I. Siniosoglou, V. Argyriou, T. Lagkas, A. Tsiakalos, A. Sarigiannidis and P. Sarigiannidis, "Covert Distributed Training of Deep Federated Industrial Honeypots," 2021 IEEE Globecom Workshops (GC Wkshps), 2021, pp. 1-6, doi: 10.1109/GCWkshps52748.2021.9682162.



# Covert Distributed Training of Deep Federated Industrial Honeypots

Ilias Siniosoglou[†], Vasileios Argyriou[‡], Thomas Lagkas[§], Apostolos Tsiakalos[¶], Antonios Sarigiannidis[¶] and Panagiotis Sarigiannidis[†]

***Abstract*—Since the introduction of automation technologies in the Industrial field and its subsequent scaling to horizontal and vertical extents, the need for interconnected industrial systems, supporting smart interoperability is ever higher. Due to this scaling, new and critical vulnerabilities have been created, notably in legacy systems, leaving Industrial infrastructures prone to cyber attacks, that can some times have catastrophic results. To tackle the need for extended security measures, this paper presents a Federated Industrial Honeypot that takes advantage of decentralized private Deep Training to produce models that accumulate and simulate real industrial devices. To enhance their camouflage, SCENT, a new custom and covert protocol is proposed, to fully immerse the Federated Honeypot to its industrial role, that handles the communication between the server and honeypot during the training, to hide any clues of operation of the honeypot other that its supposed objective to the eye of the attacker.**

***Index Terms*—Honeypots, Deep Learning, Industrial Control System, SCADA, Autoencoder, Data Generation**

## I. Introduction

In order to orchestrate, organize and make the communication of industrial environment possible, Industrial Control Systems (ICS) and Supervisory Control and Data Acquisition (SCADA) systems, undertake that task and oversee the correct operation of their respective supervised systems. These systems include, but are not limited to, water pump stations, electrical smart-grids, electrical grid substation, gas pipelines and so on that differ on their level of criticality but have additional needs to that of simple networks. As legacy systems began to upgrade to suit the modern network needs [1], they became exposed and susceptible to cyberattacks. This led to major security breaches with dire implications. Possible intruders, if attain the right access, can discover and steal important classified data, personal information of employees and services, even tinker with critical systems, for example, opening and closing the electric Supply in buildings or whole areas, messing with gas pumps and so on. As can be made clear, in the case of infrastructures like hospitals this can have catastrophic consequences that even threaten human lives.

In the last couple of years, Honeypots, the individual components deployed in Honeynets, that undertake the task of posing as active target devices or services in a network to divert, log and audit possible malicious interaction in that network, have come a long way. They are effective in capturing the interest of attackers, tracking and analysing their movements and thus giving time to security measures to be deployed and notify about the interests and motives of the impending attacks.

Unfortunately, malicious practices have also evolved. This creates the need for evermore cognitive and advanced security systems. In the case of honeypots and using techniques like network discovery and ML traffic classification, Figerprinting and so on [2], attacker can easily distinguish the security traps and avoid them, thus rendering them useless. This problem is the product of two factors, a) honeypots behave in static-predetermined manners or b) some used protocols, like Modbus,

[†] I. Siniosoglou and P. Sarigiannidis are with the Department of Electrical and Computer Engineering, University of Western Macedonia, Kozani, Greece - E-Mail: {isiniosoglou, psarigiannidis}@uowm.gr

[‡] V. Argyriou is with the Department of Networks and Digital Media, Kingston University, Kingston upon Thames, United Kingdom - E-Mail: vasileios.argyriou@kingston.ac.uk

[§] T. Lagkas is with the Department of Computer Science, International Hellenic University, Thessaloniki, Greece - E-Mail: tlagkas@cs.ihu.gr

[¶] A. Tsiakalos and A. Sarigiannidis are with Sidroco Holdings Ltd., Limassol, Cyprus - E-Mail: {atsiakalos, asarigia}@sidroco.com

support only a handful of actions. For example, if a Honeypot emulating a Modbus Programmable Logic Controller (PLC) device running modbus starts making other transaction in the network, then that produces a possible indication that the machine is not what it seems to be.

