This paper is published in IEEE Xplore: https://ieeexplore.ieee.org/abstract/document/9527936 - P. Radoglou-Grammatikis et al., "TRUSTY: A Solution for Threat Hunting Using Data Analysis in Critical Infrastructures," 2021 IEEE International Conference on Cyber Security and Resilience (CSR), 2021, pp. 485-490, doi: 10.1109/CSR51186.2021.9527936.

TRUSTY: A Solution for Threat Hunting Using Data Analysis in Critical Infrastructures

Panagiotis Radoglou-Grammatikis
University of Western Macedonia
Kozani, Greece
pradoglou@uowm.gr

Athanasios Liatifis
University of Western Macedonia
Kozani, Greece
aliatifis@uowm.gr

Elisavet Grigoriou
Sidroco Holdings Ltd
Nicosia, Cyprus
egrigoriou@sidroco.com

Theocharis Saoulidis
Sidroco Holdings Ltd
Nicosia, Cyprus
hsaoulidis@sidroco.com

Antonios Sarigiannidis
Sidroco Holdings Ltd
Nicosia, Cyprus
asarigia@sidroco.com

Thomas Lagkas
International Hellenic University
Kavala, Greece
tlagkas@cs.ihu.gr

Panagiotis Sarigiannidis
University of Western Macedonia
Kozani, Greece
psarigiannidis@uowm.gr

***Abstract*—The rise of the Industrial Internet of Things (IIoT) plays a crucial role in the era of hyper-connected digital economies. Despite the valuable benefits, such as increased resiliency, self-monitoring and pervasive control, IIoT raises severe cybersecurity and privacy risks, allowing cyberattackers to exploit a plethora of vulnerabilities and weaknesses that can lead to disastrous consequences. Although the Intrusion Detection and Prevention Systems (IDPS) constitute valuable solutions, they suffer from several gaps, such as zero-day attacks, unknown anomalies and false positives. Therefore, the presence of supporting mechanisms is necessary. To this end, honeypots can protect real assets and trap cyberattackers. In this paper, we provide a web-based platform called $TRUSTY$, which is capable of aggregating, storing and analysing the detection results of multiple industrial honeypots related to Modbus/Transmission Control Protocol (TCP), IEC 60870-5-104, BACnet, Message Queuing Telemetry Transport (MQTT) and EtherNet/IP. Based on this analysis, we provide a dataset related to honeypot security events. Moreover, this paper provides a Reinforcement Learning (RL) method, which decides about the number of honeypots that can be deployed in an industrial environment in a strategic way. In particular, this decision is converted into a Multi-Armed Bandit (MAB), which is solved with the *e-Greedy* method. The evaluation analysis demonstrates the efficiency of the proposed method.**

***Index Terms*—Cybersecurity, Dataset, Honeypot, Industrial Internet of Things, Multi-Armed Bandit, Reinforcement Learning, Thompson Sampling**

I. Introduction

In the age of the Industrial Internet of Things (IIoT), the smart technologies are not only an extension of the Critical Infrastructure (CIs) but play an indispensable role at the core of each automation process. Undoubtedly, IIoT offers multiple advantageous services in the industrial sector, such as pervasive control, self-monitoring and better utilisation of the existing resources. However, it also raises severe cybersecurity concerns, exposing CIs to new risks [1]. In particular, a cybersecurity incident against a CI can lead to disastrous consequences and economic losses, affecting public health. Characteristic examples were the Advanced Persistent Threats (APTs) Stuxnet, Duqu, Flame and Gauss [2].

Several industrial protocols, such as Modbus, IEC 60870-5-104 and BACnet, do not include essential authentication and access control measures. Hence, they are prone to a plethora of cyberattacks, including Denial of Service (DoS), unauthorised access activities and False Data Injection (FDI). Consequently, the presence of appropriate cybersecurity detection mechanisms is necessary. Typical Intrusion Detection and Prevention Systems (IDPS) utilise known signature and specification rules in order to detect possible security violations. For example, both `Snort` and `Suricata` include an appropriate language for constructing such signature/specification rules related to Modbus and the Distributed Network Protocol 3 (DNP3). However, signature/specification-based techniques can detect only known malicious patterns or generic anomalies without distinguishing details about the malicious activities. On the other side, anomaly-based detection methods are prone to False Positives (FP). Therefore, such detection mechanisms should be enhanced with other countermeasures. To this end, a honeypot is a fictitious security hole, which aims to mislead potential cyberattackers. In particular, a honeypot can hide the real assets and secondly, can gather significant information about the malicious activities.

