algorithms

MDPI

Article

V-SOC4AS: A Vehicle-SOC for Improving Automotive Security

Vita Santa Barletta [1,*], Danilo Caivano [1], Mirko De Vincentiis [1,*], Azzurra Ragone [1], Michele Scalera [1] and Manuel Ángel Serrano Martín [2]

[1] Department of Computer Science, University of Bari, 70125 Bari, Italy
[2] Departamento de Tecnologías y Sistemas de Información, University of Castilla-La Mancha, 13001 Ciudad Real, Spain
* Correspondence: vita.barletta@uniba.it (V.S.B.); mirko.devincentiis@uniba.it (M.D.V.)

Abstract: Integrating embedded systems into next-generation vehicles is proliferating as they increase safety, efficiency, and driving comfort. These functionalities are provided by hundreds of electronic control units (ECUs) that communicate with each other using various protocols that, if not properly designed, may be vulnerable to local or remote attacks. The paper presents a vehicle-security operation center for improving automotive security (V-SOC4AS) to enhance the detection, response, and prevention of cyber-attacks in the automotive context. The goal is to monitor in real-time each subsystem of intra-vehicle communication, that is controller area network (CAN), local interconnect network (LIN), FlexRay, media oriented systems transport (MOST), and Ethernet. Therefore, to achieve this goal, security information and event management (SIEM) was used to monitor and detect malicious attacks in intra-vehicle and inter-vehicle communications: messages transmitted between vehicle ECUs; infotainment and telematics systems, which provide passengers with entertainment capabilities and information about the vehicle system; and vehicular ports, which allow vehicles to connect to diagnostic devices, upload content of various types. As a result, this allows the automation and improvement of threat detection and incident response processes. Furthermore, the V-SOC4AS allows the classification of the received message as malicious and non-malicious and acquisition of additional information about the type of attack. Thus, this reduces the detection time and provides more support for response activities. Experimental evaluation was conducted on two state-of-the-art attacks: denial of service (DoS) and fuzzing. An open-source dataset was used to simulate the vehicles. V-SOC4AS exploits security information and event management to analyze the packets sent by a vehicle using a rule-based mechanism. If the payload contains a CAN frame attack, it is notified to the SOC analysts.

Keywords: automotive security; CAN Protocol; Security Operation Center; V-SOC

Citation: Barletta, V.S.; Caivano, D.; Vincentiis, M.D.; Ragone, A.; Scalera, M.; Martín, M.Á.S. V-SOC4AS: A Vehicle-SOC for Improving Automotive Security. *Algorithms* **2023**, *16*, 112. https://doi.org/10.3390/a16020112

Academic Editor: Francesco Quaglia

Received: 30 November 2022
Revised: 19 January 2023
Accepted: 8 February 2023
Published: 14 February 2023

1. Introduction

Modern vehicles are composed of electronic control units (ECUs) which are micro-controllers that deal with various functionalities of a vehicle, including safety-critical functions such as steering and braking. These ECUs are physically connected to the vehicle network and communicate with each other using different protocols: controller area network (CAN), media oriented system transport (MOST), local interconnect network (LIN), FlexRay, and autonomous Ethernet. The CAN protocol presents several advantages in comparison to others regarding cost-efficiency and flexibility. This protocol is almost like a plug-and-play system because an ECU can be connected to the intra-vehicle network with ease or no special modification of the network [1].

Since the CAN protocol was developed in 1986, the necessity of protection from cyber-attacks was not implemented at that time. The lack of security mechanisms makes the protocol vulnerable to confidentiality, integrity, and availability attacks [2]. Miller and Valasek [3] remotely took control of a Jeep Cherokee exploiting various vulnerabilities in the vehicle. Palanca et al. [4], with the Arduino Uno Rev 3 and MCP 2551 E/P attached

to Alfa Romeo Giuletta's OBD-II port, have created a malfunction in the parking sensor by fabricating messages and causing a denial of service (DoS) attack. The researchers of Keen Security Lab [5] found the SSID and password of the Infotainment System in a Mercedes-Benz model by exploiting the CAN bus using a sniffing attack since the CAN protocol does not implement any encryption techniques [6].

Consequently, different types of techniques have been proposed in the literature to remedy the security problems of the CAN protocol. Usually, these techniques involve authentication and encryption mechanisms or machine learning/deep learning techniques to develop an intrusion detection system (IDS) model. However, some hardware techniques require modification of the ECUs to encrypt the CAN payload [7,8]. Stabili et al. [9] provided an analysis of cryptographical protocol for in-vehicle networks considering the vehicle life-cycle. The authors have shown that many of the analyzed approaches may appear advantageous if deployed on a single vehicle but are disadvantageous if deployed at scale. Some works adopt techniques that consist of identifying malicious CAN payload without modifying the CAN protocol [10,11]. In the case of the IDSs, Wu et al. [12] defined some challenges regarding the machine learning techniques which are how to deploy the model in an in-vehicle system with limited computational power and how to obtain the data to train the algorithm. Moreover, these methods do not provide information about the attack, for example, where it manifested or which payload contains it.

This represents only a part of the information and vulnerabilities found inside a vehicle. Sommer et al. [13] presented a taxonomy to analyze and classify automotive attacks. In particular, the authors proposed a classification of 162 existing attacks based on this taxonomy that can be considered for the automotive security development process. Therefore, in such a scenario, it is necessary to focus not only on attack detection but to extend the analysis to a wide range of information, for example, the vehicle model, the ECU that sent the message, and the type of message sent. This information can determine malicious actions and new types of attacks based on vehicle models and ECUs and, more importantly, to provide possible response activities.

To achieve this goal, SIEM (security information and event management) and especially a SOC (security operation center) can be considered responsible for ensuring the correct identification, analysis, defense, investigation, and reporting of potential security incidents [14]. Typically, the SOC monitors and analyzes activity on networks, servers, endpoints, databases, applications, websites, other systems looking for anomalous activities that could be indicative of a security incident or compromise. The SOC is responsible for ensuring that potential security incidents are correctly identified, analyzed, defended, investigated, and reported [14]. It consists of people, processes, technology and activities and responsibilities fall into three general categories namely (i) preparation, planning, and prevention; (ii) monitoring, detection, and response; (iii) recovery, refinement, and compliance.

In this work, the vehicle is considered as a critical asset because it contains components (such as an OBD-II port, infotainment system, and USB) that can be exploited to conduct an attack. The idea is to use the SOC architecture to improve the security life cycle in the automotive context, focusing more on the response phase. Since it is not enough to work on improving the detection phase because we do not have the information about the attack (for example, the attacked component, the payload sent, and the ECU model), it is necessary to activate prevention and response actions. The prevention phase aims to block malicious activities in a component. On the other hand, the response actions can put the vehicle in emergency mode. In the literature, the proposed CAN protocol defense mechanisms take into account the detection (with intrusion detection system (IDS) and prevention phase (with cryptographic algorithms) [15] but not the response phase. Thus, the SOC architecture covers all these three phases: Detection implements the set of security controls and analysis of events that have occurred to detect possible offenses; Response defines both cyber and physical structured response plans to contain the damage from a cyber offense; Prevention implements security controls to prevent the possibility of cyber-attacks by using, among

others, cognitive technologies, and threat intelligence tools. The prevention phase also allows updating the ECU to fix the vulnerability that causes a specific attack identified in the detection phase. This update can be done using over-the-air technology to update the ECU firmware [16,17].

The paper presents a new method, vehicle-security operation center for improving automotive security (V-SOC4AS), to detect attacks and attend to the abovementioned requirements.

Two different attacks are considered to validate and confirm the research idea: DoS (denial of service), which consists in preventing legitimate access to services of a target machine by overwhelming its resources [18], and fuzzing which consists in sending a series of messages with minor randomized changes [19].

The main contributions of this paper are:

- A novel approach, vehicle-security operation center for automotive security (V-SOC4AS), to detect, respond, and prevent attacks that can occur in the automotive context. The V-SOC 4AS was developed not to replace existing solutions such as IDS, but to integrate them to support the detection, response, and prevention [14]. The goal is to improve cybersecurity and define new cyber kill chain models. This is possible as the analysis and monitoring take place in real-time and not offline, thus reducing the detection time and providing further support for response activities. The first experiment has been done considering two state-of-the-art attacks (DoS and fuzzing).
- The use of security information and event management (SIEM) to monitor each subsystem for intra-vehicle communication management, namely the controller area network (CAN), local interconnect network (LIN), FlexRay, and media oriented systems transport (MOST), and Ethernet. Each of them has vulnerabilities and security issues [6,20]. The research goal is to define a solution that can improve and redefine security controls in the three functional areas: detection, response, and prevention. Specifically, in this first step, the work focuses on the CAN protocol and the use of IBM QRadar as a SIEM.
- The possibility to identify not only the attack but also the type and the pattern of attack. This allows work on the threat, attack vector, and risk associated with a given vehicle component. In the literature, most works use binary classifiers such as IDSs for detection without defining the attack type. Therefore, V-SOC4AS allows for the investigation of how different attack types can be identified and how AI models can be integrated (in the future) to improve the security lifecycle in this context. The vehicle was simulated using an open-source dataset, specifically the exchange of CAN messages between an open-source tool and the SIEM recreating the V-SOC4AS. The proposed method can detect attacks without changing the CAN protocol. Therefore, it is also applicable to any vehicle that uses the CAN.

