Closing the Security Gaps in SOME/IP Through Implementation of a Host-Based Intrusion Detection System

Casparsen, Andreas; Sørensen, Daniel Greth; Andersen, Jeppe Nellemann; Christensen, Jonas Ingerslev; Antoniou, Panagiotis; Krøyer, Rolf; Madsen, Tatiana; Gjoerup, Karsten

Published in:
2022 25th International Symposium on Wireless Personal Multimedia Communications, WPMC 2022

DOI (link to publication from Publisher):
10.1109/WPMC55625.2022.10014951

Publication date:
2022

Document Version
Accepted author manuscript, peer reviewed version

Closing the Security Gaps in SOME/IP Through Implementation of a Host-Based Intrusion Detection System

Andreas Casparsen*, Daniel Greth Sørensen*, Jeppe Nellemann Andersen*, Jonas Ingerslev Christensen*, Panagiotis Antoniou*, Rolf Krøyer*, Tatiana Madsen*, and Karsten Gjoerup†
* Department of Electronic Systems, Aalborg University, Denmark
† MAXSYT Technologies ApS, Denmark

***Abstract*—Scalable service-Oriented MiddlewarE over IP (SOME/IP) is a protocol that provides services over the IP stack, targeted at the automobile industry that strive to implement Ethernet in future vehicles as a replacement of Controller Area Network (CAN). However, switching to Ethernet and SOME/IP changes the entire protocol stack and therefore security needs to be reconsidered. With the assumption that a malicious user has access to the in-vehicle network, four different attacks are identified that can be performed on SOME/IP. Knowing how the network and traffic shall behave in the vehicle, a set of features are used as the basis for an Intrusion Detection System (IDS). These attacks along with the respective defences are implemented and tested on a SOME/IP network. The results from the testbed have demonstrated that the proposed attacks can be performed and therefore, additional security outside the SOME/IP protocol is needed. A host-based IDS is proposed, where the detection is based on arrival time, payload values and packet contradictions. Furthermore, suggestions on how to move towards prevention are given. The purpose of this research is to improve overall SOME/IP security for in-vehicle networks.**

***Keywords*—In-Vehicle Network, SOME/IP, DoS Attack, Impersonation Attack, Security, IDS, Ethernet**

I. Introduction

In recent years the complexity of cars has been increasing rapidly. This has partly been due to the strive towards autonomous vehicles, which requires an extensive amount of electronic devices placed inside a car. Among these devices are various sensors and Electronic Control Units (ECU), which all are interconnected, creating a network, whose complexity increases with the amount of connected devices and the load of the network.

The in-vehicle network often consists of multiple Local Area Networks (LAN), which can use different communication technologies, depending on the requirements for the communication. The de facto standard for part of the network in today's vehicles is CAN [1]. CAN is a robust communication protocol, which utilizes a single bus, and allows for a bit rate up to 1 Mbit/s [2]. This works sufficiently when the amount of ECUs connected to the network is limited and the amount of data they transmit is supported by the bit rate, but when the network grows in size and becomes more complex, CAN cannot fully support the ECUs communication. Car manufacturers are starting to look into other communication technologies, such as Ethernet, since it allows for more complex networks, at up to a thousand times higher data rate than CAN. But Ethernet and CAN work very differently: CAN is a multi-master serial bus, whereas modern Ethernet uses twisted pair or fiber optic cables, and can be deployed having a switched or repeated connection.

To retain the basic philosophy of CAN while utilizing Ethernet capabilities, an option would be to use SOME/IP. SOME/IP itself is designed to provide information and services over the network using a few different mechanisms. A device can send a Remote Procedure Call (RPC) to request information or get an action performed. Alternatively devices can be configured as a Publish/Subscribe relationship to exchange information. SOME/IP is built to be used on Ethernet, and use IP and TCP/UDP as the transport and network protocols. This means that CAN compared to SOME/IP on Ethernet, not only is different on the physical medium, it is also different in terms of the protocol stack.

