A Survey of Honeypots and Honeynets for Internet of Things, Industrial Internet of Things, and Cyber-Physical Systems

Javier Franco, Ahmet Aris, Berk Canberk, *Senior Member, IEEE*, and A. Selcuk Uluagac

***Abstract*—The Internet of Things (IoT), the Industrial Internet of Things (IIoT), and Cyber-Physical Systems (CPS) have become essential for our daily lives in contexts such as our homes, buildings, cities, health, transportation, manufacturing, infrastructure, and agriculture. However, they have become popular targets of attacks, due to their inherent limitations which create vulnerabilities. Honeypots and honeynets can prove essential to understand and defend against attacks on IoT, IIoT, and CPS environments by attracting attackers and deceiving them into thinking that they have gained access to the real systems. Honeypots and honeynets can complement other security solutions (i.e., firewalls, Intrusion Detection Systems - IDS) to form a strong defense against malicious entities. This paper provides a comprehensive survey of the research that has been carried out on honeypots and honeynets for IoT, IIoT, and CPS. It provides a taxonomy and extensive analysis of the existing honeypots and honeynets, states key design factors for the state-of-the-art honeypot/honeynet research and outlines open issues for future honeypots and honeynets for IoT, IIoT, and CPS environments.**

***Index Terms*—Honeypot, honeynet, IoT, IIoT, CPS.**

I. Introduction

THE Internet of Things (IoT) is a network of Internet-connected devices, such as sensors, actuators, and other embedded devices that are able to collect data and communicate. Industrial IoT (IIoT) is the application of IoT to automation applications using industrial communication technologies [1]. Cyber-Physical Systems (CPS) on the other hand, are networks of devices such as sensors, actuators, Programmable Logic Controllers (PLCs), Remote Terminal Units (RTUs), Intelligent Electronic Devices (IEDs), and other embedded devices that monitor and control physical processes in critical and non-critical application areas. CPS includes, but is not limited to Industrial Control Systems (ICS), Smart Grid and other smart infrastructures (e.g., water, gas, building automation), medical devices, and smart cars [2], [3]. As it can be seen from the descriptions of IoT, IIoT, and CPS, these concepts do not have explicit separation points. Bordel *et al.* [2] and the National Institute of Standards and Technology's (NIST) special report by Greer *et al.* [4] analyzed the definitions of IoT and CPS in the literature and indicated that these concepts are viewed either as the same, or different but they have overlapping parts, or they are subsets of each other. Greer *et al.* [4] pointed out that IoT and CPS are similar as they both connect the physical world of engineered systems and the logical world of communications and information technology. These two worlds are connected by sensors that collect data about the physical elements of a system and transmit it to the logical elements, and to the actuators that respond to the logical elements and apply changes to the physical elements. At the same time, however, Greer *et al.* [4] stated that IoT and CPS are different in that IoT places more emphasis on information technology and networking things in the physical world, while CPS is more of a closed system and is focused more on the exchange of information for sensing and controlling the physical world. IIoT further connects the definitions of IoT and CPS, as it possesses characteristics from both.

IoT, IIoT, and CPS are converting almost every aspect of life to smart in the 21st century. Sensors, actuators, wearables, embedded devices, and many other devices are becoming ubiquitous around the world with uses in diverse contexts such as homes, buildings, cities, health, transportation, automotive, manufacturing, critical (e.g., nuclear reactors, power plants, oil refineries) and non-critical infrastructures, and agriculture. While this promises connectivity and efficiency, the various devices in IoT, IIoT, and CPS environments have their unique properties in terms of resource limitations, network lifetimes, and application Quality-of-Service (QoS) requirements which affect the security of such applications crucially [5].

IoT devices typically have constrained power, storage, computing, and communications resources which limit the accommodation of good security mechanisms [6], [7]. On the other hand, devices used in IIoT and CPS were not initially designed with security in mind and they had been considered secure, as they were isolated. This security by obscurity assumption was broken by the uncovering of the Stuxnet (2010), DuQu (2011), and Flame (2012) attacks [8]. As an increasing number of industrial environments are being connected to the Internet, security updates and patches are becoming serious problems in decades-old industrial devices [8]–[11].

In order to protect IoT, IIoT, and CPS environments from malicious entities, traditional security mechanisms such as

Manuscript received September 22, 2020; revised March 27, 2021 and June 29, 2021; accepted August 1, 2021. Date of publication August 23, 2021; date of current version December 8, 2021. This work was supported in part by the U.S. National Science Foundation Awards under Grant NSF-CAREER-CNS-1453647 and Grant NSF-1663051. *(Corresponding author: Javier Franco.)*

Javier Franco, Ahmet Aris, and A. Selcuk Uluagac are with the Cyber-Physical Systems Security Lab, Florida International University, Miami, FL 33174 USA (e-mail: jfran243@fiu.edu; aaris@fiu.edu; suluagac@fiu.edu).

Berk Canberk is with the Department of Computer Engineering, Istanbul Technical University, 34469 Istanbul, Turkey (e-mail: canberk@itu.edu.tr).

Digital Object Identifier 10.1109/COMST.2021.3106669

cryptography, firewalls, Intrusion Detection and Prevention Systems (IDS, IPS), antivirus, and anti-malware solutions can be utilized. However, they do not transparently allow security researchers to observe and analyze how attackers perform attacks and find out their behaviors [12]. Honeypots and honeynets come to the scene as viable solutions at this point, as they can provide actionable intelligence on the attackers. A honeypot is a tool that is used with the purpose of being attacked and possibly compromised [13]. Two or more honeypots implemented on a system form a honeynet [14]. Honeypots are used to attract attackers and deceive them into thinking that they gained access to real systems. Honeypots can be integrated with firewalls and IDSs to form an IPS in order to capture all the information about attackers, study all of their actions, develop ways to improve system security and prevent attacks in the future [12].

Although there exist a number of honeypot and honeynet works on IoT, IIoT, or CPS, no study exists in the literature which considers all of the honeypot and honeynet models, analyzes their similarities and differences, and extracts key points in the design and implementation of honeypots and honeynets for IoT, IIoT, and CPS. In order to fill this important research gap, we propose our comprehensive survey on honeypot and honeynet models that have been proposed for IoT, IIoT, and CPS environments over the period 2002-2020. To the best of our knowledge, our work is the first study in the literature that surveys the current state-of-the-art honeypot and honeynet models not only for IoT, but also for IIoT and CPS.

Contributions: The contributions of our survey are as follows.

- Taxonomy of honeypots and honeynets proposed for IoT, IIoT, and CPS environments,
- Comprehensive analysis of IoT, IIoT, and CPS honeypots and honeynets, and intriguing characteristics that are shared by studies,
- Statement of the key design factors for future IoT, IIoT, and CPS honeypots and honeynets,
- Presentation of open research problems that still need to be addressed in honeypot and honeynet research for IoT, IIoT, and CPS.

Organization: The paper is organized as follows: Section II gives the related work. Section III provides background information on honeypots, honeynets, and related terms. Section IV provides a methodology for the classification of honeypot and honeynet characteristics. Section V classifies and presents diverse IoT honeypot and honeynet models and research. Section VI presents a taxonomy of the proposed IoT honeypot and honeynet models. Section VII classifies and presents diverse CPS and IIoT honeypot and honeynet models and research. Section VIII presents a taxonomy of the proposed CPS and IIoT honeypot and honeynet models. Section IX provides lessons learned and design considerations for honeypot and honeynet implementations. In Section X, conclusions and future work are presented.

II. Related Work

The security of IoT, IIoT, and CPS environments is a very broad field of research, and it is possible to find a myriad of studies. Without going into much detail, we refer the readers to the works of Butun *et al.* [15] and Makhdoom *et al.* [5] for extensive overviews of vulnerabilities, threats, and attacks, the security surveys of Lee *et al.* [16] on IoT standards and Granjal *et al.* [17] on the existing IoT protocols, the study of Neshenko *et al.* [7] for a recent comprehensive IoT security survey, the study of Sikder *et al.* [18] for a survey of threats to IoT sensors, the study of Humayed *et al.* [3] for an extensive survey on the threats, vulnerabilities, attacks, and defense solutions to CPS, the survey of Al-Garadi *et al.* [19] for machine and deep learning techniques for IoT security, the comprehensive survey of Yu *et al.* [10] for CPS security and Cintuglu *et al.* [20] for CPS testbeds. There are also studies like that of Babun *et al.* [21] which develop innovative ways to protect networks with vulnerable IoT devices.

The honeypot and honeynet research has been a very active field. In terms of general honeypots and honeynets that are not specific to IoT, IIoT, or CPS, Fan *et al.* [22] proposed criteria and a methodology for the classification of honeynet solutions and analyzed the advantages and disadvantages of each criterion used in their taxonomy. In 2018, Fan *et al.* [12] expanded on their earlier research and proposed a taxonomy of decoy systems with respect to decoys and captors. There also exist other survey studies on general honeypot and honeynet solutions, which include but are not limited to [23], [24], and [25]. In addition, privacy and liability issues when honeypots are deployed were analyzed by Sokol and Andrejko [26], Sokol *et al.* [27]. In terms of honeypots and honeynets for IoT, IIoT, and CPS, only a few surveys exist in the literature. Razali *et al.* [28] analyzed types, properties, and interaction levels of IoT honeypots and classified honeynet models based on interaction, resources, purpose, and role. Dalamagkas *et al.* [29] surveyed the honeypot and honeynet frameworks for smart-grid environments. Dowling *et al.* [30] proposed a framework for developing data-centric, adaptive smart city honeynets that focus on the key values of data complexity, security, and criticality. Furthermore, Neshenko *et al.* [7] discussed the IoT and CPS honeypots in their survey on IoT security. However, they did not provide a comprehensive survey on such honeypots since the focus of their study was on the security of IoT.

In addition to proposing novel honeypot/honeynet models or surveying the existing studies, there has been research on the development of honeynet description languages and also on the detectability of honeypots. Fan *et al.* [31] presented a technology-independent, flexible honeynet description language and a tool called HoneyGen for the deployment and modification of virtual honeynets based on the VNX and Honeyd platforms. Acien *et al.* [32] analyzed the steps and requirements to deploy honeypots in IoT environments effectively in a way that they can look like real devices to attackers. Surnin *et al.* [33] focused on techniques for honeypot detection with SSH and Telnet, identifying issues of software architecture and implementation that make honeypots easily detected [34]. Zamiri-Gourabi *et al.* [35] proposed a methodology to detect the ICS honeypots deployed on the Internet by means of fingerprinting methodologies.

Differences from the existing work: While the recent years have seen an increase in honeypot and honeynet research, our study is different because it is the first comprehensive study

that analyzes the existing honeypot and honeynet models and research for IoT, IIoT, and CPS environments holistically, provides a taxonomy of honeypots and honeynets and identifies key design considerations and open issues for honeypots and honeynets in IoT, IIoT, and CPS.

III. Background Information

In this section, we give some brief information on honeypots, honeynets, and other related terms.

A. Honeypots and Honeynets

A honeypot is a tool that serves as a decoy to attract attackers and deceive them into thinking that they have gained access to a real system. There exist various views of a honeynet: A honeynet can be defined simply as two or more honeypots implemented on a system [14], or in a more narrow definition, a honeynet is a high interaction honeypot system of Generation I, II, or III [36]. Although honeypots and honeynets are defined in the mentioned ways, it is interesting to note that very few authors refer to their honeypot system as a honeynet, despite their research implementing multiple honeypots. For instance, as it will be reviewed in the following sections, only a few honeypots ([37]–[41]) in the literature were implemented with a single honeypot. For this reason, we adhered to the statements of authors about their view of their systems as honeypots or honeynets while we are reviewing the studies in this survey.

Three main architectures/generations that are used in honeynets are described in [42]. *Generation I* was developed in 1999 and is composed of a firewall and an IDS, with honeypots behind these. Generation I can capture in-depth information and unknown attacks. However, Generation I honeynets can be easily detected by attackers. *Generation II* was developed in 2002 and had a honeynet sensor that serves the purpose of the IDS sensor and of the firewall used in Generation I. This sensor works like a bridge, so it is much more difficult for attackers to detect that they are in a honeynet. *Generation III* was developed in 2004 and had the same architecture as Generation II but has improved deployment and management capabilities.

Figure 1 depicts a basic honeynet architecture. There are three essential elements to any honeynet: *data control, data capture, and data collection*. Data control involves controlling the flow of data so that the attackers do not realize they are in a honeynet and making sure that if the honeynet is compromised, it will not be used to attack other systems. The data capture involves capturing all the data regarding movements and actions within the honeynet [36]. The data collection involves the ability to securely transfer all the captured data to a centralized place [22].

Honeypots and honeynets can be deployed at various locations. They can be deployed at cloud computing environments (e.g., Amazon EC2), Demilitarized Zones (DMZ) of enterprise networks, actual application/production environments (e.g., at an IoT, IIoT, or CPS network), and private deployment environments with public IP addresses. Each of these deployment

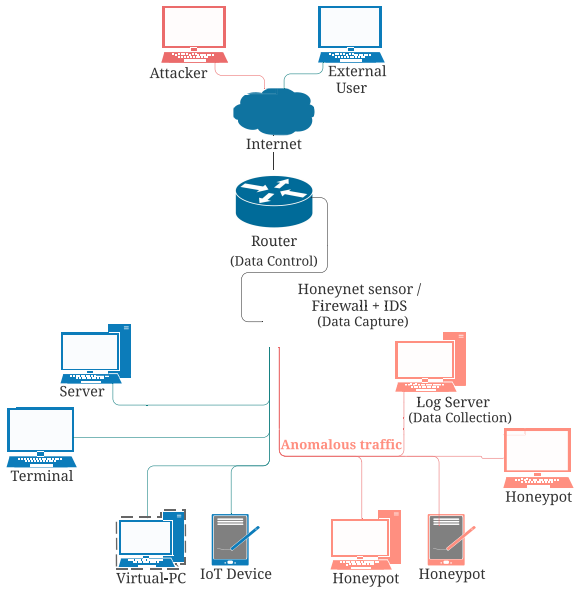

Fig. 1. Basic honeynet architecture.

options has its own advantages and disadvantages. In addition, the decision of the deployment environment may have an effect on the choice of the most appropriate type of honeypot or honeynet.

B. Other Related Terms

Other concepts and terms exist related to honeypots and honeynets for IoT, IIoT, and CPS applications. These are testbeds, network emulators, and simulation frameworks. Similar to honeypots and honeynets, such systems simulate or emulate devices, protocols, or even provide a physical environment where CPS devices operate and communicate using industrial protocols. However, unlike honeypots and honeynets, they do not act as decoy systems that aim to grab the attention of attackers and analyze their attacks. As we explain in the following sections, honeypot and honeynet researchers used such tools to create their decoy systems. The MiniCPS framework [43], the IMUNES emulator/simulator [44], the GridLab-D power distribution simulator [45], the SoftGrid smart grid security toolkit [46], the PowerWorld simulator [47], and the Mininet emulator [48] were all used in a number of studies to simulate protocols, emulate devices and scale decoy systems. Front-end and back-end are also related terms that are used in various studies. The front-end of a honeypot/honeynet system is the part attackers interact with and gathers data, while the back-end receives data from the front-end for analysis, decryption, and storage. Self-adapting refers to the ability of a honeypot to analyze information and adapt its responses or behavior accordingly in order to accomplish its purpose better.

IV. Classification Methodology

Honeypots and honeynets can be classified in various ways. In order to classify the honeypots and honeynets for

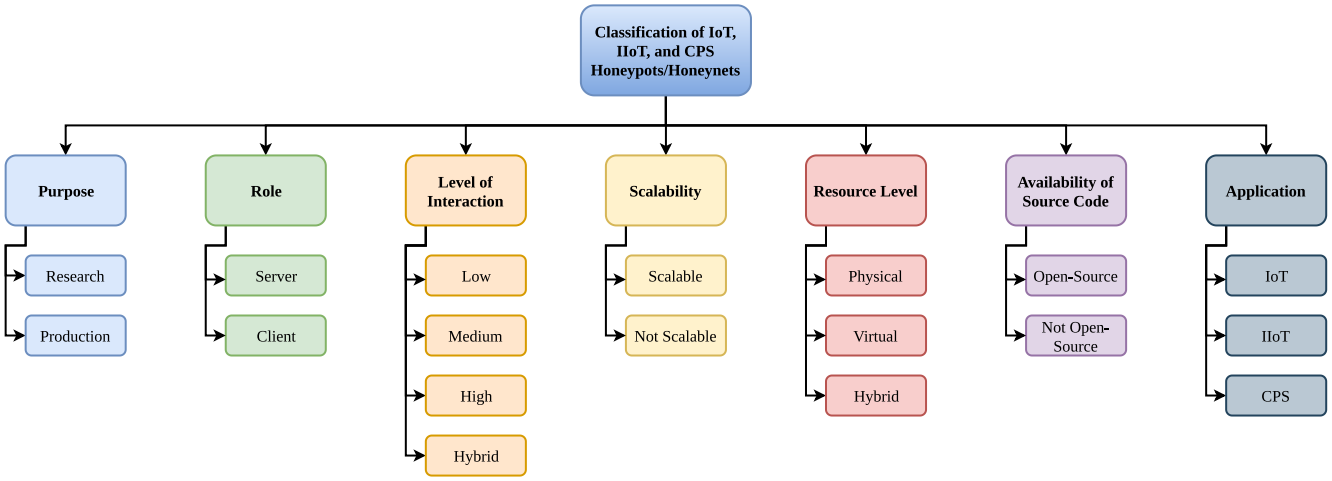

Fig. 2. Classification categories of honeypots and honeynets for IoT, IIoT, and CPS in which some of the items in the categorization build upon [12], [24], [25], [28]. Details of works corresponding to each category are tabulated in Tables I, II, and III.

IoT, IIoT, and CPS in this survey, we build upon prior surveys [12], [24], [25], [28]. However, our classification in this work improves the existing works by identifying some of the recurring key characteristics of the surveyed works. Specifically, we classify the honeypots and honeynets for IoT, IIoT, and CPS with respect to their *purpose, role, level of interaction, scalability, resource level, availability of the source code,* and their *application* as shown in Fig. 2. We also consider the simulated services, the inheritance relationships between the honeypots and honeynets, the platforms they were built on, and the programming languages they used.

Classification by Purpose: Honeypots can be categorized into two classes based on the purpose for which they were created: *research* and *production* honeypots. Research honeypots are used to gather and analyze information about attacks in order to develop better protection against those attacks. Production honeypots are more defense-focused. They are usually implemented to keep an attacker from accessing the actual system of the organization that implements it [13].

Classification by Role: Role refers to whether a honeypot actively detects or passively captures traffic. A *client* honeypot can actively initiate a request to a server to investigate a malicious program while a *server* honeypot waits for attacks. The great majority of honeypots are server honeypots [12].

Classification by Level of Interaction: Honeypots can be classified by the level of interaction that they allow to the attacker: *low interaction, medium interaction, high interaction,* and *hybrid*. Low interaction honeypots emulate one or more services with simple functions and do not give access to an operating system. The benefits of low interaction honeypots are ease of setup, low risk, low cost, and low maintenance. However, low interaction honeypots are identified much more easily by attackers because of their limitations, and the information they gather is limited and has low fidelity [28].

High interaction honeypots provide much more interaction, not only emulating services but also allowing access to an operating system [28]. While some of the research refers to high interaction when a honeypot is created using real devices, other works also include virtual environments that emulate complete devices and services as high interaction. High interaction honeypots collect information about all of the attacker's movements and actions, which is an advantage of high interaction honeypots because the information gathered has high fidelity. However, they come with high risk because everything they allow attackers to access is on real resources to gather more information. Moreover, they are more complex to set up, they collect much more data, and they are more difficult to maintain and run [28]. Once they are compromised, rebuilding them becomes necessary. Also, attackers can compromise them to attack other targets, which creates liability issues.

As the name indicates, medium interaction honeypots provide a level of interaction in-between a low and a high interaction honeypot. Although there are different perspectives on whether they have a real operating system or an emulated operating system, they do emulate more services than a low-interaction honeypot, providing for more interaction which increases risk, and makes them more difficult to detect compared to low interaction honeypots.

Figure 3 shows how the level of interaction varies in relation to the different characteristics. This should be seen as more of a fluid continuum rather than set characteristics.

A mix of honeypots with different levels of interaction implemented in the same system is called a hybrid honeynet. Hybrid honeynets are able to provide a better balance by providing the benefits of each type of honeypot [29].

Classification by Scalability: Scalability refers to the ability of a honeypot to grow and provide more decoys. An unscalable honeypot has only a certain number of decoys and cannot be changed. A scalable honeypot can expand the number of decoys it deploys and monitors [12]. Scalability is important because various honeypots implemented together in a honeynet provide greater protection, services, data collection, and variety of data compared to a single honeypot. Physical honeypots are usually harder to scale because of the resources needed. High-interaction honeypots also tend to have lower scalability because of their complexity.

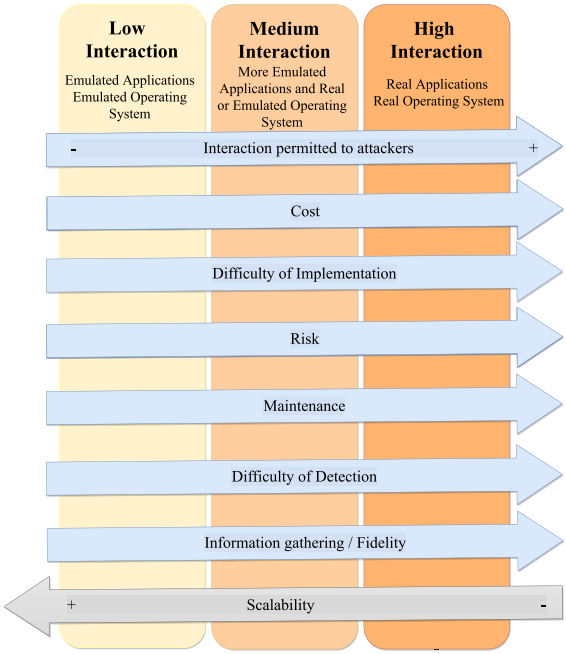

Fig. 3. Characteristics by level of interaction.

Classification by Resource Level: The type of resources used to create the honeypot system can be physical or virtual. A *physical* honeypot system is composed of several honeypots running on physical machines, while a *virtual* honeypot system is made up of virtual honeypots that are hosted on one or more physical machines. Physical honeypots have high interaction and have more data capture fidelity than virtual honeypots. However, they are more costly and require more resources to implement. Virtual honeypots require fewer resources to implement and are therefore less costly. A *hybrid* honeynet that uses both physical and virtual honeypots is able to better balance cost and data capture fidelity [22].

Classification by Availability of Source Code: Open-source refers to a software's source code being released in a way that anyone can have access to it, modify it, and/or distribute it. Open-source software allows for collaborative development. Not all of the honeypot and honeynet authors provide the source code of their decoy systems. Making the source code available allows other researchers and developers to understand and improve the existing honeypots and honeynets.

Classification by Application: Application refers to the intended application for which the honeypot system is created. In this survey, we classify the IoT honeypot systems as general use, IoT, or Smart Home IoT. General use honeypots are those that were not originally created for IoT. However, these are relevant because they have subsequently been used in research with IoT honeypots. IoT honeypots target general IoT applications. IoT Smart Home honeypots are honeypots with a specific focus on applications for Smart Home uses. We classify the CPS and IIoT honeypot systems as ICS, Smart Grid, Water System, Gas System, Building Automation System, and IIoT applications. Although the boundary between ICS and other smart infrastructures are not very obvious, we adhere to the authors' statements about their honeypots in this paper for the classification by application purposes.