In this paper, we focus our attention on (a) strategic honeypot deployment in an IIoT environment and (b) statistical analysis of research honeypots' data. First, we model the optimal number of honeypots that can support the security measures of a CI as a Multi-Armed Bandit (MAB) problem. The proposed MAB problem is solved via a Reinforcement Learning (RL) method called *e-Greedy*. takes into account both the potential costs and benefits of the defender and the attacker. Based on the number of the honeypots' security events (i.e., honeypots' logs captured when a cyberattacker interacts with them), converges to the optimal number of honeypots that can be deployed. Secondly, this paper introduces also `TRUSTY (Threat hunting using Data Analysis)`, a platform, which analyses automatically the honeypots' detection results. Based on the aforementioned remarks, the contribution of our paper is summarised in the following points:

- **Strategic Honeypot Deployment**: We model the honeypots' deployment in a CI as a MAB problem, which is solved via , considering the costs and the benefits of the defender and the potential cyberattackers.
- **Web-based Honeypot Data Analysis Platform**: We provide a web-based honeypot data analysis platform called TRUSTY, capable of analysing the honeypots' detection results. It provides geolocation information, network layer information (e.g., network flows), application-layer information (e.g., function codes, unit identifiers) and risk estimation per network flow.
- **Providing a Honeypot Dataset**: Through this work, a honeypot dataset is provided publicly, including the network traffic and logs from multiple industrial honeypots, such as Conpot and Dionaea. This dataset can be utilised for intrusion detection processes, comprising network flows statistics related to Modbus/Transmission Control Protocol (TCP), IEC 60870-5-104, BACnet, Message Queuing Telemetry Transport (MQTT) and EtherNet/IP.

The rest of this paper is organised as follows: Section II presents a background and relevant works. Section III analyses our strategic RL-based method regarding the honeypots' deployment. Section IV presents the architecture of $TRUSTY$. Section V discusses the evaluation analysis of this work. Finally, section VI concludes this paper.

II. Background and Related Work

This section provides a background on honeypots and summarises relevant honeypot implementations, relevant tools and projects.

A. Honeypot and Honeypot categories

Honeypots are assets with no production value that imitate real assets' behaviour, aiming to protect them and collect valuable information about the cyber-attackers. In particular, honeypots can be classified into two categories: (a) production honeypots and (b) research honeypots. Production honeypots are placed into an organisation's production network, trying to hide the real assets from potential malicious insiders. On the other side, research honeypots are exposed to public networks like the Internet, attracting potential cyber-attackers and collecting important information related to them. It is noteworthy that any interaction with a honeypot is considered suspicious since legitimate users do not have any reason to interact with it. Moreover, honeypots can be classified based on the interaction level as (a) Low-Interaction Honeypots (LIH) and (b) High-Interaction Honeypots (HIH). LIHP simulates a set of network services like File Transfer Protocol (FTP), Secure Shell (SSH) and Modbus realistically, without successfully implementing them. On the other hand, HIHP provides an overall operating system with pre-installed services.

B. Honeypots Implementations

Both academia and industry have implemented several honeypots. In particular, Deception Toolkit (DTK) [3] was the first honeypot released in 1997, emulating known vulnerabilities of UNIX. HoneyBOT [4] is a LIH for Windows operating systems, simulating relevant vulnerabilities. Similarly, KFSensor [5] is a commercial honeypot for Windows systems which also incorporates Snort. HoneyD [6] is probably the most known honeypot capable of emulating at the same time multiple hosts. Tiny Honeypot [7] is a server-based honeypot, which listens to all TCP ports, logging all interaction activities. Dionaea [8] is written in Python and emulates the MQTT protocol. Jackpot [7] is related to Simple Mail Transfer Protocol (SMTP) and aims to combat email spam. Cowrie [9] is a LIH emulating SSH. Conpot [10] is an industrial honeypot emulating multiple relevant protocols like Modbus and IEC 60870-5-104 [11].

C. Honeypot-related Tools

Many supporting tools have been developed in order to analyse the data retrieved from honeypots or to extend their functionalities [12]. In particular, Bait-n-Switch [13] aims to redirect all malicious traffic to a honeypot. Accordingly, Honeynet Security Console (HSC) [14] analyses, correlates and visualises honeypots logs. Honeysnap [15] processes Packet Capture (PCAP) files that were collected by server-based honeypots. GSOC-Honeyweb [7] is devoted to the management of client-based honeypots via a user-friendly environment. Moreover, TraCINg [7] aggregates data from multiple honeypots and correlates this information in order to discover possible worms.

