1.2.2* is associated with quality claim *QC:1.2.2.1*.

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 quality claim *QC:1.3.2.1* and argue (*S:1.4.2*) about possible damage scenarios that could invalidate the claims. Accordingly, we create quality claims which make sure that these damage scenarios do not happen. An example of theses claims is *QC: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.

C. Black-hat Block

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

Figure 3 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* that is shown in Figure 2. 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 quality assurance claim (*QC:1.5.1.1*).

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 quality assurance claim (*QC:1.6.1.1*) to ensure the completeness of the case.

D. 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 4 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 in order 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 4 also shows the evidence block which provides examples of evidence to justify the requirement claims. The evidence (e.g., *E:1.1*) is supported by quality evidence (e.g., *QE:1.1*) which, in tun, is complemented with requirement quality claims (e.g., *QC:1.8.1.1*) to confidently justify the associated requirement claims.

E. Generic Sub-case Block

Figure 5 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. Similarly 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 quality assurance claims (e.g., *QC:G.1.1*) as shown in Figure 5.

V. Validation

In order 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 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

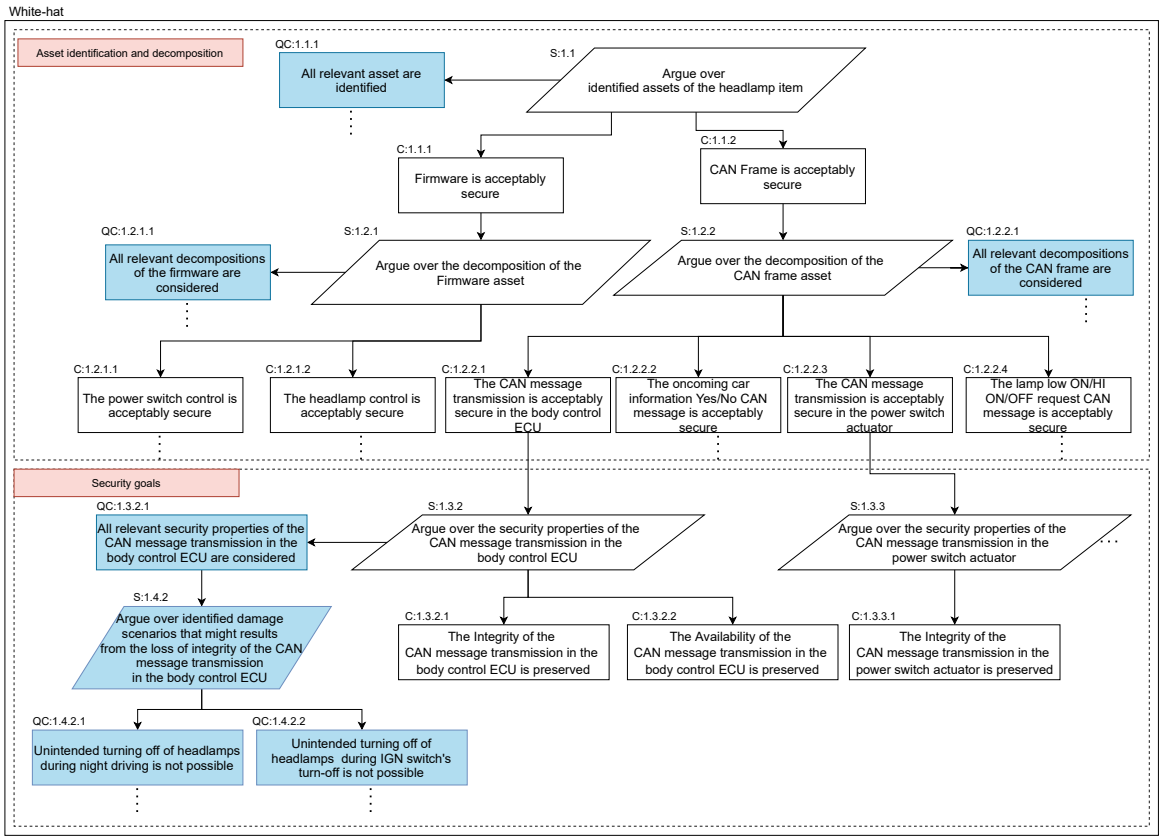

Fig. 2. White-hat block of the headlamp use case

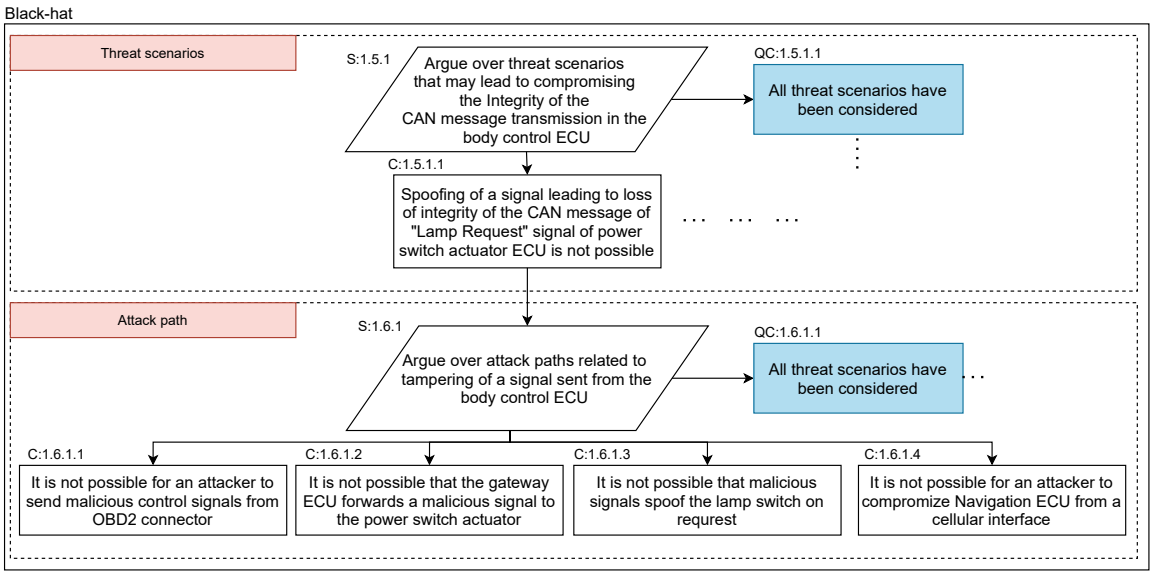

Fig. 3. Black-hat block of the headlamp use case

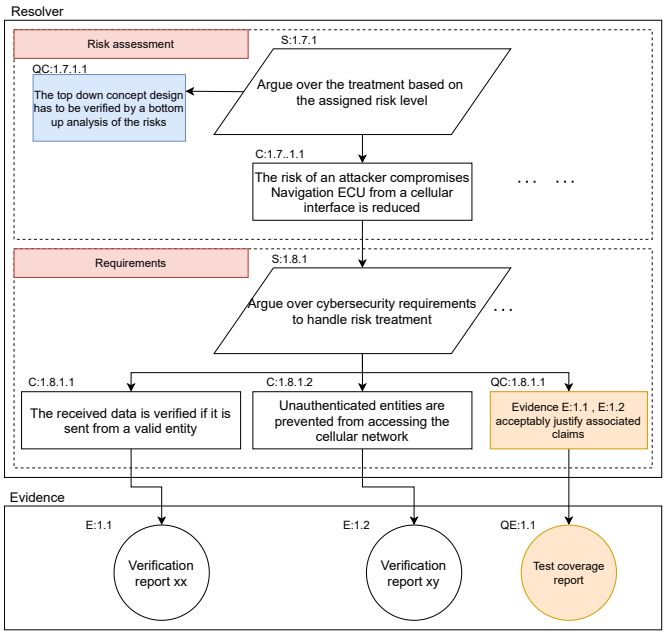

Fig. 4. Resolver and evidence blocks of the headlamp use case

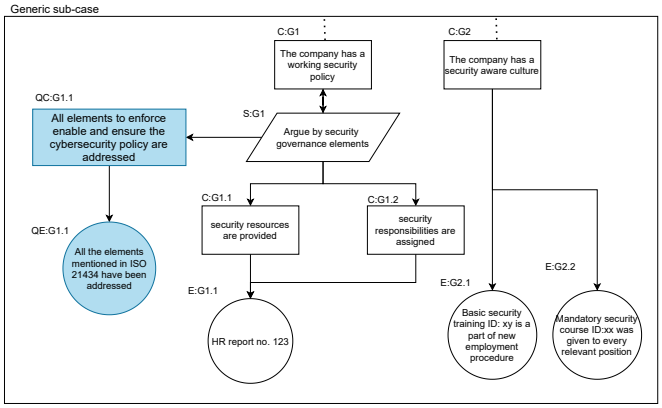

Fig. 5. Generic sub-case block of the headlamp use case

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 an SAC should look like from the company's perspective in order 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 6 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

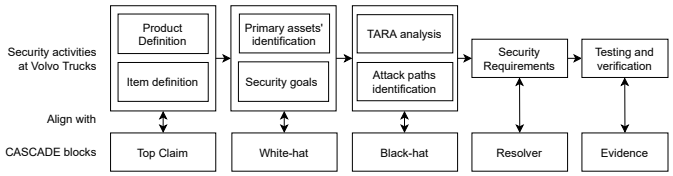

Fig. 6. Mapping of the company's security activities to CASCADE blocks

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 which communicate with the vehicles, whereas the on-board items are end user functionalities, e.g., external lighting and automated parking assistance. In order 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 an 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 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 [16]. Considering the headlamp example case, two possible damage scenarios would be "Loosing 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 as 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 6.