The paper is organized as follows: Section 2 outlines related works on identifying attacks to the CAN; Section 3 briefly describes the CAN protocol; Section 4 presents the Vehicle-SOC and how to set up the architecture to identify the attacks; Section 5 highlights the proposed architecture in conjunction with a SIEM to identify two types of attacks and experimental evaluation; Section 6 sets out the conclusions.

2. Related Work

In the literature, several research works aim to improve the security of the CAN Bus. Most of these approaches use intrusion detection systems (IDSs) or intrusion prevention systems (IPSs). An IDS is a software that automates the process of detecting an intrusion into a network. Unlike an IDS, an IPS is a software that provides the functionality of an intrusion detection system, with the added feature of attempting to block an intrusion [21]. Barletta et al. proposed a supervised Kohonen Self-Organizing Map (SOM) network to identify attack messages sent on a CAN [22]. Lee et al. [19] proposed an approach to analyzing the offset ratio and time interval between request and response messages in the CAN network.

Seo et al. [23] proposed a generative adversarial network (GAN)-based intrusion detection system using a deep-learning model. This model uses a generative model (G) and a discriminative model (D). The model encoded the CAN message into a one-hot-vector and then transformed it into a CAN image. To detect if the real-time CAN message is an attack or not, the authors also used a threshold. IDSs may be limited to detecting specific attacks; if exposed, an attacker could even use this information to avoid detection. The authors stated that this GIDS model could solve these problems related to IDSs.

Cho et al. [24] presented an anomaly based IDS that measures and then exploits the interval of periodic in-vehicle messages for fingerprinting ECUs. Lokman et al. [25] have suggested an idea to deploy IDS on an in-vehicle network component called gateway. Moreover, in this case, the risk of an attacker compromising the CAN network and gateway is greater than compromising the ECU.

Young et al. [26] reported that other IDS techniques exploited the timing when a message is sent on the CAN bus. Miller and Valasek [3] also introduced how the rate of the message can be analyzed for the in-vehicle network.

Costantino et al. [15] proposed an intrusion prevention system with challenge-based mechanisms. EARNEST, the name of the system proposed by the authors, can address replay and fuzzing attacks. When an ECU sends a message on the CAN bus, EARNEST requires the ECU to solve a challenge. If the ECU answers correctly, the message is forwarded to the bus, otherwise it is discarded. Fallstrand et al. [27] discussed the applicability of intrusion and prevention systems. The researchers said that with the anomaly detection technique, we get false positives and false negatives. In a safety-critical automotive system, it is important to evaluate the effects of these undesirable cases. The authors also discussed the importance of the post-detection phase. These schemes use warnings and log events and facilitate forensic examinations that are important for vehicle safety.

Therefore, the solution of using an IDS on an ECU involves various issues: the ECUs have limited computational resources, memory, and power so, using a machine learning or deep learning approach may not be feasible for the above problems [12]. Moreover, IDS and IPS became less and less adequate regarding the new threats [28]. Aijaz et al. [29] outline that these systems can be bypassed by zero-day or sophisticated attacks. As a consequence, to address the challenges posed by IDS and IPS, there is an increasing demand for SOC [28] because the SOC provides real-time monitoring events that help to understand if the event sent to the SOC is an attack or not from a security perspective [29] and to interpret the logs that with the IDS and/or IPS it is not easy because of their size. So, this can be useful for SOC analysist to identify anomalies or sophisticated attacks in the network by viewing the aggregated logs. In the literature, several research works use a SOC to detect attacks in a private or enterprise network such as distributed denial of service (DDoS), phishing, and SQL injection [30–32].

In contrast, the state-of-the-art in vehicle security operation center is scarce. Langer et al. [33] proposed an automotive cyber defense center aimed at creating a SOC to analyze and react to incidents. Meyer et al. [34] presented a demo regarding a security infrastructure that involves the use of a software-defined network (SDN), anomaly detection (AD), and a cyber defense center. The authors built a prototype upon a Seat Ateca that consisted of ECUs grouped into five functional domains with one CAN bus per domain. Since this work is a demo, the researchers presented the showcases but not a real implementation of a vehicle attack. Furthermore, many of the solutions developed by industries aim to manage cybersecurity automotive with a V-SOC. For example, Upstream (Available online: https://upstream.auto/solutions/vehicle-security-operations-center/ (accessed on 7 February 2023)) has a solution that aggregates and normalizes multiple property data feeds and has a full and contextual view of the vehicle owing to digital twin profiling of the connected vehicle environment and powerful AI-powered detection modeling.

In this paper, V-SOC4AS is proposed, which is responsible, starting from the limitations regarding the IDSs (detection phase) and the solutions to improve the response and prevention phases. The V-SOC4AS aims to identify anomalous and malicious activities

and analyzes them in detail. It is, thus, possible to understand in which vehicle the attack began, the information about the CAN payload, and which ECU sent the attack message; moreover, attempt to respond to an attack promptly to safeguard driver safety.

3. Controller Area Network (CAN)

The CAN protocol is a broadcast protocol that connects several electronic control units (ECUs) in-vehicle network to exchange messages among them. The messages exchanged are called CAN frames. A CAN frame is composed of the following fields:

- Arbitration field (11 bits CAN 2.0A) establishes the priority of the message, 0-bit is dominant while 1-bit is recessive. When two or more nodes transmit a data frame at the same time, if node A has a dominant bit and node B has a recessive bit, node A wins the arbitration and can send the message on the bus.
- Control field (4 bits): consists of data length control (DLC) and contains the length of the data payload.
- Data field (64 bits): contains the payload.
- The main issue of the CAN protocol is that it does not implement any security mechanism against cyber-attacks [2].

Threat on CAN Bus

Since the CAN protocol does not implement a security mechanism, an adversary could conduct various attacks that can damage the confidentiality, integrity, and availability of the data frames that are exchanged between the ECUs. Moreover, it is possible to distinguish two types of attacks: passive and active. An attacker carries out a passive attack by monitoring the bus to obtain information about the data frames. This type of attack involves the confidentiality of the payload [35]. As a way of example, a researcher of Tencent Security Keen Lab [5] found a vulnerability in the head unit of a Mercedes-Benz model that sent the passphrase and SSID (service set identifier) of Wi-Fi on the CAN bus as plaintext. Therefore, the researchers have been able to capture the data from the bus to obtain the passphrase and SSID. On the other side, an active attack occurs when an adversary can inject a data frame on the bus to cause unexpected behavior. Lokman et al. [25] presented two types of attacks on CAN packets: frequency and payload. The frequency attack consists of inserting an extra packet or erasing a legitimate packet from the CAN bus as the time interval for sending a CAN message is fixed and periodic. In addition, the author also stated that an attacker can send highest priority ID of the CAN packet in a short cycle. In the CAN packet payload, an attacker can manipulate the data content of some CAN ID. El-Rewini et al. [20] presented a survey regarding threats on vehicles and also on the CAN bus. Costantino et al. [15] introduced a different type of attack that an adversary can do when it has partial control of an ECU. Examples of these attacks are fuzzing, replay, masquerading, and information gathering. In addition, most of the attacks that can be conducted on the CAN bus required the injection of messages to change the behavior of the vehicle [36]. El-Rewini et al. [20] reported several challenges to protecting the CAN protocol against the DoS, fuzzing, spoofing, and masquerading attacks that need to be addressed. We selected two types of attacks based on the literature because the first (DoS) can occupy the bus for long periods denying other ECU to send messages; instead in the second (fuzzing), an attacker can send malicious payloads causing anomalous behavior [36]. These two well-analyzed attacks can be summarized as:

- DoS: It consists of injecting a low identifier packet to have a high priority in the bus. Most of the datasets in the literature that simulate this type of attack set the arbitration field with the value “0 x 000” as hexadecimal since it is the lowest possible.
- Fuzzing: It consists of recreating CAN messages that have a completely random value of both arbitration and data field.

4. V-SOC4AS: Vehicle-SOC for Improving Automotive Security

In information technology (IT), a security operation center (SOC) is a centralized solution that an organization uses to continuously monitor critical assets to prevent, detect, respond to, and analyze cybersecurity attacks. Usually, a SOC is formed by people, processes, and technology in order to fulfill the previous goals [14]. The SOC architecture can be divided into different subsystems. For example, Madani et al. [30] subdivided the architecture into four subsystems: *sensors* to collect traffic data; *log management* to normalize, classify, prioritize, and collect these data; *correlation engine* to analyze these events; and *response system* to react to security incidents.