D. Honeypot-related Projects

It is noteworthy that many honeypots projects have been organised, aiming to exploit at the maximum level the benefits of honeypots and mainly to discover possible zero-day attacks. In particular, the Honeynet Project [16] was started in 1999 to explore and investigate zero-day cyberattacks. Furthermore, the Leurre.com project [17] deployed multiple LIHs in more than 30 counties, aiming at collecting quantitative data related to cyberthreats and vulnerabilities. Accordingly, NoAH-Project coordinated by Foundation for Research and Technology Hellas (FORTH) deployed an HIH called Argos [18] to enhance the protection of Internet Service Providers (ISPs) and investigate potential zero-day attacks. The mw-collect Alliance project collected information about various malware by deploying multiple Nepenthes sensors [7]. Moreover, Telekom-Fruhwarnsystem [7] was started in 2013 to collect various datasets related to honeypot activities. Finally, H2020 SPEAR implemented various industrial honeypots for the smart electrical grid [19].

III. Strategic Honeypot Deployment: A Multi Armed Badnit Problem

The first functionality of $TRUSTY$ is the strategic honeypots' deployment as a MAB problem. First, we consider two antagonistic players: (a) $Attacker$ and (b) $Defender$. The goal of the $Attacker$ is to attack the real EPES assets,

while the $Defender$ aims to deploy the appropriate number of honeypots that will provide the maximum protection, taking into account the available computing resources and the behaviour of the $Attacker$. Let N_{max} be the maximum number of honeypots and real EPES assets that can be hosted in an EPES infrastructure. N_{max} is defined by $TRUSTY$. Moreover, let $N \leq N_{max}$ be the total number of the connected machines that can serve either as honeypots or real EPES assets. $TRUSTY$ is also able to define which machines will be used by honeypots. The ratio of N utilised by honeypots is symbolised by θ. $s_{A,i} \in \{0, 1\}$ represents the strategy of the $Attacker$, meaning to attack or not machine i. $s_{A,i}$ equals 1 when the cyberattack is performed against an actual EPES asset, while $s_{A,i}$ is equal to 0 when the cyberattack targets a honeypot. On the other side, $s_{D,i} \in \{-1, 1\}$ represents the strategy of the $Defender$. $s_{D,i}$ equals -1 and 1 when the cyberattack targets a real asset and a honeypot, respectively. Both strategies are characterised by some weights. a_1 denotes the benefit of the $Attacker$ for each attack against a honeypot. a_2 and a_3 denote the cost of the $Attacker$ for each attack against a honeypot and any machine (honeypot or real device), respectively. Similarly, d_1 defines the benefit related to the $Defender$ for each attack against a honeypot, while d_2 and d_3 imply the cost for each attack against a real device and the cost for each real device that is replaced by a honeypot. Finally, d_4 denotes the cost of the $Defender$ since N increases. For the sake of clarity, Table I summarises the notation.

TABLE I: Notation

Notation	Explanation
N_{max}	The maximum number of the real EPES assets and honeypots that can be simultaneously connected.
N	The number of the real EPES assets and honeypots that are connected.
a_1	The benefit of the attacker for each attack against a real EPES asset.
a_2	The cost of the attacker for each attack against a honeypot.
a_3	The cost of the attacker for each attack against any machine (honeypot or not).
d_1	The benefit of the defender for each attack against a honeypot.
d_2	The cost of the defender for each attack against a real EPES asset.
d_3	The cost of the defender for each real EPES asset which is replaced by a honeypot.
d_4	The cost of the defender as N increases.
$U_A[t]$	The utility of the $Attacker$ at the time interval t.
$U_D[t]$	The utility of the $Defender$ at the time interval t.
θ	The ratio of N utilised by honeypots.