TABLE I
LIST OF GENERAL IoT HONEYPOTS

Honeypot	Interaction Level	Simulated Services
HoneyD [50]	Low	FTP, SMTP, Telnet, IIS, POP
Dionaea [51]	Medium	Black hole, EPMAP, FTP, HTTP, Memcache, MongoDB, MQTT, MySQL, Nfq, PPTP, SIP, SMB, TFTP, UPnP
Kippo [52]	Medium	SSH
Cowrie [53]	Medium/High	SSH, Telnet, SFTP, SCP
HoneyPy [54]	Low/Medium	Created as required
AHA [55]	Low/High	SSH
AHA with Rootkit Detection [49]	Medium	SSH
RASSH [56]	Medium	SSH
QRASSH [57]	Medium	SSH

V. HONEYPOTS AND HONEYNETS FOR INTERNET OF THINGS

In this section, we give a brief overview of honeypot and honeynet studies for IoT. First, we identify some general application honeypots available. Next, we present the research with IoT honeypots and honeynets with full device emulation. Finally, we present the IoT honeypot and honeynet research focused on the type of attack. We would like to note that, unless otherwise stated, honeypots reviewed in this section are in the role of server honeypots.

A. General Application Honeypots

There are various general application honeypots that have an inheritance relationship with later research and honeypots for IoT applications. In other words, while these honeypots and honeynets were not specifically created for IoT, they are being used in research for IoT honeypots and honeynets. It is important to note that all of these are open-source, except for the Adaptive Honeypot Alternative (AHA) with Rootkit Detection [49]. Table I provides a list of the considered general IoT honeypots.

Honeyd: HoneyD [50] is an open-source software for the creation of low interaction, scalable honeypots. Honeyd creates virtual honeypots, but it also allows physical machine integration. It can simulate UDP, TCP, FTP, SMTP, Telnet, IIS, and POP services. Stafira [58] examined if HoneyD is able to create effective honeypots to attract attackers. They compared honeypots simulating IoT devices with real devices. The results showed that, although the content served by both honeypots and real devices were similar, there are significant differences between average times for query responses and Nmap scans.

Dionaea: Dionaea [51] is an open-source software for the creation of medium interaction honeypots that can simulate several services (e.g., FTP, HTTP, MongoDB, MQTT, MySQL, SIP, SMB, TFTP, UPnP, etc.) [59]. It targets adversaries that attack hosts on the Internet with vulnerable services. Since adversaries try to install malware on the infected hosts, Dionaea aims to obtain a copy of malware and help researchers to analyze it. Dionaea has a static configuration, which makes it difficult to adapt the configuration as needed to respond to events [12]. Metognon and Sadre [60] used Dionaea in their IoT honeypot research. Kaur and Pateriya [61] proposed the

setup of a cost-effective honeypot for IoT using Dionaea on Raspberry Pi and analyzed captured data using VirusTotal tool and Shodan search engine to identify the characteristics and vulnerabilities of devices in order to improve their security.

Kippo: Kippo [52] is an open-source, medium interaction, scalable honeypot. It focuses on SSH, and it logs brute force attacks, as well as interactions from automated or individual attacks [62]. Dowling *et al.* [62] modified Kippo in order to implement a ZigBee IoT honeypot. Kippo is chosen because of the high number of attacks that SSH receives. Pauna [56] used Kippo for the creation of Reinforced Adaptive SSH (RASSH) honeypot.

Adaptive Honeypot Alternative (AHA): Wagener [55] used both a low- and a high-interaction honeypot to gather data from attackers. With this data, he applied game-theory and Machine Learning (ML) techniques to develop a self-adaptive SSH honeypot called Adaptive Honeypot Alternative (AHA) [63]. While Wagener does not implement his honeypot in an IoT environment, his honeypot serves as the basis for Pauna's works [49], [56], [57], [64]. Wagener reported that attackers carried out three times more interactions when they were responding to the customized tools of an adaptive honeypot, which shows the important role that adaptive honeypots can play in honeypot research.

AHA with Rootkit Detection: In 2012, Pauna [49] improved on Wagener's adaptive honeypot, creating a medium interaction, scalable, virtual honeypot with the ability to detect rootkit malware installed by attackers. Pauna's honeypot resides on the Argos emulator as a guest OS, and utilizes Argos to detect rootkit malware. This research was followed by [56], [57], [64].

RASSH: In 2014, Pauna and Bica presented an adaptive honeypot, RASSH [56], which uses a medium-interaction Kippo honeypot integrated with two modules: Actions module and Reinforcement Learning module. RASSH interacts with attackers and takes dynamic actions (e.g., allowing, blocking, delaying, etc.) using the Reinforcement Learning module. This research was followed by [57], [64], which led to the creation of IRASSH-T [65] self-adaptive IoT honeypot.

Cowrie: Cowrie [53] is a software for the creation of medium to high interaction, scalable, virtual honeypots. As a medium interaction honeypot, it logs an attacker's shell interaction on a simulated UNIX system via emulating several commands. As a high interaction honeypot, it is a proxy for SSH and Telnet to observe an attacker's interaction on another system. To be more specific, it can act as a proxy between an attacker and a pool of virtual machines configured in a backend site which allows flexibility. Cowrie was forked from Kippo honeypot and simulates SSH, Telnet, SFTP, SCP, and TCP/IP services. It supports integration to ElasticSearch, LogStash, and Kibana for logging, storage, and visualization. It has been used in the IoT honeypot research for Metognon and Sadre [60], IRASSH-T [64], ML-Enhanced Cowrie [66], and Lingenfelter *et al.* [67].

HoneyPy: HoneyPy [54] is a software for the creation of low to medium interaction honeypots, depending on services that are simulated. HoneyPy comes with a large range of plugins that can be used for simulating services such as DNS, NTP, SIP, SMTP, Web, etc. It can also be configured to run with custom configurations as needed. HoneyPy provides researchers several options for logging, which include but are not limited to ElasticSearch, Logstash, RabbitMQ, Slack, Splunk, Twitter. In this way, external services can be used to analyze HoneyPy logs. Metognon and Sadre [60] used HoneyPy in their IoT honeypot research.

QRASSH: In 2018, Pauna *et al.* [57] proposed another SSH honeypot, namely Q Reinforced Adaptive SSH (QRASSH) [68] honeypot, which uses Cowrie and Deep Q-learning. However, Pauna *et al.* identified that the reward functions in the algorithms used in QRASSH were subjective. For this reason, they proposed further research for being able to generate optimal reward functions for the desired behavior. This study was further advanced in [64] and led to the creation of IoT Reinforced Adaptive SSH (IRASSH-T) [65] honeypot.

Metongnon and Sadre: Metongnon and Sadre [60] carried out a measurement study to observe attacks against protocols that are commonly used by IoT devices. They used a large /15 network telescope to observe large-scale events/traffic on the dark address-space of the Internet. They deployed three honeypots: Cowrie [53], HoneyPy [54], and Dionaea [59] to get more details about specific attacks. The top three most attacked protocols observed via telescope were Telnet (Ports 23 and 2323), SSH (Port 22), and HTTP(S) (Ports 80, 81, 8080, 443). The most attacked protocols observed on the honeypots were Telnet, SMB, and SSH.

TABLE II
LIST OF IOT HONEYPOTS FOR FULL DEVICE EMULATION

Honeypot	Interaction Level	Emulated Devices
FIRMADYNE [69]	High	COTS network-enabled IoT devices
ThingPot [70]	Medium	Philips Hue, Belkin, Wemo, Tplink
ML-Enhanced ThingPot [71]	Medium	General IoT devices
IoTCandyJar [72]	Intelligent	General IoT devices
Chameleon [73]	Hybrid	Any real IoT device
Honware [74]	High	CPE devices

B. Research With IoT Honeypots and Honeynets With Full Device Emulation

IoT honeypots and honeynets that provide full device emulation provide the most versatility. Full device emulation allows for greater realism and increases the difficulty for attackers to detect it as a honeypot. In this section, only those honeypots/honeynets which have the ability to fully emulate all kinds of devices are included. It is important to note that five of the six IoT honeypot/honeynet studies which are identified as providing full device emulation are also self-adaptive. Table II provides a list of the considered IoT honeypots that perform full IoT device emulation.

FIRMADYNE: Chen *et al.* [69] presented FIRMADYNE [75], an open-source, extensible, self-adaptive automated framework for discovering vulnerabilities in commercial-off-the-shelf network-enabled devices. FIRMADYNE works by emulating the full system with an instrumented kernel. It has a Web crawler component to download firmware images and their metadata, an extract

firmware filesystem, an initial emulation component, and a dynamic analysis component. FIRMADYNE was evaluated using a real-world dataset of more than 23,000 firmware images from 42 device vendors and 74 exploits. Out of 9,486 firmware images that were successfully extracted, 887 prove vulnerable to at least one exploit, and 14 previously unknown vulnerabilities were discovered.

ThingPot and ML-Enhanced ThingPot: Wang *et al.* [70] proposed ThingPot [76], a medium-interaction, scalable, virtual open-source honeypot that simulates the complete IoT platform and all supported application layer protocols. ThingPot was tested for 45 days with Extensible Messaging and Presence Protocol (XMPP) and REST API, and most of the captured requests were HTTP REST requests. The authors noted that the attackers were looking for certain devices like Philips Hue, Belkin, Wemo, and TPlink, scanning to get information about the devices, and then using more targeted attacks such as brute force or fuzzing to control them. They also noted that the attackers were using The Onion Router (TOR) network [77] to stay anonymous. Vishwakarma and Jain [71] used ThingPot to propose ML-Enhanced ThingPot, a self-adaptive honeypot solution for the detection of DDoS attacks through the Telnet port that uses unsupervised machine learning (ML) techniques in real-time.

IoTCandyJar: Luo *et al.* [72] proposed a new type of honeypot which they define as *intelligent interaction*, and has the benefits of both low and high interaction honeypots, simulating the behaviors of IoT devices without the risk of the honeypot being compromised. The honeypot uses ML with Markov Decision Process to automatically learn the behaviors of IoT devices that are publicly available on the Internet and learn which has the best response to extend the session with attackers. IoTCandyjar captured 18 million raw requests during the time of the study, including about 1 million IoT related requests. Ports 80, 7547, 8443, 81, 8080, and 88 were the most scanned, with the majority of requests being HTTP.

Chameleon: Zhou [73] proposed a self-adaptive IoT honeypot that can emulate all kinds of IoT devices. Chameleon has front-end responder, evaluator, and back-end interactor modules. The front-end responder processes requests and responds accordingly. If the request is new, the responder sends the request to the evaluator. The evaluator evaluates the security of the request with the IP whitelist. If the source is untrusted, Chameleon responds with a default response and the request is stored for manual study. The back-end interactor establishes a connection with the target IoT device and detects the open ports and services to open/start them on Chameleon. As the honeypot receives more requests, Chameleon's characteristics become more like those of the target device. Chameleon is evaluated by simulating a variety of 100 IoT devices on the Internet, and comparing this to 100 traditional honeypots using Shodan Honeyscore [78] fingerprinting tool. The honeypots simulated by Chameleon were not fingerprinted while all the traditional honeypots were.

Honware: Vetterl and Clayton [74] presented a high interaction virtual self-adaptive honeypot that emulates diverse IoT and Customer Premise Equipment (CPE) devices by processing a standard firmware image and extracting and adapting the filesystem. Honware uses Quick Emulator (QEMU) to be able to fully emulate devices, and runs this with a customized pre-built kernel and the filesystem on a host OS.

TABLE III
LIST OF IoT HONEYPOTS THAT FOCUS ON SPECIFIC ATTACKS

Target Attack(s)	Honeypots	Interaction Level
Telnet	IoTPOT [79]	Hybrid
	MTPot [80], Semic and Mrdovic [81]	Low
	Phype [82]	Medium
SSH and Telnet	Shrivastava et al. [66], IRASSH-T [65], Lingenfelter et al. [67]	Medium
Telnet, SSH, HTTP, and CPE WAN Management	Krishnaprasad [83]	Hybrid
Man-in-the-Middle	Oza et al. [84]	High
D/DoS	Anirudh et al. [85], Vishwakarma and Jain [71]	Medium
	Tambe et al. [86], Molina et al. [87]	High
Fileless attacks	HoneyCloud [88]	High
SSH on Zigbee networks	Dowling et al. [62]	Medium
UPnP	U-Pot [89]	Medium
Attacks on Authentication	HioTPot [90]	Not identified
Reconnaissance	HoneyIo4 [37]	Low
Attacks on home networks	Pot2DPI [91]	Medium
Attacks on device characteristics	Siphon [92]	High
	Metongnon and Sadre [60]	Low/Medium
	Zhang et al. [93]	Hybrid

C. Research With IoT Honeypots and Honeynets Focused on Type of Attack

This section contains all of the remaining research with IoT honeypots and honeynets, organized by their focus on attack type. Table III provides a list of the considered IoT honeypots by their target attack types.

Only Telnet Attacks: IoTPOT [79] is a hybrid honeypot proposed by Pa *et al.* [94] that simulates Telnet services for different IoT devices and focuses on Telnet intrusions. IoTPOT uses a front-end low-interaction responder that simulates IoT devices by responding to TCP requests, banner interactions, authentication, and command interactions. It is proposed to work on the back-end with a high-interaction virtual environment called IoTBOX running a Linux OS to analyze the attacks and the captured malware, and run the malware on multiple CPU architectures.

MTPot [80] is a low-interaction, unscalable, virtual IoT honeypot that was designed specifically for Mirai attacks. According to Evron [38], it detects connections on ports using Telnet, identifies Mirai based on the commands requested, alters parameters to identify Mirai attacks, and reports to a syslog server. Evron notes that while the tool can be easily fingerprinted, it is simple and can also prove useful.

Šemić and Mrdovic [81] presented a multi-component low-interaction honeypot with a focus on Telnet Mirai attacks. The front-end of their honeypot is designed to attract and interact with attackers by using a weak, generic password. Instead of using an emulation file, the front-end is programmed to generate responses based on the input from the attacker, with the logic defined in the code. The back-end is protected by a firewall and receives the information from the front-end for decryption, reporting, and storage.

Phype Telnet IoT Honeypot [82] is an open-source software for the creation of medium interaction, scalable, virtual honeypots with a focus on IoT malware. According to the Phype GitHub repository [82], Phype simulates a UNIX system shell environment. It tracks and analyses botnet connections, mapping together connections and networks. The application includes a client honeypot that accepts Telnet connections and a server to receive and analyze the information gathered about these connections.

Telnet and SSH Attacks: Shrivastava *et al.* [66] focused on the use of Cowrie Honeypot to detect attacks on IoT devices and created a Machine Learning (ML)-Enhanced Cowrie. They opened the Telnet and SSH ports, and classified requests as malicious payload, SSH attack, XOR DDoS, suspicious, spying, or clean (non-malicious). They evaluated various ML algorithms to analyze and classify data, and concluded that Support Vector Machine (SVM) gives the best results with an accuracy of 97.39%.

Based on the their prior QRASSH honeypot, Pauna *et al.* [64] proposed a self-adaptive IoT honeypot named IRASSH-T that focuses on SSH/Telnet. IRASSH-T uses reinforcement learning algorithms to identify optimal reward functions for self-adaptive honeypots to communicate with attackers and capture more information about target malware. Their evaluation shows that IRASSH-T improves on previously identified reward functions for self-adaptive honeypots and will be able to attract more attacks and enable collection of more malware from attackers.

Lingenfelter *et al.* [67] focused on capturing data on IoT botnets using three Cowrie SSH/Telnet honeypots to emulate an IoT system. Their system sets the prefab command outputs to match those of actual IoT devices and uses sequence matching connections on ports to facilitate as much traffic as possible. They analyzed remote login sessions that created or downloaded files. They also used a clustering method with edit distance between command sequences to find identical attack patterns. During their study, two Mirai attack patterns accounted for 97.7% of the attacks received on the honeypot. They concluded that botnet attacks on Telnet ports are the most common attack to download or create files, and many attacks on IoT devices are carried out with Mirai.

Telnet, SSH, HTTP, and CWMP Attacks: Krishnaprasad [83] used IoTPOT [94] as a model in creating a honeypot with a low interaction front-end. The front-end has a proxy for Telnet, SSH, HTTP, and CPE WAN Management (CWMP) protocols and gathers attack data. The high interaction backend on Krishnaprasad's model can be physical or virtual, a single machine or a network of machines, and has a module for each of the protocols. The honeypot uses Twisted [95] event-driven networking engine, and employs Logstash [96] to collect log data. The log data is pushed to Elasticsearch [97] for storage and Kibana [98] is used for visualization. For evaluation, Docker containers were setup to simulate IoT devices, and the honeypot was deployed in seven locations around the world. In seven days, the honeypot was reported to have received attacks from 6774 distinct IPs. More than half of these were Telnet attacks, followed by CWMP and SSH, with HTTP receiving significantly less attacks than the others.

Man-in-the-middle Attacks: Oza *et al.* [84] addressed the issue of Man-in-the-Middle (MitM) attacks and presented a deception and authorization mechanism called OAuth to mitigate these attacks. When a user sends a request to an IoT device in the system, if the user information is not stored in the database, it is sent to an Authenticator that sends a message to the valid user. If the request is not authenticated by the user, it is sent to the honeynet instead of sending to the IoT device.

DoS Attacks: Anirudh *et al.* [85] investigated how a DoS attack in an IoT network can be blocked by a medium-high interaction honeypot. Their system employs an IDS which passes malicious requests to the honeypot for further analysis. In order to evaluate their system, they simulated IoT data, and compared the performance of their system in blocking DoS attacks with and without the honeypot.

DDoS and Other Large Scale Attacks: Using ThingPot [70], Vishwakarma and Jain [71] proposed a self-adaptive honeypot to detect malware and identify unknown malware like those used in zero-day DDoS attacks. The proposed solution collects logs of attacks received by ThingPot honeypots and uses the logs to train ML classifiers. The authors considered deploying virtual box images of ThingPot on the IoT devices in a network, and placing the ML classifier on the router.

Tambe *et al.* [86] proposed a scalable high interaction honeypot to attract and detect large scale botnet attacks. In order to solve the scalability problem of high interaction honeypots using real devices, Tambe *et al.* used VPN tunnels which allowed a small number of real IoT devices to appear as multiple IoT devices with different IP addresses around the world. Their evaluations using commercial-off-the-shelf IoT devices showed that the devices were being detected as honeypots by Shodan Honeyscore [78]. The authors also proposed two live traffic analysis methods for the detection of large scale attacks.

Zarca *et al.* [87] presented a self-adaptive high interaction IoT honeynet as part of a full cyber-security framework. Their framework uses Network Function Virtualization (NFV) and Software Defined Networks (SDN) to emulate a network of physical devices and allow IoT systems to self-protect and self-heal from DDoS botnet attacks. The honeynet uses NFV to allow for the autonomic deployment of virtual high interaction honeypots with dynamic configuration and reconfiguration. They used SDN for connectivity, data control, traffic filtering, forwarding, and redirecting between the honeynet and the real IoT environment. This allowed them to deploy honeynets both pro-actively and reactively.

Fileless Malware Attacks: Dang *et al.* [88] presented HoneyCloud for fileless attacks on Linux-based IoT devices. HoneyCloud was implemented using both physical and virtual honeypots. The virtual honeypots provided full device emulation for the six IoT device types. They used four physical IoT honeypots (a Raspberry Pi, a Beaglebone, a Netgear R6100, and a Linksys WRT54GS) and 108 virtual IoT honeypots to attract and closely analyze the fileless attacks and to propose defense strategies. Their research revealed that approximately 9.7% of malware-based attacks on IoT devices are fileless and these attacks can be powerful. They also identified the top ten most used shell commands in fileless attacks, 65.7% of

which are launched through rm, kill, ps, and psswd commands, enabled by default on Linux-based IoT devices.

Only SSH Protocol Attacks: Dowling *et al.* [62] focused on SSH protocol attacks on ZigBee networks. A Wireless Sensor Network (WSN) was created with Arduino and XBee modules to transmit medical information in pcap files that serve as honeytokens to catch the attention of attackers. A Kippo medium interaction SSH honeypot was modified to simulate a ZigBee Gateway available through SSH to attract the maximum amount of traffic. The attacks were analyzed to see which ones were directed at ZigBee. Of all the attacks documented, only individual attacks demonstrated interest in the honeytokens, or the files, leading to the conclusion that the attacks were not geared toward ZigBee in particular. On the other hand, 94% of honeypot activity was dictionary attacks that continuously tried to access the network by sequentially trying different username and password combinations.

Attacks on UPnP Devices: U-Pot [89] is an open-source medium-interaction, virtual honeypot platform for Universal Plug and Play-based (UPnP) IoT devices. U-Pot can be used to emulate real IoT devices, and can be scaled to mimic multiple instances at once. A honeypot can even be automatically created using UPnP device description documents for a UPnP IoT device. The main benefits of U-Pot are its flexibility, scalability, and low cost.

Authentication Attacks: HIoTPot [90] is a virtual IoT honeypot created on a Raspberry Pi for both research and production. Using Raspberry Pi 3 as a server, HIoTPoT maintains a database of authenticated users. When any user attempts to gain access to the IoT network, it compares the user with the MySQL database. Unidentified users are sent to the honeypot, where their attack patterns, logs, and chat details are tracked, while the system sends an alert to notify all devices in the network of the attempted intrusion.

Reconaissance Attacks: HoneyIo4 [37] is a low-interaction virtual production honeypot that simulates four IoT devices (a camera, a printer, a video game console, and a cash register). HoneyIo4 fools network scanners conducting reconnaissance attacks by simulating IoT OS fingerprints. With this fake OS information, the attack is redirected and becomes unsuccessful.

Attacks on Home Networks: Martin *et al.* [91] presented a comprehensive system for home network defense with four major components: a local honeypot to interact with attackers and collect data, a module to capture packet patterns and recognize malicious traffic, a deep packet inspection (DPI) for signature-based filtering, and a port manager for port remapping between the router and IoT devices. HoneyD [99] low interaction honeypot is used to monitor the ports that are supposed to be inactive and Pot2DPI serves as a connection between the port manager and honeypot to inform the honeypot when packet forwarding port mapping has happened. Evaluation of the proposed system was carried out using Alman-trojan, Cerber, Fereit, and Torrentlocker pcap traces and the system was able to detect the first three with 99.84% accuracy, while it was only 48.84% accurate in detecting Torrentlocker.

Attacks Focused on Device Characteristics: Guarnizo *et al.* [92] proposed Siphon, a high-interaction, scalable, physical honeypot. Siphon was implemented on seven IoT devices (IP cameras, a network video recorder (NVR), and an IP printer). The devices were made visible as 85 geographically distributed unique services on the Internet by connecting them to Amazon, LiNode, and Digital Ocean cloud servers in different cities via creating wormholes. Zhang *et al.* [93] focused on attacks aimed at the Huawei CVE-2017-17215 vulnerability that can be exploited for remote code execution. They implemented a medium-high interaction honeypot to simulate UPnP services, a high interaction honeypot using IoT device firmware, and a hybrid multi-port honeypot using Simple Object Access Protocol (SOAP) service ports to increase the honeynet capacity and simulate honeypots. They used a Docker image to package and rapidly deploy the honeynet to capture IoT attacks.