Besides the performance aspect of in-vehicle networks, there is the **security** aspect. With increased complexity and connection to external networks, vehicles are exposed to malicious threats. For example in 2015 C. Miller et al. [3] illustrated that it is possible to remotely affect steering and the brakes of certain Fiat-Chrysler vehicles, resulting in the recall of 1.4 million vehicles. This failure has been caused by the poor design of the in-vehicle CAN network, combined with some exploits of a cellular network.

Currently, there is a lot of focus on the security of CAN and on CAN networks. But as mentioned earlier there is going to be a potential technology shift towards Ethernet as a replacement for CAN, as a backbone in in-vehicle networks. For the success of any new technology, security shall be considered from the beginning. Therefore the security aspect of SOME/IP is investigated, as it shall be considered from the start of its potential tenure as an adopted standard.

SOME/IP is designed to be used for "traditional CAN scenarios" [4], allowing for the potential recycling of security knowledge already gained from CAN. In a paper H. Lee et al. [5] showed that CAN is vulnerable to certain attacks, but also showed how the expert knowledge of consistent

behaviour in a in-vehicle network, could be used for intrusion detection. However, the entire protocol stack of SOME/IP is different, leading to an array of different attacks and defence mechanisms, which makes IDS already developed for CAN, or other network technologies impractical. The goal of this paper is to give the recommendations on how the security of SOME/IP can be enhanced.

The rest of the paper is organized as follows. Section II introduces the technical background information of SOME/IP, and related works about SOME/IP, existing car network security and IDS. Section III presents four different attack models. Section IV proposes a host-based IDS. Section V describes experiments and provides evaluation results. Then, in Section VI prevention methods are discussed and finally Section VII concludes on the results.

II. Preliminaries

A. SOME/IP

This section provides a technical description of the SOME/IP protocol [4]. Devices on a SOME/IP network have two ways of communicating. SOME/IP has Request/Response functionality, which allows a device at any given time to send a request to an ECU offering a service in the form of a RPC. This may not always trigger a response, in which case this is referred to as fire & forget, request/response communication occurs through unicast transmissions. Another way of communicating is through Publish/Subscribe, which allows for devices to subscribe to an event-group comprised of events from similar services. The subscribers will then receive notifications from the event-group, when new events occur. This communication may occur through either unicast or multicast depending on the amount of subscribers.

An important part of understanding the SOME/IP protocol is through an understanding of the frame format the protocol uses. The protocol frame is illustrated in Fig. 1.

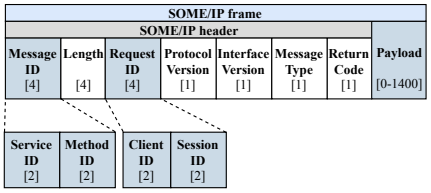

Figure 1. Structure of SOME/IP frame, the size of each field is given in [bytes]. The blue fields are further described.

- **Service ID** is the identifier for a system-wide service.
- **Method ID** is used to specify the method provided by a service for a RPC.
- **Client ID** is the ID of a client used to identify different clients.
- **Session ID** is a number that is incremented between each transmission to distinguish messages from the same sender.
- **Message type** is a parameter used to tell apart if the message is a response, request, notification or error.
- **Payload** is the field containing the data being transmitted.

B. SOME/IP-SD

SOME/IP Service Discovery (SOME/IP-SD or SD) is a protocol, which lies directly on top of the SOME/IP protocol. However, it is constrained to only work with IP based communication sent over an UDP connection. The SD protocol supports both unicast and multicast type communication.

The main purpose of the SD protocol, is to communicate and dynamically locate the available services in the in-vehicle network, and then manage the publisher/subscriber relation between ECUs. The SD protocol works by sending a predefined SOME/IP header followed by a SD header. The SOME/IP header is constructed with specific values in the following fields: *Service ID, Method ID, Protocol Version, Interface Version, Message Type and Return Code*. The values for these fields are specified by the protocol, and are shown together with the SD header on Fig. 2.

