Federated Learning-based Inter-slice Attack Detection for 5G-V2X Sliced Networks

Abdelwahab Boualouache*, and Thomas Engel
FSTM, University of Luxembourg, Luxembourg
Email: {abdelwahab.boualouache thomas.engel}@uni.lu

***Abstract*—As a leading enabler of 5G, Network Slicing (NS) aims at creating multiple virtual networks on the same shared and programmable physical infrastructure. Integrated with 5G-Vehicle-to-Everything (V2X) technology, NS enables various isolated 5G-V2X networks with different requirements such as autonomous driving and platooning. This combination has generated new attack surfaces against Connected and Automated Vehicles (CAVs), leading them to road hazards and putting users' lives in danger. More specifically, such attacks can either intra-slice targeting the internal service within each V2X Network Slice (V2X-NS) or inter-slice targeting the cross V2X-NSs and breaking the isolation between them. However, detecting such attacks is challenging, especially inter-slice V2X attacks where security mechanisms should maintain privacy preservation and NS isolation. To this end, this paper addresses detecting inter-slice V2X attacks. To do so, we leverage both Virtual Security as a Service (VSaS) concept and Deep learning (DL) together with Federated learning (FL) to deploy a set of DL-empowered security Virtual Network Functions (sVNFs) over V2X-NSs. Our privacy preservation scheme is hierarchical and supports FL-based collaborative learning. It also integrates a game-theory-based mechanism to motivate FL clients (CAVs) to provide high-quality DL local models. We train, validate, and test our scheme using a publicly available dataset. The results show our scheme's accuracy and efficiency in detecting inter-slice V2X attacks.**

***Index Terms*—5G-V2X; Network Slicing; Security; Machine learning; Misbehaving Detection Systems**

I. Introduction

The emergence of the Fifth-generation (5G) mobile systems has brought many advances for Cooperative Intelligent Transportation Systems (C-ITS). As a part of C-ITS, CAVs were designed to make our roads safer while providing traffic efficiency and user convenience. The 3GPP Release 16 [1] enables CAVs with ultra-low and ultra-reliable 5G-V2X communications in a decentralized way via the PC5 interface. In addition, CAVs are supported by a set of 5G enabling technologies, such as Software Defined Networking (SDN), Network Function Virtualization (NFV), Multi-access Edge Computing (MEC), and NS. The latter creates independent virtual networks on top of the same physical infrastructure. For CAVs, NS can enable several 5G-V2X networks with different requirements to co-exist and operate together while enjoying isolation [2]. NS then increases the exploitation degree of 5G physical infrastructure while boosting the performance of CAV applications and services. However, these benefits are underlying new cybersecurity challenges to CAVs. Indeed, V2X-NSs are vulnerable to several attack surfaces exposed by NS-enabling technologies (SDN and NFV). In addition, 5G-V2X nodes can exploit NS breaches to generate attacks against NS functioning. For example, malicious CAVs can collaboratively launch denial of V2X-NS(s) services to make them unavailable. These attacks impact not only the targeted V2X-NS(s) but also other co-existing V2X-NSs since they all share the same physical resources. Therefore, detecting and defeating V2X-NS attacks is crucial since they can put the lives of both drivers and passengers in danger by leading CAVs to high-risk situations. We can classify V2X-NS attacks into two categories: i) intra-slice attacks in which the attacker(s) and the target(s) belong to the same V2X Network Slice (V2X-NS), and (ii) inter-slice attacks in which the attacker(s) and/or the target(s) belong to different V2X-NSs. On the one hand, detecting inter-slice V2X attacks is more challenging since attackers can be distributed over NSs. Thus, security mechanisms should detect attacks while maintaining V2X-NSs isolation and preserving private information. On the other hand, V2X-NS attackers are usually part of the network (internal), which makes them resistant to cryptographic solutions. Instead, intrusion detection schemes are more efficient to detect internal V2X-NS attacks, especially when taking advantage of the latest advances in Machine Learning (ML).