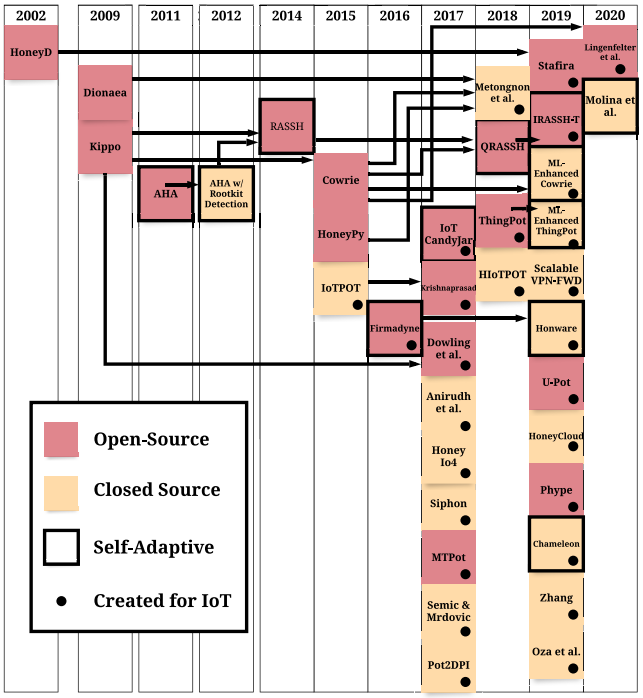

Fig. 4. Evolution of Inheritance for the IoT Honeypot and Honeynet Models and Research.

VI. Taxonomy of Honeypots and Honeynets for Internet of Things

Honeypots and honeynets proposed for IoT are listed in Table IV and the tools, implementation and attack type details of the corresponding honeypots and honeynets are also outlined in Table V. In this section, we consider all of the proposals for IoT and provide an overview of these studies based on the development of research over time, common characteristics, level of interaction, application, scalability, resource level, simulated services, most commonly used tools, availability of the source codes, and the most common attacks.

A. Development of Research Over Time

Research of honeypots specifically created for IoT begins in 2015 with the creation of IoTPOT [94]. Previous research included in this survey was originally created for general application and later built upon for IoT applications. As shown in Figure 4, about half of IoT honeypot models have

TABLE IV
CLASSIFICATION OF IOT HONEYPOTS AND HONEYNETS

Work	Year	Level of Interaction	Scalability	Resource level	Simulated services	Role	Open-source	Application
HoneyD [99]	2002	Low	✓	Virtual	FTP, SMTP, Telnet,IIS, POP	Server	Yes	General
Dionaea [59]	2009	Medium	X	Virtual	Black hole, EPMAP, FTP, HTTP, Memache, Mirror, MongoDB, MQTT, MSSQL, MySQL, nfq, PPTP, SIP, SMB, TFTP, UPnP	Server	Yes	General
Kippo [52]	2009	Medium	✓	Virtual	SSH	Server	Yes	General
Adaptive Honeypot Alternative [55]	2011	Low and High	✓	Virtual	SSH	Server	Yes	General
AHA with Rootkit Detection [49]	2012	Medium	✓	Virtual	SSH	Server	No	General
RASSH [56]	2014	Medium	✓	Virtual	SSH	Server	Yes	General
Cowrie [53]	2015	Medium/Hig	✓	Virtual	SSH, Telnet, SFTP, SCP	Server	Yes	General
HoneyPy [54]	2015	Low/Mediun	✓	Virtual	Created as required	Server	Yes	General
IoTPOT [94]	2015	Hybrid	✓	Virtual	Telnet	Server	No	IoT
Firmadyne [69]	2016	High	✓	Virtual	Full device emulation	Server	Yes	IoT
Dowling et al. [62]	2017	Medium	✓	Virtual	Zigbee, SSH, HTTP	Server	Yes	IoT
IoT CandyJar [72]	2017	Intelligent	✓	Virtual	Full device emulation	Server	Yes	IoT
Krishnaprasad [83]	2017	Hybrid	✓	Virtual	Telnet, SSH, HTTP, CWMP	Server	Yes	IoT
Anirudh et al. [85]	2017	Medium/Hig	✓	Virtual	Not identified	Server	No	IoT
HoneyIo4 Production Honeypot [37]	2017	Low	X	Virtual	SNMP, SSH, SMTP, DNS, HTTP	Server	No	IoT
Siphon [92]	2017	High	✓	Physical	HTTP, Telnet, SSH, RTSP	Server	No	IoT
MTPot [38]	2017	Low	X	Virtual	Telnet	Server	Yes	IoT
Semic and Mrdovic [81]	2017	Low	✓	Virtual	Telnet	Server	No	IoT
Pot2DPI [91]	2017	Medium	✓	Virtual	Telnet, UPnP	Server	No	Smart Home
Metongnon et al. [60]	2018	Low/Mediun	✓	Virtual	SSH, Telnet, EPMAP, FTP, HTTP, Memcache, MQTT, MSSQL, MySQL, PPTP, SIP, SMB, UPnP, TFTP, TR-069.1,TR-069.2, CoAP	Server	No	IoT
QRASSH [57]	2018	Medium	✓	Virtual	SSH	Server	Yes	General
ThingPot et al. [70]	2018	Medium	✓	Virtual	Full device emulation	Server	Yes	Smart Home
HIoTPOT [90]	2018	Not identified	✓	Virtual	Not identified	Server	No	IoT
Stafira [58]	2019	Low	✓	Physical	TCP/IP, HTTP	Server	Yes	Smart Home
IRASSH-T [64]	2019	Medium	✓	Virtual	SSH	Server	Yes	IoT
ML enhanced Cowrie [66]	2019	Medium	✓	Virtual	SSH, Telnet	Server	Yes	IoT
ML enhanced ThingPot [71]	2019	Medium	✓	Virtual	Full device emulation	Server	No	IoT
Scalable VPN-forwarded Honeypots [86]	2019	High	✓	Physical	HTTP, TFTP, Telnet, others not specified	Server	No	IoT
Zhang [93]	2019	Hybrid	✓	Physical/Virtua	UPnP, SOAP	Server	No	IoT
U-Pot [89]	2019	Medium	✓	Virtual	UPnP	Server	Yes	IoT
HoneyCloud [88]	2019	High	✓	Physical/Virtua	SSH, Telnet, SMB, HTTP, HTTPS, RDP, MySQL, SQL Server	Server	No	Smart Home
Phype [82]	2019	Medium	✓	Virtual	Telnet	Server	Yes	IoT
Oza et al. [84]	2019	High	✓	Virtual	Not identified	Server	No	IoT
Honware [74]	2019	High	✓	Virtual	Full device emulation	Server	No	IoT
Chameleon [73]	2019	Hybrid	✓	Virtual	Full device emulation	Server	Yes	IoT
Lingenfelter et al. [67]	2020	Medium	✓	Virtual	SSH, Telnet, SMTP, HTTP	Server	Yes	IoT
Molina et al. [87]	2020	High	✓	Virtual	Not identified	Server	No	IoT

a form of inheritance with each other, where a honeypot is built based on another. Cowrie [53] is the open-source honeypot with the greatest number of IoT honeypots which have been built directly from it. This could be in part because Cowrie continues to be actively maintained. In 2016, Firmadyne honeypot was the second IoT specific honeypot, and the first self-adaptive IoT honeypot. After the worldwide effects of Mirai malware in 2016, including attacks on IoT

TABLE V
TOOLS, IMPLEMENTATION AND ATTACK TYPES OF HONEYPOTS AND HONEYNETS FOR IoT

Work	Tools	Simulated services	Attack Types	Data Analyzed	Length of the Study
HoneyD [99]	N/A	FTP, SMTP, Telnet, IIS, POP	N/A	N/A	N/A
Dionaea [59]	N/A	Black TTP, Memache, Mirror, MongoDB, MQTT, MSSQL, MySQL, Nfq, PPTP, SIP, SMB, TFTP, UPnP	N/A	N/A	N/A
Kippo [52]	N/A	SSH	N/A	N/A	N/A
Adaptive Honeypot Alternative [55]	AHA Daemon	SSH	SSH-brute force	User/passwords, TTY buffer, TCP/UDP packets	8 hours
AHA with Rootkit Detection [49]	AHA Daemon, Kernel rootkit Kbeast, Argos	SSH	Rootkit malware	Keystroke logging, Rootkit malware	7 days
RASSH [56]	Pybrain RL, SARSA, Markov	SSH	SSH attack	Logs, commands offering downloading	N/A
Cowrie [53]	N/A	SSH, Telnet, SFTP, SCP	N/A	N/A	N/A
HoneyPy [54]	N/A	Created as required	N/A	N/A	N/A
IoTPOT [94]	Masscan, pcap	Telnet	DNS Water Torture, SSL attack, DoS, DDoS, UDP Flood, SYN Flood, ACK Flood, SynAck Flood, Null Flood, Telnet Scan, DNS attacks, Fake Web Hosting	PCAP analysis includes total # of packets, start/end time of packet captures, data byte/bit rate, average packet size and rate, number of victim IP address for each attack	39 days
Firmadyne [69]	Nmap, Metasploit framework, Binwalk, Scrapy, QEMU, Sasquatch, Firmware-mod-kit	HTTP, Telnet, DNS, dec-notes, HTTPS, UPnP, RIPD, Freeciv	Reconnaissance attacks, buffer overflow	Firmwares, results from web analysis, MIB files	N/A
IoT CandyJar [72]	pyLDAvis, Digital Ocean VM, Amazon AWS, MDP, Censys, ZoomEye, Shodan, MASSCAN	HTTP, RTSP, SOAP_Envelope	HTTP, HTTP_HEAD, UDP, HTTP_OPTIONS, TCP, SOAP_Envelope, RTSP, HTTP_CONNECT	Attack types and characteristics	1 month
Krishnaprasad [83]	Twisted, ELK Stack, Docker,	Telnet, SSH, HTTP, CWMP	Brute-force attack, Hajime, ZmEu attacks	Attack types and characteristics	7 days
Anirudh et al. [85]	IDS, logs	N/A	DoS attacks	IP Address, MAC Address	N/A
HoneyIo4 Production Honeypot [37]	Shodan, Nmap, Wireshark, Scapy, VM.	SNMP, SSH, SMTP, DNS, HTTP	Reconnaissance attacks	TCP, UDP and ICMP packets	N/A
Siphon [92]	Shodan, Tcpdump, Nmap	HTTP, Telnet, SSH, RTSP	Brute-force login attempts	TCP connections per wormhole, services consulted, access gained, movements statistics	60 days
MTPot [38]	N/A	Telnet	N/A	Incoming connections on any port using telnet	N/A
Semic and Mrdovic [81]	N/A	Telnet	Telnet attack	Protocols, IP addresses, logs	N/A
Pot2DPI [91]	N/A	Telnet, UPnP	Mirai and Persirai attacks protocols, ports scans	Packet traces, attack signatures, protocols, ports	N/A

devices, it is interesting to note there was a large increase in IoT honeypot and honeynet research in 2017. Of the nine studies published in 2017, seven studies explicitly refer to Mirai [37], [38], [72], [81], [83], [91], [92]. Also, as the number of IoT devices has increased rapidly in recent years, so has the research. 2019 saw a noticeable increase in the development of self-adaptive IoT honeypots. We can also see that more than half of the studies proposed independent honeypots, which may be due to shortcomings of existing honeypots to meet their needs.

TABLE V
(Continued.) TOOLS, IMPLEMENTATION AND ATTACK TYPES OF HONEYPOTS AND HONEYNETS FOR IOT

Work	Tools	Simulated services	Attack Types	Data Analyzed	Length of the Study
Metongnon et al. [60]	Eemo, Shodan	SSH, Telnet, EPMAP, FTP, HTTP, Memcache, MQTT, MSSQL, MySQL, PPTP, SIP, SMB, UPnP, TFTP, TR-069.1, TR-069.2, CoAP	Attack URL, SYN packet, Mirai and Mirai-like attacks, Harvest cryptocurrencies, Login attempts, Reconaissance	Protocols, packets per port, packets characteristics	5 months
QRASSH [57]	Deep Q-learning, Keras with Theano backend, Nmap	SSH	SSH attack	Commands(downloading, hacking, linux)	N/A
ThingPot et al. [70]	Skipfish, Nikto, Masscan	HTTP, XMPP, ZigBee	HTTP POST request, HTTP GET with URLs, scanning tools, SQL malware	HTTP request, SQL access request, scanning network	1.5 months
Stafira [58]	Nmap, Wireshark, VMWare Workstation	TCP/IP, HTTP	Only user testing	Access time, HTML code, network headers and Nmap scan	N/A
IRASSH-T [64]	Apprenticeship Learning	SSH	SSH attack	N/A	N/A
ML enhanced Cowrie [66]	Support Vector Machine (SVM), Random Forest, Naive Bayes, J48 decision tree, VirusTotal website, Weka, machine learning algorithms	SSH, Telnet	Malicious payload, SSH attack, XOR DDoS, Spying, Suspicious, Clean	System logs, IP, attack types and characteristics, commands executed, behavior analysis	40 days
ML enhanced ThingPot [71]	Linux bash scripts, Microsoft Azure, MATLAB	Telnet, MQTT, XMPP, AMQP, CoAP, UPnP, HTTP, REST	DDoS, malware, TCP SYN flood, UDP flood, HTTP GET flood	Network traffic, payload, malware samples, the toolkit by attacker	N/A
Scalable VPN-forwarded Honeypots [86]	VPN, TShark, HONAN, pcap, VM, MySQL, own script	HTTP, TFTP, Telnet, others not specified	DDoS style attacks	Protocols, packets per port, packets characteristics	16 months
Zhang [93]	Tc, own script	UPnP, SOAP	UPnP	Protocols, packets per port, timestamp, inject behaviors	7 days
U-Pot [89]	Shodan, Zmap, U-Pot	UPnP	N/A	N/A	N/A
HoneyCloud [88]	VM, Cloud storage, antivirus communities, Honeycomb, VirusTotal website	SSH, Telnet, SMB, HTTP, HTTPS, RDP, MySQL, SQL Server	Fileless attacks, malware-based attacks	Symmetry/asymmetry of data flows, packets analysis, attack types and characteristics, keystrokes, trace of network activities, CPU usage, Process list.	1 year
Phype [82]	Phype Telnet	Telnet	N/A	N/A	N/A
Chameleon [73]	Nmap, Shodan	N/A	Reconnaissance attacks	IP whitelist, received requests	N/A
Honware [74]	QEMU, Binwalk, Wireshark, Ping, Nmap, Firmadyne, Shodan	SSH, Telnet, HTTP, UPnP, DHCP, DNS, dec-notes, freciv, netbios, HTTPS, MDNS, TFTP	Reconnaissance attacks, Zero days, capture attacks traffic	Kernel logs, firmwares,	$<$ 2 months
Oza et al. [84]	OAuth2, MySQL, QEMU	N/A	Man in the Middle attacks	MAC address, Unauthorized access	N/A
Lingenfelter [67]	Filebeat, ELK stack, Logstash, VirusTotal	SSH, Telnet	IoT botnet malware	Packets per port, System logs, IPs, Brute-force scan, file hash	40 days

B. Common Characteristics

All of the honeypot/honeynet models surveyed were created for research purposes, except for HoneyIo4 [37] and the IoT honeynet presented by Molina [87], which are production, and HIoTPot [90] which is identified as both research and production. All of the decoys use Linux, and all can be classified as having a server role, except for Phype Telnet IoT Honeypot [82], which has both a client and a server role. In

addition, all of the open-source models were written in Python programming language.

C. Level of Interaction

In this study, classification based on level of interaction proved to be the most fluid of all the classifications regarding honeypots. Although most research can agree when a honeypot is low-interaction, definitions for the other levels can vary. For the purposes of this paper, level of interaction identified by the authors was used. Most research seeks to leverage the benefits of both low-interaction and high-interaction honeypots, many times calling this medium interaction. In other cases, this is done using hybrid honeynet systems.

D. Resource Level

The great majority of available research on IoT honeynets has been carried out with virtual resources rather than physical resources. Only Siphon [92], Stafira [58], and Scalable VPN forwarded Honeypots [28] were carried out with physical resources.

E. Scalability

Most of the honeypot and honeynet research was carried out using scalable honeypot systems, except for Dionaea [59] and HoneyIo4 [37], which can only deploy one simulation at a time. It is interesting to note that despite using virtual resources rather than physical resources, these two systems cannot be expanded to provide more decoys.

F. Application

Considering the application areas of honeypots and honeynets for IoT, nine of the models and research studies considered were for general use, 22 were for IoT application, and four were created for Smart Home applications.

G. Simulated Services

The most commonly simulated services in the research coincide with the top three most attacked protocols identified by Metongnon and Sadre [60]: Telnet, SSH, and HTTP(S). These are standard TCP/IP protocols, none of which are IoT specific. Two reasons for this may be that these common application protocols are targeted because they are in the most exposed and vulnerable layer and 75% of attacks on IoT devices were carried out through a router [100]. Each of the models and studies considered in this survey have their focus and specific purpose. However, there are five research models that stand out as the most versatile as they emulate full devices and are self-adaptive: IoTCandyJar [72], Chameleon [73], Firmadyne [69], Honware [74], and ML-enhanced ThingPot [71]. However, of these, only IoTCandyJar and Firmadyne are open-source.

H. Availability of Open-Source Honeypot and Honeynet Solutions

Approximately half of the IoT honeypot and honeynet models considered in this survey are open-source. This highlights the importance of open-source software in contributing to the development of improved models.

I. Most Commonly Used Tools

The three tools that were most commonly used in the honeypot research studies included in this survey are Shodan [101], Nmap [102], and MASSCAN [103]. Shodan [101] is a search engine for Internet-connected devices, which includes everything from Web cams, to medical devices, appliances, and water treatment facilities. Shodan indexes everything that is somehow connected to the Internet, their location, and their users, providing valuable information about the vulnerabilities of today's interconnected world. Shodan is used around the world, especially by corporations, researchers, security professionals, and law enforcement. Nmap [102] is an open-source, free tool for exploring networks and security auditing. It works by sending packets and then analyzing the responses. It is especially used by network administrators, auditors, and hackers to scan and determine what hosts are available on a network, the services they are offering, their operating systems, and other valuable information. The Nmap suite also has an advanced GUI, a data transfer and debugging tool called Ncat, a tool to compare scan results called Ndiff, and a packet generation and response analysis tool. Nmap is a flexible, easy, and powerful tool. Nmap is used by honeypot and honeynet developers as a tool to gain valuable information including checking for network connectivity, scanning for open ports on real or simulated devices, comparing scan results of real vs. simulated devices, and testing the fingerprintability of honeypots. MASSCAN [103] is another open-source, free tool, which is very similar to Nmap and has many similar functionalities. It is a TCP port scanner and its speed sets it apart from similar tools because it transmits 10 million packets per second, which allows it to scan the entire Internet in less than six minutes. Although there are many other diverse tools (e.g., Wireshark, VirusTotal, Pcap, Zmap, Censys, Scapy, etc.) that have been used in IoT honeynet research, these three are by far the most commonly used. This can be attributed to their availability, their low cost, their ease of use, and their effectiveness. Nmap is the most widely recognized and used network and security auditing tool and Shodan is the first and largest search engine for Internet connected devices.

J. The Most Common Attacks

The most commonly detected/tested attacks in IoT honeypots/honeynets are Telnet, SSH, DoS/DDoS, and HTTP(S) attacks. In addition, reconnaissance attacks, brute-force attacks, malware, and Mirai attacks were also detected/tested in the proposed honeypots and honeynets. Although less common than the mentioned attacks, botnet, Man-in-the-Middle, malicious cryptocurrency mining, and buffer overflow attacks were also detected/tested in the proposed systems.

VII. Honeypots and Honeynets for IIoT and CPS

In this section, we give brief overview of honeypots and honeynets proposed for IIoT and CPS applications. We group the IIoT and CPS honeypots and honeynets based on the

TABLE VI
LIST OF GENERAL ICS HONEYPOTS

Honeypots	Interaction Level	Simulated Services
CISCO [104]	Low	Modbus/TCP, Telnet, HTTP, FTP
Digital Bond [105]	Low	Modbus/TCP, Telnet, HTTP, FTP, SNMP
Conpot [108]	Low	IEC 60870-5-104, BACnet, EtherNet/IP, Guardian AST, Kamstrup, Modbus, S7comm, HTTP, FTP, SNMP, IPMI, TFTP
Zhao and Qin [110]	Medium	S7comm, Modbus, SNMP, HTTP
DiPot [111]	Low	HTTP, Modbus, Kamstrup, SNMP, IMPI, BACnet, Guardian AST, S7comm
XPOT [112]	Medium	S7comm, SNMP
HosTaGe [113]	Low	Modbus, S7comm, HTTPS, FTP, MySQL, SIP,SSH, SNMP, HTTP, Telnet, SMB, and SMT
S7CommTrace [114]	Medium	S7comm
Honeyd+ [115]	High	EtherNet/IP, HTTP
Gallenstein [116]	Low	EtherNet/IP. ISO-TSAP, HTTP
Abe et al. [117]	Low	Modbus, S7comm, BACNet, IPMI, Guardian AST, HTTP, SNMP

application types as follows: ICS, Smart Grid, Water Systems, Gas Pipeline, Building Automation Systems, and IIoT.

A. Honeypots and Honeynets for Industrial Control Systems

In this subsection, we give brief overview of honeypots and honeynets ICS. Table VI provides a list of the considered general ICS honeypots.

CISCO SCADA HoneyNet Project: The first honeynet for SCADA ICS was proposed by Pothamsetty and Franz in Cisco Systems' SCADA HoneyNet Project [104] in 2004. SCADA HoneyNet is based on the Honeyd [50] open-source honeypot framework and is a low-interaction honeynet that supports the simulation of Modbus/TCP, FTP, Telnet, and HTTP services running on a programmable logic controller (PLC).

Digital Bond SCADA Honeynet: The second honeynet for SCADA ICS was introduced by Digital Bond in 2006 under the name of SCADA Honeynet [105], [106]. It consists of two virtual machines: one of them simulates a PLC with Modbus/TCP, FTP, Telnet, HTTP, and SNMP services while the other one is a Generation III Honeywall. The Honeywall is a modified version of SCADA HoneyNet [104] that aims to monitor and control the honeypot's traffic and attacker interactions.

Wade [107] used Digital Bond's SCADA honeynet in her thesis to analyze the attractiveness of honeypots in ICS systems. Her honeypot simulated a Schneider Modicon PLC with Modbus TCP, FTP, Telnet, and SNMP services.

Conpot and Conpot-based ICS Honeypots: One of the most popular ICS honeypots that has been used by researchers is Conpot [108]. It is an open-source low-interaction honeypot that was developed under the Honeynet Project [109] and is still being maintained. Conpot supports various industrial protocols including IEC 60870-5-104, Building Automation and Control Network (BACnet), EtherNet/IP, Guardian AST, Kamstrup, Modbus, S7comm, and other protocols such as HTTP, FTP, SNMP, Intelligent Platform Management Interface (IPMI), and TFTP. It provides templates for Siemens S7 class PLCs, Guardian AST tank monitoring systems, and Kamstrup 382 smart meters.