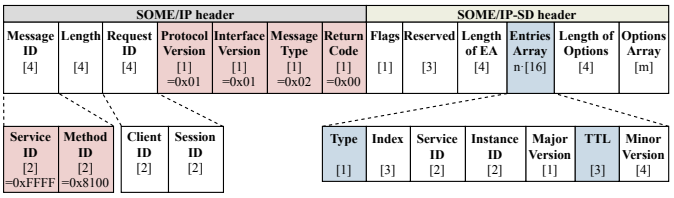

Figure 2. Structure of SOME/IP-SD message, the size of each field is given in [bytes]. The SOME/IP header fields are predefined. The SOME/IP-SD fields are further described.

The SD header contains six fields, whereas the *Entries Array* (EA) and the *Options Array* is further split up into various fields. Only some of the fields in the *Entries Array* are relevant to this investigation. These are the following.

- **Type** is one byte, which together with **Time-to-Live (TTL)** tell whether or not the ECU is trying to e.g. *find* or *offer* a service.
- **TTL** is three bytes, which describes how long the message is valid for in seconds. If it is set to 0 together with a message of **Type** *offer*, it becomes a *stop offer* message instead.

C. Related work

With the automotive industry moving to Ethernet for in-vehicle communication, new challenges regarding security are introduced, as stated by F. Sagstetter et al. [6]. They presented the challenges of integrating vehicles into Car2X and discussed that formal methods might be needed to describe what normal behavior of the vehicle is, in order to determine if an attack is ongoing.

Security flaws described by K. Koscher et al. [7] belong to the type of attacks that could be performed, if a malicious user has a physical access to a vehicle. The security of the ECUs in the vehicles under the test was poor and easily bypassed. Complete removal of the ability of the driver to brake or locking the brakes unevenly at any moment was demonstrated. Furthermore, some of the security flaws found were not supposed to occur according to the standard that the ECUs should follow.

Research for anomaly detection in CAN networks and IDS for in-vehicle networks can be grouped into two categories. The first category includes works such as H. Lee et al. [5] and uses deviations from the known traffic pattern, e.g. via message frequency and timing analysis, or deviations in the message content for identification of malicious traffic presence. The second category adopts deep-learning methods for IDS for CAN networks. An example of works from the second category is E. Seo et al. [8], who has proposed an IDS that is capable of detection of both known and unknown attacks and is based on the use of Generative Adversarial Nets.

An IDS for a SOME/IP network was made by N. Herold et al. [9], who looked into using SOME/IP on an airplane and having a rule based Network IDS (NIDS) monitoring the network. The rules were mainly based around protocol specifics that should not be violated, e.g. if a response is received by one node, then another node must have sent it within a certain time-frame. This IDS had drawbacks in regards to real time processing.

From the existing research review, it is apparent that the security aspects of SOME/IP need further investigations. The existing security in SOME/IP is only concerned with, whether a message is valid in the eyes of the protocol specification [10], but not if the contents or intentions of the message is of malicious nature. This means that additional security is needed to ensure secure communication.

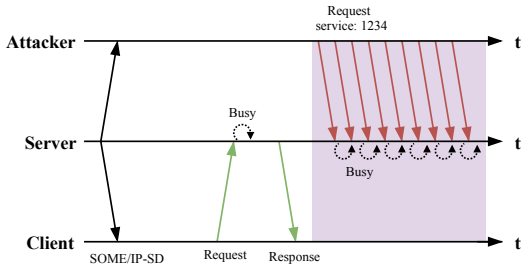

Figure 3. Message sequence diagram for normal RPC behaviour and the RPC Flooding attack.