To this end, this paper addresses detecting inter-slice V2X attacks while preserving both isolation and privacy within V2X-NSs. Our scheme leverages Virtual Security as a Service (VSaS) concept [3] and Deep learning (DL) together with Federated learning (FL) design to deploy a set of DL-empowered security Virtual Network Functions (sVNFs) over V2X-NSs. sVNFs aim not only at detecting attacks and reporting them but also at participating in the training process. Our privacy-preserving scheme is hierarchical and supports FL-based collaborative learning while continuously updating attack detection DL models. It also integrates a game-theory mechanism to incentivize FL clients (CAVs) to provide high-quality local DL models. The evaluation results show our global model's accuracy and the incentive mechanism's efficiency.

The remainder of this paper is organized as follows. Section II describes related works. The design of our scheme and targeted inter-slice V2X attacks are presented in Section III. The game theory-based model to motivate FL clients (CAVs) is described in IV. Section V depicts the performance evaluation results. Finally, Section VI concludes the paper.

II. Related Work

Several ML collaborative learning schemes have been proposed to detect V2X attacks. The authors of [4] trained an attack detection scheme based on a distributed ML approach. In this scheme, vehicles collaboratively build the global model without exchanging local datasets but instead share updates of their loss functions. The authors of [5] proposed a collaborative learning attack scheme consisting of four phases. Vehicles first build their local models based on their collected data. Then, they share models according to the requests received from the neighbors. After that, they evaluate models to detect malicious models. Finally, a collaborative model is constructed based on valid models checked in the third phase. The authors of [6] proposed an attack detection scheme based on SDN. It uses DL with generative adversarial networks to enable multiple distributed SDN controllers to jointly train the ML model for the entire network. The authors of [7] leveraged FL to propose a privacy-preserving attack detection scheme for position falsification attacks. In this scheme, vehicles serve as FL clients, who train their local models based on periodic messages from neighboring vehicles and send updates to the FL server. The authors of [8] proposed an FL-based scheme for detecting V2X position-tracking attacks. The scheme enables FL at the edge for collaborative learning while preserving the privacy of vehicles. The authors of [9] proposed a collaborative learning attack detection scheme based on roadside units. These latter select FL clients from vehicles under their coverage for training global DL models. The authors [10] proposed an FL-based attack detection scheme that offloads the learning process from servers to distributed vehicular edge nodes. The scheme trains a transformer network to learn spatial-temporal representations of vehicular traffic flows for better classification of attacks. However, previous schemes aim at detecting traditional V2X attacks. Thus, collaborative learning for detecting inter-slicing 5G V2X attacks is not yet explored. Moreover, The authors of [11] and [12] proposed a DL-based attack detection scheme for distributed denial of service (DDoS) attacks in 5G networks. However, these schemes are centralized and are only proposed to detect DDoS attacks on the core network and consider neither the 5G New Radio (NR) attacks, including V2X and the MEC, nor V2X-NS attacks. Unlike the schemes above, we exploit FL, DL, and VSaS to propose collaborative learning enabling scheme for detecting inter-slicing 5G V2X attacks. VSaS is a flexible and elastic approach, supporting the 5G security concept, which consists in integrating sVNFs in the slice life cycle [12].

III. Inter-slice V2X Attack Detection Scheme

This section describes our inter-slice V2X attack detection scheme. Figure 1 shows 5G-V2X NS architecture consisting of (i) the NR including CAVs and gNodeBs, (ii) the MEC nodes, and (iii) the 5G Core. 5G-V2X NSs are created and managed by the Network Slice Manager (NSM). During the creation of the V2X-NSs, the NSM allocates the necessary storage and processing resource to satisfy the requirements defined in the Service-Level Agreement (SLA). The NSM also implements all VNF service chaining to meet V2X-NSs needs while ensuring the isolation level determined by the SLA. The isolation level allows specifying the VNFs dedicated to V2X-NS and the ones shared between V2X-NSs. Figure 1 presents two V2X-NSs with different network requirements. These V2X-NSs share VNFs at the core level and have dedicated VNFs at the MEC and NR levels. Each CAV is equipped with a 5G network card to communicate with other CAVs via the PC5 interface and C-V2X applications via the Uu interface. To ensure the end-to-end security of V2X-NSs, the Security Operations Center (SOC) deploys a set sVNFs within V2X-NSs with the help of the NSM. These sVNFs have two tasks: (i) detect inter-slice V2X attacks and (ii) contribute to building the global FL model. sVNFs are deployed at different levels of V2X-NSs. At the 5G NR level, sVNFs are deployed at some selected CAVs. This selection can be made by obtaining the list of most trusted CAVs from SOC, and thereby, the most trusted CAVs host sVNFs. Practically, we expect that CAVs are equipped with a hypervisor/container engine, which can support one or more sVNFs. At the MEC, sVNFs are deployed according to the number of CAVs subscribed to the V2X-NS. In addition, sVNFs can migrate from a MEC node to another node, according to the mobility of CAVs. Besides, at the 5G core, sVNFs are deployed according to the isolation level defined by the SLA.