Jicha *et al.* [118] deployed Conpot honeypots at six different locations around the world via Amazon Web Services platform. The authors configured and deployed two Conpot instances at every location, one with the default configuration and the other one with gas tank level SCADA Conpot. Honeypots ran for 15 days and they analyzed the behavior of simulated protocols against Nmap scanning tool and Shodan search engine scan data. They realized that the ports identified by Shodan and Nmap may differ.

Zhao and Qin [110] improved Conpot honeypots with additional Siemens S7comm protocol functions and sub-functions support and a dynamic Human Machine Interface (HMI) for evaluation of threats to ICS environments. The authors state that their study improved the interaction level of Conpot and provided better support for the simulation of Siemens S7 class PLCs. Their 43-day long deployment received traffic from 244 valid IP addresses from 34 different countries.

Cao *et al.* proposed a distributed ICS honeypot called DiPot [111]. DiPot is based on Conpot honeypot framework. It enhances Conpot framework by adding higher-fidelity ICS protocol simulations, data capture and analysis with K-means clustering, and visualization and statistics support. The authors indicated that deployed DiPot honeypots in cloud virtual machines around the world successfully deceived Shodan search engine and were recognized as real ICS devices.

Lu *et al.* [119] deployed Conpot on a Raspberry Pi to simulate Siemens S7 class PLCs. In addition, they used an Arduino board to simulate a PLC and another Arduino for sensor simulations. However, they did not give much detail about PLC simulation on Arduino and the supported industrial protocols. The authors did not perform any deployments or tests against the proposed honeypot architecture.

Ferretti *et al.* [120] aimed to analyze the scanning traffic on the Internet that is targeting ICS. To analyze the scanners and their behaviors, the authors deployed several low interaction Conpot honeypots. Each Conpot honeypot was configured to simulate a specific ICS device with a specific communication protocol (i.e., S7comm, Modbus/TCP, IEC-61850-104, EtherNet/IP, BACnet, HTTP, FTP, and SSH). Their analysis, which covered four months of operation, showed that the majority of the scanners were legitimate (e.g., Shodan, Censys, etc.) and showed certain scanning patterns. The authors pointed out that the usage of legitimate scanner patterns could give a clue in detecting malicious scanning and attack activities targeting ICS environments.

CamouflageNet: Naruoka *et al.* [121] proposed CamouflageNet, which is a honeypot system for ICS environments. CamouflageNet creates a set of Honeyd honeypots in the ICS network with the same fingerprints (i.e., services, ports. vendors) and changes the IP addresses of the devices in the network dynamically when an intrusion attempt is caught by one of the honeypots. In terms of ICS protocols and devices, CamouflageNet does not emulate any ICS-specific protocols or simulate any ICS device.

XPOT: Lau *et al.* [112] proposed an ongoing study of a medium-interaction honeypot for ICS, namely XPOT. XPOT simulates Siemens S7-300 series PLCs, and allows the attacker to compile, interpret and load PLC program onto XPOT. It supports S7comm and SNMP protocols.

HosTaGe: Vasilomanolakis *et al.* [113] proposed HosTaGe ICS honeypot which is an ICS protocols-extended version of their earlier mobile honeypot [122]. The proposed honeypot system consists of three parts: protocol emulation, multi-stage attack detection, and signature generation for IDS. Protocol emulation supports several protocols (e.g., Modbus, S7comm, HTTP(S), FTP, SIP, SSH, SNMP, etc.). The attack detection module employs an Extended Finite State Machine model to detect multiple-stage attacks that consist of attacks applied by the same source in a serial manner within a specified time window. HosTaGe ICS honeypot can generate signatures that open-source Bro IDS can use. In addition, the honeypot system uploads attacker-injected files to VirusTotal to determine if they are malicious or not. The authors deployed HosTaGe along with a Conpot honeypot.

S7CommTrace: S7CommTrace [114] is a honeypot for ICS that uses Siemens S7comm protocol. It consists of TCP communication, S7 communications protocol, user template, and data storage modules. Compared to Conpot, S7CommTrace supports more S7comm functions and sub-fuctions. Both S7CommTrace and Conpot instances were deployed to four different locations around the world on cloud environments and analyzed for 60 days. The analysis showed that S7CommTrace was able to receive more connections and provide more attack data compared to Conpot. In addition, while all Conpot instances were fingerprinted by Shodan, S7CommTrace instances were not fingerprinted.

ICS Honeypots based on Honeyd: Disso *et al.* [123] researched the security of SCADA systems from the honeypots' point of view. They created a testbed which consists of a real PLC device as a high-interaction honeypot and a Honeyd-based low-interaction honeypot. They placed Honeynet Project's Roo honeywall [124] in front of the honeypots. They conducted latency, network traffic counter and background traffic level analysis (i.e., anti-honeypot techniques) to compare the high and low interaction honeypots.

Winn *et al.* [115] proposed Honeyd+, which aims to construct several high-interaction ICS honeypots using a proxying technique with a single physical PLC device. Honeyd+ uses Honeyd honeypots with templates that mimic ICS PLCs. The performance analysis of the authors indicated that Honeyd+ is able to simulate 75 ICS honeypots using a single PLC device with a Rapsberry Pi board. However, the analysis showed that Honeyd+ starts to see serious performance drops starting from five simultaneous connections from attackers.

Gallenstein conducted research on the automated creation and configuration of ICS PLC honeypots that would emulate PLCs from different vendors with minimum effort. In his thesis [116], he integrated Honeyd with ScriptGenE framework proposed by Warner [125]. ScriptGenE is an extended version of ScriptGen which is automated protocol replay framework proposed by Leita *et al.* [126]. Gallenstein emulated a prison ICS environment that had three PLCs from Allen-Bradley and Siemens. He tested the legitimacy of his honeypots with Shodan Honeyscore, Nmap and vendor tools (i.e., RSLinkx and STEP7).

Abe *et al.* [117] proposed an ICS honeypot system that employs Honeyd and Conpot frameworks and adds a trace-back capability to gain more information about attackers. The proposed system is able to emulate the ICS protocols and devices by means of Conpot framework, and performs basic honeypot functions by means of Honeyd. The authors implemented Nmap in the Honeyd to perform a reverse scan to the attackers and obtain useful information regarding the attack.

TABLE VII
LIST OF ICS HONEYPOTS BASED ON NETWORK SIMULATORS/EMULATORS

Honeypots	Interaction Level	Simulator/Emulator
Haney et al. [127]	High	IMUNES Simulator, JAMOD Library
Kuman et al. [128]	Low	IMUNES Simulator
Ding et al. [129]	Medium	IMUNES Simulator

ICS Honeypots based on Network Simulators/Emulators: In this category, we give brief overview of honeypots and honeynets for ICS that are based on network simulators/emulators. Table VII provides a list of the ICS honeypots that are based on network emulators/simulators.

Haney and Mauricio proposed a SCADA honeynet framework [127] based on the IMUNES Network Simulator [44]. The proposed honeynet employs a honeywall to control the activities of attackers and IMUNES-based honeypots that simulate PLCs and RTUs in SCADA systems. The authors used Java Modbus library (JAMOD) to simulate Modbus/TCP and developed Java code for PLC state variables, control logic, and process to be controlled by the simulated PLC. They also utilized Honeyd to provide Telnet, SSH, and HTTP(S) services for the virtual PLC.

Kuman *et al.* [128] utilized Conpot honeypot on top of IMUNES network emulator to analyze attacks targeting ICS networks. They modified IMUNES emulated nodes so that they were able to install Conpot instances simulating Siemens S7-300 PLCs. They also employed OSSEC host-based IDS to monitor the activities with the Conpot instances. Two weeks long experiments showed that Siemens S7-300 PLC honeypots received port scanning activities to Modbus and Web server ports.

Ding *et al.* [129] proposed an ICS honeypot that used and modified Siemens S7comm protocol on top of IMUNES network emulator. The authors paid attention to configure SNAP7 tool to provide a fingerprint like real Siemens PLC devices. They also added SNMP service support. IMUNES gives the proposed honeypot chance to use lightweight Docker container technology to quickly start up. The authors used Nmap and PLCscan tools to verify the fingerprinting results of the proposed honeypot.

Honeypots and Honeynets Using Real ICS Devices: In this category, we give brief overview of honeypots and honeynets that use real ICS devices. Table VIII provides a list of the ICS honeypots that are based on network emulators/simulators.

TABLE VIII
LIST OF ICS HONEYPOTS THAT USE REAL DEVICES

Honeypots	Interaction Level	Used Device(s)
Bodenheim [130]	High	Allen-Bradley PLCs
Piggin et al. [39]	High	Not Specified
Haney [131]	Low	Direct Logic PLC
Hilt et al. [40]	High	Siemens S7-1200, Allen-BradleyMicroLogix 1100, Omron CP1L

TABLE IX
OTHER ICS HONEYPOTS

Honeypots	Interaction Level	Simulated Services
Berman [132]	Low	Modbus/TCP
Jaromin [133]	Low	Modbus/TCP, HAP, HTTP
Holczer et al. [134]	High	S7comm, SNMP, HTTP(S)
Serbanescu et al. [135]	Low	DNP3, IEC-104, Modbus, ICCP, SNMP, TFTP, XMPP
Simões et al. [136]	Low	Modbus, SNMP, FTP
Ahn et al. [137]	Low	Modbus
Belqruch et al. [138]	Medium	SSH

Bodenheim deployed four high-interaction ICS honeypots to an integrator's site in his thesis [130]. He aimed to analyze the impact of Shodan search engine on identification of the Internet-connected ICS devices and understand if being indexed by Shodan increases the number of attacks. He used real PLCs of Allen-Bradley. He configured two honeypots with standard settings, one honeypot with obfuscated banner information and the last one with a banner that advertised itself clearly as an Allen-Bradley PLC. A 55 day deployment period showed no evidence that Shodan's indexing increased the number of attacks.

Piggin and Buffey [39] presented a high-interaction honeypot for ICS production environments. They used real PLC hardware to obtain high fidelity data and prevent being detected as a honeypot. They also implemented a process simulation for the PLC hardware. However, they did not give details in regards to the PLC used, supported protocols, and the process simulation. They also implemented a Situational Awareness and Forensics platform for honeypot traffic analysis. Their results show that a majority of the attacks were scanning activities. In addition, they realized some targeted attacks, including a password attack with default credentials, a dictionary attack, an SSH brute-force attack, an attack to the industrial protocols of the PLC, and a knowledgeable attack originating from the TOR network.

Haney proposed a hybrid high interaction honeynet framework for ICS in [131]. The proposed honeynet consists of a honeywall system, a SCADA HMI on a virtual machine, physical PLC devices, virtual nodes emulating PLC devices and services (e.g., HTTP, SNMP, SSH, Telnet), and a physical process simulation. Haney paid attention to not only the requirements of Gen III honeynets (i.e., data control, data capture and information sharing), but also other requirements of realism, scalability, and detection resistance.

Hilt *et al.* [40] constructed the most realistic ICS honeypot to date. They set up a smart-factory honeypot using four real PLC devices (i.e., Siemens S7-1200, Allen-Bradley MicroLogix 1100, and Omron CP1L) with corresponding role implementations (i.e., agitator, burner control, belt control, palletizer). They implemented HMIs for each PLC on virtual machines. In addition to PLCs and HMIs, they also accommodated robotics and engineering workstations and installed corresponding software to them. Moreover, they set up a file server and placed fake files on it. In order to monitor the honeypots, they used Ethernet taps connected to a Raspberry Pi to collect monitoring information. Their honeypot system opened Siemens S7comm, Omron FINS, EtherNet/IP, and VNC protocols and services to the Internet. To convince the attackers that they are not a honeypot, they created a fake company profile with employees, Artificial Intelligence (AI)-generated pictures, profiles, website, e-mail addresses, and phone numbers. They tried to attract the attackers using messages posted on Pastebin. Analysis of honeypot deployments for seven months shows that the honeypots received scanning activities from unknown sources as well as from legitimate scanners. They were attacked by three ransomware attacks, one malicious cryptomining operation and robotic workstation beaconing attempt. However, the PLC honeypots did not receive any targeted attacks but only scanning activities.

Other ICS Honeypots and Honeynets: In this category, we give brief overview of the other ICS honeypots and honeynets. Table IX provides a list of rest of the ICS honeypots.

Berman [132] designed and implemented a PLC emulator on a Gumstix board that could be used as an ICS honeypot. The proposed emulator was developed in Python using Scapy library and acts like an Allen-Bradley PLC running Modbus/TCP protocol. Berman evaluated the performance of the emulator via fingerprinting and standard traffic response tests.

Jaromin proposed an ICS honeypot [133] that emulated Koyo DirectLogic 405 PLC devices. His honeypot used Berman's Modbus/TCP emulator [132] and added Host Automation Products (HAP) protocol emulation and a Web server implementation. He performed several tests to compare the Gumstix deployment, laptop Personal Computer (PC) deployment, and the real Koyo DirectLogic PLC device.

Holczer *et al.* [134] proposed a high interaction PLC honeypot for ICS which simulates the main functionalities of Siemens ET 200S PLCs. It can support Siemens STEP7 PLC management service, HTTP(S) and SNMP services. The authors deployed the honeypot within the IP range of their university and realized that they did not receive much traffic from the attackers.

Serbanescu *et al.* [135] proposed a low-interaction honeynet architecture for ICS environments. Their work extends their earlier honeypot [139] with ICS protocols and creates a honeynet. Deployment of the honeynet was made on the Amazon EC2 platform with six honeypots supporting combinations of different protocols (i.e., Distributed Network Protocol 3

(DNP3), IEC-104, Modbus, ICCP, SNMP, TFTP, XMPP). Analysis of the collected data for 28 days showed that the honeypots received only the reconnaissance activities. In addition, they indicated that the popularity of the ICS protocols based on the received traffic is as follows: Modbus, ICCP, DNP3, and IEC-104 in descending order. They drew two important inferences: i) Attackers are using the Shodan results to determine which ICS systems to attack, ii) Attackers send non-ICS-specific traffic to the standard ports of ICS protocols.

Simões, *et al.* [136] proposed a SCADA honeypot architecture for ICS which is extremely similar to the honeypot architecture proposed in their previous work [9]. Their architecture simulates ICS devices and supports both ICS-specific protocols and other protocols (i.e., SNMP, FTP). It also employs a port scan module to detect the reconnaissance activities, a firewall to prevent the honeypot from being used as an attack tool, an event monitoring module to observe the traffic and attacker interactions, and a management and watchdog module for configuration of the honeypot. As a Proof-of-Concept (PoC), they implemented two low-interaction Modbus honeypots on a Rapsberry Pi, one placed in an ICS network, and the second placed in a datacenter.

Ahn *et al.* [137] proposed a security architecture for SCADA ICS systems that uses a low interaction honeypot to detect a possible intruder and performs ARP poisoning attack to poison the ARP table of the attacker later on. The authors did not provide any implementation details or any performance evaluation results.

Belqruch and Maach [138] implemented a Kippo [52] honeypot for brute-force SSH attacks on an SCADA ICS. They simulated attacks via Kali Linux tools.

B. Honeypots and Honeynets for Smart Grid

In this category, we give brief overview of the smart grid honeypots and honeynets. Table X provides a list of the smart grid honeypots.

CryPLH: Buza *et al.* [140] proposed CryPLH, a low interaction and a virtual Smart-Grid ICS honeypot simulating Siemens Simatic 300 PLC devices. CryPLH uses NGINX and miniweb Web servers to simulate HTTP(S), a Python script to simulate Step 7 ISO-TSAP protocol and a custom SNMP implementation. The authors deployed the honeypot within the university's IP range and observed scanning, pinging, and SSH login attempts.

SHaPe: Kołtyś and Gajewski proposed a low-interaction honeypot, namely SHaPe [141], for electric power substations. SHaPe is capable of emulating any IEDs in an electric power substation that is compliant with IEC 61850 standard. The proposed honeypot extended the general purpose open-source Dionaea honeypot by means of libiec61850 library.

GridPot: Redwood *et al.* [142] proposed a symbolic honeynet framework, namely SCyPH, for SCADA systems. The proposed framework aims to incorporate emulated SCADA system components with physics simulations and employ anomaly detection systems based on the changes on the data obtained from the physics simulation. In their demonstration, namely GridPot, the authors utilized GridLab-D simulator [45] for electric substation simulations and IEC 61850-based communication, and implemented Newton-Raphson power flow solver algorithm for the voltage and current flow between the actors. They utilized Conpot to emulate IEDs and also implemented GOOSE/MMS and Modbus protocols for the interactions between the devices.

TABLE X
LIST OF SMART GRID HONEYPOTS AND HONEYNETS

Honeypots	Interaction Level	Simulated Services
CryPLH [140]	Low	HTTP(S), SNMP, Step7 ISO-TSAP
SHaPe [141]	Low	IEC 61850 MMS, HTTP, FTP, SMB
GridPot [142]	Hybrid	IEC 61850 GOOSE/MMS, Modbus, HTTP
Scott [8]	Low	Modbus/TCP, HTTP, SNMP
Mashima et al. [143]	Low	IEC 60870-5-104, IEC 61850, SSH
Pliatsios et al. [144]	Low	Modbus/TCP
Mashima et al. [145]	Low	TCP port listener on IEC 61850 MMS, S7comm, Modbus/TCP, Niagara Fox, EtherNet/IP, IEC 60870-5-104, DNP3, BACnet

Kendrick and Rucker [146] deployed GridPot in their thesis to analyze the threats to smart energy grids. Their honeypot deployment emulated Modbus TCP, S7comm, HTTP, and SNMP services. Although Shodan Honeyscore detected their deployment as a honeypot, a 19-day period of data collection showed that, GridPot received heavy HTTP scanning activities, over 600 Modbus, and 102 S7comm connections.

Scott [8] implemented a SCADA honeypot that uses the open-source Conpot honeypot to simulate a Scheider Electric PowerLogic ION6200 smart meter. They deployed the honeypot in a facilities network beside other SCADA components. They configured the honeypot to send its logs to a logging server, which alerts the network administrators based on the severity of the interactions that attackers are performing. Their honeypot supports Modbus, HTTP (for HMI), and SNMP.

Mashima *et al.* [143] proposed a scalable high-fidelity honeynet system for electrical substations in smart-grid environments. The proposed honeynet consists of a virtual substation gateway that supports the standardised smart-grid communication protocols (i.e., IEC 60870-5-104 and IEC 61850) and opens the entry point to the external attackers; virtual IEDs that are represented by Mininet [48] virtual hosts and SoftGrid [46] IED simulations; and simulation of smart grid components (e.g., circuit breakers, transformers, etc.) via POWERWORLD [47] power simulator. The proposed honeynet is highly scalable and resistant to fingerprinting against Shodan and attacker tools such as Nmap.

Hyun [147] used Conpot honeypot to discover the compromise attempt indicators for ICS environments. She configured Conpot to simulate a Siemens S7-200 PLC in an electric power plant. The simulated instance supported HTTP, Modbus/TCP, S7comm, SNMP, BACnet, IMPI, and EtherNet/IP services. The deployment of the honeypot outside of the university's network for four months revealed that popular choices for compromise attempts were HTTP, Modbus, and S7comm services.

TABLE XI
LIST OF ICS HONEYPOTS FOR WATER SYSTEMS

Honeypots	Interaction Level	Simulated Services
Wilhoit [148]	Hybrid	Modbus/TCP, HTTP, FTP
Antonioli et al. [149]	High	EtherNet/IP, SSH, Telnet, VPN
Murillo et al. [150]	Low	EtherNet/IP
Petre and Korodi [151]	Medium	Modbus
MimePot [152]	High	Modbus/TCP

Pliatsios *et al.* [144] proposed a honeypot system for SmartGrid which is based on the Conpot honeypot framework. The proposed honeypot consists of real Human-Machine Interface HMI and real Remote Terminal Unit RTU devices, and two virtual machines, one for virtual HMI and the other for a Conpot-based honeypot emulating an RTU device. The Conpot honeypot uses the real traffic generated by the real RTU device in order to make the attackers believe that they are interacting with a real ICS device.

Mashima *et al.* [145] deployed low interaction smart-grid honeypots in five geographic regions via Amazon cloud platform and analyzed the traffic coming to the honeypots for six months. They did not use open-source honeypot frameworks in order to avoid fingerprinting by attackers. Instead, they set up TCP listeners on several ports for ICS protocols. They realized that their honeypot instances received SYN-flooding DoS attack on IEC 61850 and S7comm protocols' port and also scanning activity for DNP3 and Modbus/TCP protocols. Their analysis showed that the same group of attackers, using the same IP addresses, was targeting smart grid devices on their honeypot instances around the world and sometimes an attack targeting a specific honeypot instance was applied to another instance the following week.

C. Honeypots and Honeynets for Water Systems

In this category, we give brief overview of the honeypots and honeynets for water systems. Table XI provides a list of the water system honeypots.

Wilhoit [148], [153] deployed high and low interaction honeypots to understand the sources and motivations of attacks targeting ICS environments. His honeypot system mimicked a water pressure station. For high interaction honeypots, he used Nano-10 PLC and Siemens Simatic PLC. As low interaction honeypots, he created virtual HMI instances which look like controlling PLCs of an ICS. The low-interaction honeypots were deployed on Amazon EC2 cloud environments around the world.

Antonioli *et al.* [149] proposed a virtual high interaction honeypot for ICS that is based on the MiniCPS ICS simulation framework [43]. The proposed design separates the honeypot system from the real ICS, and places virtual VPN, Telnet and SSH servers as the entry points for attackers to the honeypot. Network, ICS devices and physical process simulations/emulations are performed utilizing MiniCPS framework. In addition, the authors considered to manage the bandwidth, delay, and packet loss of the emulated links in the honeypot via Tc program, and enabled EtherNet/IP communication via cpppo Python library. As a PoC , the authors implemented a water treatment ICS.

A virtual testbed environment for ICS which can be used to deploy ICS honeypots was proposed by Murillo *et al.* [150]. The presented virtual testbed environment which uses MiniCPS [43] pays attention to realistic mathematical modeling of the ICS plants and the response time of the simulated ICS devices. The authors added a nonlinear plant model to MiniCPS to create a realistic ICS plant. An emulated network of a nonlinear control system which represents three water tanks, sensors, actuators and PLC devices was developed. In addition, the authors simulated a bias injection attack on the control system and proposed a mitigation mechanism.

Petre and Korodi [151] proposed a solution for protecting water pumping stations from threats using a honeypot inside an Object Linking and Embedding Process Control (OPC) Unified Architecture (UA) wrapping structure. OPC UA [154] is a middleware that can be used to interface standard ICS protocols (e.g., Modbus) to Service Oriented Architecture (SOA) systems and Web services. The proposed honeypot uses Node-RED library to simulate a system consisting of two water pumps and two water tanks and runs in an OPC UA Wrapper.

Bernieri *et al.* [152] presented a model-based ICS honeypot, namely MimePot, that utilizes Software Defined Network SDN for traffic redirection and network address camouflage for the real devices. The proposed honeypot simulates the ICS components and control routines based on the Linear Time Invariant model. The authors provided a water distribution PoC implementation which used a simulated attacker that injects and modifies the communication between honeypot elements.

D. Honeypots and Honeynets for Gas Pipelines

Wilhoit and Hilt developed a low-interaction virtual honeypot, namely GasPot [155], for gas-tank-monitoring systems. Their honeypot represented a virtual Guardian AST gas-tank-monitoring system. Based on their deployments with physical IP addresses in seven countries around the world, they realized reconnaissance attempts and DDoS attacks were performed by attackers.