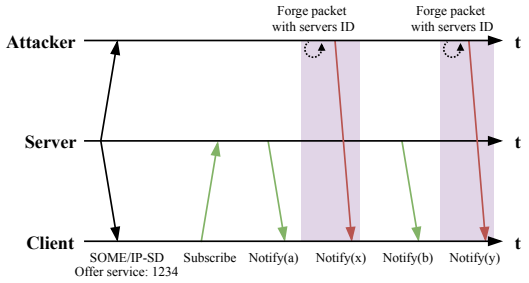

Figure 4. Message sequence diagram for normal Publish/Subscribe behaviour and the False Notify attack.

III. Attack Model

A. Adversary Model

In this paper the protocol specific vulnerabilities of SOME/IP and SOME/IP-SD are examined, how they can be exploited, the impact they have on the system and how attacks on such vulnerabilities can be detected. For the examined attacks, an attacker that has bypassed the first level of defense is assumed. An example of an entry point can be found through the Bluetooth connection between the car and a phone, shown by S. Checkoway et al. [11]; they also mention several alternative entry points. The attacker is able to capture broadcast and multicast transmissions in a switched network, meaning that SOME/IP-SD packets are available to them. This applies for Publish/Subscribe scenarios where the publisher(server) is configured to multicast SOME/IP packets to the subscribers(clients). In the case of Request/Response, the attacker can only capture the server's and client's SOME/IP-SD messages, since after a client is subscribed to a service the communication is unicasted. The protocol specific attacks proposed in this paper are the RPC Flooding, False Notify, Session Stealing and Stop Offer Attack.

B. Attack Scenarios

For the **RPC Flooding** shown on Fig. 3, when a service is offered, the server multicasts SOME/IP-SD packets on the network. The Service ID is included in the SOME/IP-SD header and can be extracted by the attacker. After getting the Service ID the attacker floods the server with RPC requests, making the server unable to respond to other tasks. This attack is categorized as a DoS attack.

A **False Notify** attack shown on Fig. 4, is an impersonation attack taking place in the Publish/Subscribe scenario. The attacker captures the multicasted SOME/IP-SD messages transmitted from the server containing the server's information. The attacker forges a packet with the same SOME/IP header information as the server, fills it with fake SOME/IP payload and transmits it to the network's multicast address. All clients subscribed to that service will receive the packet, extract the fake payload and update their real value with the false one.

Session Stealing shown on Fig. 5, is an impersonation attack, where the Server/Client connection is hijacked. The objective of the attacker is to steal the current Session ID of the communication and impersonate the Server. In a Publish/-Subscribe scenario, the transmission happens on the multicast channel. This means that each time the server sends a new notification event, the attacker also receives this packet. By inspecting the packet the attacker can extract the current Session ID located in SOME/IP header. Afterwards, the attacker retransmits the packet with the Session ID incremented. All the clients subscribed to this event will receive and process the forged packet with the increased Session ID. When the server sends a new notification, the message is to be ignored by the subscribers, since the Session ID is marked as invalid and the server gets cut out of the connection. The attacker acts as the server and feeds the subscribers with false data.

The **Stop Offer Attack** shown on Fig. 6, is a sophisticated

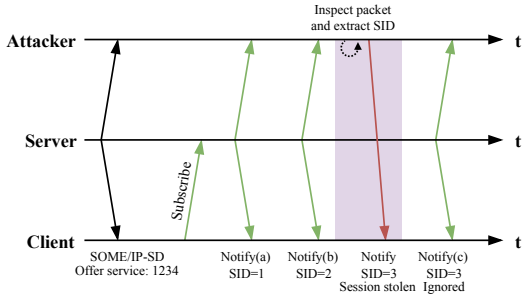

Figure 5. Message sequence diagram for normal Publish/Subscribe behaviour and the Session Stealing attack.