Taking into account the above problem formulation, this work strives to tackle the mentioned limitations by extending an Industrial Modbus Honeypot that incorporates a Deep Neural Network. The contributions of this work are as follows:

- Presents an updated Neural Honeypot based on LTSM-Autoencoders for better behaviour accumulation and replication for the Modbus protocol
- Extents this Honeypot to a Federated Modbus Honeypot to be used in decentralized and distributed Honeynets
- Presents modbuS Covert fEderated traiNing proTocol (SCENT), a protocol for secure covert Federated Training in Honeynet networks

The rest of this paper is organized as follows. Section II describes the tools used. Section III explains the methodology used for the solution while Section IV provides the evaluation results of the presented work. Section V concludes this work.

## II. Background

### A. Modbus Communication Protocol

Modbus [3] is an openly available protocol, widely used in industrial applications for data communication between industrial equipment and control systems. It is a simple protocol, that supports the communication of industrial network entities like PLC and Remote Terminal Units (RTUs) over serial and Transmission Control Protocol (TCP) communication. In this work the Modbus protocol is utilized for both the traffic replication through the deep accumulation of Modbus memory frames and for the proposed covert Federated Training protocol.The basic Modbus entities comprising the Modbus industrially ecosystem are the i) Modbus Clients, the ii) Modbus Servers and the iii) Modbus Slaves. A Client is a remote query terminal, such as an Human-Machine Interface (HMI), requesting information from the Modbus servers and sending control information to them. Servers usually represent either PLC or RTU controllers in the network. Those controllers supervise Modbus Slave entities, such as Acquisition Blocks, that oversee the field devices. Each Server can have multiple Slaves with unique slave IDs attached to them.

### B. Honeypot Technology

As mentioned, Honeypots represent security traps deployed in a network. This is done to attract the interest of possible attackers and safeguard the integrity of real devices in a network. In this work a research Honeypot implementation is proposed based on the Conpot honeypot [4], a well known industrial honeypot. A research Honeypot, having the role of gathering and analyzing information in addition to a being a simple decoy, consists of more intricate components and logging systems. This work mainly focuses on the Modbus protocols since it is widely used in industrial applications, in both modern and legacy ICSs.

Except Conpot, numerous Honeypot technologies have seen the light in the last few years, since the need for enchanced seucutity measure becomes stronger. For example, the work performed in [5]. The authors present a NeuralPot, a Deep Neural Honeypot that is trained on modbus data of RTU an PLC controllers. They subsequently test the data against two DL architectures, an Autoencoder and a GAN network and compare their results on generating realistic responses. A drawback of this method is that the models have to be manually trained. Furthermore, the models used are not configured to capture the temporal attributes of the data. Similarly, in [6] realizes a honeypot extending [7] that uses Q-Learning to interact and adapt to the attacker's behaviour.

### C. Federated Learning

Federated Learning is a distributed ML methodology [8] that orchestrates and trains Deep Learning models in a big corpus of edge devices [9]. Models are trained locally on edge devices and then their weights are send to a central server where they are fused to a global model using an algorithm like Federated Averaging [10]. The global model is then sent back to the remote devices to be used. The central server disseminates a global model $w^0_{Global}$ to a Federated population $P_f \in [1, N]$ where $N \in \mathbb{N}^*$. Every node holds a set of local data $D_{i \in N}$

and local models $w_l^i$. The distributed models are trained on the on-device data $D_i$ and then the model weights $w_{Global}^i$ are retrieved by central server to be fused utilizing the Federated Averaging (1), or a similar fusion algorithm to produce thew global model $w_{Global}^k$ [11] containing the newly collected knowledge.

$$w_G^k = \frac{1}{\sum_{i \in N} D_i} \sum_{i=1}^{N} D_i w_i^k \tag{1}$$

Here $w_G^k$ is the global model on the $k_{th}$ iteration and $w_i^k$ is the remote $i_{th}$ model at that iteration.