Sensors generate events from a critical asset, for example, these events can be a user typing an incorrect password, or a user log-in secure shell (SSH). When the events are generated by sensors, these will be aggregated and modified into a standard format to obtain homogeneous data and saved into a database to perform statistical and forensic analysis. Usually, these operations are done using security information and event management (SIEM) that identifies the attack and provides a response to reduce the impact of that compromise. Therefore, considering these components and how they communicate with each other, the V-SOC4AS architecture can be represented in Figure 1. This architecture is suitable in several general contexts, such as military or civilian vehicles. In our work, the vehicle is simulated using an open-source tool (ICSim https://github.com/zombieCraig/ICSim (accessed on 7 February 2023)).

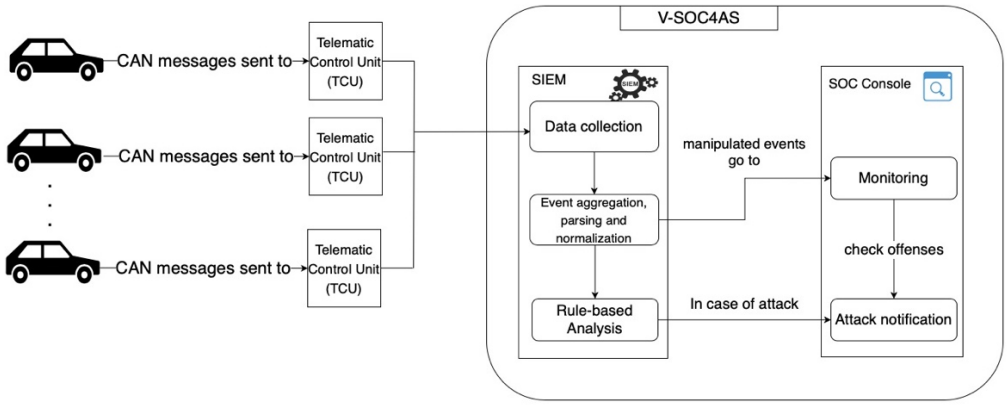

Figure 1. V-SOC4AS architecture.

The V-SOC4AS architecture consists of two main components: SIEM and SOC console.

- SIEM collects the raw events that will be aggregated, parsed, normalized, and analyzed to be displayed correctly. It can analyze data from different types of vehicles. Owing to this component, we can parse these data without adding further physical components to the car. After the events are parsed and normalized, the rule-based component allows control over whether the events sent to SIEM correspond to attacks through custom rules. For example, in the information technology (IT) field, rules could detect if we have multiple logins failed attempts on a login page and excessive firewall denies.
- SOC Console: the parsed events will be shown on the SOC console component and monitored by a cybersecurity specialist. If a rule-based mechanism triggers an offense, it will appear on this component so that the analyst can analyze it and potentially move on to the response phase. An advantage of the SOC console is that if an attack is detected, the cybersecurity specialist can check if it is a false positive or a real attack. Furthermore, the SOC analyst (a person in the organization who has knowledge of the automotive domain and automotive attack), when monitoring the data flow from the vehicle, could identify anomalies or attacks that the SIEM has not identified and subsequently add or modify a rule. In the next subsection, we will show an example of an event displayed on the SOC console.

The following is an example of how a V-SOC4AS works (Figure 1). When an ECU generates a CAN frame, it sends the CAN frame to the telematic control unit (TCU). The TCU converts the messages into JSON format and sends them to the SIEM. When the SIEM receives these events, it aggregates and displays them on the SOC console. Simultaneously, the SIEM, with a rule-based mechanism, checks whether these events can be considered as attacks or not. Events that are considered attacks will be displayed on the SOC console.

Communication between the vehicle and the SIEM is simulated using an open-source tool called ICSim. This tool emulates an instrument cluster, creates the data, and sends them to a virtual channel using the SocketCAN interface. In particular, the instrument cluster represents the dashboard of a vehicle and includes a speedometer, door lock indicators, turn signal indicators, and a control panel. The data generated by the instrument cluster are created by the ICSim. The *vcan* (virtual can) interface was used to allow communication between the ICSim and the CAN bus. Ubuntu 20.04 was used to run ICSim.

The TCU is a Python module that captures the messages from the vcan0 and sends them to the SIEM in a JSON format.

4.1. IBM QRadar SIEM

Starting from the proposed architecture, IBM QRadar SIEM was used in order to identify attacks in an automotive environment [37]. This choice was based on the following reasons:

- Good position, "Leader", in Gartner Magic Quadrant 2022 [38] in comparison to other SIEMs.
- Possibility to create a custom Log Source different from the IT context. IBM QRadar, using the Syslog protocol, allows information to be received from systems that are different from each other, for example, a vehicle and a computer.
- Possibility to use a set of default rules to be customized for the automotive environment.
- Risk priority modelling that determines the priority level of each offense based on local and enriched threat context, tactics and techniques observed and learned offense disposition patterns within the automotive environment.

QRadar consists of a modular three-layer architecture that provides, in real-time, the security status of the IT infrastructure. These three-layer are [39]:

- Data Collection: Collects data such as events or flows from a specific asset. IBM QRadar SIEM accepts data from IDS/IPS, firewall, Syslog, and other sources. The data are parsed and normalized before they are passed to the processing layer. Raw data are parsed and then normalized to present them in a structured and usable format. In Figure 1, this task is carried out by the "Data Collection", "Event aggregation parsing & normalization", and "Rule-based Analysis" modules.
- Data processing: In this layer, the data are run through the custom rule engine (CRE), which generates offenses and alerts, and then are written to the storage. In Figure 1 this task is carried out by the "rule-based" module.
- Data searches: Data collected and processed by QRadar are available to users for searches, analysis, reporting, and alerts or offense investigations. In Figure 1, this task is carried out in the SOC console.

IBM QRadar uses the Syslog protocol [40] to receive generic events. For this reason, in the proposed architecture, the CAN messages are sent using this protocol in JSON format. When IBM QRadar obtains data from a generic log source (would be the TCU or other vehicle components in the context of automotive) using the Syslog protocol, it manipulates them using the three-tier architecture described above. In this work, IBM QRadar SIEM Community Edition v7.3.3 was used. Specifically, V-SOC4AS combines the tools found in QRadar to obtain messages from a vehicle, parse them, and check using rule-based mechanism whether these messages are attacks. An important consideration is that we send messages to IBM QRadar using a local network and not through the cloud

environment. Figure 2 shows the Log Source created to receive the messages. It contains various parameters: *ID* that is automatically generated; *Name* is a generic name that can be defined by the user; *Description* can be empty; *Enabled* whether it is to be activated or not; *Log Source Type* identifies the type of the event; *Protocol Type* to set the appropriate format for different contexts. In this example, Syslog was chosen.

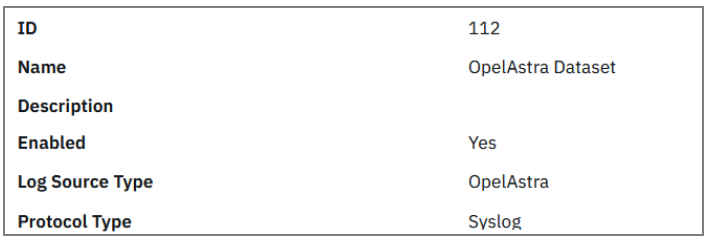

Figure 2. Log source created to receive the CAN messages.

To properly analyze the data from the vehicle, a plug-in file called device support module (DSM) supported by QRadar was used. This plug-in collects the raw data and transforms them into a specific format defined by the SOC analyst so that it can be shown on log activity. The DSM extracts the value contained in the JSON with a specific key and creates six properties. An example of a property created by DSM is shown in Figure 3.

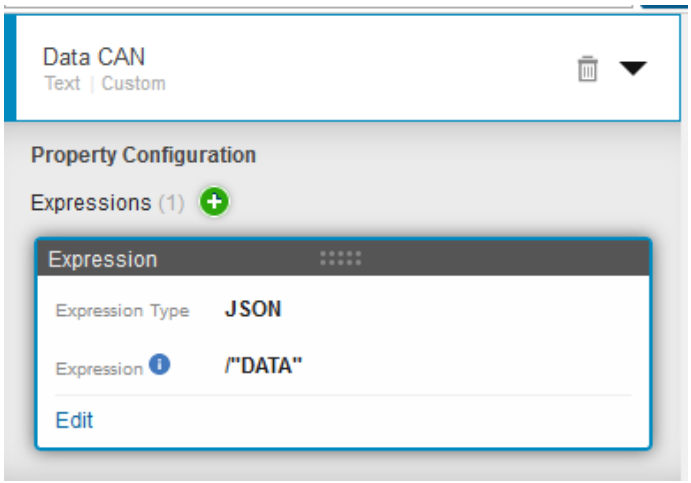

Figure 3. An example of property created on QRadar to obtain the value of key "DATA".