Zamiri-Gourabi *et al.* [35] proposed an enhanced version of GasPot honeypot for ICS. Their upgrade applied patches to GasPot so that it will not be detected as a honeypot on the Internet. They fixed the incomplete command support for ATG protocol, made response times more realistic, and patched the problem of responding with static inventory values and the output formatting issue which can help an attacker to understand that it is a honeypot.

E. Honeypots and Honeynets for Building Automation Systems

Litchfield *et al.* [41] stated that high interaction honeypots are unsuitable for CPS due to safety risks, costs, and limitations with the usefulness of the honeypot without the physical part of the CPS. Therefore, they suggested the use of hybrid interaction honeypots in which real CPS devices

interact with the simulation of the physical part of the CPS, and proposed HoneyPhy. HoneyPhy consists of three modules: Internet interface(s), process model(s), and device model(s). A PoC implementation of HoneyPhy was given where a Heating, Ventilation and Air Conditioning (HVAC) honeypot is constructed by means of a physical SEL-751A relay, a black-box simulation model of a physical relay and a heating and cooling process simulation model. The extendability of the proposed honeypot framework for other CPS applications is limited since device and process models for the corresponding CPS application are needed.

F. Honeypots and Honeynets for IIoT

Ammar and AlSharif [156] proposed a model called HoneyIo3, composed of three honeynets carried out with three Raspberry Pi devices with Linux OS and Honeeepi sensor, that mimic IIoT/ICS services. Services/Protocols used in HoneyIo3 model are IPMI, S7comm, HTTP, Kamstrup, SNMP and SSH.

Du and Wang [157] focused on DDoS attacks on SDNs in IIoT environments. They identified a new kind of attack that could identify a honeypot being used in an SDN and disable it. Analyzing attacker strategies, they presented a pseudo-honeypot game strategy to dynamically protect SDNs. The evaluation was performed on a testbed using servers and hybrid honeypots, and showed that the proposed strategy can protect against DDoS attacks.

VIII. Taxonomy of Honeypots and Honeynets for IIoT and CPS

Honeypots and honeynets proposed for IIoT and CPS are listed in Table XII and the tools, implementation, and attack type details of the corresponding honeypots and honeynets are also outlined in Table XIII. In this section, we consider all of the proposals for IIoT and CPS and provide an overview of these studies based on the development of research over time, common characteristics, scalability, simulated services, most commonly used tools, availability of the source codes, and the most common attacks.

A. Development of Research Over Time

We analyzed the studies and depicted the development of research over time and also the inheritance relationship between the honeypots and honeynets for IIoT and CPS in Figure 5. As shown in the figure, honeypots and honeynets for IIoT and CPS started with the SCADA HoneyNet [104] project of Pothamsetty and Franz from Cisco Systems in 2004. This project was followed by Digital Bond's SCADA Honeynet [105] in 2006. In terms of the honeypot and honeynet research for IIoT and CPS systems in the literature, we can see that Berman's thesis [132] in 2012 was the first study. His thesis was followed by another thesis conducted by Jaromin [133] the following year. It is interesting to note that both studies were performed in the U.S. Air Force Institute of Technology. This also corresponds to a time in which notorious malware (i.e., Stuxnet (2010), DuQu (2011), Night Dragon (2011) and Flame (2012)) appeared in the wild against nations' critical infrastructure environments, and quickly grabbed the attention of military/defense organisations. In the same year, 2013, the most popular ICS honeypot - Conpot [108] started and Wilhoit from Trend Micro Research published the white paper of their low interaction ICS honeypots [148]. After these works, honeypot and honeynet research and practice in IIoT and CPS gained a momentum.

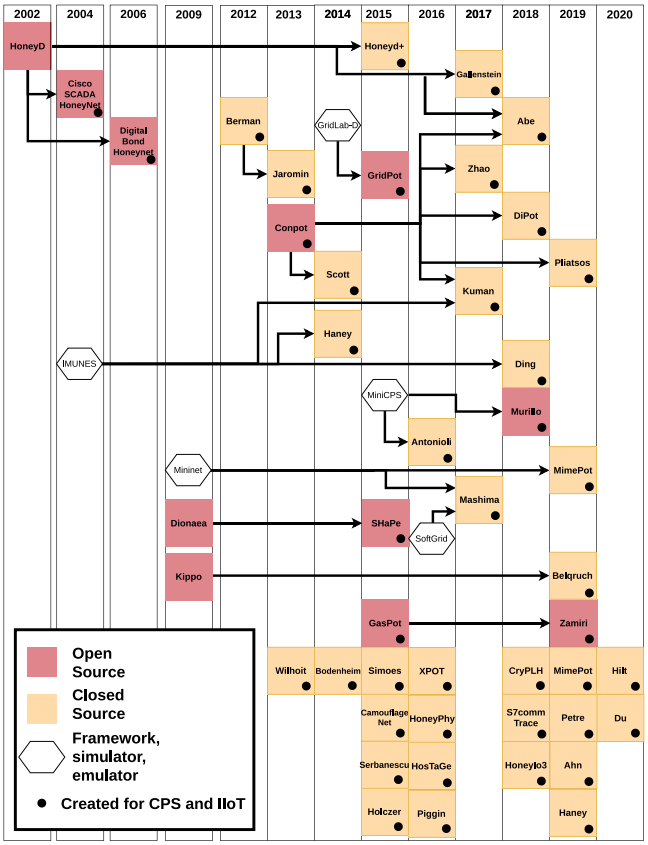

Fig. 5. Evolution of Inheritance for the honeypots and honeynets of IIoT and CPS.

As shown in Figure 5, more than one-third of works have a form of inheritance relationship with each other, where a honeypot is built based on another. In this respect, Conpot [108] is the leading honeypot, as six honeypots were developed based on Conpot (this number does not include the studies that do not extend Conpot but only use it). The underlying reasons can be manifold. Conpot is open-source and is still being actively maintained. It supports several industrial and non-industrial protocols. In addition, it is being developed under the umbrella of Honeynet Project [109], which has a significant background with honeypots such as Honeyd, Honeywall CDROM, Dionaea and Kippo.

In addition to extending the existing honeypots, researchers also considered to employ simulators, emulators, or frameworks as the main building block for their studies. As Figure 5 shows, Mininet and IMUNES emulators, GridLab-D, SoftGrid and POWERWORLD simulators, and MiniCPS framework were utilized in a number of honeypot/honeynet studies.

Apart from extending honeypots or using simulators, emulators and frameworks, we can see that half of the studies proposed independent honeypots. This may be due to the

TABLE XII
CLASSIFICATION OF HONEYPOTS AND HONEYNETS FOR IIOT AND CPS

Work	Year	Level of Interaction	Scalability	Resource level	Simulated services	Role	Application
CISCO [104]	2004	Low	✓	Virtual	Modbus/TCP, Telnet, HTTP, FTP	Server	ICS
Digital Bond [105]	2006	Low	✓	Virtual	Modbus/TCP, Telnet, HTTP, FTP, SNMP	Server	ICS
Conpot [108]	2013	Low	✓	Virtual	IEC 60870-5-104, BACnet, EtherNet/IP, Guardian AST, Kamstrup, Modbus, S7comm, HTTP, FTP, SNMP, IPMI, TFTP	Server	ICS
Zhao and Qin [110]	2017	Medium	✓	Virtual	S7comm, Modbus, SNMP, HTTP	Server	ICS
DiPot [111]	2018	Low	✓	Virtual	HTTP, Modbus, Kamstrup, SNMP, IMPI, BACnet, Guardian AST, S7comm	Server	ICS
CamouflageNet [121]	2015	Low	✓	Virtual	N/A	Server	ICS
XPOT [112]	2016	Medium	✓	Virtual	S7comm, SNMP	Server	ICS
HosTaGe [113]	2016	Low	✓	Virtual	Modbus, S7comm, HTTPS, FTP, MySQL, SIP, SSH, SNMP, HTTP, Telnet, SMB and SMT	Server	ICS
S7CommTrace [114]	2018	Medium	✓	Virtual	S7comm	Server	ICS
Disso et al. [123]	2013	Hybrid	Limited	Hybrid	N/A	Server	ICS
Honeyd+ [115]	2015	High	Limited	Hybrid	EtherNet/IP, HTTP	Server	ICS
Gallenstein [116]	2017	Low	✓	Virtual	EtherNet/IP. ISO-TSAP, HTTP	Server	ICS
Abe et al. [117]	2018	Low	✓	Virtual	Modbus, S7comm, BACNet, IPMI, Guardian AST, HTTP, SNMP	Server	ICS
Haney et al. [127]	2014	Low	✓	Virtual	Modbus/TCP, Telnet, SSH, HTTP(S)	Server	ICS
Kuman et al. [128]	2017	Low	✓	Virtual	Modbus/TCP	Server	ICS
Ding et al. [129]	2018	Medium	✓	Virtual	S7comm, SNMP	Server	ICS
Bodenheim [130]	2014	High	Limited	Physical	HTTP, EtherNet/IP, SNMP	Server	ICS
Piggin et al. [39]	2016	High	X	Physical	SSH, HTTP, RDP	Server	ICS
Haney [131]	2019	High	✓	Hybrid	Modbus/TCP, SSH, Telnet, SNMP, HTTP	Client, Server	ICS
Hilt et al. [40]	2020	High	X	Hybrid	S7comm, Omron FINS, EtherNet/IP, VNC	Server	ICS
Berman [132]	2012	Low	Limited	Virtual	Modbus/TCP	Server	ICS
Jaromin [133]	2013	Low	Limited	Virtual	Modbus/TCP, HAP, HTTP	Server	ICS
Holczer et al. [134]	2015	High	✓	Virtual	S7comm, SNMP, HTTP(S)	Server	ICS
Serbanescu et al. [135]	2015	Low	✓	Virtual	DNP3, IEC-104, Modbus, ICCP, SNMP, TFTP, XMPP	Server	ICS
Simões [136]	2015	Low	✓	Virtual	Modbus, SNMP, FTP	Server	ICS
Ahn et al. [137]	2019	Low	✓	Virtual	Modbus	Server	ICS
Belqruch et al. [138]	2019	Medium	✓	Virtual	SSH	Server	ICS
SHaPe [141]	2015	Low	✓	Virtual	IEC 61850 MMS, HTTP, FTP, SMB	Server	Smart Grid
GridPot [142]	2015	Hybrid	✓	Virtual	IEC 61850 GOOSE/MMS, Modbus, HTTP	Server	Smart Grid
Scott [8]	2014	Low	✓	Virtual	Modbus/TCP, HTTP, SNMP	Server	Smart Grid
Mashima et al. [143]	2017	Medium / High	✓	Virtual	IEC 60870-5-104, IEC 61850, SSH	Server	Smart Grid
CryPLH [140]	2018	Low	✓	Virtual	HTTP(S), SNMP, Step7 ISO-TSAP	Server	Smart Grid
Pliatsios et al. [144]	2019	Low	Limited	Hybrid	Modbus/TCP	Client, Server	Smart Grid
Mashima et al. [145]	2019	Low	✓	Virtual	TCP port listener on IEC 61850 MMS, S7comm, Modbus/TCP, Niagara Fox, EtherNet/IP, IEC 60870-5-104, DNP3 and BACnet ports	Server	Smart Grid
Murillo et al. [150]	2018	Low	✓	Virtual	EtherNet/IP	Server	Water System
Petre et al. [151]	2019	Medium	✓	Virtual	Modbus	Server	Water System
Wilhoit [148]	2013	Hybrid	Limited	Hybrid	Modbus/TCP, HTTP, FTP	Server	Water System
Antonioli et al. [149]	2016	High	✓	Virtual	EtherNet/IP, SSH, Telnet, VPN	Server	Water System
MimePot [152]	2019	High	✓	Virtual	Modbus/TCP	Client, Server	Water System
GasPot [155]	2015	Low	✓	Virtual	N/A	Server	Gas System
Zamiri et al. [35]	2019	Medium	✓	Virtual	Veeder-Root ATG	Server	Gas System
HoneyPhy [41]	2016	Hybrid	X	Hybrid	DNP3	Server	Building Auto.
HoneyIo3 [156]	2018	Hybrid	✓	Hybrid	IPMI, S7comm, Kamstrup, SNMP, HTTP(S), Ntopng, SSH	Server	IIoT
Du and Wang [157]	2020	Hybrid	✓	Virtual	Not identified	Server	IIoT

shortcomings of existing honeypots to support CPS and IIoT environments or fingerprintability of them from attackers' point of view.

B. Common Characteristics

Honeypots and honeynets proposed for IIoT and CPS have several characteristics in common.

TABLE XIII
SUMMARY OF TOOLS, IMPLEMENTATION, AND ATTACK TYPES OF HONEYPOTS AND HONEYNETS FOR IIOT AND CPS

Work	Tools	Ports	Attack Types	Data Analyzed	Length of the Study
CISCO [104]	N/A	Modbus/TCP (502), Telnet (23), HTTP (80), FTP (21)	N/A	N/A	N/A
Digital Bond [105]	Sebek, Argus, Walleye, Snort IDS	Modbus/TCP (502), SNMP (161), Telnet (23), HTTP (80), FTP (21)	N/A	N/A	N/A
Conpot [108]	N/A	IEC 60870-5-104 (2404), BACnet (47808), EtherNet/IP (44818), Guardian AST (10001), Kamstrup (1025, 50100), Modbus (502), S7comm (102), HTTP (80), FTP (21), SNMP (161), IPMI (623), TFTP (69)	N/A	N/A	N/A
Zhao and Qin [110]	Flask framework, Wireshark	N/A	Traffic from 244 IP addresses from 34 countries	Types, sources, requests from IPs	43 days
DiPot [111]	N/A	N/A	Modbus and Kamstrup scan, Modbus over-length packets	Access sequences to protocols and their IPs	6 months
Camouflage [121]	Nmap, Kali Linux	N/A	Scanning	N/A	N/A
XPOT [112]	Nmap, nfqueue	N/A	N/A	S7comm handshakes and queries	1 month
HosTaGe [113]	Wireshark, Bro IDS, snp4j	Modbus (502)	Multi-stage attacks consisting of different scanning and attack attempts	Attacks to Modbus, S7comm, HTTP, Telnet and IP addresses targeting HosTaGe and Conpot	12 weeks
S7CommTr [114]	N/A	S7comm (102)	N/A	Indexing in Shodan, valid and invalid requests, function coverage of S7comm, received IP address diversity	60 days
Disso et al. [123]	N/A	N/A	N/A	Link latency, network traffic counting and connection limiting, background network traffic	N/A
Honeyd+ [115]	Nmap, Zenmap, Wget	EtherNet/IP (44818, 2222), HTTP (80)	Scanning	Fingerprints of Honeyd+ hosts, error rates and protocol data rates	N/A
Gallenstein [116]	Nmap, Shodan Honeyscore, RSLinx, STEP7, Wget, Wireshark	EtherNet/IP (44818), ISO-TSAP (102), HTTP (80)	Scanning	Nmap fingerprint similarity, Honeyscore performance, RSLinx and STEP7 PLC module discovery performance, comparison of responses to Wget requests	N/A
Abe et al. [117]	Nmap	Modbus (502), S7comm (102), BACNet (47808), IPMI (623), Guardian AST (10001), HTTP (80), SNMP (161)	Havex RAT, Modbus Stager, PLC blaster	Behavior against Havex RAT, Modbus Stager and PLC blaster attacks	N/A
Haney et al. [127]	IMUNES, JAMOD Library, Snort IDS, Snort daemon logger, Sebek, Honeywall	Modbus/TCP (502), HTTP (80), HTTPS (443), Telnet (23), SSH (22)	Network and port scan, Modbus packet capture, injection and out of band packets	N/A	N/A
Kuman et al. [128]	OSSEC host-based IDS, PLCScan, Shodan, iptables	Modbus/TCP (502)	Port scans on Modbus and HTTP protocols	Conpot logs	2 weeks
Ding et al. [129]	Nmap, snmpwalk, STEP7 software, PLCscan	S7comm (102)	Scanning	Scanning result	N/A

In terms of purpose of the honeypots, we can see that the majority of the honeypots and honeynets outlined in Table XII and Table XIII have research purposes. The only studies which have production purposes are Antonioli *et al.* [149], Piggin and Buffey [39], and Scott [8]. This is understandable since IIoT and CPS environments have unique features that make it hard for security tools including honeypots to be actively deployed in such areas. Equipments in SCADA environments work continuously, and interruptions and downtimes are highly refrained from [8], [9]. In addition to this,

TABLE XIII
(Continued.) SUMMARY OF TOOLS, IMPLEMENTATION, AND ATTACK TYPES OF HONEYPOTS AND HONEYNETS FOR IIOT AND CPS

Work	Tools	Ports	Attack Types	Data Analyzed	Length of the Study
Bodenheim [130]	Nmap, TCPdump, SSH, Tshark, Wireshark, Shodan API, Security Onion Linux, Snort, netcat	EtherNet/IP (44818), HTTP (80) SNMP (161)	Scanning	Shodan's functionality and indexing, effect of being indexed on the received traffic, effect of modifying device service banners	55 days
Piggin et al. [39]	Google Dorks	N/A	Scanning, password attack, execute malicious program, SSH brute-force, an attack originated from TOR network, DoS on the PLC	Origin and target protocols of the attacks	N/A
Haney [131]	SecurityOnion, iptables, SnortSam, Sebekd, Argus, JAMOD, IMUNES, LabVIEW, Matlab Simulink	Modbus/TCP (502), HTTP (80), SSH (22), SNMP (161)	Modbus scanning via Shodan, brute-force login	The most common usernames and passwords used for attacks, attack origins	2 weeks
Hilt et al. [40]	Tshark, Moloch, Chaosreader, VNCLogger, Suricata, Syslog	S7comm (102), Omron FINS (9600), EtherNet/IP (44818), VNC (5900, 5901)	Scanning, ransomware, malicious cryptomining, robotic workstation beaconing attempt	Unique IP addresses, amount of traffic, protocol-specific traffic and commands to PLCs, communication with scanners, VNC screen recording, attacker's downloads	7 months
Berman [132]	Nmap, Wireshark, SSH, TCPDump, Syslog, Triangle MicroWorks Protocol Test Harness	Modbus/TCP (502)	Scanning, invalid ICS traffic	Modbus/TCP traffic tests, response statistics, fingerprint analysis, response to invalid ICS traffic, logging capabilities	N/A
Jaromin [133]	Nmap, Metasploit, NetEdit3, DirectSOFT5, iptables and netfilter modules, libpcap library, HAP API, Syslog	Modbus/TCP (502), HAP (28784), HTTP (80)	Brute-force password guessing, fingerprinting	Packet level accuracy and logging capability, OS fingerpring accuracy, Metasploit attack performance, response timing	N/A
Holczer et al. [134]	Step7, Szilu SSL, MiniWeb, iptables	S7comm (102), HTTP (80), HTTPS (443), SNMP (161)	Pings, port scans, SSH scans	Attack origins, logs	50 days
Serbanescu et al. [135]	Snort, Matlab, Amazon EC2 environment	N/A	Scanning	Modbus traffic (connections, requests, port scans, activity types, country of origin), impact of Shodan listing the devices, attractiveness of ICS protocols	28 days
Simões [136]	Modbus-tk, Pymodbus and Libpcap libraries, NET-SNMP, VSFTPd	N/A	N/A	Resource usage of honeypot, response time, reliability	N/A
Ahn et al. [137]	N/A	Modbus (502)	ARP poisoning	N/A	N/A
Belqruch et al. [138]	Kali Linux	SSH (22)	SSH brute force	Username-password combinations, password attempts	N/A
SHaPe [141]	libiec61850	N/A	N/A	TCP connection information (connection ID, source and destination IPs and ports), Dionaea logs	N/A
GridPot [142]	ETSY Skyline project anomaly detection modules, GridLab-D, hpfeeds logging	N/A	IED switching attack	Physics impact of the attack	N/A
Scott [8]	Tenable Nessus, Splunk Enterprise, Rsyslog	Modbus/TCP (502), HTTP (80), SNMP (161), Syslog (514), Splunk (8000), SMTP (25)	Scanning attack	Alerts generated by Splunk	N/A

industrial devices typically have real-time constraints with guaranteed response times [134]. For these reasons, it is very difficult to insert a honeypot in an ICS production environment which may affect the ICS communication and has the danger of being compromised (if it is a high-interaction honeypot).

TABLE XIII
(Continued.) SUMMARY OF TOOLS, IMPLEMENTATION, AND ATTACK TYPES OF HONEYPOTS AND HONEYNETS FOR IIoT AND CPS

Work	Tools	Ports	Attack Types	Data Analyzed	Length of the Study
Mashima et al. [143]	VirtualBox, Mininet, SoftGrid and POWERWORLD simulators, SOCAT port forwarding, OpenMUC	IEC 60870-5-104 (2404), IEC 61850 (102), SSH (22)	Nmap scan, Shodan	Fingerprinting, latency, scalability and cost analysis	N/A
CryPLH [140]	Nessus, Nmap, Backtrack Linux, Miniweb, NGINX, SNMPWalk	ISO-TSAP (102), HTTP (80), HTTPS (443), SNMP (161)	Attack tests with Backtrack Linux (Kali Linux), Nmap, nessus	Honeypot logs	38 days
Pliatsios et al. [144]	Wireshark, Tshark	N/A	N/A	N/A	N/A
Mashima et al. [145]	Wireshark, ELK stack, Amazon Cloud	IEC 61850 MMS and S7comm (102), Modbus/TCP (502), Niagara Fox (1911, 4911), EtherNet/IP (ENIP) (2222, 44818), IEC 60870-5-104 (2404), DNP3 (19999, 20000), BACnet (47808)	SYN-flooding DoS, scanning	Access trends, protocol specific attempts, correlation of honeypots' data, attack origin dynamics	6 months
Murillo et al. [150]	Mininet, MiniCPS, Odeint solver	N/A	Bias injection attack	Tank levels and plant behavior without attack, with attack and defense	N/A
Petre et al. [151]	Node-RED, Softing OPC UA Client, SQLite	N/A	Unauthorized access	Database entries	N/A
Wilhoit [148 [153]	Snort, tcpdump, Pastebin, Amazon EC2	Modbus/TCP (502), HTTP (80), FTP (21)	Scanning, spearphishing, unauthorized access and modification, Modbus traffic modification, CPU fan speed modification on the water pump, malware exploitation	Attack types and origins	28 days
Antonioli et al. [149]	Mininet, MiniCPS, ocserv VPN, sshd, telnetd, tc link shaping, cpppo EtherNet/IP emulation	Ethernet/IP (44818), HTTP (80), SSH (22), Telnet (23)	DoS, Man in the Middle, port scan, service enumeration, physical process attacks (i.e., tank overflow)	Network metrics (address, packet loss, delay, bandwidth, topology, protocols, etc.) and physical metrics (realistic mathematical model, sensor and actuator operations, etc.)	Capture the Flag Competition
MimePot [1	Mininet, Scapy	N/A	Man in the Middle and integrity attack	Tank water levels, Mime Estimation and Control status by time, water pump status, flows between tanks	N/A
GasPot [155]	N/A	N/A	Reconnaissance, DDoS	Connection attempts, commands, attack origins	N/A
Zamiri et al. [35]	Nmap	Veeder-Root ATG (10001)	N/A	N/A	N/A
HoneyPhy [41]	OpenDNP3 library, LabVIEW	N/A	N/A	Heating and cooling curve from both physical system and the process model	N/A
Du and Wang [157]	SDN testbed	N/A	DDoS attacks, SYN Flood attack, FTP flow	Protocols, packets per port, packet characteristics	N/A
HoneyIo3 [156]	Shodan, Nmap	IPMI (623), S7comm (102), Kamstrup (1025), SNMP (161), HTTP (80), HTTPS (443), ntopng (3000), SSH (9002)	Reconnaissance attacks	Protocols, packets per port, packets characteristics	N/A

Considering the roles of honeypots, we see that the overwhelming majority of the proposals have server roles. The honeypots and honeynets that have components which act like clients are Haney [131], Pliatsios *et al.* [144], and MimePot [152].