## III. Methodology

### A. LSTM Autoencoder Design

This work takes advantage of the Auto-Encoder Architecture [12] that can encode a data space $X$ to a manifold $F$ and then decompress it to space $F$ predicting values $P$ using an Encoder-Decoder. The data leveraged in this work, are defined as a collection of time-dependant Modbus memory block updates. Since they are time-dependant, meaning that the current samples is based on the previous samples with an inclined direction (higher/lower), to better represent the device that the data are coming from and in extend its function correctly, the augmentative neural network needs to take in account the order of the accumulated data and their correlation. To that end, this work implements an Long Short-Term Memory (LSTM) Autoencoder [13], taking advantage of the LSTM's [14] ability to capture the temporal structure of given sequences and the manifold recreation property of the Autoencoder. Figure **??** shows the according structure.

The produced model consists of two mirroring partitions, the LSTM-Encoder and the LSTM-Decoder containing two layers each with scaling features in opposite directions, as can be seen in Figure **??**. The model is compiled with the Mean Squared Error (MSE) loss function, eq. 2 and the Adam Optimizer [15].

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y}_i)^2 \tag{2}$$

Here, $n$ represents the number of predictions, while $y$ and $\tilde{y}$ denote the samples and predicted values, respectively. To train the network, the input data of $N$ features are rearranged in windows of $t$ time-steps and are fed to the model.

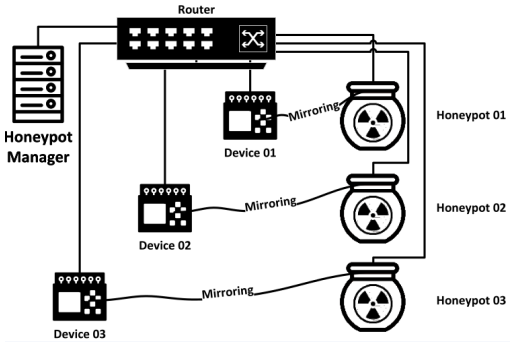

Fig. 1: Industrial FL Architecture

### B. Honeypot Federated Architecture

The Federated Learning architecture extends the training to a distributed ML ecosystem. Pursuing the notion of Deep Neural Honeypots for dynamic asset simulation, a Honeypot Network (Honeynet) now becomes a grid of clever honeypots emulating real devices, Figure 1. This knowledge can be utilised to produce a generalized global model that can generate realistic data that adhere to a whole corpus of similar edge devices in a macro-Federated Environment.

In this scheme, the Honeypot manager, Figure 1, is in charge of deploying and managing honeypots in the industrial network, but also has the role of the Federated Server (FL Server). This means that the Honeypot manager will decide when a Federated Training session should take place, with which population, in case there are more than one categories of Deep Neural Honeypots in the network. The honeypots mirror their corresponding real devices to get the needed data for the training. Beginning the Training process, the remote workers (the honeypots) register to the FL server so that they can participate. The training process follows the described FL process. After the training process has ended, the model passes through a personalization step in order to personalize the received models to the device the honeypot mirrors.

### C. SCENT: Modbus Federated Training Protocol

This work, to complement the Federated Honeynet Design, proposes a custom covert Federated

Training protocol, SCENT (modbuS Covert fEderated traiNing proTocol), that relies on the Modbus scheme to orchestrate the FL procedure and communication with the involved parties under the pretext of industrial data exchange. In a real industrial environment a modbus device, can usually communicate using only the modbus protocol. The Federated process is a mainly communicational scheme that demands the constant exchange of information between involved parties. An attacker, would probably uncover the honeypot's intentions seeing that a device makes long non-modbus transaction over the network. In contrast, during the whole Federated process SCENT communicates only through modbus.