The other properties created are:

- Vehicle ID: unique identifier of the vehicle (expressed in the payload with a Universally Unique Identifier (UUID). It can be used in the response phase when an attack is detected.
- Event ID and Event Category: are used to map an event with Qradar Identifier (QID). QID is a numeric representation of a specific event and includes name, description, severity, and low-level category. Figure 4 shows the QID, ECU data transfer created with these properties.
- Log Source Time: when the payload arrived at Qradar.
- ID: CAN ID sent on the bus.
- DATA: data value in hexadecimal.

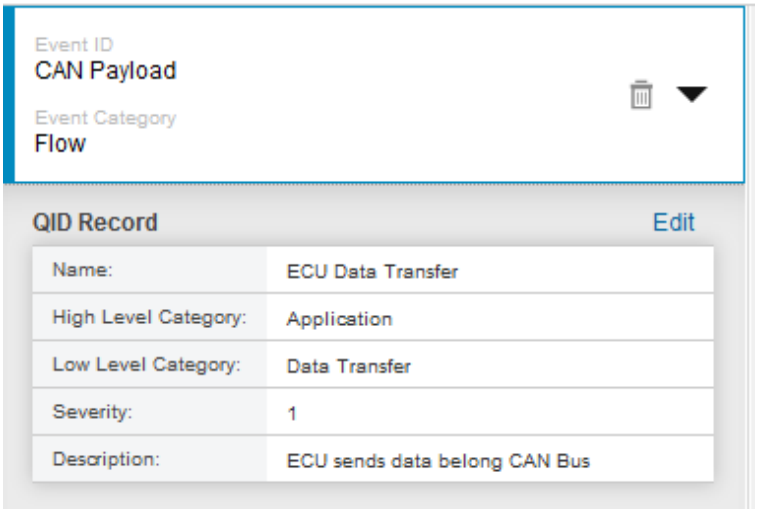

Figure 4. Event ID and event category used to create a new QID named ECU data transfer.

4.2. Communication between Vehicle and SIEM

The first approach is simulating the communication between the vehicles and Qradar. As described in the previous section, the ICSim generates the CAN messages and sends them to the vcan0, and a component called TCU transforms these messages into a JSON format. After this, using the Syslog protocol, the TCU sends the data to IBM Qradar. The data generated by ICSim are shown in Figure 5.

```
vcan0  333   [7]  00 00 00 00 00 00 1E
vcan0  095   [8]  80 00 07 F4 00 00 00 35
vcan0  1A4   [8]  00 00 00 08 00 00 00 2F
vcan0  1AA   [8]  7F FF 00 00 00 00 67 20
vcan0  1B0   [7]  00 0F 00 00 00 01 66
vcan0  1D0   [8]  00 00 00 00 00 00 00 0A
vcan0  166   [4]  D0 32 00 09
vcan0  244   [5]  00 00 00 01 D3
vcan0  158   [8]  00 00 00 00 00 00 00 0A
vcan0  161   [8]  00 00 05 50 01 08 00 0D
vcan0  191   [7]  01 00 90 A1 41 00 30
vcan0  133   [5]  00 00 00 00 98
vcan0  136   [8]  00 02 00 00 00 00 00 1B
vcan0  13A   [8]  00 00 00 00 00 00 00 19
vcan0  13F   [8]  00 00 00 05 00 00 00 1F
vcan0  164   [8]  00 00 C0 1A A8 00 00 31
vcan0  17C   [8]  00 00 00 00 10 00 00 12
vcan0  18E   [3]  00 00 5C
```

Figure 5. The CAN messages generated by ICSim that are sent on the vcan0.

When the ICSim generates and sends the messages, TCU remains listening on the vcan0 channel waiting for new messages. If a new message is received, the component transforms it into JSON format (Figure 6) and sends it to the SIEM (Figure 7). In addition, considering Figure 7, we can see that the Source IP is the IP of the vehicle that sent the CAN message. With this information, if an attack is detected by the SIEM, it is possible to respond to it by temporarily resetting the ECU, sending a notification to the driver to pull the vehicle over, or disabling some ECU functionality until the vehicle is shut down.

```
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.935275", "ID": "0333", "DATA": "00 00 00 00 00 00 1E"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.936347", "ID": "0095", "DATA": "80 00 07 F4 00 00 00 35"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.937420", "ID": "01a4", "DATA": "00 00 00 08 00 00 00 2F"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.937436", "ID": "01aa", "DATA": "7F FF 00 00 00 00 67 20"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.937439", "ID": "01b0", "DATA": "00 0F 00 00 00 01 66"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.937441", "ID": "01d0", "DATA": "00 00 00 00 00 00 00 0A"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.938495", "ID": "0166", "DATA": "D0 32 00 09"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.940095", "ID": "0244", "DATA": "00 00 00 01 D3"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.940616", "ID": "0158", "DATA": "00 00 00 00 00 00 00 0A"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.940624", "ID": "0161", "DATA": "00 00 05 50 01 08 00 0D"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.940625", "ID": "0191", "DATA": "01 00 90 A1 41 00 30"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.940627", "ID": "0133", "DATA": "00 00 00 00 98"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.941690", "ID": "0136", "DATA": "00 02 00 00 00 00 1B"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.941723", "ID": "013a", "DATA": "00 00 00 00 00 00 19"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.941725", "ID": "013f", "DATA": "00 00 00 05 00 00 00 1F"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.941729", "ID": "0164", "DATA": "00 00 C0 1A A8 00 00 31"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.941730", "ID": "017c", "DATA": "00 00 00 00 10 00 00 12"}
{"UUID": "53cd3082a0", "eventID": "CAN Payload", "eventCategory": "Flow", "Timestamp": "1661872106.942793", "ID": "018e", "DATA": "00 00 5C"}
```


Figure 6. Messages generated by ICSim and transformed into JSON format by the TCU component.

Event Name	EventID (custom)	Log Source Time	Source IP	Vehicle ID (custom)	ID CAN (custom)	Data CAN (custom)
ECU Data Transfer	CAN Payload	Aug 30, 2022, 5:30:40 PM	192.168.1.231	53cd3082a0	01aa	7F FF 00 00 00 00 67 20
ECU Data Transfer	CAN Payload	Aug 30, 2022, 5:30:37 PM	192.168.1.231	53cd3082a0	01a4	00 00 00 08 00 00 00 2F
ECU Data Transfer	CAN Payload	Aug 30, 2022, 5:30:34 PM	192.168.1.231	53cd3082a0	0095	80 00 07 F4 00 00 00 35
ECU Data Transfer	CAN Payload	Aug 30, 2022, 5:30:31 PM	192.168.1.231	53cd3082a0	0333	00 00 00 00 00 00 1E

Figure 7. Result after creating all properties. On console it can be seen the correctly formatted payload.

Figure 8 summarizes the communication between ICSim and IBM Qradar that has been described.

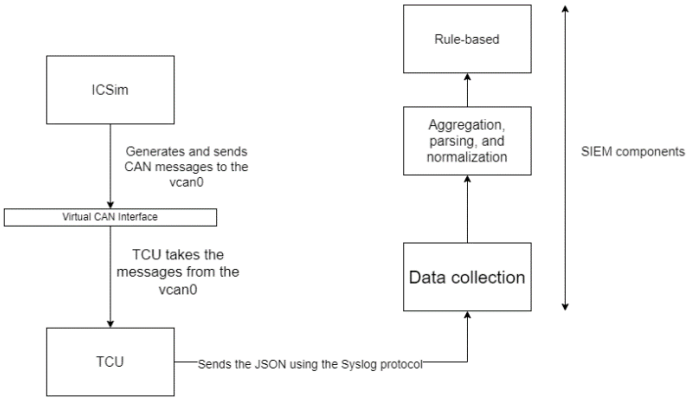

Figure 8. The simulated architecture using ICSim and IBM Qradar.

5. Experimental Results and Discussion

To validate the proposed architecture, two attacks were considered in the experimental setting: DoS and fuzzing. In the identification phase, messages from the two different datasets were sent to Qradar and these two datasets represent the vehicles in real context. The logs were modified in JSON format and then sent using the Syslog protocol. Figure 9 shows the approach used to send the CAN messages to IBM Qradar. If a message is identified as an attack, an offense will be raised.