Therefore, based on the aforementioned remarks, the utility function of the $Attacker$ in a time interval t ($U_A[t]$) is given by Equation 1. In particular, $U_{A[t]}$ increases based on $\sum_{i=1}^{N} \frac{1+S_{D,i}}{2} s_{A,i}$ and decreases according to $\sum_{i=1}^{N} \frac{1-S_{D,i}}{2} s_{A,i}$ and $\sum_{i=1}^{N} s_{A,i}$. $\sum_{i=1}^{N} \frac{1+S_{D,i}}{2} s_{A,i}$ refers to the overall benefit of the $Attacker$ in terms of the real EPES assets attacked. In contrast, $\sum_{i=1}^{N} \frac{1-S_{D,i}}{2} s_{A,i}$ and $\sum_{i=1}^{N} \frac{1+S_{D,i}}{2} s_{A,i}$ denote the overall damage of the $Attacker$ in terms of the honeypots attacked and the entire number of the cyberattacks. The more cyber attacks the more possibilities to reveal the identity of the $Attacker$.

$$U_A[t] = f(a_{i\in\{1,2,3\}}, \sum_{i=1}^{N} \frac{1+S_{D,i}}{2} s_{A,i}, \sum_{i=1}^{N} \frac{1-S_{D,i}}{2} s_{A,i}, \sum_{i=1}^{N} s_{A,i}) \tag{1}$$

Supposing the previous terms progress linearly, Equation 1 can be expressed with Equation 2.

$$U_A[t] = a_1 \sum_{i=1}^{N} \frac{1+S_{D,i}}{2} s_{A,i} - a_2 \sum_{i=1}^{N} \frac{1-S_{D,i}}{2} s_{A,i} - a_3 \sum_{i=1}^{N} s_{A,i}) \tag{2}$$

In a similar manner, the utility function of the Defender in a time interval t i.e., ($U_D[t]$) is given by Equation 3. Equation 3 increases in terms of $\sum_{i=1}^{N} \frac{1-S_{D,i}}{2} s_{A,i}$ and decreases in terms of $\sum_{i=1}^{N} \frac{1+S_{D,i}}{2} s_{A,i}$, $\sum_{i=1}^{N}$ and N.

$$U_D[t] = g(d_{i\in\{1,2,34\}}, \sum_{i=1}^{N} \frac{1-S_{D,i}}{2} s_{A,i}, \sum_{i=1}^{N} \frac{1+S_{D,i}}{2} s_{A,i}, \sum_{i=1}^{N} \frac{1+s_{D,i}}{2}, N) \tag{3}$$

If we assume that the terms of Equation 3 progress linearly, the Equation 3 can be written in the form of Equation 4.

$$U_D[t] = d_1 \sum_{i=1}^{N} \frac{1-S_{D,i}}{2} s_{A,i} - d_2 \sum_{i=1}^{N} \frac{1+S_{D,i}}{2} s_{A,i} - d_3 \sum_{i=1}^{N} \frac{1+s_{D,i}}{2} - d_4 N \tag{4}$$

Therefore, based on the security events received by the Suricata our goal is to set the appropriate ratio θ in order to maximise $U_D[t]$ each time (Equation 4). The previous modelling relies on our previous work in [20]. To re-define, the appropriate number of θ for each security event in the time interval t can be expressed as a MAB problem, where exploitation intends to maximise $U_D[t]$ (Equation 5) and exploration aims to test different values of θ to discover more information for the $Attacker$ in terms of Equation 4. $TRUSTY$ plays the role of the gambler and the various values of $theta$ represent the slot machines. To solve the MAB problem, we adopt the $e - Greedy$ method, where we commonly select that mean of $theta$ providing the maximum value $U_D[t]$ (Equation 5) and there is a small probability e where other values of θ are selected in order to discover how Equation 4 ranges. Algorithm 1 reflects how $TRUSTY$ decides to deploy θ honeypots, utilising $e - Greedy$.

$$max(U_D[t]) = max(d_1 \sum_{i=1}^{N} \frac{1 - S_{D,i}}{2} s_{A,i} - d_2 \sum_{i=1}^{N} \frac{1 + S_{D,i}}{2} s_{A,i} - d_3 \sum_{i=1}^{N} \frac{1 + s_{D,i}}{2} - d_4 N) \quad (5)$$

Algorithm 1: TRUSTY Honeypot Deployment

Data: N_{max}, N, UD_Matrix, sum_θ_Matrix, $mean\_\theta\_Matrix$, max_mean, securityEventCounter, a_1, a_2, a_3, d_1, d_2, d_3, d_4