Linux is by far the most popular operating system environment choice of honeypot and honeynet developers. Apart from Linux, we see that only Haney and Papa [127] used FreeBSD.

In terms of the programming languages used for the development of honeypots and honeynets for IIoT and CPS, we

note that Python is the most favored one. Aside from Python, C/C++ and Java are also used by the studies. We believe that this has a relation with the library support that these languages have for industrial protocols. In this regard, Modbus-tk, pymodbus and cpppo EtherNet/IP libraries of Python; libiec61850 and OpenDNP3 libraries of C/C++ and JAMOD Modbus library of Java are utilized by the developers in the studies. In addition, Conpot - the most popular open-source honeypot for IIoT and CPS is also written in Python.

C. Level of Interaction

Honeypots and honeynets proposed for IIoT and CPS environments exhibit all possible interaction levels. In this respect, as Table XII shows, half of the works allow low interaction capabilities to an attacker. On the other hand, numbers of medium, high and hybrid interaction honeypots are almost equal to each other. We had to make a decision on setting the interaction level for some of the studies since not every author explicitly stated that information in their proposals. Low interaction honeypots in IIoT and CPS systems can provide valuable information in terms of scanning, target protocol, attack origin and brute-force attempts. On the other hand, it is possible to see other more advanced attacks and industrial protocol and process specific attacks only with medium and high interaction honeypots. However, one has to be extremely careful when deploying a high interaction honeypot especially in IIoT and CPS environments since they allow attackers to compromise the system and then apply other operations using the honeypot (e.g., downloading malware, trying to exploit other devices on the same network, performing attacks on behalf of the attacker).

D. Resource Level

In terms of resource levels of honeypots and honeynets for IIoT and CPS, we can see that most of the decoy systems use virtual resources. However, honeypots and honeynets utilizing real industrial devices and a combination of real and virtual devices also exist. One of the biggest driving factor for researchers to propose virtual honeypots may be the high cost of actual industrial devices. As several researchers ([115], [116], [143] and [158]) highlighted, components of an industrial system such as PLCs have high costs in the order of tens of thousands of dollars.

E. Scalability

The majority of the honeypots and honeynets for IIoT and CPS have scalable designs. This is also related to these honeypots having virtual resources. As we explained in Section IV, physical honeypots are difficult to scale as they need more physical resources, and real industrial environments can have several industrial devices. For instance, Mashima *et al.* [143] noted the number of substations in a power grid in Hong Kong as 200. In order to propose a realistic decoy system, scalable honeypot design gains importance.

F. Target IIoT and CPS Application

As target application areas of the existing honeypots, we can state that more than half of the works targeted ICS environments. However, considerably fewer decoys exist for specific CPS and IIoT applications such as smart grid, water, gas, and building automation systems. Although the majority of the studies are for ICS, we would like to note the fact that the similar industrial devices (e.g., PLCs) can be used both by ICS and smart infrastructures (e.g., grid, water, gas).

G. Industrial Process Simulations

In terms of industrial process simulations, we see that only five studies considered to employ some form of simulations. For water management CPS environments, Antonioli *et al.* [149] used equation of continuity from hydraulics and Bernoulli's principle for the trajectories (for drain orefice), Murillo *et al.* [150] utilized a nonlinear model with Luenberger observer, and Bernieri *et al.* [152] employed linear time invariant model for plant simulation. GridPot [142] made use of Newton-Raphson power flow solver for electrical grid process. Lastly, for building automation systems, Litchfield *et al.* [41] considered Newton's Law of Cooling for the building process model.

H. Simulated Services

Honeypots and honeynets for IIoT and CPS support a wide variety of protocols and services that are both specific and not specific to industrial environments. The protocols and services supported by the honeypots and honeynets are shown in Table XII while ports that are exposed for such protocols in the honeypots are outlined in Table XIII. When we consider the protocols, we can see that Modbus, HTTP, SNMP, and S7comm are the most popular protocols among the studies. Our findings are also validated by a number of researchers [3], [127], [135], [144], [151], [159], [160] who cite Modbus as the most widely used industrial protocol. Popularity of industrial protocols along with number of honeypots supporting them can be expressed as follows: Modbus (22), S7comm (12), EtherNet/IP (8), IEC 60870-5-104 (4), BACnet (4), Kamstrup (4), DNP3 (3), Guardian AST (3), IEC 51850 (3), and ISOTSAP (2). The popularity of non-industrial protocols, HTTP and SNMP are very reasonable. HTTP is used as the interface of HMIs of industrial systems [8], [149] and also it enables the remote configuration of industrial components such as PLCs [113], [134]. For these reasons, it is stated as the target of scanning activities performed by malicious entities [146]. SNMP on the other hand is used for monitoring and management purposes in industrial environments [115], [134].

I. Availability of Open-Source Honeypot and Honeynet Solutions

There exist eight honeypot and honeynet studies that provide their implementation openly. In this respect, CISCO SCADA HoneyNet [104] source code is still available. However, the last shared version was in 2015. Unfortunately, Digital Bond's

SCADA Honeynet [105] is not reachable right now. Conpot on the other hand, is open-source and is still being actively maintained. Considering the rest of the honeypot and honeynet studies, only the honeypot of Zamiri-Gourabi *et al.* [35] is actively maintained. However, their study was performed in 2019 and it is not known if they will continue to actively maintain it. The implementations of GridPot [142] and SHaPe [141] are still available, but their last update was in 2015. The last update for GasPot [155] was in 2016, and honeypot-like testbed of Murillo *et al.* [150] was maintained in 2018.

J. Most Commonly Used Tools

The most commonly used tool for IIoT and CPS honeypot and honeynet studies is Nmap, which is followed by Wireshark, Snort IDS, Shodan tools, Mininet, iptables, tshark, TCPDUMP, and syslog. Researchers used Nmap to obtain fingerprints of their honeypots and to identify the exposed ports. Wireshark was used for traffic capture and analysis. Snort IDS is used for attacker control attempts especially in honeywall configurations. Shodan tools were used to find out indexing information, honeypot's fingerprint from Shodan's point of view, and also to find out if Shodan detects the decoy system as a honeypot or not.

K. Most Common Attacks

The most commonly detected/tested attacks in IIoT and CPS honeypots/honeynets are scanning attacks. Majority of the studies detected/tested scanning attacks to the IIoT and CPS environments. In addition to DoS/DDoS, SSH, brute-force, and Man-in-the-Middle attacks were also detected/tested in the proposed honeypots and honeynets. Although less common than the mentioned attacks, ransomware, malicious cryptocurrency mining, malware and ICS specific attacks such as HAVEX RAT, PLC Blaster, and tank overflow attacks were also detected/tested in the proposed systems.

IX. Lessons Learned and Open Issues

Considering the honeypot and honeynets for IoT, IIoT, and CPS environments, we believe that it is crucial to stress the importance of key points. This is valuable to interpret the state-of-the-art and to motivate for further research and practice.

A. Lessons Learned

Any honeypot/honeynet developer and researcher for IoT, IIoT, and CPS needs to consider a few key factors at the very beginning of his/her work. The key factors that should be taken into account are target application area, purpose of the honeypot/honeynet, cost, deployment location, intended level of interaction with the attacker, resource level, services that will be provided, simulated, or emulated, and their realistic service to the attackers, tools that will be used, fingerprintability and indexing, and liability issues that may come up.

Target Application Area Selection: IIoT and CPS environments have their own characteristics which may affect the entire honeypot/honeynet design. Devices, communication channel characteristics, protocols, traffic rates, application QoS requirements, and many other factors can be different for each unique application. CPS and IIoT devices have quite different characteristics from regular IoT devices. In addition, they work with industrial protocols which are not used in traditional ICT or IoT environments. Such industrial devices have life-times in the order of decades and work with real-time constraints which strictly require them to work without interruptions [9], [134]. Critical infrastructures of nations are controlled by such industrial devices. While typical IoT applications do not have any physical processes to be continuously monitored and controlled, it is very common for IIoT and CPS applications. For these reasons, it is extremely important to determine the target application and its characteristics.

Purpose of the Honeypot/Honeynet: The purpose of a honeypot or honeynet significantly affects the measures that need to be taken to ensure that attacks on the honeynet do not compromise the infrastructure on which it is implemented. In a research environment, this can be done by isolating the honeynet system. For example, by implementing it in a DMZ. However, if production honeypots are to be deployed in IIoT and CPS environments where industrial devices monitor and control critical plant processes, then extra care has to be given to the decoy system design. Such production honeypots in industrial environments need to ensure that they cannot be compromised by attackers, as well as ensure that they do not interfere with the communication and control processes (i.e., operational resources) of the existing industrial devices. In addition, one has to note that honeypots and honeynets do not stop attacks [8]. For this reason, the alerts or logs created by them have to be considered by administrators.

Deployment Location: While deployment location can have an important effect on honeypot activity, only twelve of the reviewed studies stated their deployment locations. Two CPS studies [130], [134] deployed their honeypots within the IP range of universities, which may call the attention of attackers who check the IP address spaces of their targets. Another two CPS studies [139], [145] and three IoT studies [74], [88], [93] chose cloud environments as deployment targets. Such an approach would provide a global view of attacks to honeypot/honeynet owners and also may be more attractive to attackers than the university option. However, attackers can still find out that the target system operates within the IP range of a cloud provider. Additionally, two CPS studies [40], [155] and three IoT studies [63], [86], [92] use public IP addresses which is the better option. In addition to this, Guarnizo *et al.* [92] identified that geographical location selection, in terms of country or city of deployment, or at least the location shown to attackers, is an important consideration. This is because attackers might seek to attack devices in certain cities if they are looking for a point to start targeted attacks or if they have an interest in reselling IPs after they are infected.

Cost: Cost is a crucial consideration in developing honeypots and honeynets. Setting up a honeypot or honeynet can be very expensive if physical resources and closed source tools are used instead of virtual resources and open source tools. Also, it is important to note that the PLCs, IEDs, RTUs, and RIOs used in industrial applications are considerably more expensive than Commercial of-the-shelf (COTS)

IoT devices. In addition, complexity of a honeypot, especially a honeynet, can be another contributing factor for the cost of the system. Complexity is directly proportional to the level of interaction provided and also the number of services/protocols supported. As the interaction level and number of supported services increase for honeypots and honeynets, higher fidelity data in high volume is collected, which requires more resources to store and process. Moreover, deployment locations can have an effect on the cost of the system. To be more specific, although deployment of a honeypot or a honeynet in a university IP address space can be cost efficient for research, it can easily call the attention of adversaries. Honeypot/honeynet deployments in cloud environments would be significantly more costly compared to university environments. However, attackers can still determine that the IP addresses are in the cloud provider space. The third option would be renting private IP addresses to avoid suspicion by attackers, but such an option can be more costly than the cloud option. For these reasons, honeypot/honeynet developers and researchers need to consider how resource and interaction levels as well as deployment environment and complexity affect the cost.

Level of Interaction Considerations: The level of interaction of a honeypot/honeynet affects many different aspects, as explained in Section IV. Considering the existing honeypots and honeynets for IoT, IIoT, and CPS, almost every possible level of interaction choice can be seen as reviewed in Section VI and Section VIII. However, high interaction is needed in order to identify complex attacks that may target IoT, IIoT, and CPS devices and understand possible effects on industrial processes and critical infrastructures. Although COTS IoT devices are more affordable, industrial devices in the order of thousands of dollars can be a significant issue to consider. Therefore, resource level choice and realistic simulation/emulation become important considerations. These are further discussed in the following categories.

Resource Level Selection: The question of whether real, simulated, or both types of devices are to be used in honeypots/honeynets for IoT, IIoT, and CPS is quite a vital one. Real devices can act as high-interaction honeypots and provide high fidelity information. In addition, they would be almost impossible to be detected as a honeypot by outsiders. However, as explained earlier, costs of real devices can change based on the target application area and constructing a realistic honeynet with a realistic number of industrial devices may cost a fortune. These important factors motivated researchers and developers to design honeypots/honeynets with virtual components. Virtualization enables scalability, heterogeneity, easy maintenance and cost-effective deployment of IoT, IIoT, and CPS honeypots. In this respect, Dang *et al.* [88] found that approximately 92.1% of malware-based attacks target multiple IoT device architectures and emphasized the need for a virtual IoT honeypot solution. At the same time, they identified that virtual honeypots attracted 37% fewer suspicious connections and 39% fewer attacks than physical honeypots. Also, the variety of attacks virtual honeypots captured were more than with physical honeypots. Dang *et al.* [88] also pointed out that a virtual honeypot costs 12.5x less to maintain than a physical honeypot. These factors should be weighed in considering honeypot/honeynet design. Balancing the benefits of both physical and virtual resources in a hybrid solution is an important consideration. In addition to this, the choice of which model of devices to select, either real or simulated, can play a factor in attracting attackers. Guarnizo *et al.* [92] identified that models with known vulnerabilities tend to be attacked more frequently.

Choice of Services to Provide/Simulate and Realism: Choice of services to provide or simulate, and ensuring realism in such services are very critical factors in honeypot/honeynet design. These considerations get even more important for IIoT and CPS systems. Which services will be provided? Is it logical to support all of the protocols and services in the target application area? If not, how to choose among the set of protocols/services? Scott [8] pointed out that honeypots and honeynets should simulate only the services that the mimicked device would usually accommodate. If the mimicked device does not have a certain service or does not support it, but the honeypot does, then attackers may realise that they are interacting with a decoy system. After determining the services/protocols to be supported, then comes another important aspect: realism.

One of the principal considerations when deploying a honeypot or honeynet system for IoT, IIoT, or CPS is how to simulate a real system effectively in order to avoid hackers and search engines from identifying that they are interacting with a decoy system. This is vital for the honeypot system to be able to attract attackers and to gather as much information as possible from their interactions. In order to avoid detection more effectively for a honeynet deployed in an IoT environment, Surnin *et al.* [33] recommended the following: a limited number of services should be run to simulate a more realistic environment, ping command host requests should yield an existing host, files created by attackers should not be deleted, commands for utilities should return a list of running processes, no hardcoded values should be used, simulated Linux utilities should have full functionality from the origin, and attacker file requests should be sent to a sandbox with a specified delay before checking them on external services such as VirusTotal. Zamiri-Gourabi *et al.* [35] pointed out the fact that default hardcoded configurations, missing features of the simulated services or protocols, unusual or unrealistic behaviors, fingerprintability of the hosting platform and response times can be the possible fingerprints of honeypots and honeynets. Simulations of plant processes in a realistic way comes to the scene for IIoT and CPS honeypots/honeynets. Unfortunately, only a small portion of honeypots/honeynets considered this vital issue with IIoT and CPS honeypots.With IoT honeypots, this factor was considered by various studies. In fact, the most commonly used tools for IoT honeypot and honeynet research were all tools which were used to check the available services, realism in responses, including response times, and other factors that affect fingerprintability, which will be discussed in the following sections.

Choice of Tools: A honeypot or honeynet designer should consider the deployment area or target application area characteristics when he/she is choosing the tools such as scanners. Not every tool may support all of the IoT, IIoT, and CPS

applications, their corresponding protocols and services. In addition, tools that also support vulnerability checks should be considered to be employed [8]. A designer should also consider how to pair their honeypot or honeynet with tools that will best complement the honeynet for effective attack mitigation. While medium and high interaction honeypots enable more interactivity for attackers, attackers may have tools to check whether they are interacting with a virtual environment and whether their activities are being recorded/logged. Tools such as Sebek are used by researchers in order to seamlessly log the activities of the attackers.

Appearance on the Search Engines and Fingerprintability: One of the most important factors in honeypot/honeynet design is ensuring appearance on the search engines while not being fingerprinted as a decoy system. For this reason, honeypot/honeynet owners have to monitor IoT search engines which identify and detect devices and honeypots on the Internet, such as Shodan. Different views exist in the literature whether being indexed by such search engines has an effect on the attacks to be received. For example, Guarnizo *et al.* [92] identified that the number of attacks on a device increase significantly in the first few weeks after they are listed on Shodan. Nevertheless, such indexing services can make the jobs of attackers easier by pointing out Internet-connected ready-to-attack targets. Being indexed by such search engines verifies the accessibility of the honeypot/honeynet system. Being listed as a real system rather than a honeypot/honeynet is an achievement that helps honeypot owners to reach their ultimate goal.

Comparison of IoT, IIoT, and CPS Honeypots/Honeynets: Honeypot and honeynet research for IoT, IIoT, and CPS environments is an important research area. Although we summarized the studies and provided taxonomies in the previous sections, comparison of the decoy systems for IoT, IIoT, and CPS, and highlighting their similarities and differences can be very crucial. The first significant difference arises from the supported services. While the IoT decoys considered mostly support Telnet, SSH, and HTTP which are not IoT-specific, the CPS decoys considered mostly support industrial protocols such as Modbus, S7comm, EtherNet/IP, and non industry-specific protocols such HTTP and SNMP. Since there are only two decoys for IIoT and only one of them is disclosing its services, we can see that IIoT decoys stay in the intermediary position in this regard, supporting both industrial and non-industrial protocols. The second difference arises from the process simulations. While some CPS decoys employ simulations of industrial processes for ICS plants, water management, electrical grid, and building HVAC systems, we do not see such process simulations in the proposed IoT decoys. The third difference arises from the interaction level of proposed honeypots and honeynets. While the majority of the decoys proposed for IoT are medium interaction decoys (10 studies), the majority of the decoys for CPS are low interaction (16 studies). The cost of physical ICS devices and difficulty of realistic process simulations play an important role in the interaction level choice of CPS honeypots and honeynets. Considering the similarities, we see that decoys with virtual resources and server roles are common between IoT, IIoT, and CPS environments.

Control and Liability: When deploying a honeypot or honeynet for IoT, IIoT, and CPS environments, control and liability issues are the aspects that are greatly overlooked, but designers should always consider. The greater the level of interaction a honeypot allows, the greater the risk that it could be compromised and used by attackers for harming other systems in the network or even launching attacks on other networks. Scott [8] advised to be familiar with laws before deployment of honeypots since honeypots are interpreted as entrapment by jurisdictions in some places. Haney [131] emphasized the importance of taking liability and legal issues into account and putting data control as a first priority, even if this means data capture may be affected. Haney proposed setting up both automated as well as manual data control mechanisms, with at least two protection mechanisms to always have a second option if one data control method fails. Sokol [26] highlighted that a honeynet should contain the following parts in order to address security, data control, and liability issues: a firewall with only the necessary network ports opened, a dynamic (re)connection mechanism to determine if a connection is trusted and can be allowed, a testbed for analysis, an emulated private virtual network to restrict attackers, and a control center to monitor connections and respond to issues quickly.

Improving Security of IoT, IIoT, and CPS Devices: The information gathered from research with honeypots and honeynets can lead to innovative ways to improve the security of IoT devices despite their constraints. One example of this is the proposal by Dang *et al.* [88] of a series of measures called IoTCheck to increase the security of IoT devices, which include asking whether the IoT device has a unique strong password, whether the default system user is a non-root user, and whether there are unnecessary components on the devices which can be eliminated. The same authors also suggest for manufacturers to disable shell commands that are enabled by default on Linux-based IoT devices but are not necessary, as these are used for attacks.

B. Open Issues

Honeypots and honeynets for IoT, IIoT, and CPS have been a very active field of research during the last decade. We studied 79 honeypots/honeynets in greater detail in this study. However, there are still open issues which need to be addressed by researchers.

Emerging Technologies/Domains: In terms of decoy systems for IoT, we see that there are honeypots/honeynets for Smart Home, but not for emerging domains or technologies such as wearable devices, medical devices, and smart city. In terms of decoy systems for IIoT and CPS, we see that there are honeypots/honeynets for general ICS, smart grid, water, gas and building automation systems. However, we do not see such decoys for other IIoT and CPS applications such as smart city, transportation, nuclear plants, medical devices. As smart medical devices in modern healthcare applications are becoming more prevalent and are threatened by various attacks [161]–[163], decoy systems for modern healthcare applications are needed. In addition, to

the best of our knowledge, there is only one honeypot system for building automation systems. Considering the rapid increase of notorious ransomware attacks [164], cryptocurrency mining attacks [165], [166], and attacks to enterprise IoT systems [167]–[170], we believe that further research is needed which may enable us to protect smart buildings from ransomware attacks. We would like to note that building honeypots and honeynets for the unexplored IoT, IIoT and CPS applications may require realistic process models (e.g., patient vitals models, vehicle operation models, nuclear process models, etc.) in case virtual or hybrid decoy systems are targeted.

Unexplored Protocols: Existing IoT, IIoT, and CPS honeypots/honeynets support a wide range of ICT, IoT, and industrial protocols. Various IoT honeypots emulate full devices. However, one cannot claim that the state-of-the-art honeypot/honeynet research considered every protocol or service. In addition to this, very few current studies focus on IoT specific protocols. There are also protocols and services that still need to be addressed by honeypot research. For instance, we did not find any study that supports Highway Addressable Remote Transducer (HART) and WirelessHART [171] industrial protocols. In addition, Enterprise IoT environments can employ various proprietary communication protocols that rely on security through obscurity [170]. For this reason, decoy designs for such type of proprietary solutions are needed. Researching unexplored protocols and services may provide valuable information for honeypot/honeynet research and practice. A potential solution to unexplored protocols for IoT, IIoT, and CPS honeypots/honeynets could be extending open source honeypots and honeynets such as Conpot, Honeyd, Dionaea, Kippo, etc. for the unexplored protocols. Although open source libraries for the unexplored well-known protocols can be found, researchers would have to perform reverse engineering for the proprietary communication protocols.

Emerging Platforms: In the recent years, several platforms were proposed/developed by both researchers and vendors for the management of the IoT devices [172]. In this regard, platforms such as openHAB, Samsung SmartThings, thingworx, Amazon AWS IoT, IBM Watson IoT, Apple HomeKit, etc. emerged for IoT applications. Such platforms have different characteristics in terms of supported IoT devices, communication protocols and network topologies, data processing and event handling approaches, and security. Although there exist decoy systems for generic IoT applications, one does not see any studies focusing on honeypot and honeynet design for the mentioned emerging IoT platforms. Since popularity of such platforms is increasing in recent years, IoT applications that are built on top of such platforms can be sweet spots for the adversaries. Therefore, there is a need for honeypot/honeynet research for the emerging IoT platforms. In order to propose novel decoy systems for the emerging platforms, researchers can benefit from the existing IoT honeypot/honeynet research and extend the open source IoT decoys.

Optimized Deployment Location: Honeypots and honeynets proposed for IoT, IIoT, and CPS employed various deployment locations (i.e., university, cloud, private locations) as explained in the previous sections. Each deployment location option has its own benefits and pitfalls in terms of fingerprintability [21], [173], suitability for IoT, IIoT, or CPS application, complexity, and cost. Although a few studies investigated how a limited set of deploy locations attract attackers, one does not see any study in the literature that aims to optimize the deployment location for the decoy system with respect to a set of constraints. We believe that, this is an important gap in the honeypot/honeynet research and there is a need for extensive analysis and novel frameworks in order to optimize deployment location decisions. Although this problem is hard to tackle, researchers can employ relaxation strategies in order to approximate the optimal deployment location solution for the IoT, IIoT, and CPS decoys.