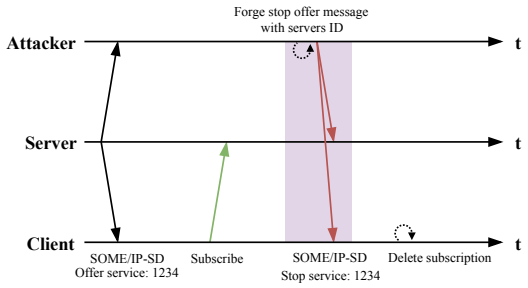

Figure 6. Message sequence diagram for normal Publish/Subscribe behaviour and the Stop Offer Attack.

DoS attack and a SOME/IP-SD protocol specific attack. When the attacker captures an Offer Service packet on the multicast channel, it forges a duplicate of this packet and changes the Time-to-Live field in SOME/IP-SD header from 0x00*i*, where *i* is Time-to-live of the offer for the specific service, to 0x000 which means that this service is not offered anymore and multicasts it. The clients already subscribed to this service will then remove their subscription to the server, when they receive the stop offer message, and clients will be unable to subscribe to this service since, to their knowledge, it is no longer available.

IV. Host-based Intrusion Detection System

A Host-based IDS (HIDS) is designed to inspect the ongoing traffic of the network in order to verify the validity of transmitted packets. The proposed IDS defence mechanisms are derived from Packet arrival time and by doing Packet Inspection.

Depending on the role of the ECU (Server/Client) a different rule-set is applied. The rules set up in the IDS can be created by someone with expert knowledge about how the in-vehicle communication behaves. That means that the boundaries of changes occurring in the payload and periodicity of the communication are predefined. How much the communication varies during normal conditions also determines how strict the rules can be set, combined with the amount of false positives that can be allowed.

The detection coverage is system-wide meaning that every ECU has its own IDS, independently trying to detect potential intrusions. Each Host-Based IDS is not cooperating with the rest of the system's IDSs. Below the principles of the intrusion detection mechanisms utilized are presented.

***Packet arrival frequency**:* For traffic in the vehicle that adhere to a deterministic traffic pattern, looking into the packet arrival frequency is one of the proposed rules, which will notice **RPC Flooding**, from the server side. The threshold for the packet interarrival time during normal behaviour is used to set up the alarm threshold. The packet interarrival time (alternatively estimation can be done by using packet frequency) dropping below a specified threshold would activate an alarm.

***Deep packet inspection**:* (Client Side) To detect the **False Notify Attack**, a threshold of the values of the notification event is defined. For example, the angle of the steering wheel can be limited to be less than 180 degrees. Whenever the clients receive a notification event, the IDS extracts the SOME/IP payload and compares the value contained with the boundaries defined. If the value exceeds the threshold, the IDS throws an intrusion alert.

An attacker might also send malicious messages within the allowed range. A method to detect this is to investigate the rate of change in the payload, where they may not differ more than a pre-defined amount per transmission, since there is limits to what the equipment is capable of doing. This works for periodic traffic patterns, however for non-periodic traffic the equivalent is to evaluate it as change over time.

***Session ID monitoring**:* For devices with Session ID handling enabled for their communication the IDS can be used to detect **Session Stealing Attacks**, on client side, by keeping a counter of the current Session ID of each session. This is done by extracting the Session ID field in the SOME/IP header. Detection is based using comparison of the received Session ID with the expected one.

***Contradictory messages**:* Detection of a **Stop Offer Attack** is based on traffic monitoring on any device in the Publish/-Subscribe setup. A device captures everything on the multicast channel, with the goal of detecting if anything is contradictory. This can be done in two ways, one is to find discrepancies between what is being multicasted by a device and what is received by the same device. If a message is received and it is seemingly from itself, but the device did not send that message, an alarm is activated. Alternatively, discrepancies can be found by inspecting only the received messages. An alarm is activated, if stop offer messages are received, but the server in the same time period is still multicasting notify messages.