Result: $\theta_{selected}$

$size\_\theta\_Matrix = []$, $UD\_Matrix = []$, $sum\_\theta\_Matrix = []$, $mean\_\theta\_Matrix = []$, securityEventCounter = 0, max_mean = 0, $\theta_{selected}$ = 0, a_1, a_2, a_3, d_1, d_2, d_3, d_4 = init();

while *True* **do**

  Receive a security event;

  securityEventCounter = securityEventCounter +1;

  max_mean = 0;

  p = random number in [0,1];

  if $p < e$ **then**

    $\theta_{selected}$ = random integer number in [1, N];

    $UD\_Matrix[\theta] = d_1 \sum_{i=1}^{N} \frac{1-S_{D,i}}{2} s_{A,i} - d_2 \sum_{i=1}^{N} \frac{1+S_{D,i}}{2} s_{A,i} - d_3 \sum_{i=1}^{N} \frac{1+s_{D,i}}{2} - d_4 N$;

    $sum\_\theta\_Matrix[\theta] = sum\_\theta\_Matrix[\theta] + UD\_Matrix[\theta]$;

    $mean\_\theta\_Matrix = sum\_\theta\_Matrix[\theta]$ / securityEventCounter;

  end

  else

    for $\theta \leftarrow 1$ **to** N **by** 1 **do**

      $UD\_Matrix[\theta] = d_1 \sum_{i=1}^{N} \frac{1-S_{D,i}}{2} s_{A,i} - d_2 \sum_{i=1}^{N} \frac{1+S_{D,i}}{2} s_{A,i} - d_3 \sum_{i=1}^{N} \frac{1+s_{D,i}}{2} - d_4 N$;

      $sum\_\theta\_Matrix[\theta] = sum\_\theta\_Matrix[\theta] + UD\_Matrix[\theta]$;

      $mean\_\theta\_Matrix = sum\_\theta\_Matrix[\theta]$ / securityEventCounter;

      if $mean\_\theta\_Matrix[\theta] > max\_mean$ **then**

        $max\_mean = mean\_\theta\_Matrix[\theta]$;

        $\theta_{selected} = \theta$;

      end

    end

  end

end

IV. TRUSTY: A WEB-BASED DATA ANALYSIS PLATFORM

This section is devoted to the *TRUSTY* architecture. As illustrated in Fig. 1, *TRUSTY* consists of two main components, namely (a) Honeypot Sensors and (b) Honeypot Analyser Server. Honeypot Sensors represent honeypots, while the Honeypot Analyser Server collects and analyses their outcomes. The following subsections provide more information about each component.

A. Honeypot Sensors

Honeypot Sensors are Virtual Machines (VMs) hosting the honeypot applications: (a) `Conpot` and (b) `Dionaea` as well as external tools like `Tshark`, `CICFlowmeter`, `Scapy` and `Suricata` to capture network traffic data periodically, the network flows and the Suricata logs, respectively. In particular, `Conpot` is an industrial honeypot emulating multiple industrial protocols: Modbus, IEC 60870-5-104, BACnet and EtherNet/IP. `Dionaea` is another popular honeypot that offers an easy to use Python Application Programming Interface (API) and emulates many protocols including MQTT. The network traffic related to honeypot applications is captured using `hark`. For extracting the network flow statistics, `CICFlowMeter`. `Suricata` is applied in order to identify potential threats based on the pcap files captured by `Tshark`. `Scapy` is a network packet manipulation tool, which allows to process the network packets captured by `Tshark`, extracting relevant statistics based on the application-layer protocols. Finally, the Honeypot Sensors calculate the risk related to their data, by applying Equation 6 [7].

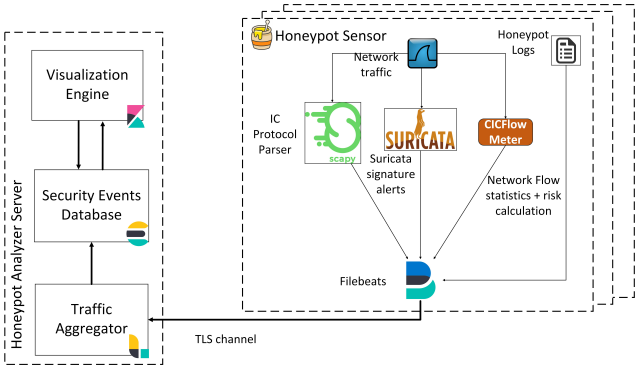

Fig. 1: TRUSTY architecture

$$Risk = log(nPackets) + log(nBytes) + log(duration + 1) \quad (6)$$

B. Description of Honeypot Analyser Server

The honeypot applications usually produces text-based entries with a complex structure. Consequently, the scalable processing and visualisation of the honeypot-related data are valuable capabilities for the security administrators. The Honeypot Analyzer Server is responsible for aggregating and processing the honeypot data. It consists of three modules: (a) the Traffic Aggregator, (b) the Security Events Database and (c) the Visualization Engine. The Traffic Aggregator relies on Logstash and is responsible for receiving data from multiple honeypots applications. Moreover, it performs a series of transformation of this data and forwards the processed information to the Security Event Database, which uses the