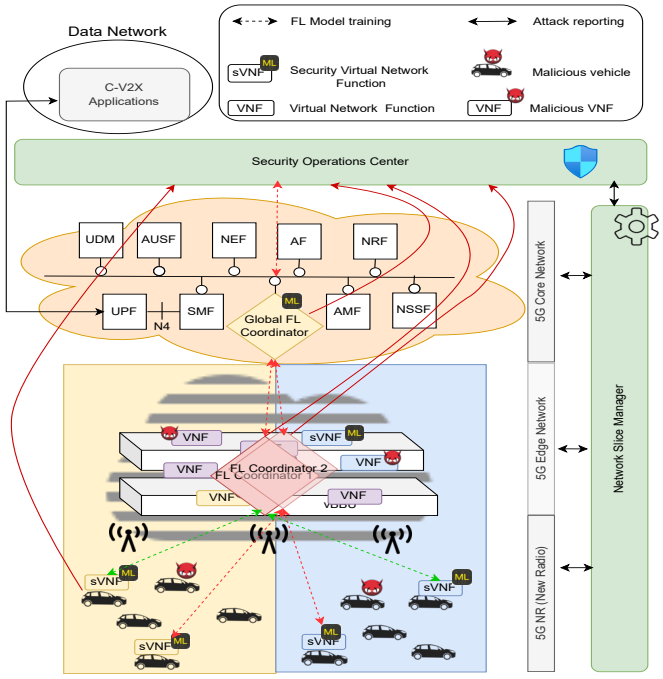

Fig. 1: FL-based inter-slice attack detection scheme for 5G-V2X NS

We distinguish three types of sVNFs: (i) Ordinal sVNFs deployed on CAVs aim to detect inter-slice V2X attacks and train local DL models, (ii) sVNFs deployed on the MEC, called FL coordinators, aim at detecting inter-slice V2X attacks and aggregate local DL models received from CAVs,

and (iii) An sVNF deployed on the 5G Core, called Global FL coordinator, aims to detect V2X inter-slice attacks and aggregate DL models received from FL coordinators. To this end, sVNFs embed two engines: (i) DL model learning engine, which is used for training local DL models in the case of ordinal sVNFs, and for aggregating DL models in the FL coordinators, and (ii) Attack detection engines, which uses a DL model to detect the inter-slice attacks.

Our scheme thus includes three processes: (i) inter-slice V2X attack detection process, (ii) FL collaborative learning process, and (iii) FL global model update process. While the attack detection process is always active, collaborative learning and model update processes are on-demand triggered by the SOC. In the following, we briefly describe each of these processes:

1) **Attack detection:** In this process, sVNFs use DL models, which are deployed on their sVNFs to detect inter-slice V2X attacks and report them to the SOC. Initially, the SOC installs a basic version of the DL model on sVNFs, as shown in Figure 2 (stage 1), which is updated after each FL collaborative learning process.
2) **FL Collaborative learning:** As shown in Figure 2 (stage 2), during this process, the SOC pushes an initial DL model via secure communication channels to all FL coordinators along with some parameters such as the number of rounds to run and the rewarding price, which is used to motivate FL clients (CAVs). The FL coordinators select FL clients for performing the FL training process while encouraging them by using the rewarding price for providing high-accuracy local DL models. The FL coordinators also aggregate all the local DL models received from FL clients. Once its model is ready, the FL coordinator sends it to the global FL coordinator. Once the latter gets all the global models from all FL coordinators, it aggregates them and sends the resulting model to the SOC for evaluation.
3) **Model update:** As shown in Figure 2 (stage 3), the DL model update process follows the FL collaborative training process. Thus, the SOC forwards the DL model to all sVNFs to update their attack detection engine to adapt to the adversary environment and enhance the detection rate accordingly.