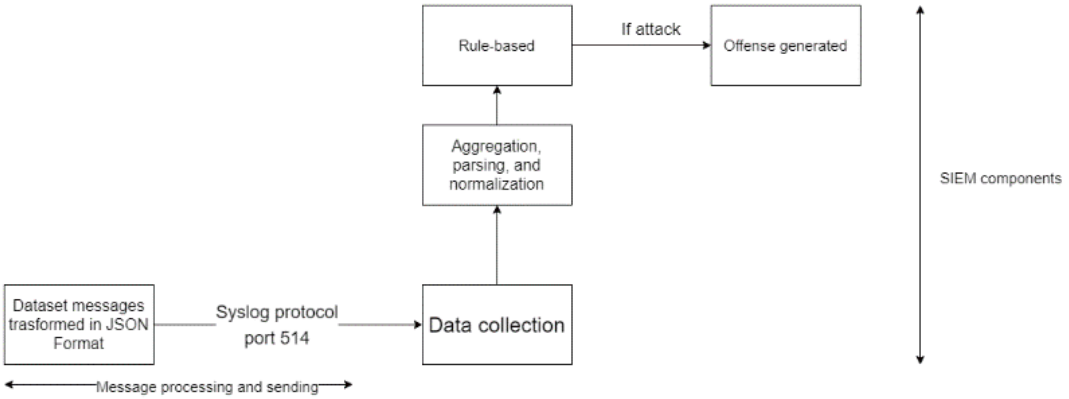

Figure 9. Process of sending, receiving, and handling messages.

For the experimental analysis, the JSON sent to IBM Qradar is changed from the one depicted in Figure 6. Two new attributes are added into the new JSON message:

- UUID_ECU: A unique identifier for each ECU. It can be used to check which ECU is under attack. For this experiment, two unique identifiers will be used: one for the DoS attack and another for the Fuzzing.
- vehicleModel: To represent the vehicle model. Two different models were used to simulate two different vehicles: OpelAstra and RenaultClio.

The attribute UUID will be modified based on the vehicleModel. For the OpelAstra, the UUID 53cd3082a0 will be used. Instead, for the RenaultClio, the UUID 6fab142fe3 will be used.

5.1. Dataset

Dupont et al. [41] proposed a dataset for CAN IDSs and in this work, the DoS and fuzzing attacks were considered to validate the V-SOC4AS solution.

For the DoS dataset considering the OpelAstra vehicle, the authors inject messages with CAN ID "000" for 10 s to simulate a flood of a 500 Kbps on the CAN bus. The packets injected are in the form "000\#0000000000000000". The total number of DoS packets injected is 40015, the first packet injected is "(1536574995.000091) slcan0 000\#0000000000000000", and the last injected packet is "(1536575004.999811) slcan0 000\#0000000000000000". The total number of messages are 827555.

For the fuzzing experiment, the "fuzzing_canid.log" dataset was used, which was comprised of an injection of 10 messages in total, with illegitimate CAN ID values. In this dataset, the data field is populated with hexadecimal values that are set to "FF". Table 1 shows the data injected in the Fuzzing dataset. The researchers used the same injected messages to create another dataset called RenaultClio [41].

Table 1. Fuzzing messages injected by the authors [41].

CAN ID	Data Field
111	FFFFFFFFFFFFFFFF
111	FFFFFFFFFFFFFFFF
111	FFFFFFFFFFFFFFFF
222	FFFFFFFFFFFFFFFF
222	FFFFFFFFFFFFFFFF
222	FFFFFFFFFFFFFFFF
333	FFFFFFFFFFFFFFFF
333	FFFFFFFFFFFFFFFF
333	FFFFFFFFFFFFFFFF
444	FFFFFFFFFFFFFFFF

To prove that V-SOC4AS can also be used to identify threats in multiple vehicles, the fuzzing dataset was divided into two splits. In the first five messages (first split) the attribute *vehicleModel* was set to "OpelAstra" and for the last five messages (second split) the attribute *vehicleModel* was set to "RenaultClio". The attribute *UUID_ECU* was the same for both splits to simulate the case in which two different vehicles have the same ECU produced by a specific original equipment manufacturer (OEM).

5.2. DoS Attack

Since CAN does not have inherent authentication mechanisms, an attacker can send high-priority messages (for example, "000") to manipulate the bus. To detect this attack, a rule was created in QRadar that checks if the CAN ID of a specific event contains all zeros.

When true, the rule raises an offense. Figure 10 shows the DoS rule. The first row is used to filter out the other events not related to DoS. For example, messages with QIDs belonging to ECU data transfer (see Figure 4) are filtered for further analysis using this rule. The second row indicates a regular expression (Regex), a string of text that allows searching a pattern in the text. If the CAN ID has all zeros, an offense is generated.

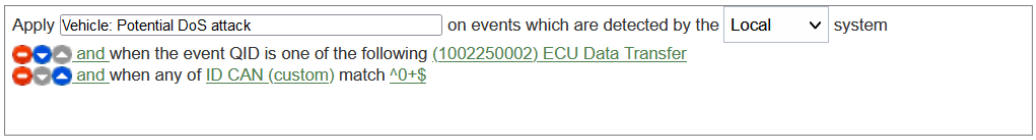

Figure 10. DoS rule in QRadar.

To test the DoS rule, the dataset "dosattack.log" contained in the OpelAstra dataset was used. The messages were transformed into JSON format with additional fields: UUID_ECU and vehicleModel. Figure 11 shows some of the messages that were sent to IBM QRadar.

Figure 11. Example of JSON messages that correspond to DoS attack.

To generate an offense, IBM QRadar uses a component called *Rule Wizard*. Figure 12 shows the actions chosen to raise the offense regarding the DoS attack. "Dispatch New Event" defines the field used to generate the offense such as the Event Name and Event Description. The SIEM offers the possibility of sending an e-mail when an attack occurs or adding the event in a Reference to save the results, and execute custom actions. These actions could be used for the response phase, for example, to send an email to alert the driver if an attack occurs. By choosing the action "Add to a Reference Set" and "Add to a Reference Data", the values regarding the attack will be saved to understand where the attack occurs and what payload has been sent by the attacker. The role of the "Reference Set" is to save the CAN ID(s) that are sent as an attack on the bus. Therefore, it is possible to resolve the problem by fixing the ECU source code. In IBM QRadar SIEM, the Reference Set is a collection of unique values. The role of the "Reference Data" is to save the vehicle model and the ECU ID.

The reference data are important for two reasons. First, since V-SOC can analyze multiple vehicles, it is possible to determine based on the vehicle model, if the attacker is attacking multiple vehicles or if the attack has already been identified in other vehicles. If the attack has been identified in other vehicles, we have the knowledge about the attack and, hence know, how to thwart it. Second, with the ECU ID we know where the attack began and if other vehicles are mounted with the same ECU. With this information, the developers can fix the ECU source code and extend the fix not only to the attacked vehicle but also to others using the same ECU. In this way, it is possible to prevent an adversary from attacking other similar vehicles and ECUs.

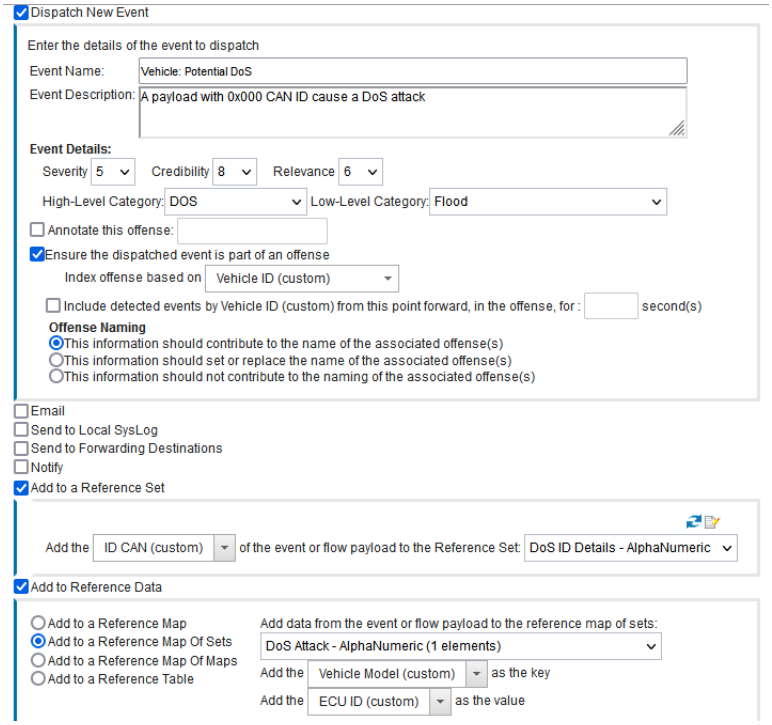

Figure 12. Rule Wizard with the field used to identify the DoS attack.