V. Experiments and evaluation

Based on the attacks described, a test setup is created, composing of multiple Linux based Raspberry Pi 3B+'s connected through a switch and a Linux based laptop connected through the same Ethernet switch. Each of the Raspberry Pi's is setup to act as ECUs performing communication over SOME/IP with the implementation *vsomeip v2.14.16* [12]. The ECUs normal behaviour is defined by a dataset of CAN traffic in a Kia Soul [13]. In this vehicle most CAN network traffic is periodic with a mean time period between packets of 20ms.

For the impersonation attacks that target the Publish/Subscribe mode, a Raspberry Pi is setup, through the *vsomeip* implementation, as a Publisher node, by configuring it to advertise a service through the SD protocol, and then publish a service through the multicast channel. A Subscriber node is also setup on a Raspberry Pi, and configured to subscribe the advertised service. The service is published every 20ms. A laptop is then set, as an attacker, to send a SOME/IP packet that resemble the one published by the Publisher node, but with a malicious payload.

For the RPC-flooding a Raspberry Pi is setup, as the service node, also through *vsomeip*, to advertise a service through the SD protocol. Another Raspberry Pi node is then set to request this service at an interval of 20ms. A laptop is then set, as an attacker, to send request calls to the service node as fast as possible. Each of four tests start under normal control conditions, with no attacker active (illustrated as a grey region in the performance graphs), then after a few seconds, the attacker gets activated.

A. Detecting RPC Flooding

The RPC Flooding is a DoS attack and is detected by monitoring the packet arrival frequency feature. The packet arrival frequency and how it changes over time is shown as a graph in Fig. 7. The packet frequency is measured on the Server ECU, that generates responses to the requests. As soon as the attack starts, with at least 75.000 additional requests per second, less time between each request is observed.

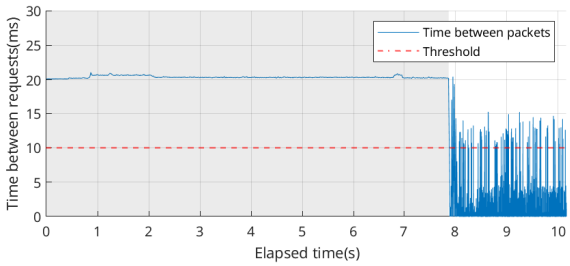

Figure 7. Results for time between each packet arrival. Shown in the gray area is the control state, and the white area is during the RPC Flooding. The threshold value used for detection is shown with a red dashed line.

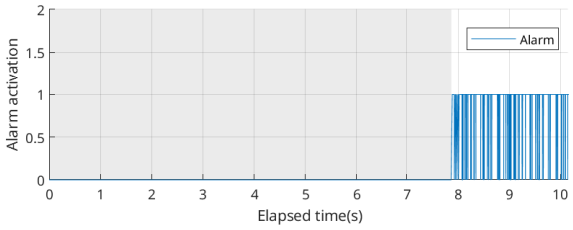

Figure 8. RPC Flooding detection results showing the alarm state for each packet in the control state marked gray, and the first part of the attack state. Here a value of zero means no alarm and a value of 1 means alarm.

Using packet arrival timing as a detection method against RPC Flooding shows the capability to detect anomalies during an attack, and also zero false alarms during normal operating conditions (see Fig. 8). This approach is applicable to the scenarios with a well defined periodic communication pattern for normal operating conditions. But the implemented detection cannot differentiate trustworthy messages from malicious messages during the RPC Flooding, limiting the IDS to just give an alarm that something is wrong and not which messages are wrong.

B. Detecting False Notify

The **False Notify Attack** is an impersonation attack and the detection is based on monitoring the value of the payload, looking both at the value boundaries and rate of change.

Figure 9 illustrates the test. The Server sends a notify message with a sinusoidal shape within the boundary of $(50, 100)$ under normal operation conditions. The malicious messages have a payload with a value within the permitted range, but the values are randomly selected following uniform distribution. The bottom graph in Fig. 9 shows that the detection of the ongoing attack is possible, since the rate of change exceeds the allowing minimum and maximum values (being -5 and 5 for the designed test). However, some of the malicious packets will still be accepted as legitimate, since they do not violate the boundaries, nor the allowed rate of change.

