FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

HONEYPOT-BASED SECURITY ENHANCEMENTS FOR INFORMATION SYSTEMS

A dissertation submitted in partial fulfillment of the

requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

by

Javier R. Franco

2022

To: Dean John L. Volakis
College of Engineering and Computing

This dissertation, written by Javier R. Franco, and entitled Honeypot-based Security Enhancements for Information Systems, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Kemal Akkaya

Alexander Perez-Pons

A. Selcuk Uluagac, Major Professor

Date of Defense: June 14, 2022

The dissertation of Javier R. Franco is approved.

Dean John L. Volakis
College of Engineering and Computing

Andrés G. Gil
Vice President for Research and Economic Development
and Dean of the University Graduate School

Florida International University, 2022

DEDICATION

To my family. Thank you for your love, sacrifices, and unwavering support.

ACKNOWLEDGMENTS

I would like to thank my major professor, Dr. Selcuk Uluagac, Director of the Cyber-Physical Systems Security (CSL) lab, for all of his support, mentorship, and guidance during my graduate studies. I would also like to express my gratitude to my committee members, Dr. Kemal Akkaya and Dr. Alexander Perez-Pons for their contributions. A very special thanks to Dr. Ahmet Aris, Dr. Abbas Acar, and Dr. Leonardo Babun for their insights and guidance.

ABSTRACT OF THE DISSERTATION

HONEYPOT-BASED SECURITY ENHANCEMENTS FOR INFORMATION SYSTEMS

by

Javier R. Franco

Florida International University, 2022

Miami, Florida

Professor A. Selcuk Uluagac, Major Professor

The purpose of this thesis is to explore honeypot-based security enhancements for information systems. First, we provide a comprehensive survey of the research that has been carried out on honeypots and honeynets for Internet of Things (IoT), Industrial Internet of Things (IIoT), and Cyber-physical Systems (CPS). We provide a taxonomy and extensive analysis of the existing honeypots and honeynets, state key design factors for the state-of-the-art honeypot/honeynet research and outline open issues. Second, we propose S-Pot, a smart honeypot framework based on open-source resources. S-Pot uses enterprise and IoT honeypots to attract attackers, learns from attacks via ML classifiers, and dynamically configures the rules of SDN. Our performance evaluation of S-Pot in detecting attacks using various ML classifiers shows that it can detect attacks with 97% accuracy using J48 algorithm. Third, for securing host-based Docker containers from cryptojacking, using honeypots, we perform a forensic analysis to identify indicators for the detection of unauthorized cryptomining, present measures for securing them, and propose an approach for monitoring host-based Docker containers for cryptojacking detection. Our results reveal that host temperature, combined with container resource usage, Stratum protocol, keywords in DNS requests, and the use of the container's ephemeral ports are notable indicators of possible unauthorized cryptomining.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The digital transformation has been converting all aspects of life in recent years. The ever-growing number of Internet of Things (IoT) devices has exacerbated demand on traditional networks, making it increasingly complex to manage and scale. Furthermore, enterprise networks are becoming increasingly heterogeneous where enterprise devices and IoT devices coexist, requiring tools for effective management and security. Software-Defined Networking (SDN) has emerged in response to such needs of modern networks. SDN emerged in response to these needs, transforming networking infrastructure, moving the brain from network devices to a centralized software controller [LSFF16]. More and more enterprise networks and global network infrastructures are moving to SDN [The20b], and the market size of SDN is expected to reach $59 billion by 2023 [Mar]. Alongside these fast-paced changes, constantly evolving cyber threats are increasing in quantity and impact. The cost of cybercrime is expected to reach $6 trillion in 2021, and reach $10.5 trillion in 2025 [Mor]. At the same time, blockchain-based cryptocurrencies have transformed financial transactions and created opportunities to profit from generating new coins through cryptomining. This has led to cybercriminals stealthily using their victim's computational power and resources for their own profit. Recent trends point to an increase in targeting devices with greater processing power, such as host-based Docker engines, through cryptojacking for faster and greater profit. Docker [Sas21] has become one of the top three most popular development platforms [Sta20], and thus a target for cyber criminals to maximize profit using host-based cryptojacking.