The SIEM allows the addition of information about the severity, credibility, and relevance of the attack. The severity responds to the question: "How high is the potential damage to the destination?"; the credibility responds to the question: "How valid is the information from that source?"; and the relevance responds to the question: "How important is the destination?". This information can be used to understand the threat level and required response time. For example, the fuzzing attack has a severity level higher than the DoS attack because it could activate anomalous behavior in the vehicle. For this reason, the fuzzing attack has a higher priority than the DoS attack. The generation of these three values depends on the organization; it comes with the experience of the SOC analyst, the common vulnerability scoring system (CVSS), or other parameters. In this paper, the generation of these values was considered as general as possible. For example, considering Figure 12 in the case of the DoS attack, the severity is five because this attack could flood the network. However, it could be non-destructive to the behavior of the vehicle. The credibility is set at value eight because it was considered that the attacker had direct access to the bus. Finally, the relevance is set at value six because this depends on the network that is being protected. For example, the powertrain will have more impact in case of an attack than the window management system because it involves acceleration, transmission, and other critical components. In this context, it was considered as general as possible.

When a payload with CAN ID 000 is sent to IBM QRadar, an offense is generated. Figure 13 shows the total number of DoS messages identified by the SIEM (red rectangle).

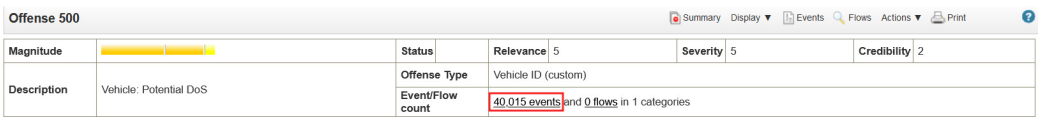

Figure 13. Generated offenses when a DoS attack is detected.

Figure 12 shows that the reference set saves the unique CAN_ID detected as the DoS attack. Instead, the reference data, in particular reference map of sets, is used to save the vehicleModel (as a key) and the UUID_ECU (as a value). To clarify, the reference set

does not contain all CAN IDs because the set is an abstract data type that can store only unique values.

5.3. Fuzzing Attack

In this attack, an attacker sends a random ID and a CAN data frame. The following approach was considered to detect the fuzzing attack with SIEM: each ECU has its identifier determined at the design phase and consequently, the SOC analyst could create the database with ECU identifiers. For this reason, a reference set was created on QRadar by putting CAN IDs without attacks. Usually, a reference set contains all unique values that can also be used to store business data such as IP address, and username. The reference set contains the unique CAN IDs for the OpelAstra and RenaultClio dataset.

These IDs have been taken from "full_data_capture.log" present in the OpelAstra and RenaultClio dataset that contains 2690069 packets captured with a total of 85 unique CAN IDs for the first dataset, and 386567 packets captured with 55 unique CAN IDs for the second dataset [41]. In Figure 14, the JSON messages regarding the Fuzzing attack were created in this way:

- The first five messages are used for the OpelAstra vehicle. The last five for the RenaultClio vehicle. The UUID related to the OpelAstra will not be changed, while a new one is generated for the RenaultClio.
- The UUID_ECU is the same for both vehicles. In this way, it is possible to simulate two vehicles that mount the same ECU and are attacked at the same time.


```
{"UUID": "53cd3082a0", "UUID_ECU": "8b470861", "vehicleModel": "OpelAstra", "eventID": "CAN Payload", "
    eventCategory": "Flow", "Timestamp": "1536574998.963000", "ID": "111", "DATA": "FFFFFFFFFFFFFFFF"}
{"UUID": "53cd3082a0", "UUID_ECU": "8b470861", "vehicleModel": "OpelAstra", "eventID": "CAN Payload", "
    eventCategory": "Flow", "Timestamp": "1536575008.043000", "ID": "111", "DATA": "FFFFFFFFFFFFFFFF"}
{"UUID": "53cd3082a0", "UUID_ECU": "8b470861", "vehicleModel": "OpelAstra", "eventID": "CAN Payload", "
    eventCategory": "Flow", "Timestamp": "1536575036.002000", "ID": "111", "DATA": "FFFFFFFFFFFFFFFF"}
{"UUID": "53cd3082a0", "UUID_ECU": "8b470861", "vehicleModel": "OpelAstra", "eventID": "CAN Payload", "
    eventCategory": "Flow", "Timestamp": "1536575061.281000", "ID": "222", "DATA": "FFFFFFFFFFFFFFFF"}
{"UUID": "53cd3082a0", "UUID_ECU": "8b470861", "vehicleModel": "OpelAstra", "eventID": "CAN Payload", "
    eventCategory": "Flow", "Timestamp": "1536575081.616000", "ID": "222", "DATA": "FFFFFFFFFFFFFFFF"}

{"UUID": "6fab142fe3", "UUID_ECU": "8b470861", "vehicleModel": "RenaultClio", "eventID": "CAN Payload", "
    eventCategory": "Flow", "Timestamp": "1536575109.908500", "ID": "222", "DATA": "FFFFFFFFFFFFFFFF"}
{"UUID": "6fab142fe3", "UUID_ECU": "8b470861", "vehicleModel": "RenaultClio", "eventID": "CAN Payload", "
    eventCategory": "Flow", "Timestamp": "1536575133.263300", "ID": "333", "DATA": "FFFFFFFFFFFFFFFF"}
{"UUID": "6fab142fe3", "UUID_ECU": "8b470861", "vehicleModel": "RenaultClio", "eventID": "CAN Payload", "
    eventCategory": "Flow", "Timestamp": "1536575165.597000", "ID": "333", "DATA": "FFFFFFFFFFFFFFFF"}
{"UUID": "6fab142fe3", "UUID_ECU": "8b470861", "vehicleModel": "RenaultClio", "eventID": "CAN Payload", "
    eventCategory": "Flow", "Timestamp": "1536575181.757100", "ID": "333", "DATA": "FFFFFFFFFFFFFFFF"}
{"UUID": "6fab142fe3", "UUID_ECU": "8b470861", "vehicleModel": "RenaultClio", "eventID": "CAN Payload", "
    eventCategory": "Flow", "Timestamp": "1536575200.992100", "ID": "444", "DATA": "FFFFFFFFFFFFFFFF"}
```


Figure 14. The JSON messages created to test the Fuzzing rule.

Figure 15 shows the rule created to identify the fuzzing attacks. The first row works similar to the DoS rule. The second row is used to avoid being recognized as an event of DoS attack. In the third row, if the ID of the current payload sent to QRadar does not exist in the table, the offense will be raised.

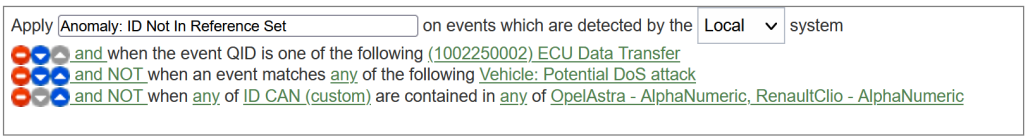

Figure 15. Fuzzing rule in QRadar.

The actions chosen for the fuzzing attack are shown in Figure 16. The severity value was set to eight because in this case, this attack can impair the behavior of the vehicle by sending random identifiers. Credibility and relevance are the same as the DoS attack.

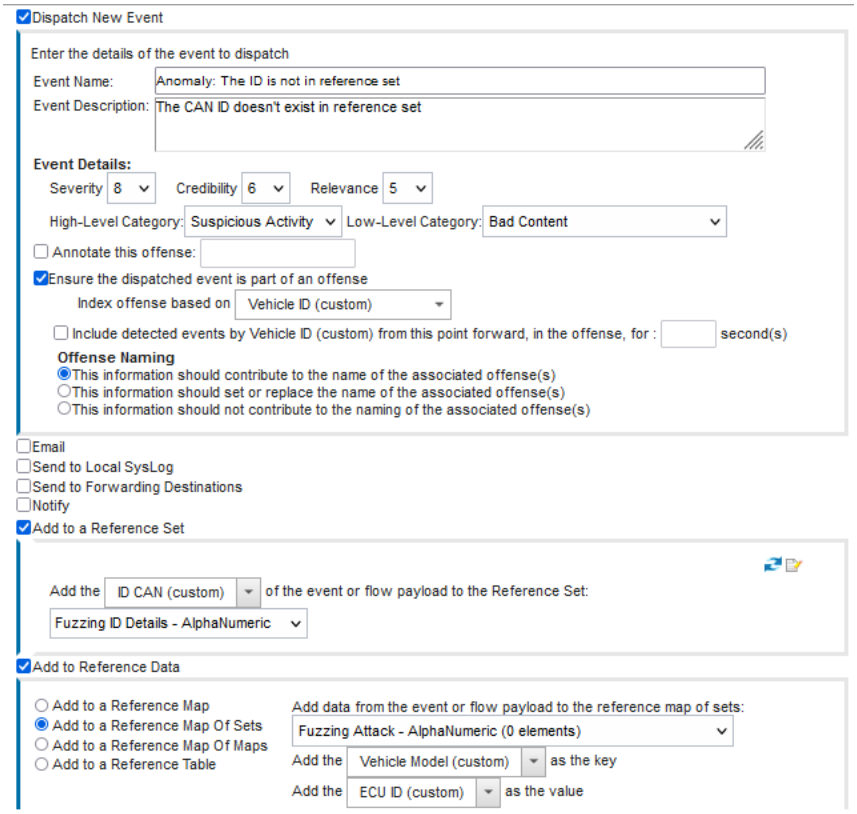

Figure 16. The actions chosen for the fuzzing attack.

