Here comes SAID: A SOME/IP Attention-based mechanism for Intrusion Detection

Natasha Alkhatib, Maria Mushtaq, Hadi Ghauch, Jean-Luc Danger
Télécom Paris, IP Paris, Palaiseau, France
{natasha.alkhatib, maria.mushtaq, hadi.ghauch, jean-luc.danger}@telecom-paris.fr

***Abstract*—The increasing connectivity among vehicles along with their rising complexity increases their attack surface and challenges their security. In this paper, we consider the problem of intrusion detection for SOME/IP protocol and present "SAID" a novel technique for the detection of anomalies from a large sequence of exchanged SOME/IP network packets. The proposed detector leverages a self-attention-based neural network to model the contextual dependencies between SOME/IP packets. For this purpose, we evaluate our proposed approach, by generating a simulated and manually annotated SOME/IP dataset, with several categories of attacks. The results of the extensive experiments indicate that our technique detects (with high accuracy) the majority of SOME/IP's protocol violations, e.g., with an area-under-the-curve ≈ 0.8, and inference time ≈ 0.3 ms. A comparative study, including various state-of-the-art benchmark algorithms, shows that SAID shows better performance in detecting intrusions and enables parallelization. Our source code and data are available at: https://github.com/Alkhatibnatasha/supervised_detection_some_ip/**

***Index Terms*—Security, SOME/IP, Automotive Ethernet, In-Vehicle Network, Deep Learning, Attention**

I. Introduction

As the bandwidth requirements in automotive keep increasing, a migration to a cost-efficient high-speed switched network in automotive that lifts the bandwidth and higher layer protocols restrictions is necessary. Established by the OPEN Alliance, Automotive Ethernet - or more correctly "Ethernet-based communications" - was since then adopted to enable the development of new automotive protocols for specific layers within the ISO/OSI models while allowing the reuse of protocols for the remaining others.

As the increasing automotive software complexity necessitates switching to service-based in-vehicle communication. The Scalable service-Oriented MiddlewarE over IP (SOME/IP) is an automotive middleware protocol which operates at the higher layers of the ISO/OSI layer model. Its key advantages lie in the complexity reduction of the Ethernet-based in-vehicle network by providing serialization, remote procedure call (RPC), and service discovery, among other features. Researchers [1] have recently found relevant vulnerabilities in SOME/IP protocol which in turn can lead to exploitation and car hacking,i.e., man-in-the-middle (MITM) attack in which an attacker can intercept, manipulate, and interrupt the communication between different ECUs.

To mitigate these risks, the international regulation UN R-155 mandates cybersecurity monitoring to assess vehicle data and records for vulnerabilities and cyberattacks. Consequently, a wide array of security controls, (e.g. authentication, encryption, firewalls, and secure updates) have been proposed to guarantee the security of SOME/IP protocol, and mitigate, and prohibit targeted attacks from growing substantially [10]. In this paper, we consider developing a SOME/IP-specific Intrusion Detection System (IDS) that is able to monitor suspicious SOME/IP network traffic behavior and detect potential intrusions.

While the proposed state-of-the-art solutions leverage specification-based techniques [9] [6] [3] which in turn require substantial human effort for crafting suitable specification rules or features, our research overcomes this key challenge by leveraging suitable deep learning techniques that extract high-level, abstract features, from raw sequences of exchanged SOME/IP packets and which in turn facilitate the real-time intrusion detection task. More importantly, given the problems of the large volume of SOME/IP network traffic due to the emergent dynamical structure of the in-vehicle network, the deep learning approaches are regarded as a good candidate for intrusion detection owing to their ability to process high-dimensional data.

In [8], we employed a Recurrent Neural Network (RNN) for offline classification of attacks on SOME/IP protocol. In this work, however, we adapt Vaswani's transformer model [11], which allows for more parallelization, making the approach more suitable for the real-time intrusion detection. Although we use the transformer's model "self-attention" mechanism to enable modeling of the interdependencies between different elements of the input network sequence regardless of their distance, our proposed neural network architecture is considerably simpler as it is not built on the typical Encoder-Decoder architecture format for language translation.