Elasticsearch database. The honeypot log entries are analysed, stored and indexed. Finally, the Visualization Engine utilises Kibana and is responsible for presenting the data through interactive visualisations in a web-based environment.

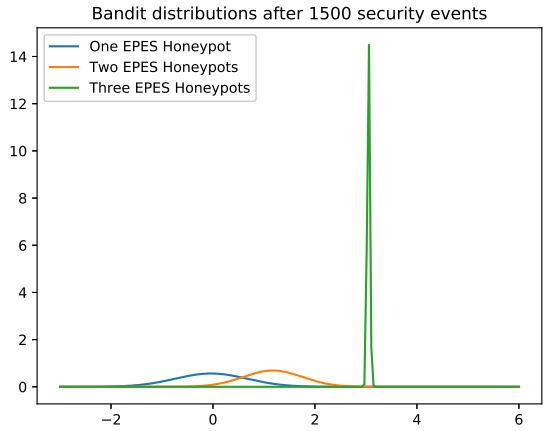

Fig. 2: Posterior Probability of $U_D[t]$ after 1500 honeypot-related security events

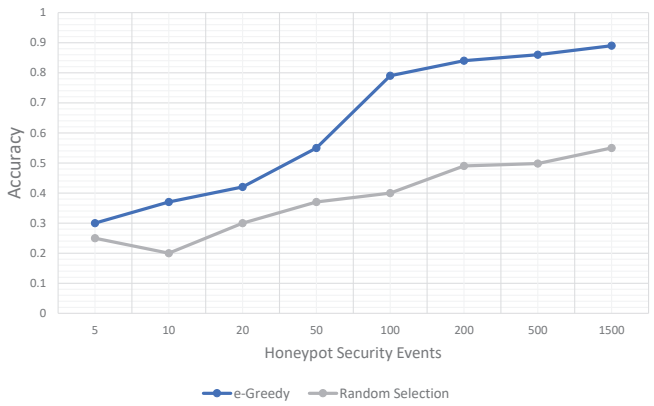

Fig. 3: e-Greedy accuracy compared to the accuracy of a random choice

V. Results and Discussion

This section focuses on the evaluation analysis related to the proposed method for the strategic honeypot deployment. To this end, we use a power plant simulation environment and a dataset with honeypot-related security events. The dataset was constructed by deploying $TRUSTY$ with five Honeypot Sensors during one year period. The honeypot applications emulate industrial devices that use (a) Modbus/TCP, (b) IEC 60870-5-104, (c) MQTT, (d) BACnet and (e) EtherNet/IP, respectively. Thus, we created a dataset with honeypot-related security events, including pcap files, honeypots' logs and Comma-Separated Values (CSV) with the network flow statistcis. The dataset is provided publicly though this work. The following sub-sections describe the honeypot dataset and the evaluation results related to the proposed method for the strategic honeypot deployment.

A. Honeypot Dataset Analysis

By inspecting the map in figure 4, we can observe that countries such as the USA and Netherlands are popular choices for deploying bots. This is also validated by the alerts generated by `Suricata` on each honeypot sensor. Countries such as Russia and China are ranked third and fourth respectively. The majority of the network flows was short TCP sessions. This is a strong indication related to reconnaissance cyberattacks. Table II lists the application-layer data collected for each industrial protocol mentioned above. It is worth mentioning that a considerable portion of the network traffic data (85%) originate from the popular search engine `Shodan`. Using reverse Domain Name System (DNS) queries, $TRUSTY$ is able to identify sub-domains of the `shodan.io`.