Figure 17a shows the generated offense when the five fuzzing messages are sent and analyzed by the SIEM. This offense corresponds to the OpelAstra (green rectangle corresponds to the UUID). Instead, Figure 17b shows the offense generated for the RenaultClio.

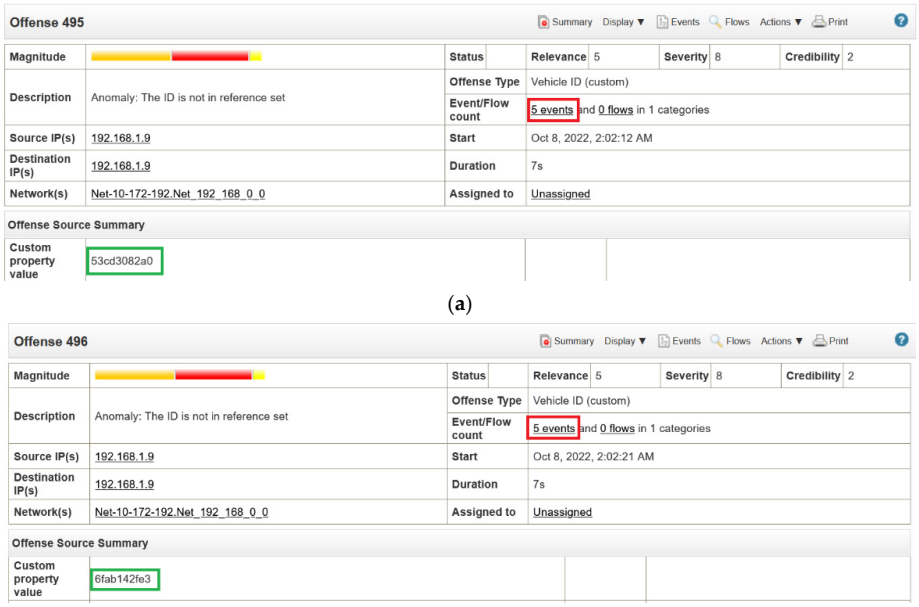

Figure 17. Generated offenses when a fuzzing attack is detected. (**a**) Shows the offense regarding the vehicle model OpelAstra with five identified attacks (red rectangle). In (**b**) the figure refers to the RenaultClio with five identified attacks.

The reference set contains the CAN_IDs detected by the SIEM as a fuzzing attack. In addition, the reference map of set contains the vehicleModel as a key and the UUID_ECU as a value. In this approach, using the reference map of set, if the attack occurs in multiple vehicles, the information of where the attack happened and whether multiple vehicles are involved are available. For example, with the UUID of the ECU, the SOC analyst could control if other vehicles mount the same ECU and proceed to implement security measures before the attack is carried out in these vehicles.

The metrics accuracy, precision, recall, and F1-score are used to evaluate the SIEM's detection performance [22]. For the DoS attack, the number of injected messages analyzed was 40.015. As we see in Section 5.2 Figure 13, with the created rule, the SIEM successfully analyzes all the messages. In this case, the SIEM obtained 100% accuracy, precision, recall, and F1-score because all 40.015 messages were recognized correctly (see Section 5.1 for the dataset details).

Considering the fuzzing attacks, Section 5.3 Figure 17a,b show that SIEM was able to detect all 10 CAN ID attacks for both vehicle datasets. In addition, IBM QRadar SIEM reaches 100% accuracy, precision, recall, and F1-score. Table 2 summarizes the results obtained with IBM QRadar SIEM. The results show that the SIEM, while using a rule-based system, can identify attacks that might occur on the CAN bus.

Table 2. Evaluation results obtained with IBM QRadar SIEM using a rule-based method.

Attack	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
DoS	100	100	100	100
Fuzzing	100	100	100	100

Finally, with the proposed architecture, it is possible to prevent and respond to the attack with the vehicle IP. In addition, the ECU_UUID and the vehicleModel parameters add greater detail to identify the vulnerability that caused the attack

6. Conclusions

Automotive security is critical to ensure driver safety and comfort in an interconnected vehicle. However, as vehicle connectivity becomes commonplace, new security risks emerge because communication protocols, such as the CAN network, are still insecure and vulnerable to attacks.

Therefore, in this work, a vehicle-SOC for improving automotive security (V-SOC4AS) was proposed. The V-SOC4AS aims to work alongside current systems of IDSs and IPSs to improve attack detection but, more importantly, to define a response phase in the automotive context.

In this work, two types of attacks, DoS and fuzzing, were analyzed to validate the proposed approach and to detect these attacks, two rules were developed in IBM QRadar SIEM. An advantage of the V-SOC4AS is that, with the UUID_ECU and vehicleModel parameters discussed in Section 5, it is possible to figure out which vehicles are attacked and the specific ECU that is vulnerable. In this way, it is possible to identify and fix the vulnerability that caused the attack and update vehicles of the same model, and those that mount the same ECU model. This can be helpful to prevent an attack that can occur on other vehicles. An additional advantage of the architecture presented in the paper is that through the vehicle IP on the SOC console, a proper response could be delivered according to the different strategies presented in Section 4.2.

Finally, it is possible to generalize the presented approach not only to identify and detect attacks but also to analyze the entire vehicle and not just the CAN protocol, as the SIEM is customizable with different information to be compatible with other protocols.

Author Contributions: Conceptualization, V.S.B. and D.C.; methodology, V.S.B. and M.D.V.; validation, M.S., M.Á.S.M., and D.C.; formal analysis, V.S.B. and M.D.V.; investigation, V.S.B. and M.D.V.; writing—original draft preparation, V.S.B. and M.D.V.; writing—review and editing, D.C., M.Á.S.M., M.S., and A.R.; visualization, M.D.V.; supervision, V.S.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the following projects: SSA (Secure Safe Apulia—Regional Security Center, Codice Progetto 6ESURE5) and KEIRETSU (Codice Progetto V9UFIL5) funded by "Regolamento regionale della Puglia per gli aiuti in esenzione n. 17 del 30 September 2014 (BURP n. 139 suppl. del 06 October 2014) TITOLO II CAPO 1 DEL REGOLAMENTO GENERALE "Avviso per la presentazione dei progetti promossi da Grandi Imprese ai sensi dell'articolo 17 del Regolamento".

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Dibaei, M.; Zheng, X.; Jiang, K.; Abbas, R.; Liu, S.; Zhang, Y.; Xiang, Y.; Yu, S. Attacks and Defences on Intelligent Connected Vehicles: A Survey. *Digit. Commun. Netw.* **2020**, *6*, 399–421. [CrossRef]
2. Bozdal, M.; Samie, M.; Jennions, I. A Survey on Can Bus Protocol: Attacks, Challenges, and Potential Solutions. In Proceedings of the 2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE), Southend, UK, 16–17 August 2018; IEEE: New York, NY, USA, 2018; pp. 201–205.
3. Miller, C.; Valasek, C. A Survey of Remote Automotive Attack Surfaces. *Black Hat USA* **2014**, *2014*, 94.
4. Palanca, A.; Evenchick, E.; Maggi, F.; Zanero, S. A Stealth, Selective, Link-Layer Denial-of-Service Attack against Automotive Networks. In Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Bonn, Germany, 6–7 July 2017; Springer: Cham, Switzerland, 2017; pp. 185–206.
5. Tencent Security Keen Lab Experimental Security Assessment of Mercedes-Benz Cars, Mercedes-Benz MBUX Security Research Report. Available online: https://keenlab.tencent.com/en/whitepapers/Mercedes_Benz_Security_Research_Report_Final.pdf (accessed on 7 February 2023).
6. Martínez-Cruz, A.; Ramírez-Gutiérrez, K.A.; Feregrino-Uribe, C.; Morales-Reyes, A. Security on In-Vehicle Communication Protocols: Issues, Challenges, and Future Research Directions. *Comput. Commun.* **2021**, *180*, 1–20. [CrossRef]
7. Doan, T.P.; Ganesan, S. CAN Crypto FPGA Chip to Secure Data Transmitted through CAN FD Bus Using AES-128 and SHA-1 Algorithms with a Symmetric Key; SAE Technical Paper 2017-01-1612, WCX™ 17: SAE World Congress Experience. 2017. Available online: https://www.sae.org/publications/technical-papers/content/2017-01-1612/ (accessed on 7 February 2023).
8. Siddiqui, A.S.; Gui, Y.; Plusquellic, J.; Saqib, F. Secure Communication over CANBus. In Proceedings of the 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), Boston, MA, USA, 6–9 August 2017; IEEE: New York, NY, USA, 2017; pp. 1264–1267.
9. Stabili, D.; Ferretti, L.; Marchetti, M. Analyses of Secure Automotive Communication Protocols and Their Impact on Vehicles Life-Cycle. In Proceedings of the 2018 IEEE International Conference on Smart Computing (SMARTCOMP), Taormina, Italy, 18–20 June 2018; pp. 452–457.
10. Cheng, K.; Bai, Y.; Zhou, Y.; Tang, Y.; Sanan, D.; Liu, Y. CANeleon: Protecting CAN Bus with Frame ID Chameleon. *IEEE Trans. Veh. Technol.* **2020**, *69*, 7116–7130. [CrossRef]
11. Kornaros, G.; Bakoyiannis, D.; Tomoutzoglou, O.; Coppola, M.; Gherardi, G. TrustNet: Ensuring Normal-World and Trusted-World CAN-Bus Networking. In Proceedings of the 2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), Beijing, China, 21–23 October 2019; pp. 1–6.
12. Wu, W.; Li, R.; Xie, G.; An, J.; Bai, Y.; Zhou, J.; Li, K. A Survey of Intrusion Detection for In-Vehicle Networks. *IEEE Trans. Intell. Transp. Syst.* **2019**, *21*, 919–933. [CrossRef]
13. Sommer, F.; Dürrwang, J.; Kriesten, R. Survey and Classification of Automotive Security Attacks. *Information* **2019**, *10*, 148. [CrossRef]
14. Baldassarre, M.T.; Barletta, V.S.; Caivano, D.; Raguseo, D.; Scalera, M. Teaching Cyber Security: The HACK-SPACE Integrated Model. In Proceedings of the ITASEC, Pisa, Italy, 13–15 February 2019.
15. Costantino, G.; Matteucci, I.; Morales, D. EARNEST: A Challenge-Based Intrusion Prevention System for CAN Messages. In Proceedings of the 2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Coimbra, Portugal, 12–15 October 2020; IEEE: New York, NY, USA, 2020; pp. 243–248.