Hence, the contribution of this work is three-fold. For one, we introduce "SAID", a neural network-based approach that leverages attention-based mechanisms for identifying anomalies in long sequences of exchanged network packets. Moreover, we simulate and manually annotate a SOME/IP dataset, with different attack ratios, to evaluate our proposed detector. Finally, we compare "SAID" with other deep learning algorithms to validate its outstanding performance.

The remainder of this paper is organized as follows. Section II discusses main publications that are related to SOME/IP security countermeasures. In Section III, we present an overview of the SOME/IP protocol. In Section IV, we present the leveraged threat model and the different considered attacks.

The generated and labeled dataset is presented in Section V. In Section VI, we present our proposed attention-based detector "SAID". In Section VII, we explain the considered evaluation metrics used for performance evaluation. We discuss our experimental results in Section VIII, followed by a conclusion that summarizes our study.

II. Related Work

Koyama et al. [3] proposed an IDS that achieves high accuracy by combining two whitelist-based anomaly detection algorithms. Alkhatib et al. [8] presented a deep learning-based sequential model for offline intrusion detection on SOME/IP application layer protocol. Tobias et al. [9] propose an architecture for a SOME/IP intrusion detection system, discuss its security properties and report preliminary experimental results. Herold et al. [6] presented an anomaly detection system using the Esper complex event processing engine, thus applying a domain-specific rule set to a stream of SOME/IP packets. Tan et al. [13] present a new technique based on the neural attention mechanism for real-time attack detection since it uses time slot-based features. Nam et al. [14] Alkhatib et al. [15] proposed attention-based IDS to detect attacks on the traditional in-vehicle network known as CAN.

III. Overview of SOME/IP protocol

Over the course of the past few decades, there has been a shift away from electronic control units (ECUs) that are solely devoted to performing a single function and toward high-performance domain controllers (DC) and vehicle computers (VC), both of which integrate a wide variety of software functions. To facilitate the coordination and interchange of data amongst them, the Scalable-service Oriented Middleware Over IP (SOME/IP) [5] protocol was thereby adopted as an automotive middleware that runs on top of the ISO/OSI model. Several automotive use cases have been developed as a result of its key properties, ranging from its support for a service-based communication approach, its compatibility with AUTOSAR, its scalability for use on platforms of varying sizes, and to its adaptability to common automotive operating systems such as OSEK and QNX.

A. SOME/IP Packet Structure

The individual elements of SOME/IP are considered for detecting intrusions on SOME/IP protocol. The header, 16 bytes long, is made up of the following fields:

- **Message ID** The first 16 bits of the Message ID identify the service used. The service provides the overall structure for middleware communication. Each service needs to have a unique **Service ID**, which the system integrator assigns. A service can consist of a set of methods, events, and fields, which are identified in the *Method ID*. The 16 bits for the Method ID represent the other half of the Message ID.
- **Length** The length field uses 32 bits to specify the number of bytes starting from the **Client ID** until the end of the SOME/IP message.
- **Request ID** The Request ID allows a client to differentiate between multiple calls of the same method. The first 16 bits of the Request ID represent the **Client ID** which identifies the application triggering a specific request. The second 16 bits of the Request ID represent the **Session ID** which is an identifier incremented whenever a new message is sent.
- **Protocol Version** An 8-bit field that identifies the SOME/IP version.
- **Interface Version** These 8 bits identify the major version of the service interface.
- **Message Type** This field differentiates between the different possible types of messages such as REQUEST, REQUEST-NO-RETURN, NOTIFICATION, RESPONSE, and ERROR.
- **Return Code** The 8 bits of the Return Code signal whether a request was successfully processed or an occurred error.