As previously mentioned, detected inter-slice attacks are reported to the SOC, which is in charge of reacting in response to them and updating the list of trusted CAVs. Besides, we focus on detecting inter-slice V2X attacks. More precisely, two classes of attacks are considered in this paper.

- **Class 1 (Distributed Denial of Slice Service (DDoSS)):** The attackers of this class try to prevent V2X-NS members from having ordinal access to V2X-NS services. DDoSS is a variant of the DoSS attack that involves multiple attackers belonging to multiple V2X-NSs, which can collaborate and synchronize to perform the attack. DDoSS can target different parts of V2X-NSs, ranging from the NR to the 5G core. Moreover, attackers can exploit various network protocols of the protocol stack, such as ARP, UDP, and HTTP.
- **Class 2 (Unauthorized access to V2X slices):** Attackers of this class try to have unauthorized access to V2X-NSs that are not attached to them. For example, two malicious CAVs attached to two different V2X-NSs can create a tunnel for sharing slice-sensitive information between them. Unauthorized access to V2X-NS is a multi-stage attack that starts by infiltrating the targets by exploiting protocol flaws.

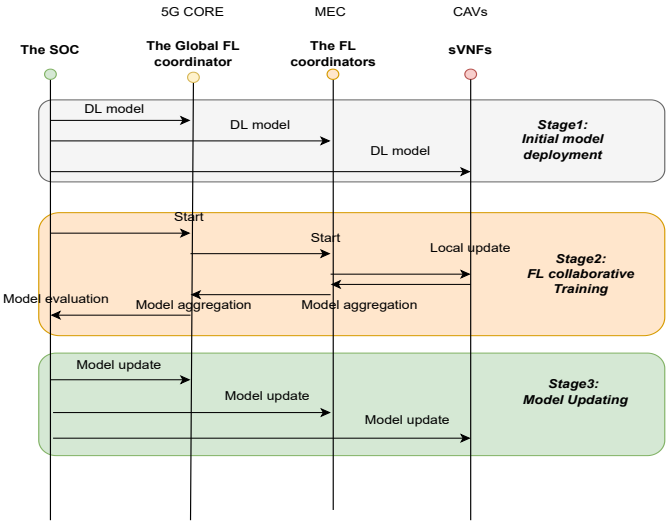

Fig. 2: FL collaborative learning and model update processes

IV. FL Clients Incentive Mechanism

This section describes our game-theory-based incentive mechanism to motivate FL clients (CAVs) for providing high-accuracy local DL models. We formulate the process of building a global FL model as a Stackelberg game. This game consists of the SOC acting as the leader and several CAVs acting as followers.

A. Problem Formulation

We assume that an FL model building process is performed between the SOC and a set of n $CAVs$: $\mathbb{D} = \{CAV_1, CAV_2, ..., CAV_n\}$, where an FL coordinator C is acting as a broker between the SOC and the set of $CAVs$. Each CAV_i is rational and independently decides the level of contribution in terms of the amount of data dt_{CAV_i} used to train its local DL model for serving the SOC. To ensure fair a game, the reward that each CAV_i gets is proportional to the amount of data dt_{CAV_i} in the total data used to train the global model. However, training local DL models generate a processing overhead as well as consumes energy. Thus, the utility of CAV_i should also consider the energy consumption and the processing overhead together with the reward getting for the SOC. Let α_i represent the amount of consumed energy per unit size to locally train the model and send it to C. Thus, the overall energy consumption for CAV_i is $\alpha_i \, dt_{CAV_i}$. Moreover,

due to the resource limitation of CAVs, local training of the DL model generates additional processing overhead that may cause unnecessary inconvenience to different processes running on CAV_i. We formulate this processing overhead as $\beta\ dt_{CAV_i}$, where β is the processing overhead factor. To this end, the utility function of CAV_i is given by formula 1.

$$u_{CAV_i} = \frac{dt_{CAV_i}}{\sum_{j=1}^{N} dt_{CAV_i}}\mathbb{P} - \alpha_i dt_{CAV_i} - \beta dt_{CAV_i}^2 \quad (1)$$

where $\mathbb{P}$ is the price set by the SOC for building the global FL model process. On the other hand, the utility function of the SOC (U_{SOC}) depends on the monetary costs it pays ($\mathbb{P}$) and the accuracy of the global FL model that it obtains. Thus, the utility gain of the SOC is simultaneously related to the total amount of data dt_{CAV_i} used to train the local model and the corresponding accuracy of the local models l_i, which is denoted by $\sum_{i=1}^{N} l_i dt_{CAV_i}$. To this end, U_{SOC} is given by formula 2.