In the SCENT scheme, the FL Server takes the role of a Human Machine Interface (HMI) that polls the FL Client being a Modbus Master or controller. At this point it is important to note that Modbus is a One-Way request protocol. This means that the Modbus master can only receive modbus requests from an HMI and can only reply with a corresponding response but can never make requests on its own. To solve this drawback, the protocol relies on a wait-for-request-and-update policy described in a finite state machine pipeline. The honeypot reserves some addresses for the FL communication among its usual addresses or deploys an extra modbus slave reserved for the FL. These addresses are used for control signals and payload transactions. The addresses can have variable offsets but cannot dynamically change since the FL Server must strictly know the exact location and lengths of the address blocks. Two address blocks are used in SCENT, the Control Block and the Payload Block composed of modbus Holding Registers. The former is used for signaling the various states in the protocol stack and information about them like acknowledging a successful read and the latter is primarily used for the transference of the model's weights. The payload address block can also consist of Coils instead of Holding Registers in the case of Binary Neural Network (BNNs) for compression and optimization purposes.

The communication takes place by the FL Server writing to the control addresses and the FL Client acting upon the control context. Since the client cannot sent messages to the server to notify about the stages of the FL procedure, it writes the needed information in the control block registers. The Server pols the registers and thus gets notified about the state of the process. The orchestration pipeline stack can be seen in Figure 2. The weight transmission follows the same principle.

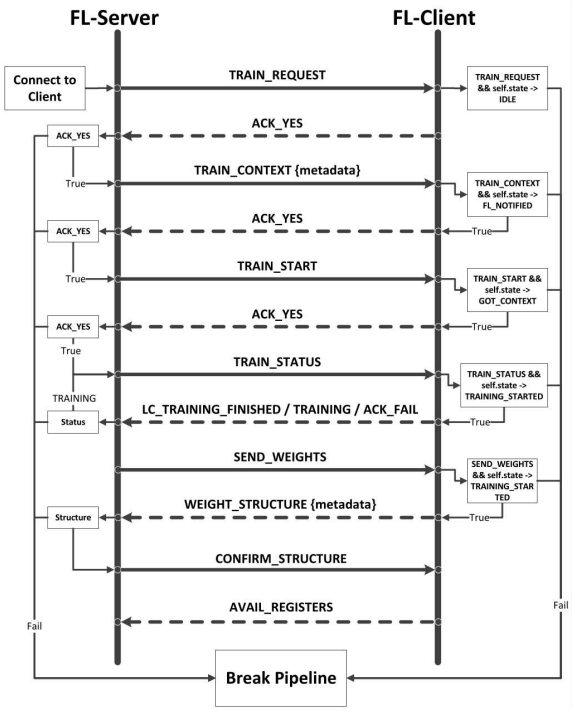

Fig. 2: SCENT orchestration Protocol Stack

The complex part of the process occurs with the weight transmission. Usually, the weights are stored in matrices, each representing a layer of the network. This means that we have heterogeneously distributed vectors of weights. Another problem is that the maximum number of unsigned integer values that modbus can carry in one response is 125. To solve these problems, the weight matrix is segmented to layers and each layer is segmented to half of the available transmission registers. This way the transmission process is distributed to phases and each weight vector, is divided to pages that are transmitted separately. The segmentation in half of the available registers occurs because before they are transmitted the weights are converted to unsigned integers. Since in the modbus protocol each register is 16bit and the integer and float values are represented by 16bits and 32bits, respectively, a float value takes two registers. The transactions that need to occur to transmit each weight block

are calculated so both sides know how many data pages to transmit/receive. Equation 3 shows the transaction calculation formula.

$$tr = \frac{2\prod_{i=1}^{n} d_i}{a_r}, \tag{3}$$

$$\begin{cases} a_r mod 2 = 0, & tr \\ a_r mod 2 > 0, & tr + 1 \end{cases} \tag{4}$$

Here, $d_i$ is the $i^{th}$ dimension of the layer weights, and $a_r$ denotes the available register. It should be noted that in the case of the Holding Registers, the $a_r$ should be an even number, equation 4. To calculate the total weight transmissions over the weight communication we refer to equation 5,

$$Tr = w_d \frac{2\prod_{i=1}^{n} d_i}{a_r} \tag{5}$$

where $Tr$ are the total transactions and $w_d$ are the weight layers. These are calculated without the confirmation sequences or the metadata transmissions.