B. SOME/IP Remote Procedure Calls

SOME/IP defines a service by its Service Interface, i.e., the activities of the client and server, based on the defined communication principles [**?**]. In fact, a server is an ECU which offers a service instance that is used by an ECU client. The defined communication principles, considered in this work, cover methods with the response (request/response), messages without response (fire and forget), and events,i.e., a message from the server to the client when something happens.

- **Request/Response** defines a two-way exchange in which a client sends a Request message to the server in order to invoke a method, and the server sends back a Response message with the outcome of the method call.
- **Fire and Forget** entails a one-way communication by not requiring the callee to send back a response. The client only requests a remote node to perform an action and disregards the result.
- **Events** describes a communication in which the server sends messages with specific information to the client, either periodically or when there is a change (event), without expecting any message back from the client.

IV. Threat model

While SOME/IP has useful features, it is lacking essential security properties like authentication and encryption. As a result, we've used a threat model previously implemented by [6] in which an adversary behaves as a man-in-the-middle attacker and does not follow the SOME/IP protocol definition. Consequently, we explored a SOME/IP-based network without service discovery characteristics in our study. Clients are aware of all information about the servers, including their MAC and IP addresses. Each client is only authorized to use a restricted number of services and approaches. Some servers transmit notifications to certain clients on a regular basis. The precise intervals utilized for these services, as well as the services and methods associated with them, are known ahead of time. As [6] specified, an attacker can do the following:

- Compromises a known device within the system. Thus, the attacker has a legitimate MAC address, IP address, and service ID.
- Sniffs SOME/IP network traffic exchanged in the in-vehicle network
- Transmits packets to all network participants,i.e., clients and servers, and thereby impersonates other SOME/IP devices and services.

A. Attacks

We have considered four intrusion types in this work, detailed below :

- **Requests without Response**: Requests have to be answered with either a response or an error message. If a request was never answered, it means that an attacker has relayed the communication between the client and the server who believe that they are directly communicating with each other.
- **Response without Request**: A response should only be delivered in response to an open, previous request. As a result, a normal request with message type 0x00 should be answered by a single response with message type 0x80. Two replies to a single request break the protocol and may indicate the existence of an attacker attempting to impersonate the server and inject extra packets.
- **Error on Error**: Based on AUTOSAR standard specification, an error message should not be answered with another error message. Hence an incoming error that doesn't have a corresponding request (or another packet) with the same settings indicates the presence of a network intrusion.
- **Error on Event**: Notifications should not be answered with an error message. Thus, a notification replied to with an error depicts a network intrusion between the client and the server.

V. Dataset Generation

Given a sequence of SOME/IP packets, we aim to detect whether this sequence is normal or anomalous, i.e., a SOME/IP sequence is anomalous if it contains at least one abnormal (i.e., injected/out of order/replayed) packet. Hence, we have generated and labeled a dataset that contains benign and malicious SOME/IP packets using the SOME/IP Generator developed by [6], implemented in Python 3 and available in Github [7]. The generator models the behavior of different clients and servers assumed to behave according to the AUTOSAR standard specification, as well as an attacker carrying out a variety of protocol violation attacks described in Section IV.

A. Dataset configuration

To build our dataset, we have considered an in-vehicle network composed of 8 servers, 8 clients and one attacker. The amount of services to be exchanged is limited to 3. Moreover, each client of a subscribed service and method can generate 500 packets for a SOME/IP network session. The response time of the attacker is ranged between 1 and 3 seconds.