$$U_{SOC} = \sum_{j=1}^{N} l_j dt_{CAV_j} - \mathbb{P} \quad (2)$$

B. Analysis of Stackelberg Equilibrium

Building an FL model between the SOC and n $CAVs$ is formulated as a Stackelberg game. The SOC acts as a leader while n $CAVs$ are regarded as followers. For motivating the $CAVs$ to participate in building the global FL model, the SOC stimulates all the $CAVs$ with reward parameter $\mathbb{P}$. According to the given reward parameter $\mathbb{P}$, the $CAVs$ determine the amount of data (dt_{CAVs_i}) to be used for training their local DL models to maximize their utilities. In our scheme, we assume that the FL coordinator (C) is fully aware of the strategies and actions of $CAVs$. Thus, the SOC can be replaced with a broker C to determine the optimal reward parameter $\mathbb{P}^*$. For a given reward $\mathbb{P}^*$, each CAV decides the best response $dt^*_{CAV_i}$ to maximize the payoffs. The goal of the proposed game is to find the unique Stackelberg equilibrium, where both the SOC and $CAVs$ have no motivations to change their strategies unilaterally [13]. Thus, the Stackelberg equilibrium is defined as follows.

Definition 1: we consider a series of decisions ($dt^*_{CAV_i}$, $\mathbb{P}^*$) as the Stackelberg equilibrium, when and only when it meets the following set of inequalities.

$$\forall dt_{CAV_j}, u_{CAV_j}(dt^*_{CAV_j}, \mathbb{P}^*) \geq u_{CAV_j}(dt_{CAV_j}, \mathbb{P}^*) \quad (3)$$

$$\forall \mathbb{P}, U_{SOC}(dt^*_{CAV_i}, \mathbb{P}^*) \geq U_{SOC}(dt^*_{CAV_i}, \mathbb{P}) \quad (4)$$

Theorem: The unique Stackelberg equilibrium exists between the SOC and $CAVs$.

Proof: We mathematically prove that the second derivative of u_{CAV_i}, $\frac{\partial^2 u_{CAV_i}}{\partial dt^2_{CAV_i}} < 0$. Thus, in the response of a given reward parameter $\mathbb{P}$, each CAV has its unique optimal strategy $dt^*_{CAV_i}$. In addition, since C has full knowledge of all the best responses of $\forall\ CAV_i \in \mathbb{D}$, $dt^*_{CAV_i}$, the utility function of the CAV can be adjusted accordingly. The maximum utility of the CAV, $\mathbb{P}^*$ can be found using the formula 5, which is the best strategy under given the optimal responses from all $CAVs$, with $\partial^2 U_{SOC}/\partial \mathbb{P}^2 < 0$.

$$\mathbb{P}^* = \frac{\bar{l}^2(n-1)^2 - (\sum_{i=1}^{n} \alpha_i)^2}{8\beta(n-1)} \quad (5)$$

To this end, both the leader and followers are satisfied with their decisions ($dt^*_{CAV_i}$,$\mathbb{P}^*$) and have no motivation to change their strategies. Thus, the unique Stackelberg equilibrium is reached in this game.

V. Performance Evaluation

This section evaluates the performance of our scheme. We first evaluate the performance of our FL collaborative training model. We then perform an equilibrium analysis of the FL model-building game.

A. FL model evaluation

In this section, we train and test our FL global model for detecting inter-slice V2X attacks. The FL architecture was implemented using Tensorflow and Keras Python libraries. The global model was trained on the Google Colab platform, using Compute Engine backend (TPU). FL clients (CAVs) have been implemented as Tensorflow instances running local models. For the multi-class classification, we use a DL model, which will be embedded in sVNFs. We selected the CSE-CIC-IDS-2018 dataset [14] to train our DL model since it adequately covers different types of inter-slice V2X attacks, addressed in this paper. To train our DL model, we have a dataset containing a total of $1,959,893$ instances (rows). The dataset originally includes 80 features on network flows. But after converting the timestamp feature to the date format, the number of features becomes 85. Table I shows the dataset distribution per each attack type. This dataset contains several features with different scales, which slows down the training process. To address this issue, we also rescaled the dataset using MinMaxScaler, which normalizes dataset features to values in the range of [0,1]. Before passing the training process, the dataset was split into training, validation, and test sub-datasets. Since this dataset contains more than 1 million rows, which is in the order of big data, we thus apply recommendations given in [15], by choosing 1% of the whole dataset as a validation dataset, and 1% as a test dataset.