Remote Management: Several tools can be utilized to manage honeypots/honeynets locally or remotely. While the decoys with virtual resources can be managed locally or remotely without much efforts, the decoys with physical resources may require researchers to be physically present in such locations for maintenance purposes. However, the Covid-19 pandemic caused lockdowns all around the world which forced researchers to perform their tasks remotely. Extraordinary times like the current pandemic, natural hazards, etc. can cause similar situations that can force people to remotely manage their decoy systems. We believe that researchers have to consider such conditions while designing and deploying their decoy systems for IoT, IIoT, and CPS. Remote management of decoy systems require employment of secure tools and secure configurations. However, vulnerabilities of such tools that are considered to be secure can be found, as in the case of SolarWinds [174], which require continuous efforts to check for vulnerabilities and patch.

Anti-Detection Mechanisms: Honeypots and honeynets that are using virtual resources have been widely used in IoT, IIoT, and CPS environments. Such an approach has several advantages as discussed in the previous section. However, from the malware research domain we know that virtual environment detection techniques are frequently used by malicious software developers. When we checked the honeypot/honeynet studies for IoT, IIoT, and CPS that use virtual resources, we did not see any study which considers this important issue. In addition, the analysis parts of the studies did not mention detecting an attacker which uses such techniques. Although research did not observe the existence of a sample case, we think that attackers will be using such methods in the near future. For this reason, future honeypots and honeynets for IoT, IIoT, and CPS should consider to employ anti-detection mechanisms in their medium/high interaction virtual decoy systems. Researchers in this regard can benefit from existing anti-detection research from the malware analysis domain such as hiding the artifacts regarding the analysis environment, moving analysis logic to lower levels such as hypervisors or bare-metal, etc. [175].

Vulnerabilities of Industrial Devices: IoT, IIoT, and CPS environments consist of several devices produced by different vendors. Vulnerabilities with device firmware, OS and other software are often found and listed in vulnerability databases such as Common Vulnerabilities and Exposures (CVE) [176]. As explained earlier, devices with such vulnerabilities attract the attackers and stand as vulnerable targets to compromise.

Considering the honeypots and honeynets, we see that there exist studies which take such vulnerabilities into account when designing honeypots. However, we did not encounter any proposal for IIoT and CPS that considers vulnerabilities of industrial devices. We believe that a research gap exists in the literature in regards to whether attackers really pay attention to industrial device vulnerabilities or not when choosing targets. A potential way to address this open problem could be deploying honeypots for IIoT and CPS environments that advertise both vulnerable and patched versions of ICS device firmware or management software. In this way, it would be possible to understand if adversaries pay attention to disclosed vulnerabilities when choosing their targets.

Insider Attacks: The target users for IoT, IIoT, and CPS are very diverse and have very different skill levels for deploying honeypot/honeynet systems. However, none of the IoT research studies consider how the systems being proposed could be implemented on a wider scale in the future, taking into account the need for simple deployment. In addition to this, none of the current research places focus on attacks initiated and carried out from inside the network. These types of attacks could be carried out by disgruntled employees or for corporate espionage. However, researchers may not deploy physical or virtual honeypots on a network in a straightforward way since insiders may have a chance to reach the decoys physically or virtually. We believe that virtualization technologies such as Network Function Virtualization (NFV) and containers, and SDN technologies can be utilized to develop moving target defense-like honeypot solutions for insider attackers.

Machine Learning: Another open issue is the employment of ML and AI techniques for honeypot design. Considering the studies, we see that ML techniques have been employed by a limited number of honeypot/honeynet works for configuration and data analysis purposes. Although eight studies [[55]–[57], [64], [66], [67], [70], [72]] employed ML for IoT honeypots/honeynets, we see that only one study [111] used ML techniques for IIoT and CPS honeypots. We believe that future IoT, IIoT, and CPS honeypots and honeynets can benefit from ML techniques to propose smarter decoy systems that can i) adapt themselves based on the actions of attackers, ii) discriminate known attacks from new attacks thus enable researchers to focus more on novel threats, and iii) increase the efficiency and prevalence of honeypots and honeynets.

Discrimination of Benign Decoy Traffic: Honeypots and honeynets are traditionally assumed to receive only malicious traffic which are in fact helpful for the existing IDS and IPS elements in the network to increase their true positive rates. However, IoT honeypots and honeynets employing physical IoT devices can receive benign traffic from vendors. For instance smart home devices provided by Google, Apple, Samsung, and Amazon can receive benign traffic from their vendors with application-specific motivations (e.g., cloud connectivity, health check, updates, etc.). Such benign traffic originating from device vendors targeting the decoy system, as well as the traffic generated by benign bots such as Shodan and Censys to index the Internet-connected devices, break the aforementioned assumption of incoming traffic to decoy systems. For this reason, researchers have to take such benign traffic into account while analyzing the decoy traffic. We believe that IP address lookup for the traffic sources can provide information on the benign origins of the decoy traffic. In addition, analysis of Ferretti *et al.* [120] on the scanning patterns of legitimate scanners such as Shodan can give clues to researchers on discriminating legitimate traffic.

Production Decoys: Considering the reviewed honeypots and honeynets for IoT, IIoT, and CPS environments, we see that the majority of the reviewed works are research honeypots. Although research honeypots are important to understand the attacks and new tactics of attackers, they do not actively participate in securing an IoT, IIoT, or CPS environment. For this reason, more production honeypots are needed that can actively participate in securing IoT, IIoT, and CPS networks. Efforts in combining honeypots/honeynets with IDS solutions are noteworthy in this regard. Researchers can employ open source IDS solutions such as Snort, Zeek, Suricata etc., malware analysis platforms such as Cuckoo, and next generation networking technologies such as SDN to propose novel decoy solutions for IoT, IIoT, and CPS environments.

X. Conclusion

In this paper, we provided a comprehensive survey of honeypots and honeynets for IoT, IIoT, and CPS environments. We provided a taxonomy of honeypots and honeynets based on purpose, role, level of interaction, scalability, resource level, availability of source code and target IoT, IIoT, or CPS application. In addition, we analyzed the existing honeypots and honeynets extensively and extracted the common characteristics of state-of-the-art honeypots and honeynets for IoT, IIoT, and CPS. Moreover, we outlined and discussed the key design factors for honeypots and honeynets for IoT, IIoT, and CPS applications. We also summarized the open research problems that can be addressed by future honeypot and honeynet studies. As future work, we are planning to propose novel honeypot/honeynet systems for IoT and CPS environments that build upon this survey.

Acknowledgment

The views expressed are those of the authors only, not of the funding agencies.

References

[1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018.

[2] B. Bordel, R. Alcarria, T. Robles, and D. Martín, "Cyber–physical systems: Extending pervasive sensing from control theory to the Internet of Things," *Pervasive Mobile Comput.*, vol. 40, pp. 156–184, 2017.

[3] A. Humayed, J. Lin, F. Li, and B. Luo, "Cyber-physical systems security—A survey," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1802–1831, Dec. 2017.

[4] C. Greer, M. Burns, D. Wollman, and E. Griffor, "Cyber-physical systems and Internet of Things," NIST, Gaithersburg, MD, USA, Rep. 1900-202, Mar. 2019. [Online]. Available: https://doi.org/10.6028/NIST.SP.1900-202

[5] I. Makhdoom, M. Abolhasan, J. Lipman, R. P. Liu, and W. Ni, "Anatomy of threats to the Internet of Things," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1636–1675, 2nd Quart., 2019.

[6] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, "IoT: Internet of threats? a survey of practical security vulnerabilities in real IoT devices," *IEEE Internet of Things J.*, vol. 6, no. 5, pp. 8182–8201, Oct. 2019.

[7] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on Internet-scale IoT exploitations," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2702–2733, 3rd Quart., 2019.

[8] C. Scott, "Designing and implementing a honeypot for a SCADA network," SANS, Bethesda, MD, USA, White Paper, Jun. 2014. [Online]. Available: https://www.sans.org/reading-room/whitepapers/detection/designing-implementing-honeypot-scada-network-35252

[9] P. Simões, T. Cruz, J. Gomes, and E. Monteiro, "On the use of honeypots for detecting cyber attacks on industrial control networks," in *Proc. 12th Eur. Conf. on Inf. Warfare Security (ECIW)*, 2013, pp. 263–270.

[10] Z. Yu, Z. Kaplan, Q. Yan, and N. Zhang, "Security and privacy in the emerging cyber-physical world: A survey," *IEEE Commun. Surveys Tuts.*, early access, May 18, 2021, doi: 10.1109/COMST.2021.3081450.

[11] J. Lopez, L. Babun, H. Aksu, and S. Uluagac, "A survey on function and system call hooking approaches," *J. Hardw. Syst. Security*, vol. 1, pp. 114–136, Sep. 2017.

[12] W. Fan, Z. Du, D. Fernández, and V. A. Villagrá, "Enabling an anatomic view to investigate honeypot systems: A survey," *IEEE Syst. J.*, vol. 12, no. 4, pp. 3906–3919, Dec. 2018.

[13] L. Spitzner. (Oct. 2001). *The Value of Honeypots, Part One:Definitions and Values of Honeypots*. Accessed: Apr. 14, 2020. [Online]. Available: http://www.symantec.com/connect/articles/value-honeypots-part-onedefinitions-and-values-honeypots/

[14] P. Kumar and R. Verma, "A review on recent advances & future trends of security in honeypot," *Int. J. Adv. Res. Comput. Sci.*, vol. 8, no. 3, pp. 1108–1113, Mar./Apr. 2017.

[15] I. Butun, P. Österberg, and H. Song, "Security of the Internet of Things: Vulnerabilities, attacks, and countermeasures," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 616–644, 1st Quart., 2020.

[16] E. Lee, Y.-D. Seo, S.-R. Oh, and Y.-G. Kim, "A survey on standards for interoperability and security in the Internet of Things," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1020–1047, 2nd Quart., 2021.

[17] J. Granjal, E. Monteiro, and J. S. Silva, "Security for the Internet of Things: A survey of existing protocols and open research issues," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 3, pp. 1294–1312, 3rd Quart., 2015.

[18] A. K. Sikder, G. Petracca, H. Aksu, T. Jaeger, and A. S. Uluagac, "A survey on sensor-based threats and attacks to smart devices and applications," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1125–1159, 2nd Quart., 2021.

[19] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for Internet of Things (IoT) security," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1646–1685, 3rd Quart., 2020.

[20] M. H. Cintuglu, O. A. Mohammed, K. Akkaya, and A. S. Uluagac, "A survey on smart grid cyber-physical system testbeds," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 446–464, 1st Quart., 2017.

[21] L. Babun, H. Aksu, L. Ryan, K. Akkaya, E. Bentley, and A. S.Uluagac, "Z-IoT: Passive device-class fingerprinting of ZigBee and Z-wave IoT devices," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2020, pp. 1–7.

[22] W. Fan, Z. Du, and D. Fernández, "Taxonomy of honeynet solutions," in *Proc. SAI Intell. Syst. Conf. (IntelliSys)*, Nov. 2015, pp. 1002–1009.

[23] A. Mairh, D. Barik, K. Verma, and D. Jena, "Honeypot in network security: A survey," in *Proc. Int. Conf. Commun. Comput. Security*, 2011, pp. 600–605.

[24] R. M. Campbell, K. Padayachee, and T. Masombuka, "A survey of honeypot research: Trends and opportunities," in *Proc. 10th Int. Conf. Internet Technol. Secured Trans. (ICITST)*, 2015, pp. 208–212.

[25] L. Zobal, D. Kolář, and R. Fujdiak, "Current state of honeypots and deception strategies in cybersecurity," in *Proc. 11th Int. Congr. Ultra Mod. Telecommun. Control Syst. Workshops (ICUMT)*, 2019, pp. 1–9.

[26] P. Sokol and M. Andrejko, "Deploying honeypots and honeynets: Issues of liability," in *Computer Networks*. Cham, Switzerland: Springer Int., 2015, pp. 92–101.

[27] P. Sokol, M. Husak, and F. Lipták, "Deploying honeypots and honeynets: Issue of privacy," in *Proc. 10th Int. Conf. Availability Rel. Security*, 2015, pp. 397–403.

[28] M. F. Razali, M. N. Razali, F. Z. Mansor, G. Muruti, and N. Jamil, "IoT honeypot: A review from researcher's perspective," in *Proc. IEEE Conf. Appl. Inf. Netw. Security (AINS)*, Nov. 2018, pp. 93–98.

[29] C. Dalamagkas *et al.*, "A survey on honeypots, honeynets and their applications on smart grid," in *Proc. IEEE Conf. Netw. Softw. (NetSoft)*, Jun. 2019, pp. 93–100.

[30] S. Dowling, M. Schukat, and H. Melvin, "Data-centric framework for adaptive smart city honeynets," in *Proc. Smart City Symp. Prague (SCSP)*, 2017, pp. 1–7.

[31] W. Fan, D. Fernández, and V. A. Villagrá, "Technology independent honeynet description language," in *Proc. 3rd Int. Conf. Model Driven Eng. Softw. Develop. (MODELSWARD)*, Feb. 2015, pp. 303–311.

[32] A. Acien, A. Nieto, G. Fernandez, and J. Lopez, "A comprehensive methodology for deploying IoT honeypots," in *Trust, Privacy and Security in Digital Business*, Sep. 2018, pp. 229–243.

[33] O. Surnin *et al.*, "Probabilistic estimation of honeypot detection in Internet of Things environment," in *Proc. Int. Conf. Comput. Netw. Commun. (ICNC)*, Feb. 2019, pp. 191–196.

[34] O. Surnin. *Honeypot*. Accessed: Apr. 1, 2020. [Online]. Available: https://gitlab.com/legik/honeypot

[35] M.-R. Zamiri-Gourabi, A. R. Qalaei, and B. A. Azad, "Gas what? i can see your gaspots. studying the fingerprintability of ICs honeypots in the wild," in *Proc. 5th Annu. Ind. Control Syst. Security (ICSS) Workshop*, 2019, pp. 30–37.

[36] Honeynet Project. (Apr. 2001). *Know Your Enemy: Honeynets*. Accessed: Apr. 2, 2020. [Online]. Available: http://www.symantec.com/connect/articles/knowyour-enemy-honeynets

[37] A.G. Manzanares, "HoneyIo4: The construction of a virtual, low-interaction IoT honeypot," Ph.D. dissertation, Dept. School Eng., Universitat Politècnica de Catalunya, Barcelona, Spain, 2017. [Online]. Available: https://pdfs.semanticscholar.org/3124/456d251e3657746de4c34472224f5b2d8efe.pdf

[38] G. Evron. (Nov. 2018). *Mirai Open-Source IoT Honeypot: New Cymmetria Research Release*. Accessed: Apr. 16, 2020. [Online]. Available: https://cymmetria.com/blog/mirai-open-source-IoT-honeypot-new-cymmetria-research-release/

[39] R. Piggin and I. Buffey, "Active defence using an operational technology honeypot," in *Proc. 11th Int. Conf. Syst. Safety Cyber Security (SSCS)*, 2016, pp. 1–6.

[40] S. Hilt, F. Maggi, C. Perine, L. Remorin, M. Rösler, and R. Vosseler, "Caught in the act: Running a realistic factory honeypot to capture real threats," Trend Micro, Shibuya City, Japan, White Paper, 2020. [Online]. Available: https://documents.trendmicro.com/assets/white_papers/wp-caught-in-the-act-running-a-realistic-factory-honeypot-to-capture-real-threats.pdf

[41] S. Litchfield, D. Formby, J. Rogers, S. Meliopoulos, and R. Beyah, "Rethinking the honeypot for cyber-physical systems," *IEEE Internet Comput.*, vol. 20, no. 5, pp. 9–17, Sep./Oct. 2016.

[42] A. D. Oza, G. N. Kumar, and M. Khorajiya, "Survey of snaring cyber attacks on IoT devices with honeypots and honeynets," in *Proc. 3rd Int. Conf. Converg. Technol. (I2CT)*, Apr. 2018, pp. 1–6.

[43] D. Antonioli and N. O. Tippenhauer, "Minicps: A toolkit for security research on CPS networks," in *Proc. 1st ACM Workshop Cyber Phys. Syst. Security Privacy*, 2015, pp. 91–100.

[44] M. Zec, "Implementing a clonable network stack in the freebsd kernel," in *Proc. FREENIX Track USENIX Annu. Techn. Conf.*, San Antonio, TX, USA, Jun. 2003, pp. 137–150.

[45] (2020). *GridLab-D Simulation Software*. Accessed: Apr. 7, 2020. [Online]. Available: https://www.gridlabd.org/

[46] P. Gunathilaka, D. Mashima, and B. Chen, "SoftGrid: A software-based smart grid testbed for evaluating substation cybersecurity solutions," in *Proc. 2nd ACM Workshop Cyber Phys. Syst. Security Privacy*, 2016, pp. 113–124.

[47] Powerworld. (2020). *PowerWorld Simulator*. Accessed: May 15, 2020. [Online]. Available: https://www.powerworld.com/

[48] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proc. 9th ACM SIGCOMM Workshop Hot Topics Netw.*, 2010, p. 19.

[49] A. Pauna, "Improved self adaptive honeypots capable of detecting rootkit malware," in *Proc. 9th Int. Conf. Commun. (COMM)*, Jun. 2012, pp. 281–284.

[50] N. Provos. (2017). *Honeyd*. Accessed: Apr. 2, 2020. [Online]. Available: https://github.com/DataSoft/Honeyd

[51] DinoTools. *Dionaea*. Accessed: Apr. 2, 2020. [Online]. Available: https://github.com/DinoTools/dionaea

[52] Kippo. (2016). *Kippo- SSH Honeypot*. Accessed: Apr. 2, 2020. [Online]. Available: https://github.com/desaster/kippo

[53] Cowrie. (2019). *Cowrie SSH and Telnet Honeypot*. Accessed: Apr. 2, 2020. [Online]. Available: https://www.cowrie.org/

[54] foospidy. (2013). *Honeypy*. Accessed: Apr. 30, 2020. [Online]. Available: https://github.com/foospidy/HoneyPy

[55] G. Wagener, "Self-adaptive honeypots coercing and assessing attacker behaviour," Ph.D. dissertation, Dept. Comput. Sci., Institut National Polytechnique de Lorraine, Vandœuvre-lès-Nancy, France, 2011. [Online]. Available: https://tel.archives-ouvertes.fr/tel-00627981/file/thesis_gerard_wagener_after_defense.pdf

[56] A. Pauna and I. Bica, "RASSH—Reinforced adaptive SSH honeypot," in *Proc. 10th Int. Conf. Commun. (COMM)*, May 2014, pp. 1–6.

[57] A. Pauna, A.-C. Iacob, and I. Bica, "QRASSH—A self-adaptive ssh honeypot driven by Q-learning," in *Proc. Int. Conf. Commun. (COMM)*, Jun. 2018, pp. 441–446.

[58] L. Stafira, "Examining effectiveness of web-based Internet of Things honeypots," Ph.D. dissertation, Dept. Elect. Comput. Eng., Air Force Inst. Technol., Wright-Patterson AFB, OH, USA, 2019. [Online]. Available: https://scholar.afit.edu/etd/2284

[59] Dionaea. (2015). *Service*. Accessed: Apr. 2, 2020. [Online]. Available: https://dionaea.readthedocs.io/en/latest/introduction.html

[60] L. Metongnon and R. Sadre, "Beyond telnet: Prevalence of IoT protocols in telescope and honeypot measurements," in *Proc. WTMC*, Aug. 2018, pp. 21–26.

[61] B. Kaur and P. K. Pateriya, "A survey on security concerns in Internet of Things," in *Proc. 2nd Int. Conf. Intell. Comput. Control Syst. (ICICCS)*, Madurai, India, Jun. 2018, pp. 27–34.

[62] S. Dowling, M. Schukat, and H. Melvin, "A ZigBee honeypot to assess IoT cyberattack behaviour," in *Proc. 28th Irish Signals Syst. Conf. (ISSC)*, Jun. 2017, pp. 1–6.

[63] *G. Wagener*. (2018). *Adaptive Honeypot Alternative (AHA)*. Accessed: Apr. 23, 2020. [Online]. Available: http://git.quuxlabs.com

[64] A. Pauna, I. Bica, F. Pop, and A. Castiglione, "On the rewards of self-adaptive IoT honeypots," *Ann. Telecommun.*, vol. 74, pp. 501–515, Jul. 2019.

[65] A. Pauna. (2018). *Irassh*. Accessed: Apr. 23, 2020. [Online]. Available: https://github.com/adpauna/irassh/

[66] R. K. Shrivastava, B. Bashi, and C. Hota, "Attack detection and forensics using honeypot in IoT environment," in *Proc. Int. Conf. Distrib. Comput. Internet Technol.*, Bhubaneswar, India, Jan. 2019, pp. 402–409.

[67] B. Lingenfelter, I. Vakilinia, and S. Sengupta, "Analyzing variation among IoT botnets using medium interaction honeypots," in *Proc. 10th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Las Vegas, NV, USA, 2020, pp. 0761–0767.

[68] A. Pauna. (2018). *Qrassh*. Accessed: Apr. 16, 2020. [Online]. Available: https://github.com/adpauna/qrassh/

[69] D. Chen, M. Egeley, M. Woo, and D. Brumley, "Towards automated dynamic analysis for linux-based embedded firmware," in *Proc. NDSS*, Feb. 2016, pp. 21–24.

[70] M. Wang, J. Santillan, and F. Kuipers, "ThingPot: An interactive Internet-of-Things honeypot," Jul. 2018. [Online]. Available: http://arxiv.org/abs/1807.04114.

[71] R. Vishwakarma and A. K. Jain, "A honeypot with machine learning based detection framework for defending IoT based botnet DDoS attacks," in *Proc. 3rd Int. Conf. Trends Electron. Informat. (ICOEI)*, Apr. 2019, pp. 1019–1024.

[72] T. Luo, Z. Xu, X. Jin, Y. Jia, and X. Ouyang, "IoTCandyjar: Towards an intelligent-interaction honeypot for IoT devices," in *Proc. Black Hat*, 2017, pp. 1–11.

[73] Y. Zhou, "Chameleon: Towards adaptive honeypot for Internet of Things," in *Proc. ACM Turing Celebration Conf. China*, May 2019, pp. 1–5.

[74] A. Vetterl and R. Clayton, "Honware: A virtual honeypot framework for capturing cpe and IoT zero days," in *Proc. APWG Symp. Electron Crime Res. (eCrime)*, 2019, pp. 1–13.

[75] D. Chen, M. Egeley, M. Woo, and D. Brumley. (2016). *Firmadyne*. Accessed: Apr. 30, 2020. [Online]. Available: https://github.com/firmadyne/firmadyne

[76] M. Wang. (2017). *ThingPot*. Accessed: May 14, 2020. [Online]. Available: https://github.com/Mengmengada/ThingPot

[77] Tor Project Inc. *Tor Project*. Accessed: Jul. 26, 2020. [Online]. Available: https://www.torproject.org/

[78] Shodan. *Honeyscore*. Accessed: Jul. 26, 2020. [Online]. Available: https://honeyscore.shodan.io/

[79] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow. (Jun. 2016). *IoTPOT: Analysing the Rise of IoT Compromises*. Accessed: Apr. 2, 2020. [Online]. Available: https://ipsr.ynu.ac.jp/IoT/

[80] Cymmetria. *MTPot*. Accessed: Apr. 1, 2020. [Online]. Available: https://github.com/Cymmetria/MTPot

[81] H. Šemić and S. Mrdovic, "IoT honeypot: A multi-component solution for handling manual and Mirai-based attacks," in *Proc. 25th Telecommun. Forum (TELFOR)*, 2017, pp. 1–4.