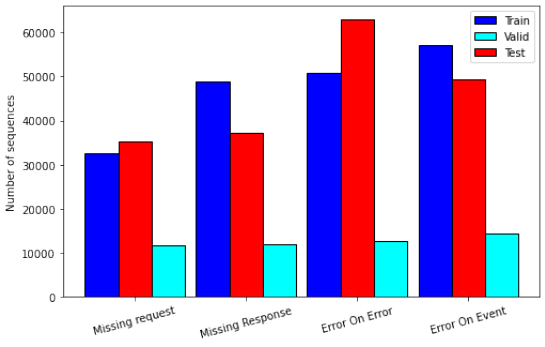

Fig. 1. Dataset

Furthermore, through our work, we have mainly considered four protocol violation attacks on SOME/IP protocol. It's noteworthy to mention that the simplicity of the attacks used in the dataset does not necessarily limit the effectiveness of deep learning for intrusion detection. By training on a diverse set of attacks, deep learning models can learn to recognize common attack patterns and generalize to new and more complex attacks that may leverage the same underlying techniques. For training and testing our deep learning-based IDS, we have generated several PCAP files corresponding to different attack types. During the training, we balance the data to eliminate bias, in the learning. In fact, we consider the same number of normal and abnormal SOME/IP sequences, when training the different approaches. Additionally, the abnormal set contains different attack types. However, during testing, we evaluate the performance of SAID for each attack separately. The dataset statistics are represented in Figure 1.

B. Dataset representation

Since the IDS will be monitoring the SOME/IP network traffic each 100ms, we collect sequences composed of 128 SOME/IP packets using the **Feature-based Sliding Window (FSW)**, where T represents the window size or the total number of packets per window and the slide size is 1. Hence, each sequence of ordered packets is defined as $S = \{\mathbf{p}_1, ..., \mathbf{p}_t, ..., \mathbf{p}_T\}$, where $\mathbf{p}_t$ indicates a transmitted SOME/IP packet at time t and $\mathbf{y}_t \in \{0, 1\}$ its corresponding label, with 1 indicating a malicious packet and 0 otherwise. Each packet $\mathbf{p}_t$ in the SOME/IP network traffic is represented by 58 features, each of which has an integer value between 0 and 255. Moreover, we label each SOME/IP sequence using the following criteria:

$$y = \begin{cases} 0 \text{ (normal)} & \text{if } \mathbf{y}_t = 0, \forall t \in \{1, .., T\} \\ 1 \text{ (abnormal)} & \text{otherwise} \end{cases}$$

$y \in \{0, 1\}$ is the SOME/IP sequence's label, which is labeled as anomalous if there is at least one anomalous packet.

VI. Proposed Framework: SAID

We now thoroughly explain the architecture of the SOME/IP Attention-based IDS (SAID) which is inspired by the transformer's ability to handle ordered sequences of data [11]. For

an overview please refer to Figure 2. In contrast to Vaswani's model [11] which leverages an encoder-decoder structure for machine translation, our proposed detector employs only the encoder network followed by a sigmoid activation layer for binary classification of SOME/IP packets' sequences.