16. Asokan, N.; Nyman, T.; Rattanavipanon, N.; Sadeghi, A.-R.; Tsudik, G. ASSURED: Architecture for Secure Software Update of Realistic Embedded Devices. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2018**, *37*, 2290–2300. [CrossRef]
17. Mbakoyiannis, D.; Tomoutzoglou, O.; Kornaros, G. Secure Over-the-Air Firmware Updating for Automotive Electronic Control Units. In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, Limassol, Cyprus, 8–12 April 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 174–181.
18. Catalano, C.; Paiano, L.; Calabrese, F.; Cataldo, M.; Mancarella, L.; Tommasi, F. Anomaly Detection in Smart Agriculture Systems. *Comput. Ind.* **2022**, *143*, 103750. [CrossRef]
19. Lee, H.; Jeong, S.H.; Kim, H.K. OTIDS: A Novel Intrusion Detection System for In-Vehicle Network by Using Remote Frame. In Proceedings of the 2017 15th Annual Conference on Privacy, Security and Trust (PST), Calgary, AB, Canada, 28–30 August 2017; IEEE: New York, NY, USA, 2017; pp. 57–5709.
20. El-Rewini, Z.; Sadatsharan, K.; Selvaraj, D.F.; Plathottam, S.J.; Ranganathan, P. Cybersecurity Challenges in Vehicular Communications. *Veh. Commun.* **2020**, *23*, 100214. [CrossRef]
21. Scarfone, K.; Mell, P. *Guide to Intrusion Detection and Prevention Systems (IDPS)*; Special Publication 800-94; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2007.
22. Barletta, V.S.; Caivano, D.; Nannavecchia, A.; Scalera, M. A Kohonen SOM Architecture for Intrusion Detection on In-Vehicle Communication Networks. *Appl. Sci.* **2020**, *10*, 5062. [CrossRef]
23. Seo, E.; Song, H.M.; Kim, H.K. GIDS: GAN Based Intrusion Detection System for In-Vehicle Network. In Proceedings of the 2018 16th Annual Conference on Privacy, Security and Trust (PST), Belfast, Ireland, 28–30 August 2018; IEEE: New York, NY, USA, 2018; pp. 1–6.
24. Cho, K.-T.; Shin, K.G. Fingerprinting Electronic Control Units for Vehicle Intrusion Detection. In Proceedings of the 25th USENIX Security Symposium (USENIX Security 16), Austin, TX, USA, 10–12 August 2016; pp. 911–927.
25. Lokman, S.-F.; Othman, A.T.; Abu-Bakar, M.-H. Intrusion Detection System for Automotive Controller Area Network (CAN) Bus System: A Review. *EURASIP J. Wirel. Commun. Netw.* **2019**, *2019*, 184. [CrossRef]
26. Young, C.; Zambreno, J.; Olufowobi, H.; Bloom, G. Survey of Automotive Controller Area Network Intrusion Detection Systems. *IEEE Des. Test* **2019**, *36*, 48–55. [CrossRef]
27. Fallstrand, D.; Lindström, V. Applicability Analysis of Intrusion Detection and Prevention in Automotive Systems. Master's Thesis, Chalmers University of Technology, Gothenburg, Sweden, 2015.
28. Falk, E.; Repcek, S.; Fiz, B.; Hommes, S.; State, R.; Sasnauskas, R. VSOC-a Virtual Security Operating Center. In Proceedings of the GLOBECOM 2017—2017 IEEE Global Communications Conference, Singapore, 4–8 December 2017; IEEE: New York, NY, USA, 2017; pp. 1–6.
29. Aijaz, L.; Aslam, B.; Khalid, U. Security Operations Center—A Need for an Academic Environment. In Proceedings of the 2015 World Symposium on Computer Networks and Information Security (WSCNIS), Hammamet, Tunisia, 19–20 September 2015; IEEE: New York, NY, USA, 2015; pp. 1–7.
30. Madani, A.; Rezayi, S.; Gharaee, H. Log Management Comprehensive Architecture in Security Operation Center (SOC). In Proceedings of the 2011 International Conference on Computational Aspects of Social Networks (CASoN), Salamanca, Spain, 19–20 October 2011; IEEE: New York, NY, USA, 2011; pp. 284–289.
31. Bidou, R. Security Operation Center Concepts & Implementation. 2005. Available online: https://www.researchgate.net/publication/228587242_Security_Operation_Center_Concepts_Implementation (accessed on 7 February 2023).
32. Shahjee, D.; Ware, N. Designing a Framework of an Integrated Network and Security Operation Center: A Convergence Approach. In Proceedings of the 2022 IEEE 7th International conference for Convergence in Technology (I2CT), Mumbai, India, 7–9 April 2022; IEEE: New York, NY, USA, 2022; pp. 1–4.
33. Langer, F.; Schüppel, F.; Stahlbock, L. Establishing an Automotive Cyber Defense Center. In Proceedings of the 17th Escar Europe: Embedded Security in Cars, Stuttgart, Germany, 19–20 November 2019.
34. Meyer, P.; Hackel, T.; Langer, F.; Stahlbock, L.; Decker, J.; Eckhardt, S.A.; Korf, F.; Schmidt, T.C.; Schüppel, F. A Security Infrastructure for Vehicular Information Using Sdn, Intrusion Detection, and a Defense Center in the Cloud. In Proceedings of the 2020 IEEE Vehicular Networking Conference (VNC), New York, NY, USA, 16–18 December 2020; IEEE: New York, NY, USA, 2020; pp. 1–2.
35. Tommasi, F.; Catalano, C.; Taurino, I. Browser-in-the-Middle (BitM) Attack. *Int. J. Inf. Secur.* **2022**, *21*, 179–189. [CrossRef]
36. Stabili, D.; Ferretti, L.; Andreolini, M.; Marchetti, M. DAGA: Detecting Attacks to In-Vehicle Networks via N-Gram Analysis. *IEEE Trans. Veh. Technol.* **2022**, *71*, 11540–11554. [CrossRef]
37. IBM. IBM QRadar Security Intelligence. Available online: https://www.ibm.com/products/qradar-siem (accessed on 7 February 2023).
38. Magic Quadrant for Security Information and Event Management. Available online: https://www.gartner.com/doc/reprints?id=1-2BDC4CEU&ct=221010&st=sb (accessed on 7 February 2023).
39. IBM Architecture and Deployment Guide. Available online: https://www.ibm.com/docs/en/SS42VS_7.4/pdf/b_siem_deployment.pdf (accessed on 7 February 2023).

40. Gerhards, R. *The Syslog Protocol*. Available online: https://www.rfc-editor.org/rfc/rfc5424 (accessed on 9 February 2023).
41. Dupont, G.; Lekidis, A.; Den Hartog, J.; Etalle, S. *Automotive Controller Area Network (CAN) Bus Intrusion Dataset V2*; 4TU.Centre for Research Data: Delft, The Netherlands, 2019.