B. Strategic Honeypot Deployment

In order to evaluate the strategic honeypot deployment process, we use a power plant simulation environment where a_1, a_2, a_3, d_1, d_2, d_3 and d_4 are defined experimentally by security experts and electrical operators. The simulation ran in an Ubuntu 18.04. Long-Term Support (L) computing system with (a) Central Processing Unit (CPU): Intel Core i7-6700 at 3.40 GHz, (b) Random Access Memory (RAM): 16 GB and (c) Solid State Drive (SSD): 240 GB. Furthermore, regarding the advent of honeypot-related security events, we utilise the honeypot dataset provided through this paper. First, we show how the Probability Density Function (PDF) of $U_D[t]$ ranges. Based on the available computing resources, we consider the deployment of one, two and three EPES honeypots. Moreover, e was defined to 0.1. Fig. 2 shows how PDF of $U_D[t]$ ranges for 1500 honeypot-related security events. We can observe that the proposed method converges briefly to the deployment of three EPES honeypots. Finally, Fig. 3 shows the accuracy of the proposed e-Greedy technique for the deploying the EPES honeypots compared to a random selection. The accuracy of e-Greedy method reaches 0.89 after 1500 honeypot-related security events, while the accuracy of the random choice reaches 0.55.

TABLE II: Industrial Protocol Data collected from the Analysis Process

Protocol	**Collected Data**
MQTT	topic, QOS
Modbus	Function code, Unit ID, length, Type
EnIP	command ID, length, session, status, sender, context, start, ADPU length, testfr_connection, testfr_action,
options IEC104	stopdt_con, stopdt_act, startdt_con, startdt_act, octet_1_1_2 octet_2, octet_3

VI. Conclusions

The rise of IoT offers multiple benefits and raises new cybersecurity risks that require appropriate intrusion detection and prevention mechanisms. Although the typical IDPS solutions can detect and mitigate a plethora of known cyberthreats, they face critical challenges, such as zero-day attacks, unknown anomalies, FP and multi-step attack scenarios. To this end, they should be enhanced with additional supportive

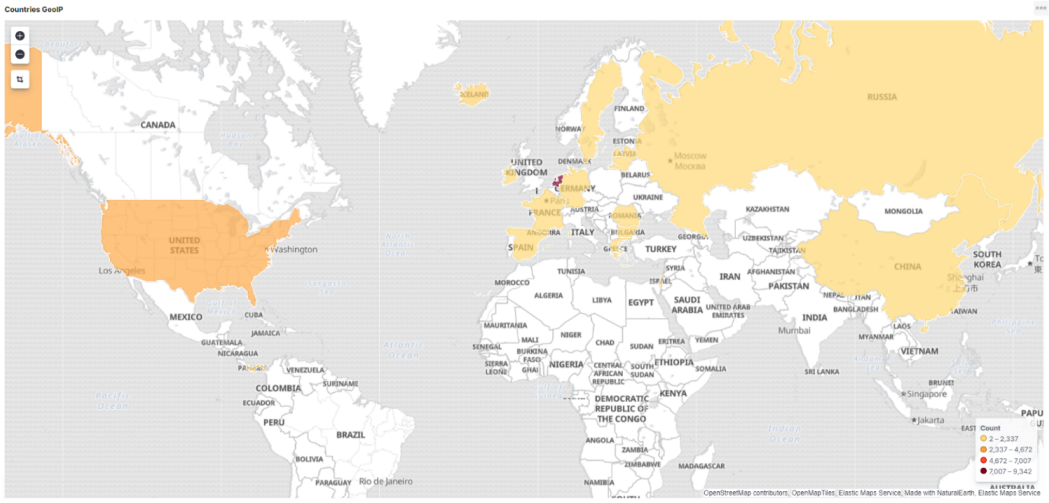

Fig. 4: Honeypot Geo-location information

mechanisms. Honeypots consitute an emerging technology that can trap potential cyberattackers and gather valuable information about their malicios activities. In this paper, we focus our attention on a strategic and dynamic way for deploying honeypots in an industrial environment, taking into account the costs and benefits of the defender and the cyberattacker. Moreover, we provide a platform called $TRUSTY$ which comprises industrial honeypot applications and analyses their detection outcomes in terms of network traffic data, network flow statistics and honeypots' logs. Through $TRUSTY$, we created a dataset with honeypot-related security events, including the information mentioned earlier. This dataset is provided publicly via this work. Finally, the evaluation results demonstrate the efficiency of our work.

Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 833955.

References

[1] P. Radoglou-Grammatikis, P. G. Sarigiannidis, and I. D. Moscholios, "Securing the internet of things: challenges, threats and solutions," *Internet of Things*, vol. 5, pp. 41–70, 2019.