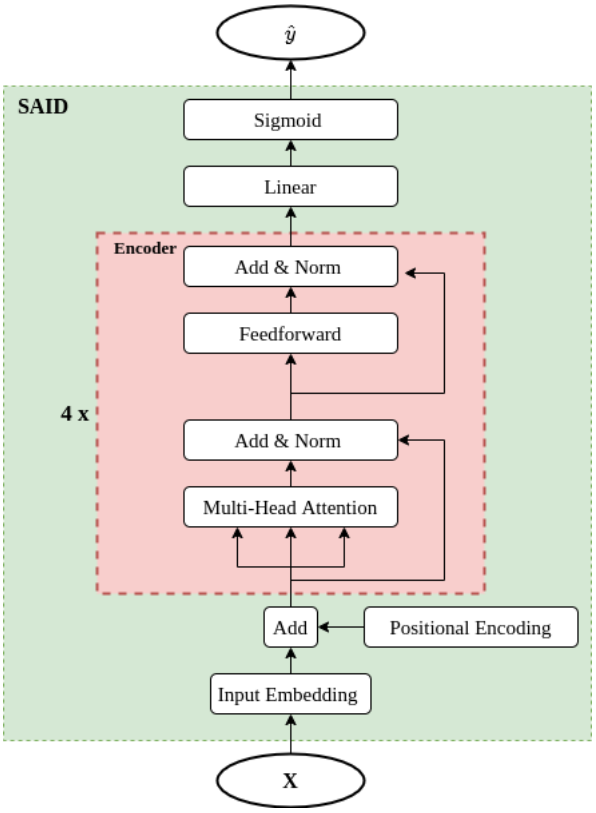

Fig. 2. The overall structure of our proposed SOME/IP intrusion detector "SAID". We first project every SOME/IP packet into an embedding space and then inject positional encodings and apply dropout. Next, the embeddings are fed to L = 4 stacked attention layers. Finally, we detect intrusions by feeding the resulting output into the sigmoid activation function.

A. Input Embedding and Positional Encoding

The SOME/IP packets' sequence $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_T\} \in \mathbb{R}^{T\times d}$, is fed firstly into the SAID's "Input Embedding" module. $\mathbf{x}_t$ denotes a d-dimensional SOME/IP packet transmitted in the Ethernet-based in-vehicle network at time t. The Input embedding projects $\mathbf{X}$ into a h-dimensional space using a linear function: $\mathbf{X}_I = \mathbf{X}\cdot\mathbf{W}_0 + \mathbf{B}_0$.The weights are represented as $\mathbf{W}_0 \in \mathbb{R}^{d\times h}$, $\mathbf{B}_0 \in \mathbb{R}^{T\times h}$ is the bias, h is the hidden size. The parameters of the embedding, $\mathbf{W}_0, \mathbf{B}_0$, are randomly initialized, and updated with the other parameters of the model during the training phase.

We additionally inject a notion of ordering by adding sinusoidal positional encoding to the SOME/IP packets' embeddings following [11]. The tth SOME/IP packet's position embedding can be represented by the following equations:

$$\mathbb{PE}_{t,2i} = sin(t/10000^{2i/d}),\ \forall i \in \{1,2,...,d/2\} \quad (1)$$

$$\mathbb{PE}_{t,2i+1} = cos(t/10000^{2i/d}),\ \forall i \in \{1,2,...,d/2\} \quad (2)$$

where i is an index, d is in the input dimension.

The resulting embedding $\mathbf{X}_E = \mathbf{X}_I + \mathbf{X}_P$ is thus fed into the next module with $\mathbf{X}_P \in \mathbb{R}^{T\times h}$ is the positional embeddings of the SOME/IP packets' sequence.

B. Encoder block

The output $\mathbf{X}_E \in \mathbb{R}^{T\times h}$ obtained from the previous module is passed to a stack of L attention blocks where we apply the "self-attention" attention to update the embeddings. Similarly to [11], [13], we use the scaled dot-product attention, requiring a matrix of keys $\mathbf{K} \in \mathbb{R}^{T\times h}$, a matrix of values $\mathbf{Q} \in \mathbb{R}^{T\times h}$, and a matrix of "queries" $\mathbf{Q} \in \mathbb{R}^{T\times h}$. The attention operation also involves a weight matrix, $\mathbf{V} \in \mathbb{R}^{T\times h}$.

$$Attn(\mathbf{Q},\mathbf{K},\mathbf{V}) = \tau(\frac{\mathbf{Q}\cdot\mathbf{K}^T}{\sqrt{h}})\cdot\mathbf{V} = \mathbf{AV} \quad (3)$$

where $\tau(-)$ is a matrix function (applied element-wise) chosen as the softmax function. The resulting output context contains information about the intrinsic dependencies between latent representations of a sequence of SOME/IP network packets $\mathbf{X}$ within a data sample. It will subsequently be used to assist in the predictions of $\hat{\mathbf{y}}$. Additionally, a residual connection is applied around the attention mechanism, followed by a layer normalization. The output of the normalization , $\mathbf{X}_A \in \mathbb{R}^{T\times h}$ is given as,