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 which are considered as supporting assets. These requirements are converted to claims in the black-hat block of CASCADE, as shown in Figure 6.

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 experts. For instance, they emphasised 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. Lastly, the experts stress 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 this is covered by the generic sub-case block of CASCADE.

To summarise the validation, we showed that CASCADE aligns well with respect to the way of working at Volvo Trucks. The structure of SAC built with CASCADE follows the structure of work done at the design phase at Volvo to a large extent and the generic sub-case and quality blocks help to serve the abstraction and completeness requirements of the company. We have identified some limitations in the approach which will be the basis for future work.

VI. Conclusion and Future work

We have presented CASCADE, an approach to build security assurance cases driven by assets and geared towards automotive companies that want to conform to the upcoming ISO/SAE-21434. We illustrated the approach using an example case from the standard and validated it at an industrial OEM. We found that the way of working at the company aligns with our approach.

As a future work, we plan to extend the approach to take into consideration the maintenance of SAC. We will also look into requirements sources other than ISO/SAE-21434 to better cover the needs of the automotive industry. Additionally, we plan to further validate the approach by including a larger community of automotive companies and automotive security experts. Additionally, we plan to create a systematic methodology to create SAC in the automotive industry by mapping CASCADE to the requirements and work products of ISO/SAE-21434.

References

[1] International Organization for Standardization, "ISO 26262 Road vehicles – Functional safety, 2nd Edition," Geneva, Switzerland, 2018.

[2] International Organization for Standardization and Society of Automotive Engineers, "ISO / SAE 21434 Road vehicles – Cybersecurity Engineering, CD Draft," 2018.

[3] T. S. Ankrum and A. H. Kromholz, "Structured assurance cases: Three common standards," in *Ninth IEEE International Symposium on High-Assurance Systems Engineering (HASE'05)*. IEEE, 2005, pp. 99–108.

[4] L. Cyra and J. Gorski, "Supporting compliance with security standards by trust case templates," in *2nd International Conference on Dependability of Computer Systems (DepCoS-RELCOMEX'07)*. IEEE, 2007, pp. 91–98.

[5] J. Goodenough, H. Lipson, and C. Weinstock, "Arguing security - creating security assurance cases," 2007.

[6] J. Knight, "The importance of security cases: Proof is good, but not enough," *IEEE Security Privacy*, vol. 13, no. 4, pp. 73–75, July 2015.

[7] R. Alexander, R. Hawkins, and T. Kelly, "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*, 2011.

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

[9] "Claims, arguments and evidence (cae)." [Online]. Available: https://www.adelard.com/asce/choosing-asce/cae.html

[10] T. Rosenstatter, K. Strandberg, R. Jolak, R. Scandariato, and T. Olovsson, "Remind: A framework for the resilient design of automotive systems," in *2020 IEEE Secure Development (SecDev)*. IEEE, 2020, pp. 81–95.

[11] B. Xu, M. Lu, and D. Zhang, "A layered argument strategy for software security case development," in *2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, Oct 2017, pp. 331–338.

[12] N. Luburić, G. Sladić, B. Milosavljević, and A. Kaplar, "Demonstrating enterprise system security using an asset-centric security assurance framework," in *8th International Conference on Information Society and Technology*, 2018, p. 16.

[13] R. Hawkins, T. Kelly, J. Knight, and P. Graydon, "A new approach to creating clear safety arguments," in *Advances in systems safety*. Springer, 2011, pp. 3–23.

[14] C. Haley, R. Laney, J. Moffett, and B. Nuseibeh, "Security requirements engineering: A framework for representation and analysis," *IEEE Transactions on Software Engineering*, vol. 34, no. 1, pp. 133–153, 2008.

[15] M. Howard and S. Lipner, *The security development lifecycle*. Microsoft Press Redmond, 2006, vol. 8.

[16] International Organization for Standardization, "ISO 26262 Information technology — Security techniques — Information security risk management," Geneva, Switzerland, 2018.