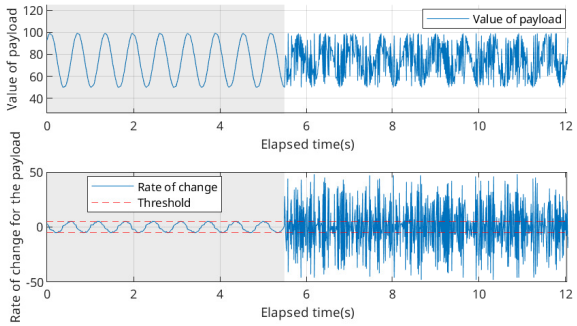

Figure 9. The top graph show the measured payload value over time in the control state marked gray and during the False Notify attack. The bottom graph show the rate of change calculated from the top graph over time, also for both the control and attack period.

C. Detecting Session Stealing

Session Stealing Attack could not be tested due to limitations in vsomeip implementation. Session handling for multicast messages can not be enabled, because it is not an implemented feature. Therefore monitoring how much the session ID deviated from expected value is not tested either.

D. Detecting Stop Offer Attack

The **Stop Offer Attack** is a DoS attack and it is performed by sending stop offer messages right after the offer service messages, or at a faster rate than the offer service messages in order to disable the service. This test is done by sending stop offer messages every 2 seconds. This test shows that when the attack starts, the client almost instantaneously stops receiving notify messages. On Fig. 10 the effect of the attack is shown, displaying how many packets of different types are registered on the clients network interface over time.

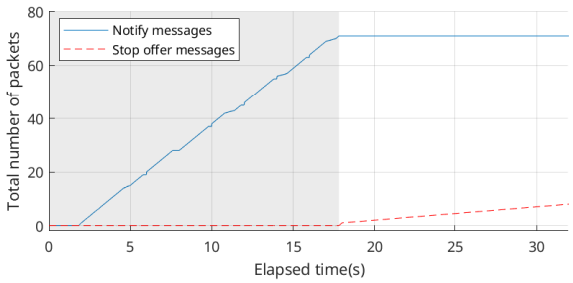

Figure 10. Measured amount of packets over time on the client, for both the notify type(blue) and stop offer type(red dashed). The gray area is the state with no attack and the white area is with the Stop Offer Attack active.

The detection of this attack uses contradictions in messages between messages sent and received. Detection is implemented by observing if any stop offer messages appear while the server is active, since the server is not set up to send stop offer messages. The server side detection will work, as long as the server can keep track of its own messages, it can guaranty the ability to tell the impersonated messages apart from the real ones.

VI. Prevention

RPC Flooding: It is difficult to fully prevent this attack, because the amount of packets that needs to be processed is high. Therefore one way to ensure that the ECU will not constantly be interrupted and incapable of fulfilling its tasks, can be to throttle the packet throughput and only allow a certain amount through. This applies to both incoming and outgoing traffic, the throttling or blocking can be done in both contexts and will help to not overload an ECU.

Session Stealing: To prevent an ongoing Session Stealing attack, its presence will need to be detected first, once this is done the ECU shall try to resubscribe to the service to start a new instance, and thereby be unaffected by the attack on the other instance's session handling.

False Notify: If the notify messages are sent using unicast, then the only time an attacker can see the information needed to target a service is during the service discovery. To conceal the information, a negotiation on a private channel can be used, or a static setup of the SOME/IP network can be applied. In the context of a vehicle having a few milliseconds slower startup using a more complicated service discovery negotiation might not matter. If the notify messages are sent using multicast, it is recommended that session handling is enabled, because then the attack can be avoided using session stealing prevention.