$$\mathbf{X}_A = N_1(\mathbf{Q} + \mathbf{AV}) \quad (4)$$

where $N_1(-)$ is a matrix function (applied element-wise) modeling the layer normalization. After the layer normalization, we pass the output to a fully connected Feed Forward (FF) network and a subsequent layer normalization with the residual connection. Following [11], [13], the feed-forward block consists of two linear projections separated by a ReLU activation:

$$FFN(\mathbf{W}_1,\mathbf{W}_2,\mathbf{B}_1,\mathbf{B}_2) = \sigma(\mathbf{X}_A\cdot\mathbf{W}_1+\mathbf{B}_1)\cdot\mathbf{W}_2+\mathbf{B}_2 \quad (5)$$

$$\mathbf{X}_{FF} = N_2(\mathbf{X}_A + FFN(\mathbf{W}_1,\mathbf{W}_2,\mathbf{B}_1,\mathbf{B}_2)) \quad (6)$$

The output $FFN(\mathbf{W}_1,\mathbf{W}_2,\mathbf{B}_1,\mathbf{B}_2) \in \mathbb{R}^{T\times h}$, depends on the shared parameters $\mathbf{W}_1 \in \mathbb{R}^{h\times p}$, $\mathbf{W}_2 \in \mathbb{R}^{p\times h}$, $\mathbf{B}_1 \in \mathbb{R}^{T\times p}$, and $\mathbf{B}_2 \in \mathbb{R}^{T\times h}$, to be optimized in training. $\sigma(-)$ is a matrix ReLU function applied element-wise. Moreover, $\mathbf{X}_{FF} \in \mathbb{R}^{T\times h}$ denotes the FF output that undergoes layer normalization, modeled by the matrix function $N_2(-)$.

C. Output layer

The output $\mathbf{X}_{FF}$ given by the Encoder module passes through a final fully connected layer with the sigmoid activation function which outputs the binary class $\hat{\mathbf{y}} \in \{0,1\}$ of the SOME/IP sequence, with the value 1 representing the possibility of intrusion in a sequence of SOME/IP packets and 0 otherwise.

VII. Evaluation metrics

For measuring the performance of our proposed detector, we adopt the False Positive Rate (FPR), the True Positive Rate (TPR), and the Area Under Curve (AUC) measures, explained below.

False Positive Rate (FPR), also called False Alarm Rate, is the ratio of incorrectly classified normal instances as an attack of all observations in the actual class.

$$FPR = \frac{FP}{FP + TN} \tag{7}$$

True Positive Rate (TPR) or Recall is the ratio of correctly predicted abnormal observations of all observations in the actual class.

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

Where: TP= True Positive; FP=False Positive; TN= True Negative; FN=False Negative.

We additionally use the AUC metric which stands for "Area under the receiver operating characteristic Curve". It measures the entire two-dimensional area underneath the entire ROC curve. A random intrusion detector will have an area under curve of 0.5. Meanwhile, the perfect detector will have a large predictive power if the FPR value is near to 0, the TPR score is approaching 1 and the AUC score is also near to 1. The higher the AUC, the generally better the IDS is.

VIII. Experimental Results

A. Implementation

In this experiment, we evaluate the attack detection capabilities of SAID on the generated dataset and perform a comparative analysis with other deep learning techniques. The models have been implemented using Python programming language and Pytorch framework on the Tesla V100S-PCIE-32GB machine. We set the window size as 128 throughout our experiment, which considers the temporal and contextual information between SOME/IP packets, memory, and computation efficiency. The deep learning models will fail if the window size is smaller for intrusion detection. Note that, although a larger window size makes the model learn complex contextual interdependencies, it results in larger memory costs for the embedded systems.