These realities demand further research to seek dynamic solutions. In order to protect IoT, Industrial Internet of Things (IIoT), Cyber-Physical System (CPS), and enterprise environments from malicious entities, honeypots and honeynets can

be used in conjunction with traditional security mechanisms to allow security researchers to observe, analyze, and continuously learn from attacks to develop effective defense mechanisms [FDFV18]. In essence, a honeypot is a decoy that is used to lure and deceive attackers into thinking they have accessed a real system to gather information about their interaction with the honeypot [FACU21]. Research honeypots have been a very active field of research during the last decade. However, there have been few research studies on the use of honeypot data for developing security solutions for production purposes [FACU21]. Despite the wealth of research in SDN security, none of the studies have benefited from honeypots for production purposes for the security of SDN-based enterprise networks. Moreover, the current literature has not explored cryptojacking targeting Docker containers and its detection methods considering honeypots.

1.1 Objectives

The objectives of this thesis are summarized as follows:

- To provide a detailed analysis of honeypots and honeynets proposed for IoT, IIoT, and CPS environments and a comprehensive analysis of IoT, IIoT, and CPS honeypots and honeynets, including intriguing characteristics that are shared by studies.
- To identify key design factors for future IoT, IIoT, and CPS honeypots and honeynets, and present open research problems that still need to be addressed in honeypot and honeynet research for IoT, IIoT, and CPS.
- To design, implement, and evaluate a novel, smart honeypot framework based on open-source resources, that benefits from enterprise and IoT honeypots,

IDPS, and ML classifiers for securing SDN-based hybrid enterprise networks through dynamic rules configuration.

- To identify key indicators for the detection of cryptomining in host-based Docker containers through forensic analysis using honeypots, propose an approach for monitoring host-based Docker containers, and present countermeasures to protect them from cryptojacking attacks.

1.2 Organization of the Thesis

As part of this thesis, Chapter 2 provides a comprehensive survey on honeypot and honeynet models that were proposed for IoT, IIoT, and CPS environments. Next, Chapter 3 introduces a smart honeypot framework based on open-source resources, that benefits from enterprise and IoT honeypots and integrates the use of IDPS and ML for securing SDN-based hybrid enterprise networks through dynamic rules configuration. Chapter 4 presents a forensic analysis for detecting unauthorized cryptomining using honeypots, measures for securing host-based Docker containers, and an approach for monitoring host-based Docker containers for cryptojacking detection. Finally, in Chapter 5, we present concluding remarks and areas of interest for future research.

CHAPTER 2

A SURVEY OF HONEYPOTS AND HONEYNETS FOR INTERNET OF THINGS, INDUSTRIAL INTERNET OF THINGS, AND CYBER-PHYSICAL SYSTEMS

2.1 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 [SSH+18]. 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 [BARM17, HLLL17]. As it can be seen from the descriptions of IoT, IIoT, and CPS, these concepts do not have explicit separation points. Border et al. [BARM17] and the National Institute of Standards and Technology's (NIST) special report by Greer et al. [GBWG19] 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. [GBWG19] 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. [GBWG19] 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 [MAL+19].

IoT devices typically have constrained power, storage, computing, and communications resources which limit the accommodation of good security mechanisms [MCZ+19, NBHC+19]. 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 [Sco14]. 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 [SCGM13, Sco14, YKYZ21, LBAU17].

In order to protect IoT, IIoT, and CPS environments from malicious entities, traditional security mechanisms such as 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 [FDFV18]. 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 [Spi01]. Two or more honeypots implemented on a system form a honeynet [KV17]. 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 [FDFV18].

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 chapter is organized as follows: Section 2.2 gives the related work. Section 2.3 provides background information on honeypots, honeynets, and related terms. Section 2.4 provides a methodology for the classification of honeypot and honeynet characteristics. Section 2.5 classifies and presents diverse IoT honeypot and honeynet models and research. Section 2.6 presents a taxonomy of the proposed IoT honeypot and honeynet models. Section 2.7 classifies and presents diverse CPS and IIoT honeypot and honeynet models and research. Section 2.8 presents a taxonomy of the proposed CPS and IIoT honeypot and honeynet models. Section 2.9 provides lessons learned and design considerations for honeypot and honeynet implementations. In Section 2.10, conclusions and future work are presented.