TABLE I: Dataset distribution for muli-class classification

Attack class	Attack type	Support
	Benign	499717
Class 1	DDOS attack-HOIC	343006
	DDoS attacks-LOIC-HTTP	288096
	Bot	286191
Class 2	FTP-BruteForce	193360
	SSH-Bruteforce	187589
	Infilteration	161934

Our model consists of (i) an inputs layer with 84 neuron nodes, (ii) two hidden layers with 85 and 42 hidden nodes

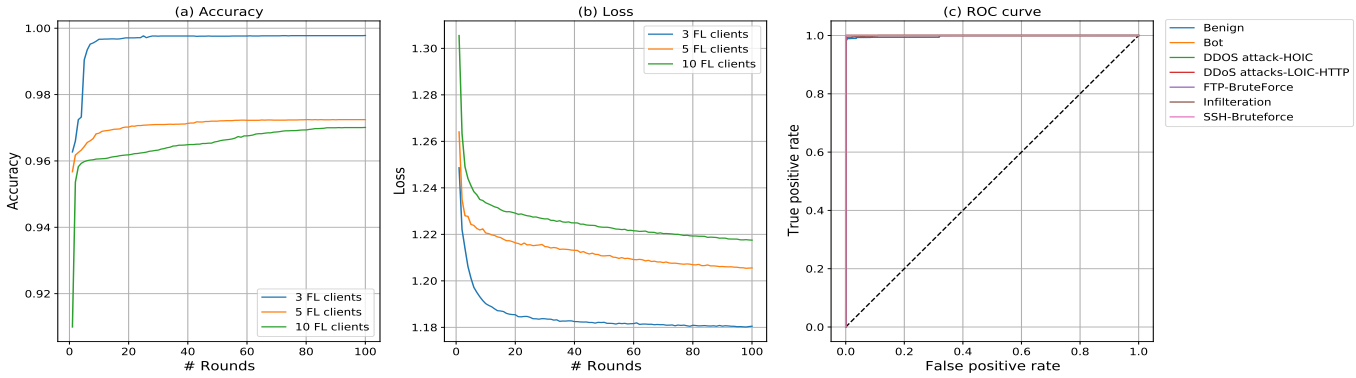

Fig. 3: Model performance

for each of them, respectively, with a dropout rate set to 0.75, (iii) an output layer with 7 nodes based on one hot encoding to detect and identify inter-slice V2X attacks. The ReLU activation function is used for the hidden nodes, while the softmax function is used for the output layer. To calculate the weights of local models we use the Stochastic Gradient Descent (SGD) with a learning rate equal to 0.01. We also optimize the decay parameter, which controls how the learning time change over time. Indeed, we decay the learning rate with respect to the number of rounds rather than the number of epochs. The Federated Averaging is used after each round to calculate the weight of the global model. During the training, we consider mini-batches of size 32. Table II lists the hyperparameters of the model.

TABLE II: Training parameters of the global model

Parameter	**Value**
Optimizer	SGD
Learning rate	0.01
Batch size	32
Dropout	0.75
Ratio of validation/test dataset	1%
# Rounds	100

We considered several evaluation metrics, including accuracy, precision, recall, F1-score, and the Area Under the Curve (AUC). We also take several deployment parameters of our DL model into the account, such as the training time, the size of the model, and the inference time. Figure 3 (a) and Figure 3 (b) show the obtained accuracy values and loss values, respectively, versus the number of rounds, which is limited to 100. This evaluation considers three configurations of FL clients: 3, 5, and 10. Figure (b) shows that the loss decreases as the number of rounds increases. In addition, Figure (a) shows that obtained accuracy surpasses 96% for all the configurations since the first rounds. Moreover, we can see that the accuracy drops with the rise of FL clients. For example, accuracy converges to more than 99% when 3 FL clients are selected, while it still reaches 97% in the case of 10 FL clients. The nature of FL architecture can explain this. Indeed, the number of local models to be aggregated increases the chances of divergence/conflicts in the aggregation process [16].