Stop Offer: One way to completely avoid the Stop Offer Attack can be to reject all stop offer messages. In a vehicular context, most of the ECUs would be turned off at the same time, when a car is turned off, and therefore having them turn off nicely is not needed, even though the specification states that ECUs shall always, on shutdown, send stop offer messages.

VII. Conclusion

In this paper SOME/IP protocol vulnerabilities are exposed and ways of detecting them are proposed, by implementing an HIDS for the ECUs in in-vehicle networks. The presented attacks are shown to be detectable by utilizing different features. Approaches to prevent these attacks are discussed in the paper. Most of these are proven on hardware. An overview of the research is shown in Table I.

Table I
Shows the four attacks, if theory is provided, whether or not they have been proved on hardware and if detection and prevention are possible.

Attacks	Theory	Detection possible	Prevention possible	Proved on hardware
RPC Flooding	✓	✓	✓	✓
False Notify	✓	✓	✓	✓
Session Stealing	✓	✓	✓	×
Stop Offer Attack	✓	✓	✓	✓

The designed attacks prove that additional security for SOME/IP is required and that the detection methods proposed are the first steps towards developing an IDS for in-vehicle ECUs. The prevention suggestions is believed to contribute in closing the security gaps of SOME/IP.

References

[1] R. Lange & R. S. de Oliveira, "Hybrid FlexRay/CAN Automotive Networks", in *Embedded Computing Systems: Applications, Optimization, and Advanced Design*, 2013. pp. 323 - 324.

[2] "Controller Area Network (CAN) Overview". [Online]. Available: ni.com/da-dk/innovations/white-papers/06/controller-area-network--can--overview.html. [Accessed Dec. 16, 2019].

[3] C. Miller, C. Valasek "Remote Exploitation of an Unaltered Passenger Vehicle", academia.edu, 2015.

[4] L. Völker "Scalable service-Oriented MiddlewarE over IP (SOME/IP)". [Online]. Available: some-ip.com, 2019. [Accessed Dec. 16, 2019].

[5] H. Lee, S.H. Jeong & H.K. Kim, "OTIDS: A Novel Intrusion Detection System for In-vehicle Network by Using Remote Frame", in *15th Annual Conference on Privacy, Security and Trust (PST)*, 2017. pp. 57 - 66.

[6] F. Sagstetter, M. Lukasiewycz, S. Steinhorst, M. Wolf, A. Bouard, W. R. Harris,S. Jha, T. Peyrin, A. Poschmann & S. Chakraborty. "Security Challenges in Automotive Hardware/Software Architecture Design", in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2013. pp. 458 - 463.

[7] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham & S. Savage, in "Experimental Security Analysis of a Modern Automobile", *IEEE Symposium on Security and Privacy*, 2010. pp. 447 - 462.

[8] E. Seo, H.M. Song & H.K. Kim, "GIDS: GAN based Intrusion Detection System for In-Vehicle Network", in *16th Annual Conference on Privacy, Security and Trust (PST)*, 2018. pp. 1 - 6.

[9] N. Herold, S.A. Posselt, O. Hanka & G. Carle, "Anomaly Detection for SOME/IP using Complex Event Processing", in *IEEE/IFIP Network Operations and Management Symposium*, 2016. pp. 1221 - 1226.

[10] AUTOSAR "SOME/IP Protocol Specification". [Online]. Available: autosar.org. [Accessed Dec. 16, 2019].

[11] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner & T. Kohno, "Comprehensive Experimental Analyses of Automotive Attack Surfaces" in *SEC'11 Proceedings of the 20th USENIX conference on Security*, 2011.

[12] "An implementation of Scalable service-Oriented MiddlewarE over IP". [Online]. Available: https://github.com/GENIVI/vsomeip. [Accessed Dec. 16, 2019].

[13] "HCRL-Hacking and Countermeasure Research Lab". [Online]. Available: ocslab.hksecurity.net/Datasets. [Accessed Dec. 16, 2019].