[82] Phype. (2019). *Telnet IoT Honeypot*. Accessed: Apr. 2, 2020. [Online]. Available: https://github.com/Phype/telnet-IoT-honeypot

[83] P. Krishnaprasad, "Capturing attacks on IoT devices with a multi-purpose IoT honeypot," Ph.D. dissertation, Dept. Comput. Sci. Eng., Ind. Inst. Technol. Kanpur, Kanpur, India, 2017. [Online]. Available: https://security.cse.iitk.ac.in/sites/default/files/15111021.pdf

[84] A. Oza, G. Kumar, M. Khorajiya, and V. Tiwari, *Snaring Cyber Attacks on IoT Devices With Honeynet*. Singapore: Springer Nat., 2019.

[85] M. Anirudh, S. A. Thileeban, and D. J. Nallathambi, "Use of honeypots for mitigating DoS attacks targeted on IoT networks," in *Proc. Int. Conf. Comput. Commun. Signal Process. (ICCCSP)*, Jan. 2017, pp. 1–4.

[86] A. Tambe *et al.*, "Detection of threats to IoT devices using scalable VPN-forwarded honeypots," in *Proc. 9th ACM Conf. Data Appl. Security and Privacy (CODASPY)*, Mar. 2019, pp. 85–96.

[87] A. M. Zarca, J. B. Bernabe, A. Skarmeta, and J. M. A. Calero, "Virtual IoT HoneyNets to mitigate cyberattacks in SDN/NFV-enabled IoT networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1262–1277, Jun. 2020.

[88] F. Dang *et al.*, "Understanding fileless attacks on linux-based IoT devices with honeycloud," in *Proc. 17th Annu. Int. Conf. Mobile Syst. Appl. Serv.*, Nov. 2019, pp. 482–493.

[89] M. A. Hakim. *U-POT*. Accessed: Apr. 1, 2020. [Online]. Available: https://github.com/azizulhakim/u-pot/

[90] U. D. Gandhi, P. M. Kumar, S. Kadu, R. Varatharajan, G. Manogaran, and R. Sundarasekar, "HIoTpot: Surveillance on IoT devices against recent threats," *Wireless Pers. Commun.*, vol. 103, no. 2, pp. 1179–1194, 2018.

[91] V. Martin, Q. Cao, and T. Benson, "Fending off IoT-hunting attacks at home networks," in *Proc. 2nd Workshop Cloud Assist. Netw.*, Dec. 2017, pp. 67–72.

[92] J. D. Guarnizo *et al.*, "Siphon: Towards scalable high-interaction physical honeypots," in *Proc. Cyber Phys. Syst. Security Workshops (CPSS)*, Apr. 2017, pp. 57–68.

[93] W. Zhang, B. Zhang, Y. Zhou, H. He, and Z. Ding, "An IoT honeynet based on multiport honeypots for capturing IoT attacks," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 3991–3999, May 2020.

[94] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "IoTPOT: Analysing the rise of IoT compromises," in *Proc. 9th USENIX Workshop Offensive Technol. (WOOT)*, Washington, DC, USA, Aug. 2015, p. 9.

[95] Twisted Matrix Labs. (Sep. 2014). *Welcome to the Twisted Documentation*. Accessed: Apr. 9, 2020. [Online]. Available: https://twistedmatrix.com/documents/current/

[96] Elastic. (2020). *Getting Started with Logstash*. Accessed: Apr. 9, 2020. [Online]. Available: https://www.elastic.co/guide/en/logstash/current/getting-started-with-logstash.html

[97] Elastic. (2017). *Elasticsearch 5.2.2*. Accessed: Apr. 9, 2020. [Online]. Available: https://www.elastic.co/downloads/past-releases/elasticsearch-5-2-2/

[98] Elastic. (2020). *Kibana: Your Window into the Elastic Stack*. Accessed: Apr. 9, 2020. [Online]. Available: https://www.elastic.co/kibana

[99] N. Provos. (May 2007). *Honeyd Frequently Asked Questions*. Accessed: Apr. 1, 2020. [Online]. Available: http://www.honeyd.org/faq.php

[100] "Internet security threat report (ISTR) 2019," vol. 24, Symantec, San Jose, CA, USA, Rep., Feb. 2019. [Online]. Available: https://docs.broadcom.com/doc/

[101] (2020). *Shodan*. Accessed: May 14, 2020. [Online]. Available: https://www.shodan.io/

[102] (2020). *Nmap*. Accessed: May 14, 2020. [Online]. Available: https://nmap.org/

[103] R. Graham. (2019). *Masscan*. Accessed: May 14, 2020. [Online]. Available: https://github.com/robertdavidgraham/masscan/

[104] V. Pothamsetty and M. Franz. (2004). *SCADA HoneyNet Project: Building Honeypots for Industrial Networks*. Accessed: May 2, 2020. [Online]. Available: http://scadahoneynet.sourceforge.net/

[105] D. Peterson. (2006). *SCADA Honeywall: Use Your Own PLC As The Target*. Accessed: May 2, 2020. [Online]. Available: https://dale-peterson.com/2008/07/08/scada-honeywall-use-your-own-plc-as-the-target/

[106] D. Bond. (2011). *Digital Bond SCADA Honeynet*. Accessed: May 2, 2020. [Online]. Available: https://web.archive.org/web/20111215085656/http:www.digitalbond.com/tools/scada-honeynet/

[107] S. M. Wade, "SCADA Honeynets: The attractiveness of honeypots as critical infrastructure security tools for the detection and analysis of advanced threats," M.S. thesis, Dept. Elect. Comput. Eng., Iowa State Univ., Ames, IA, USA, 2011. [Online]. Available: https://lib.dr.iastate.edu/etd/12138

[108] L. Rist, J. Vestergaard, D. Haslinger, A. De Pasquale, and J. Smith. (2020). *Conpot ICS/SCADA Honeypot*. Accessed: May 2, 2020. [Online]. Available: http://conpot.org/

[109] (2020). *Honeynet Project*. Accessed: May 2, 2020. [Online]. Available: https://www.honeynet.org/

[110] C. Zhao and S. Qin, "A research for high interactive honepot based on industrial service," in *Proc. 3rd IEEE Int. Conf. Comput. Commun. (ICCC)*, 2017, pp. 2935–2939.

[111] J. Cao, W. Li, J. Li, and B. Li, "DiPot: A distributed industrial honeypot system," in *Smart Computing and Communication*, M. Qiu, Ed. Cham, Switzerland: Springer Int., 2018, pp. 300–309.

[112] S. Lau, J. Klick, S. Arndt, and V. Roth, "POSTER: Towards highly interactive honeypots for industrial control systems," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 1823–1825.

[113] E. Vasilomanolakis, S. Srinivasa, C. G. Cordero, and M. Mühlhäuser, "Multi-stage attack detection and signature generation with ICS honeypots," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp.*, 2016, pp. 1227–1232.

[114] F. Xiao, E. Chen, and Q. Xu, "S7commTrace: A high interactive honeypot for industrial control system based on S7 protocol," in *Information and Communications Security*. Cham, Switzerland: Springer Int., 2018, pp. 412–423.

[115] M. Winn, M. Rice, S. Dunlap, J. Lopez, and B. Mullins, "Constructing cost-effective and targetable industrial control system honeypots for production networks," *Int. J. Crit. Infrastruct. Prot.*, vol. 10, pp. 47–58, Sep. 2015.

[116] J. K. Gallenstein, "Integration of the network and application layers of automatically-configured programmable logic controller honeypots," M.S. thesis, Dept. Elect. Comput. Eng., Air Force Instit. Technol. Air Univ., Wright-Patterson AFB, OH, USA, Mar. 2017. [Online]. Available: https://apps.dtic.mil/dtic/tr/fulltext/u2/1054643.pdf

[117] S. Abe, Y. Tanaka, Y. Uchida, and S. Horata, "Developing deception network system with traceback honeypot in ICS network," *SICE J. Control Meas. Syst. Integr.*, vol. 11, no. 4, pp. 372–379, 2018.

[118] A. Jicha, M. Patton, and H. Chen, "SCADA honeypots: An in-depth analysis of conpot," in *Proc. IEEE Conf. Intell. Security Informat. (ISI)*, Tucson, AZ, USA, Sep. 2016, pp. 196–198.

[119] K.-C. Lu *et al.*, "Evaluation and build to honeypot system about scada security for large-scale IoT devices," *J. Robot. Netw. Artif. Life*, vol. 6, pp. 157–161, Dec. 2019.

[120] P. Ferretti, M. Pogliani, and S. Zanero, "Characterizing background noise in ICS traffic through a set of low interaction honeypots," in *Proc. ACM Workshop Cyber Phys. Syst. Security Privacy*, 2019, pp. 51–61.

[121] H. Naruoka *et al.*, "ICS honeypot system (CamouflageNet) based on attacker's human factors," *Procedia Manuf.*, vol. 3, pp. 1074–1081, Jan. 2015.

[122] E. Vasilomanolakis *et al.*, "This network is infected: HosTaGe—A low-interaction honeypot for mobile devices," in *Proc. 3rd ACM Workshop Security Privacy Smartphones Mobile Devices*, 2013, pp. 43–48.

[123] J. P. Disso, K. Jones, and S. Bailey, "A plausible solution to SCADA security honeypot systems," in *Proc. 8th Int. Conf. Broadband Wireless Comput. Commun. Appl.*, Compiegne, France, 2013, pp. 443–448.

[124] Honeynet Project. (2011). *Honeywall CDROM*. Accessed: May 15, 2020. [Online]. Available: https://www.honeynet.org/projects/old/honeywall-cdrom/

[125] P. C. Warner, "Automatic configuration of programmable logic controller emulators," M.S. thesis, Dept. Elelct. Comput. Eng., Air Force Inst. Technol. Air Univ., Wright-Patterson AFB, OH, USA, Mar. 2015. [Online]. Available: https://apps.dtic.mil/dtic/tr/fulltext/u2/a620212.pdf

[126] C. Leita, K. Mermoud, and M. Dacier, "ScriptGen: an automated script generation tool for honeyd," in *Proc. 21st Annu. Comput. Security Appl. Conf. (ACSAC)*, Tucson, AZ, USA, 2005, p. 12.

[127] M. and M. Papa, "A framework for the design and deployment of a SCADA honeynet," in *Proc. 9th Annu. Cyber Inf. Security Res. Conf.*, 2014, pp. 121–124.

[128] S. Kuman, S. Groš, and M. Mikuc, "An experiment in using IMUNES and conpot to emulate honeypot control networks," in *Proc. 40th Int. Convention Inf. Commun. Technol. Electron. Microelectron. (MIPRO)*, Opatija, Croatia, 2017, pp. 1262–1268.

[129] C. Ding, J. Zhai, and Y. Dai, "An improved ICS honeypot based on SNAP7 and IMUNES," in *Cloud Computing and Security*. Cham, Switzerland: Springer Int., 2018, pp. 303–313.

[130] R. C. Bodenheim, "Impact of the Shodan computer search engine on Internet-facing industrial control system devices," M.S. thesis, Dept. Elect. Comput. Eng., Air Force Inst. Technol. Air Univ., Wright-Patterson AFB, OH, USA, Mar. 2014. [Online]. Available: https://apps.dtic.mil/docs/citations/ADA601219

[131] M. Haney, "Leveraging cyber-physical system honeypots to enhance threat intelligence," in *Critical Infrastructure Protection XIII*. Cham, Switzerland: Springer Int., 2019, pp. 209–233.

[132] D. J. Berman, "Emulating industrial control system field devices using Gumstix technology," M.S. thesis, Dept. Elect. Comput. Eng., Air Force Inst. Technol. Air Univ., Wright-Patterson AFB, OH, USA, Jun. 2012. [Online]. Available: https://scholar.afit.edu/etd/1080/

[133] R. M. Jaromin, "Emulation of industrial control field device protocols," M.S. thesis, Dept. Elect. Comput. Eng., Air Force Inst. Technol. Air Univ., Wright-Patterson AFB, OH, USA, Mar. 2013. [Online]. Available: https://apps.dtic.mil/dtic/tr/fulltext/u2/a582482.pdf

[134] T. Holczer, M. Felegyhazi, and L. Buttyan, "The design and implementation of a plc honeypot for detecting cyber attacks against industrial control systems," in *Proc. Int. Conf. Comput. Security Nucl. World Expert Discussion Exchange*, 2015.

[135] A. V. Serbanescu, S. Obermeier, and D.-Y. Yu, "ICS threat analysis using a large-scale honeynet," in *Proc. 3rd Int. Symp. ICS SCADA Cyber Security Res.*, 2015, pp. 20–30.

[136] P. Simões, T. Cruz, J. Proença, and E. Monteiro, *Specialized Honeypots for SCADA Systems*. Cham, Switzerland: Springer Int., 2015, pp. 251–269.

[137] S. Ahn, T. Lee, and K. Kim, "A study on improving security of ICS through honeypot and ARP spoofing," in *Proc. Int. Conf. Inf. Commun. Technol. Converg.*, Jeju, South Korea, Oct. 2019, pp. 964–967.

[138] A. Belqruch and A. Maach, "SCADA security using SSH honeypot," in *Proc. 2nd Int. Conf. Netw. Inf. Syst. Security*, Mar. 2019, pp. 1–5.

[139] A. V. Serbanescu, S. Obermeier, and D.-Y. Yu, "A flexible architecture for industrial control system honeypots," in *Proc. 12th Int. Joint Conf. e-Bus. Telecommun.*, vol. 4. Colmar, France, 2015, pp. 16–26.

[140] D. I. Buza, F. Juhász, G. Miru, M. Félegyházi, and T. Holczer, "CryPLH: Protecting smart energy systems from targeted attacks with a PLC honeypot," in *Smart Grid Security*. Cham, Switzerland: Springer Int., 2014, pp. 181–192.

[141] K. Kołtyś and R. Gajewski, "SHaPe: A honeypot for electric power substation," *J. Telecommun. Inf. Technol.*, vol. nr 4, pp. 37–43, Apr. 2015.

[142] O. Redwood, J. Lawrence, and M. Burmester, "A symbolic honeynet framework for SCADA system threat intelligence," in *Critical Infrastructure Protection IX*. Cham, Switzerland: Springer Int., 2015, pp. 103–118.

[143] D. Mashima, B. Chen, P. Gunathilaka, and E. L. Tjiong, "Towards a grid-wide, high-fidelity electrical substation honeynet," in *Proc. IEEE Int. Conf. Smart Grid Commun. (SmartGridComm)*, Dresden, Germany, Oct. 2017, pp. 89–95.

[144] D. Pliatsios, P. Sarigiannidis, T. Liatifis, K. Rompolos, and I. Siniosoglou, "A novel and interactive industrial control system honeypot for critical smart grid infrastructure," in *Proc. IEEE 24th Int. Workshop Comput.-Aided Model. Design Commun. Links Netw. (CAMAD)*, Limassol, Cyprus, Sep. 2019, pp. 1–6.

[145] D. Mashima, Y. Li, and B. Chen, "Who's scanning our smart grid? empirical study on honeypot data," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Waikoloa, HI, USA, Dec. 2019, pp. 1–6.

[146] M. M. Kendrick and Z. A. Rucker, "Energy grid threat analysis using honeypots," M.S. thesis, Naval Postgraduate School, Monterey, CA, USA, Jun. 2019. [Online]. Available: https://calhoun.nps.edu/handle/10945/62843

[147] D. Hyun, "Collecting cyberattack data for industrial control systems using honeypots," M.S. thesis, Naval Postgraduate School, Monterey, CA, USA, Mar. 2018. [Online]. Available: http://hdl.handle.net/10945/58316

[148] K. Wilhoit, "Who's really attacking your ICS equipment?" Trend Micro, Shibuya City, Japan, White Paper, 2013. [Online]. Available: https://www.trendmicro.com.tr/media/wp/whos-really-attacking-your-ics-equipment-whitepaper-en.pdf

[149] D. Antonioli, A. Agrawal, and N. O. Tippenhauer, "Towards high-interaction virtual ics honeypots-in-a-box," in *Proc. 2nd ACM Workshop Cyber Phys. Syst. Security Privacy*, 2016, pp. 13–22.

[150] A. F. Murillo, L. F. Cómbita, A. C. Gonzalez, S. Rueda, A. A. Cardenas, and N. Quijano, "A virtual environment for industrial control systems: A nonlinear use-case in attack detection, identification, and response," in *Proc. 4th Annu. Ind. Control Syst. Security Workshop*, 2018, pp. 25–32.

[151] C.-A. Petre and A. Korodi, "Honeypot inside an OPC UA wrapper for water pumping stations," in *Proc. 22nd Int. Conf. Control Syst. Comput. Sci. (CSCS)*, Bucharest, Romania, 2019, pp. 72–77.

[152] G. Bernieri, M. Conti, and F. Pascucci, "MimePot: A model-based honeypot for industrial control networks," in *Proc. IEEE Int. Conf. Syst. Man Cybern. (SMC)*, Bari, Italy, Oct. 2019, pp. 433–438.

[153] K. Wilhoit, "The SCADA that didn't cry wolf who's really attacking your ICS equipment? (Part 2)," Trend Micro, Shibuya City, Japan, White Paper, 2013. [Online]. Available: https://www.trendmicro.de/cloud-content/us/pdfs/security-intelligence/white-papers/wp-the-scada-that-didnt-cry-wolf.pdf

[154] (2020). *OPC Unified Architecture*. Accessed: May 2, 2020. [Online]. Available: https://opcfoundation.org/about/opc-technologies/opc-ua/

[155] K. Wilhoit and S. Hilt, "The GasPot experiment: Unexamined perils in using gas-tank-monitoring systems," in *Proc. Black Hat USA*, 2015, pp. 1–24.

[156] Z. Ammar and A. AlSharif, "Deployment of IoT-based honeynet model," in *Proc. 6th Int. Conf. Inf. Technol. IoT Smart City*, Dec. 2018, pp. 134–139.

[157] M. Du and K. Wang, "An SDN-enabled pseudo-honeypot strategy for distributed denial of service attacks in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 16, no. 1, pp. 648–657, Jan. 2020.

[158] B. Green, A. Lee, R. Antrobus, U. Roedig, D. Hutchison, and A. Rashid, "Pains, gains and PLCs: Ten lessons from building an industrial control systems testbed for security research," in *Proc. 10th USENIX Workshop Cyber Security Exp. Test*, Vancouver, BC, Canada, Aug. 2017, p. 4.

[159] T. Alves, R. Das, and T. Morris, "Virtualization of industrial control system testbeds for cybersecurity," in *Proc. 2nd Annu. Ind. Control Syst. Security Workshop*, 2016, pp. 10–14.

[160] S. Almulla, E. Bou-Harb, and C. Fachkha, "Cyber security threats targeting CPS systems: A novel approach using honeypot," in *Proc. SECURWARE 12th Int. Conf. Emerg. Security Inf. Syst. Technol.*, Dec. 2018, pp. 85–91.

[161] A. I. Newaz, A. K. Sikder, M. A. Rahman, and A. S. Uluagac, "A survey on security and privacy issues in modern healthcare systems: Attacks and defenses," *ACM Trans. Comput. Healthcare*, vol. 2, no. 3, pp. 1–44, 2020.

[162] A. I. Newaz, A. K. Sikder, L. Babun, and A. S. Uluagac, "HEKA: A novel intrusion detection system for attacks to personal medical devices," in *Proc. IEEE Conf. Commun. Netw. Security (CNS)*, Avignon, France, 2020, pp. 1–9.

[163] A. I. Newaz, A. K. Sikder, M. A. Rahman, and A. S. Uluagac, "HealthGuard: A machine learning-based security framework for smart healthcare systems," in *Proc. 6th Int. Conf. Soc. Netw. Anal. Manage. Security (SNAMS)*, 2019, pp. 389–396.

[164] H. Oz, A. Aris, A. Levi, and A. Selcuk Uluagac, "A survey on ransomware: Evolution, taxonomy, and defense solutions," Feb. 2021. [Online]. Available: arXiv:2102.06249.

[165] F. Naseem, A. Aris, L. Babun, E. Tekiner, and S. Uluagac, "MINOS: A lightweight real-time cryptojacking detection system," in *Proc. 28th Annu. Netw. Distrib. Syst. Security Symp. (NDSS)*, Feb. s2021, pp. 1–15.

[166] E. Tekiner, A. Acar, A. S. Uluagac, E. Kirda, and A. A. Selcuk, "SoK: Cryptojacking Malware," Mar. 2021. [Online]. Available: arXiv:2103.03851.

[167] L. P. Rondon, L. Babun, K. Akkaya, and A. S. Uluagac, "HDMI-walk: Attacking HDMI distribution networks via consumer electronic control protocol," in *Proc. 35th Annu. Comput. Security Appl. Conf.*, 2019, pp. 650–659.

[168] L. C. PucheRondon, L. Babun, K. Akkaya, and A. S. Uluagac, "HDMI-watch: Smart intrusion detection system against HDMI attacks," *IEEE Trans. Netw. Sci. Eng.*, early access, Aug. 28, 2020, doi: 10.1109/TNSE.2020.3020084.

[169] L. P. Rondon, L. Babun, A. Aris, K. Akkaya, and A. S. Uluagac, "Poisonivy: (in)secure practices of enterprise IoT systems in smart buildings," in *Proc. 7th ACM Int. Conf. Syst. Energy Efficient Build. Cities Transp.*, 2020, pp. 130–139.

[170] L. P. Rondon, L. Babun, A. Aris, K. Akkaya, and A. S. Uluagac, "Survey on enterprise Internet-of-Things systems (E-IoT): A security perspective," Feb. 2021. [Online]. Available: arXiv:2102.10695.

[171] FieldComm Group. (2020). *HART Communication Protocol*. Accessed: May 14, 2020. [Online]. Available: https://fieldcommgroup.org/technologies/hart

[172] L. Babun, K. Denney, Z. B. Celik, P. McDaniel, and A. S. Uluagac, "A survey on IoT platforms: Communication, security, and privacy perspectives," *Comput. Netw.*, vol. 192, Jun. 2021, Art. no. 108040.

[173] H. Aksu, A. S. Uluagac, and E. S. Bentley, "Identification of wearable devices with bluetooth," *IEEE Trans. Sustain. Comput.*, vol. 6, no. 2, pp. 221–230, Jun. 2021.

[174] Center for Internet Security. (2021). *The SolarWinds Cyber-Attack: What You Need to Know*. Accessed: Mar. 26, 2021. [Online]. Available: https://www.cisecurity.org/solarwinds/

[175] A. Afianian, S. Niksefat, B. Sadeghiyan, and D. Baptiste, "Malware dynamic analysis evasion techniques: A survey," *ACM Comput. Surveys*, vol. 52, no. 6, pp. 1–28, Nov. 2019.

[176] The MITRE Corporation. (2020). *Common Vulnerabilities and Exposures*. Accessed: May 17, 2020. [Online]. Available: https://cve.mitre.org/