Table III shows the performance of the global model built with 5 FL clients on the test dataset. We have obtained 97% and 96% for accuracy and F1-score, respectively. We also got 99% for the AUC. These results demonstrate the efficiency of our model to detect not previously seen attack instances and distinguish between attack classes.

TABLE III: Attack detection results (FL clients = 5)

Accuracy	Precision	Recall	F1-score	AUC
0.97	0.98	0.95	0.96	0.99

Table IV shows the detection results of our global model per each type of attack. These results demonstrate the high capability of our DL model to distinguish the benign traffic network from the malicious one. Indeed, our DL model with 95% of F1-score regarding identifying benign network traffic. In addition, as shown in the Receiver Operator Characteristic (ROC) curve, illustrated in inf Figure 3 (c), attack curves are closer to the top-left corner, which further proves high performance to classify inter-slice V2X attacks. Note that the backline illustrates the performance of a basic classifier. Moreover, Table IV shows high accuracy (>=99% of F1-score) to detect most of the attacks (5/6). We obtain less accuracy for the infiltration attack due to the lack of sufficient traces to recognize this attack accurately. However, future FL processes will enhance the accuracy, especially in the presence of FL clients with traces of this attack.

TABLE IV: Multi-class results of the DL model

Traffic type	Precision	Recall	F1-score	Support
Benign	0.90	0.99	0.95	4997
Bot	0.99	1.00	0.99	2862
DDOS attack-HOIC	1.00	1.00	1.00	3430
DDoS attacks-LOIC-HTTP	1.00	1.00	1.00	2881
FTP-BruteForce	1.00	1.00	1.00	1934
Infilteration	0.99	0.68	0.80	1619
SSH-Bruteforce	1.00	1.00	1.00	1876

Table V gives our model deployment parameters. It takes 19.84 seconds on average to locally train local DL models,

which is a good time for collaborative training of the model. Our model also has a small storage size (less than 1 MB), making them lightly deployable in sVNFs, at different levels from CAVs to the 5G core. Moreover, the Inference Time is short. It takes less than 57.7 ms to decide if an event is an attack or not, which demonstrates the fast detection of our scheme, leading to an immediate reaction after detecting an attack.

TABLE V: Model deployment parameters

Training time (s)	Size (KB)	Inference time (ms)
19.84	65.6	57.7

B. Incentive evaluation

In this section, we analyze the Stackelberg game for the FL model. More specifically, we analyze the best responses of the SOC and CAVs considering different model parameters. This evaluation considers that the FL process consists of 50 CAVs. We also consider that model parameters α and β are in the range $[0.1-0.9]$. In addition, we assume that $\mathbb{P} \in [1-100]$. In Figure 4, we evaluate the utility U_{SOC} with the variation of both the reward ($\mathbb{P}$) and the local model accuracy ($\bar{l}$). As we can see in Figure 4, U_{SOC} is influenced by different reward values $\mathbb{P}$. In addition, U_{SOC} increases with the accuracy of the local models $\bar{l}$. The dashed lines in Figure 4 also show the best strategy of the SOC $\mathbb{P}^*$ to have the maximum utility for model accuracy $\bar{l}$. Thus, the SOC should increase its $\mathbb{P}^*$ for encouraging CAVs to provide higher local model accuracy. For example, to increase the accuracy of the local model $\bar{l}$ from 0.85 to 0.95, the SOC should increase the value of $\mathbb{P}^*$ from 56 to 78 i.e. 39%.