## IV. Evaluation

### A. Evaluation Data

The utilized data were produced by a real network traffic of industrial modbus devices. The traffic was mirrored from the device and was stored in the pcap file format. The traffic consists of information retrieval requests and command exchanges between an HMI service and the corresponding RTU device running Modbus. The data were augmented to simulate a number of similar devices to be used in the Federated Training. Each of the files represents the traffic to and from an industrial edge node for Federation. The data include queries to 41 modbus addresses with different distribution of data for each address.

### B. Centralized Model Evaluation

Evaluating the proposed LSTM-Autoencoder model used for the traffic accumulation and the subsequent Modbus value generation in the honeypot, the model seems to converge at 180-200 iterations and saturated after that, for the given data. The model seems to capture the data's tendencies to change over time producing real-like results. Figure 3 depicts the mean and standard deviation of the predicted against the real data. As seen, the model captures the spacial characteristics of the data and can reproduce them to emulate the real device.

TABLE I: Autoencoder Training Results

| Model | Mean Dist | FID | MAE | MSE | RMSE | R2 | EVS |
|---|---|---|---|---|---|---|---|
| AE-NeuralPot | 0.0285 | 11.6361 | 0.1283 | 0.0415 | 0.1736 | -1.5517 | -1.3489 |
| AE-Simple | 0.0294 | 11.9394 | 0.0855 | 0.0148 | 0.1036 | -0.0228 | -0.0024 |
| AE-LSTM | 0.0086 | 13.1457 | 0.0631 | 0.0159 | 0.0848 | -0.0109 | -0.0001 |
| AE-LSTM FL | 0.0044 | 13.0914 | 0.0661 | 0.1042 | 0.0870 | 0.4792 | 0.4878 |

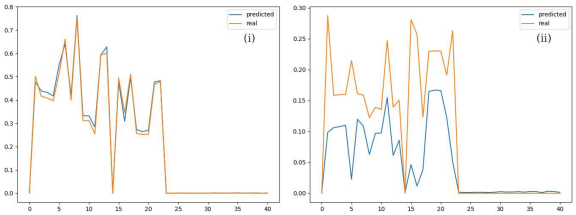

Fig. 3: Predicted against the Real data. (i) Mean, (ii) Std

### C. Federated Deployment Evaluation

For the Federated testbed, 4 remote workers or honeypots were emulated. In order to emulate the data originating from a Federated environment, the data of two devices were divided to be used by the four remote workers in a Federated dataset. As can be seen in Table I, both the central and federated models produce better results. Noticeable are the Mean Dist and FID scores showing the proximity of the models's output to the real data, establishing their close emulation.The models were trained for 5 Federated iterations and 30 local epochs while taking 64 sample batches. The federated models seem to converge faster than the centralized ones.

### D. SCENT protocol Evaluation

Concerning the robustness and security of the proposed scheme, SCENT is a rather strict protocol as it depends on the Modbus protocol as a channel for communication. When not participating in a FL session, the reserved addresses are used as passive decoys. When federating, most to all unauthorised actions would lead to the pipeline breaking and the FL process stopping. Then, the Failure status is written in the control registers and the state of the control slave is reverted to idle awaiting for the next FL request, Figure 2. All the transaction and states of the SCENT protocol are logged by the hopeypot and can subsequently audited. On it network performance, to send a weight vector of

size $n$ and considering eq. 3, using $r$ registers then there would be at least $2n/r$ transactions to archive the weight transference. Since weights are multi-layer we face thousands of parameters x2 to be send only using 125 addresses, which is not very optimal. On the other side, even with its criticallity level, the protocol showed only around a $10\%$ of failures in consecutive weight transmissions. Forgoing its performance shortcomings, SCENT completely camouflaged the FL process as no other protocol trace other than Modbus was shown in network captures parallel to the process.