After hyperparameter tuning, we consider the depicted parameters in Table I for SAID. We consider known neural network models to be compared with our proposed detector:

- The **RNN (LSTM)** model is compoe of 2 layers, with the first layer's hidden size set at 16 and the second layer's hidden size set at 8.
- For the **CNN** model, we convert each sequence into an image. The input is thereby fed to the CNN model composed of 2 convolutional layers, having each a number of channels of 8 and 4 respectively. The kernel size is set to 3 and the stride size is set to 2.
- For the **CNN-LSTM** model, the same image is fed to the CNN model composed of three convolutional layers with 3, 6, and 3 as their number of channels. Their kernel size is set to 3 and their stride size is 1. After that, the output of the CNN is fed to an LSTM neural network composed of three layers with 16, 8, and 4 as their number of features in the hidden states.

TABLE I
SAID model configuration

Parameter	Value
L	4
$d_{model}(h)$	8
d_{ff} (p)	32
$\#$ *heads*	1
P_{drop}	0.1
Optimizer	Adam
Adam β_1	0.9
Adam β_2	0.999
Learning rate	0.001
Batch size	32
$\#$ Epochs	200
Patience	10

B. Results

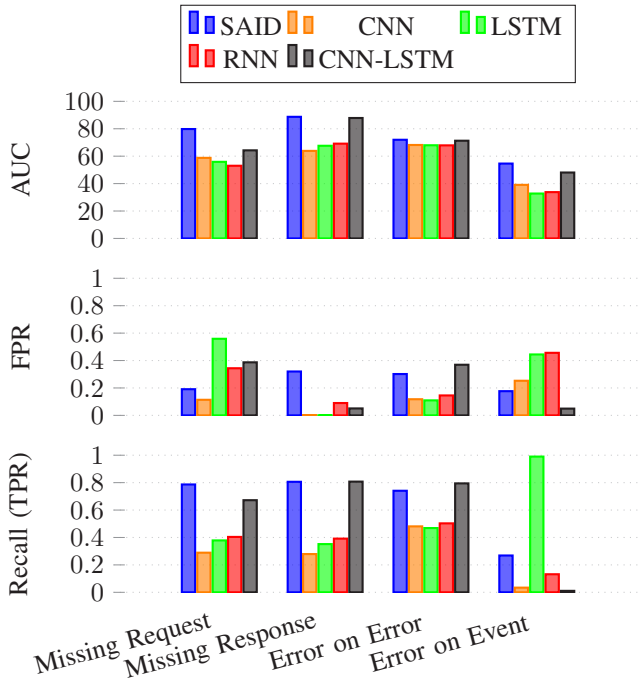

Fig. 3. High attack ratio dataset

Figure 3 provide the TPR, FPR, and AUC scores for SAID and other deep learning models for the generated dataset. As shown, SAID (Missing Request AUC $\approx$ 0.80, Missing Response AUC $\approx$ 0.89, Error on Error AUC $\approx$ 0.72) outperforms other deep learning algorithms for all attacks in terms of AUC scores (Missing Request $\overline{AUC} \approx 0.62$, Missing Response $\overline{AUC} \approx 0.75$, Error on Error $\overline{AUC} \approx 0.69$). Our proposed detector goes beyond the point-wise representation learned

TABLE II
MODELS COMPLEXITY

Model	Time (ms)	Size (MB)	# Parameters
SAID	0.30 ± 0.30	0.12	1,329
CNN	0.07 ± 0.05	0.16	1,985
LSTM	4.583 ± 0.60	0.08	11,401
RNN	4.786 ± 0.13	0.06	2,857
CNN-LSTM	0.37 ± 0.34	15.53	3,750,896

by the conventional deep learning techniques and models the more informative interdependencies between SOME/IP packets. However, although the "Error on Event" attack ratio has increased, all models also perform poorly when detecting this protocol violation ($\overline{AUC} \approx 0.42$). A general comparison of the results of the results implies that SAID not only improves the AUC and recall rate of SOME/IP intrusion detection but also decreases the false positive rate for the majority of intrusions.