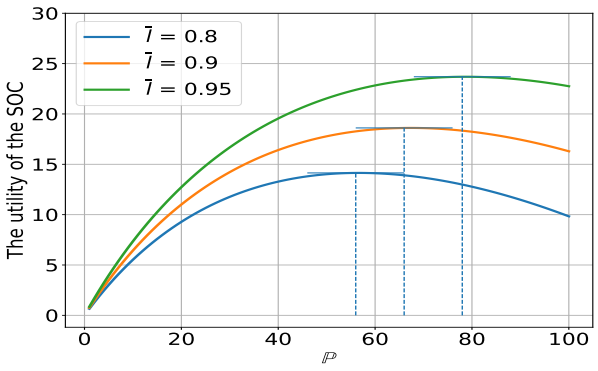

Fig. 4: The utility of the SOC with the variation of $\mathbb{P}$ and $\bar{l}$

VI. Conclusion

The failure in detecting inter-slice 5G-V2X attacks could threaten the safety of both drivers and passengers. In this paper, we have proposed a scheme for detecting inter-slice V2X attacks. Our scheme combines the flexibility of virtual security as a service concept and the power of deep learning and federated collaborative learning to efficiently detect the attack while maintaining privacy preservation and the V2X-NSs isolation. We plan to perform featuring engineering as future work to enhance the performance further.

Acknowledgment

This work was supported by the 5G-INSIGHT bilateral project (ID: 14891397) / (ANR-20-CE25-0015-16), funded by the Luxembourg National Research Fund (FNR), and by the French National Research Agency (ANR).

References

[1] 3GPP TS 23.287, "Architecture enhancements for 5G System (5GS) to support Vehicle-to-Everything (V2X) services," Jul 2020.

[2] C. Campolo, A. Molinaro, and V. Sciancalepore, "5G Network Slicing for V2X Communications: Technologies and Enablers," *Radio Access Network Slicing and Virtualization for 5G Vertical Industries*, pp. 239–257, 2021.

[3] Y. Khettab, M. Bagaa, D. L. C. Dutra, T. Taleb, and N. Toumi, "Virtual security as a service for 5G verticals," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2018, pp. 1–6.

[4] T. Zhang and Q. Zhu, "Distributed privacy-preserving collaborative intrusion detection systems for vanets," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 1, pp. 148–161, 2018.

[5] F. A Ghaleb, F. Saeed, M. Al-Sarem, B. Ali Saleh Al-rimy, W. Boulila, A. Eljialy, K. Aloufi, and M. Alazab, "Misbehavior-aware on-demand collaborative intrusion detection system using distributed ensemble learning for vanet," *Electronics*, vol. 9, no. 9, p. 1411, 2020.

[6] J. Shu, L. Zhou, W. Zhang, X. Du, and M. Guizani, "Collaborative intrusion detection for VANETs: a deep learning-based distributed SDN approach," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[7] A. Uprety, D. B. Rawat, and J. Li, "Privacy Preserving Misbehavior Detection in IoV using Federated Machine Learning," in *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2021, pp. 1–6.

[8] A. Boualouache and T. Engel, "Federated learning-based scheme for detecting passive mobile attackers in 5g vehicular edge computing," *Annals of Telecommunications*, pp. 1–20, 2021.

[9] H. Liu, S. Zhang, P. Zhang, X. Zhou, X. Shao, G. Pu, and Y. Zhang, "Blockchain and Federated Learning for Collaborative Intrusion Detection in Vehicular Edge Computing," *IEEE Transactions on Vehicular Technology*, 2021.

[10] M. Abdel-Basset, N. Moustafa, H. Hawash, I. Razzak, K. M. Sallam, and O. M. Elkomy, "Federated Intrusion Detection in Blockchain-Based Smart Transportation Systems," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[11] A. Thantharate, R. Paropkari, V. Walunj, and C. Beard, "Deepslice: A deep learning approach towards an efficient and reliable network slicing in 5g networks," in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. IEEE, 2019, pp. 0762–0767.

[12] N. A. E. Kuadey, G. T. Maale, T. Kwantwi, G. Sun, and G. Liu, "Deepsecure: Detection of distributed denial of service attacks on 5g network slicing-deep learning approach," *IEEE Wireless Communications Letters*, 2021.

[13] J. Zhang and Q. Zhang, "Stackelberg game for utility-based cooperative cognitive radio networks," in *Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, 2009, pp. 23–32.

[14] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, pp. 108–116, 2018.

[15] DeepLearning.AI, "Setting up your ml application: Train/dev/test sets," *Coursera. Available online: https://cs230.stanford.edu/files/C2M1.pdf (accessed on 15 April 2022)*, 2022.

[16] M. S. Ozdayi, M. Kantarcioglu, and R. Iyer, "Improving accuracy of federated learning in non-iid settings," *arXiv preprint arXiv:2010.15582*, 2020.