[2] P. I. Radoglou-Grammatikis and P. G. Sarigiannidis, "Securing the smart grid: A comprehensive compilation of intrusion detection and prevention systems," *IEEE Access*, vol. 7, pp. 46 595–46 620, 2019.

[3] M. H. Almeshekah and E. H. Spafford, "Cyber security deception," in *Cyber deception*. Springer, 2016, pp. 23–50.

[4] C. Irvene, D. Formby, S. Litchfield, and R. Beyah, "Honeybot: A honeypot for robotic systems," *Proceedings of the IEEE*, vol. 106, no. 1, pp. 61–70, 2017.

[5] D. Kumar and V. Vashishtha, "Snort based h-ids with kf sensor and weka," *International Journal*, vol. 2, no. 5, 2012.

[6] M. Tsikerdekis, S. Zeadally, A. Schlesener, and N. Sklavos, "Approaches for preventing honeypot detection and compromise," in *2018 Global Information Infrastructure and Networking Symposium (GIIS)*. IEEE, 2018, pp. 1–6.

[7] M. Nawrocki, M. Wählisch, T. C. Schmidt, C. Keil, and J. Schönfelder, "A survey on honeypot software and data analysis," *CoRR*, vol. abs/1608.06249, 2016. [Online]. Available: http://arxiv.org/abs/1608.06249

[8] V. Sethia and A. Jeyasekar, "Malware capturing and analysis using dionaea honeypot," in *2019 International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2019, pp. 1–4.

[9] R. K. Shrivastava, B. Bashir, and C. Hota, "Attack detection and forensics using honeypot in iot environment," in *International Conference on Distributed Computing and Internet Technology*. Springer, 2019, pp. 402–409.

[10] S. Gokhale, A. Dalvi, and I. Siddavatam, "Industrial control systems honeypot: A formal analysis of conpot." *International Journal of Computer Network & Information Security*, vol. 12, no. 6, 2020.

[11] P. Radoglou-Grammatikis, P. Sarigiannidis, I. Giannoulakis, E. Kafetzakis, and E. Panaousis, "Attacking iec-60870-5-104 scada systems," in *2019 IEEE World Congress on Services (SERVICES)*, vol. 2642. IEEE, 2019, pp. 41–46.

[12] I. Koniaris, G. Papadimitriou, and P. Nicopolitidis, "Analysis and visualization of ssh attacks using honeypots," in *Eurocon 2013*. IEEE, 2013, pp. 65–72.

[13] A. Tiwari and D. Kumar, "Comparitive study of various honeypot tools on the basis of their classification & features," *Available at SSRN 3565078*, 2020.

[14] S. Manchekar, M. Kadam, and K. Jamdaade, "Application of honeypot in cloud security: A review," *International Journal on Future Revolution in Computer Science & Communication Engineering*, vol. 4, no. 6, pp. 63–65, 2018.

[15] A. K. Seewald and W. N. Gansterer, "On the detection and identification of botnets," *Computers & Security*, vol. 29, no. 1, pp. 45–58, 2010.

[16] W. Zhang, B. Zhang, Y. Zhou, H. He, and Z. Ding, "An iot honeynet based on multiport honeypots for capturing iot attacks," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 3991–3999, 2019.

[17] C. Leita, V. Pham, O. Thonnard, E. Ramirez-Silva, F. Pouget, E. Kirda, and M. Dacier, "The leurre.com project: Collecting internet threats information using a worldwide distributed honeynet," *2008 WOMBAT Workshop on Information Security Threats Data Collection and Sharing*, 2008.

[18] G. Portokalidis, A. Slowinska, and H. Bos, "Argos," *Proceedings of the 2006 EuroSys conference on - EuroSys 06*, 2006.

[19] P. Radoglou-Grammatikis, P. Sarigiannidis, E. Iturbe, E. Rios, A. Sarigiannidis, O. Nikolis, D. Ioannidis, V. Machamint, M. Tzifas, A. Giannakoulias *et al.*, "Secure and private smart grid: The spear architecture," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2020, pp. 450–456.

[20] P. Diamantoulakis, C. Dalamagkas, P. Radoglou-Grammatikis, P. Sarigiannidis, and G. Karagiannidis, "Game theoretic honeypot deployment in smart grid," *Sensors*, vol. 20, no. 15, p. 4199, 2020.