Eventually, as our solution will be deployed on Electronic Control Units (ECUs) which have tight resource constraints including computing, memory, and important power budget, we assessed their computational performance. The results are listed in Table II. As depicted, while the LSTM and RNN-based models take longer time to detect attacks on SOME/IP protocol, the CNN, CNN-LSTM, and SAID detect intrusions between 0.07 and 0.4 ms. Using SAID allows much faster detection than recurrent methods by parallelizing inference on GPUs. Despite CNN's low inference time, it poorly detects attacks when compared to both CNN-LSTM and SAID. The numerical results show the advantage of the proposed framework, verify its low detection error against the majority of intrusions, and outperforms the other benchmarks.

IX. CONCLUSION

As the in-vehicle network is increasingly scaling due to the emerging SOME/IP protocol, the conventional intrusion detection techniques can no longer be adopted as they require substantial human effort for crafting suitable specification rules or features, respectively. Additionally, due to scaling, the high-dimensional nature of transmitted interdependent SOME/IP network packets is particularly challenging for intrusion detection, as many corresponding dimensions may be noisy and irrelevant for anomaly detection. Our research aims to overcome these challenges by the deployment of effective security solutions that adapt to the recent dynamic in-vehicle environment. More concretely, we present a novel deep learning-based IDS for SOME/IP able to learn features with different levels of abstraction at different processing layers without human intervention. Using the self-attention mechanism, our approach can model the contextual dependencies between SOME/IP packets and which are crucial for the detection of contextual intrusions. The tests performed on SOME/IP datasets show that our proposed model is computationally efficient and achieves an overall superior performance than the prevailing sequential models. For future work, we aim to investigate the effectiveness of the attention mechanism in improving our model's interpretability and in detecting stealthier attacks.

REFERENCES

[1] Du, Jinze, Rui Tang, and Tao Feng. "Security Analysis and Improvement of Vehicle Ethernet SOME/IP Protocol." Sensors 22.18 (2022): 6792.

[2] Supervised Intrusion Detection for SOME/IP protocol, https://github.com/Alkhatibnatasha/supervised_detection_some_ip

[3] Koyama, Takuma, et al. "SOME/IP Intrusion Detection System Using Real-Time and Retroactive Anomaly Detection." 2022 IEEE 95th Vehicular Technology Conference:(VTC2022-Spring). IEEE, 2022.

[4] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

[5] Scalable service-Oriented MiddlewarE over IP (SOME/IP) documentation, https://some-ip.com/

[6] Herold, Nadine, et al. "Anomaly detection for SOME/IP using complex event processing." NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium. IEEE, 2016.

[7] Herold, Nadine and Posselt, Stephan-A and Hanka, Oliver and Carle, Georg, https://github.com/Egomania/SOME-IP_Generator

[8] Alkhatib, Natasha, Hadi Ghauch, and Jean-Luc Danger. "SOME/IP intrusion detection using deep learning-based sequential models in automotive ethernet networks." 2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON). IEEE, 2021.

[9] Gehrmann, Tobias, and Paul Duplys. "Intrusion detection for some/ip: Challenges and opportunities." 2020 23rd Euromicro Conference on Digital System Design (DSD). IEEE, 2020.

[10] Rumez, Marcel, et al. "An overview of automotive service-oriented architectures and implications for security countermeasures." IEEE access 8 (2020): 221852-221870.

[11] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

[12] Ma, Bin, et al. "An authentication and secure communication scheme for in-vehicle networks based on SOME/IP." Sensors 22.2 (2022): 647.

[13] Tan, Mengxuan, et al. "A neural attention model for real-time network intrusion detection." 2019 IEEE 44th conference on local computer networks (LCN). IEEE, 2019.

[14] Nam, Minki, Seungyoung Park, and Duk Soo Kim. "Intrusion detection method using bi-directional GPT for in-vehicle controller area networks." IEEE Access 9 (2021): 124931-124944.

[15] Alkhatib, Natasha, et al. "CAN-BERT do it? Controller Area Network Intrusion Detection System based on BERT Language Model." 2022 IEEE/ACS 19th International Conference on Computer Systems and Applications (AICCSA). IEEE, 2022.