## V. Conclusion

This paper extends a classic Honeypot based on Conpot that utilizes DNNs to simulate devices running the Modbus protocol, to train its DNNs in a Federated way. To that end, first a new updated Deep Network is designed, in particular, an LSTM-Autoencoder, that shows its ability to learn both the data of the modbus devices and their temporal sequencing and is tested in a Federated environment. To complement the training and fully camouflage it from possibly "clever" intruders, SCENT, a new custom FL training protocol is proposed, that depends on the Modbus protocol. As explained, using this protocol the FL training network transactions are fully veiled into seemingly normal modbus traffic, leaving no trace of the actual distributed communication of the FL environment with the honeypots, adding to their concept of "Just" industrial equipment.

## VI. Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 957406 (TERMINET).

## References

[1] S. Bera, S. Misra, S. K. Roy, and M. S. Obaidat, "Soft-wsn: Software-defined wsn management system for iot applications," *IEEE Systems Journal*, vol. 12, no. 3, pp. 2074–2081, 2018.

[2] N. Naik, P. Jenkins, R. Cooke, and L. Yang, "Honeypots that bite back: A fuzzy technique for identifying and inhibiting fingerprinting attacks on low interaction honeypots," in *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2018, pp. 1–8.

[3] P. Huitsing, R. Chandia, M. Papa, and S. Shenoi, "Attack taxonomies for the modbus protocols," *International Journal of Critical Infrastructure Protection*, vol. 1, pp. 37–44, 12 2008.

[4] A. Jicha, M. Patton, and H. Chen, "Scada honeypots: An in-depth analysis of conpot," in *2016 IEEE conference on intelligence and security informatics (ISI)*. IEEE, 2016, pp. 196–198.

[5] I. Siniosoglou, G. Efstathopoulos, D. Pliatsios, I. D. Moscholios, A. Sarigiannidis, G. Sakellari, G. Loukas, and P. Sarigiannidis, "Neuralpot: An industrial honeypot implementation based on deep neural networks," in *2020 IEEE Symposium on Computers and Communications (ISCC)*, 2020, pp. 1–7.

[6] S. Suratkar, K. Shah, A. Sood, A. Loya, D. Bisure, U. Patil, and F. Kazi, "An adaptive honeypot using q-learning with severity analyzer," *Journal of Ambient Intelligence and Humanized Computing*, 04 2021.

[7] W. Cabral, C. Valli, L. Sikos, and S. Wakeling, "Review and analysis of cowrie artefacts and their potential to be used deceptively," in *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2019, pp. 166–171.

[8] M. Tiwari, S. Misra, P. K. Bishoyi, and L. T. Yang, "Devote: Criticality-aware federated service provisioning in fog-based iot environments," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10 631–10 638, 2021.

[9] S. Misra, S. Chatterjee, and M. S. Obaidat, "On theoretical modeling of sensor cloud: A paradigm shift from wireless sensor network," *IEEE Systems Journal*, vol. 11, no. 2, pp. 1084–1093, 2017.

[10] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," ser. Proceedings of Machine Learning Research, 2017, pp. 1273–1282.

[11] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y. C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[12] C. Zhou and R. Paffenroth, "Anomaly detection with robust deep autoencoders," 08 2017, pp. 665–674.

[13] H.-D. Nguyen, K. P. TRAN, S. Thomassey, and M. Hamad, "Forecasting and anomaly detection approaches using lstm and lstm autoencoder techniques with the applications in supply chain management," *International Journal of Information Management*, 11 2020.

[14] A. Diro and N. Chilamkurti, "Leveraging lstm networks for attack detection in fog-to-things communications," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 124–130, 2018.

[15] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.