2.2 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. [BÖS20] and Makhdoom et al. [MAL+19] for extensive overviews of vulnerabilities, threats, and attacks, the security surveys of Lee et al. [LSOK21] on IoT standards and Granjal et al. [GMSS15] on the existing IoT protocols, the study of Neshenko et al. [NBHC+19] for a recent comprehensive

IoT security survey, the study of Sikder et al. [SPA+21] for a survey of threats to IoT sensors, the study of Humayed et al. [HLLL17] for an extensive survey on the threats, vulnerabilities, attacks, and defense solutions to CPS, the survey of Al-Garadi et al. [AGMAA+20] for machine and deep learning techniques for IoT security, the comprehensive survey of Yu et al. [YKYZ21] for CPS security and Cintuglu et al. [CMAU17] for CPS testbeds. There are also studies like that of Babun et al. [BAR+20] 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. [FDF15] 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. [FDFV18] 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 [MBVJ11, CPM15], and [ZKF19]. In addition, privacy and liability issues when honeypots are deployed were analyzed by Sokol et al. [SA15,SHL15]. In terms of honeypots and honeynets for IoT, IIoT, and CPS, only a few surveys exist in the literature. Razali et al. [RRM+18] analyzed types, properties, and interaction levels of IoT honeypots and classified honeynet models based on interaction, resources, purpose, and role. Dalamagkas et al. [DSI+19] surveyed the honeypot and honeynet frameworks for smart-grid environments. Dowling et al. [DSM17a] 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. [NBHC+19] 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. [FFV15] 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. [ANFL18] 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. [SHH+19] focused on techniques for honeypot detection with SSH and Telnet, identifying issues of software architecture and implementation that make honeypots easily detected [Sur]. Zamiri-Gourabi et al. [ZGQA19] 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.

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 [SSH+18]. 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 [BARM17, HLLL17]. As it can be seen from the descriptions of IoT, IIoT, and CPS, these concepts do not have explicit separation points. Border et al. [BARM17] and the National Institute of Standards and Technology's (NIST) special report by Greer et al. [GBWG19] 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. [GBWG19] 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. [GBWG19] 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 [MAL+19].

IoT devices typically have constrained power, storage, computing, and communications resources which limit the accommodation of good security mechanisms [MCZ+19, NBHC+19]. 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 [Sco14]. 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 [SCGM13, Sco14, YKYZ21, LBAU17].

In order to protect IoT, IIoT, and CPS environments from malicious entities, traditional security mechanisms such as 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 [FDFV18]. 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 [Spi01]. Two or more honeypots implemented on a system form a honeynet [KV17]. 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 [FDFV18].

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 chapter is organized as follows: Section 2.2 gives the related work. Section 2.3 provides background information on honeypots, honeynets, and related terms. Section 2.4 provides a methodology for the classification of honeypot and honeynet characteristics. Section 2.5 classifies and presents diverse IoT honeypot and honeynet models and research. Section 2.6 presents a taxonomy of the proposed IoT honeypot and honeynet models. Section 2.7 classifies and presents diverse CPS and IIoT honeypot and honeynet models and research. Section 2.8

presents a taxonomy of the proposed CPS and IIoT honeypot and honeynet models. Section 2.9 provides lessons learned and design considerations for honeypot and honeynet implementations. In Section 2.10, conclusions and future work are presented.

2.3 Background Information

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

2.3.1 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 [KV17], or in a more narrow definition, a honeynet is a high interaction honeypot system of Generation I, II, or III [Hon01]. 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 ([A.G17, Evr16, PB16, HMP+20, LFR+16]) 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 [OKK18]. *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 2.1 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 [Hon01]. The data collection involves the ability to securely transfer all the captured data to a centralized place [FDF15].

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 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.

2.3.2 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 frame-

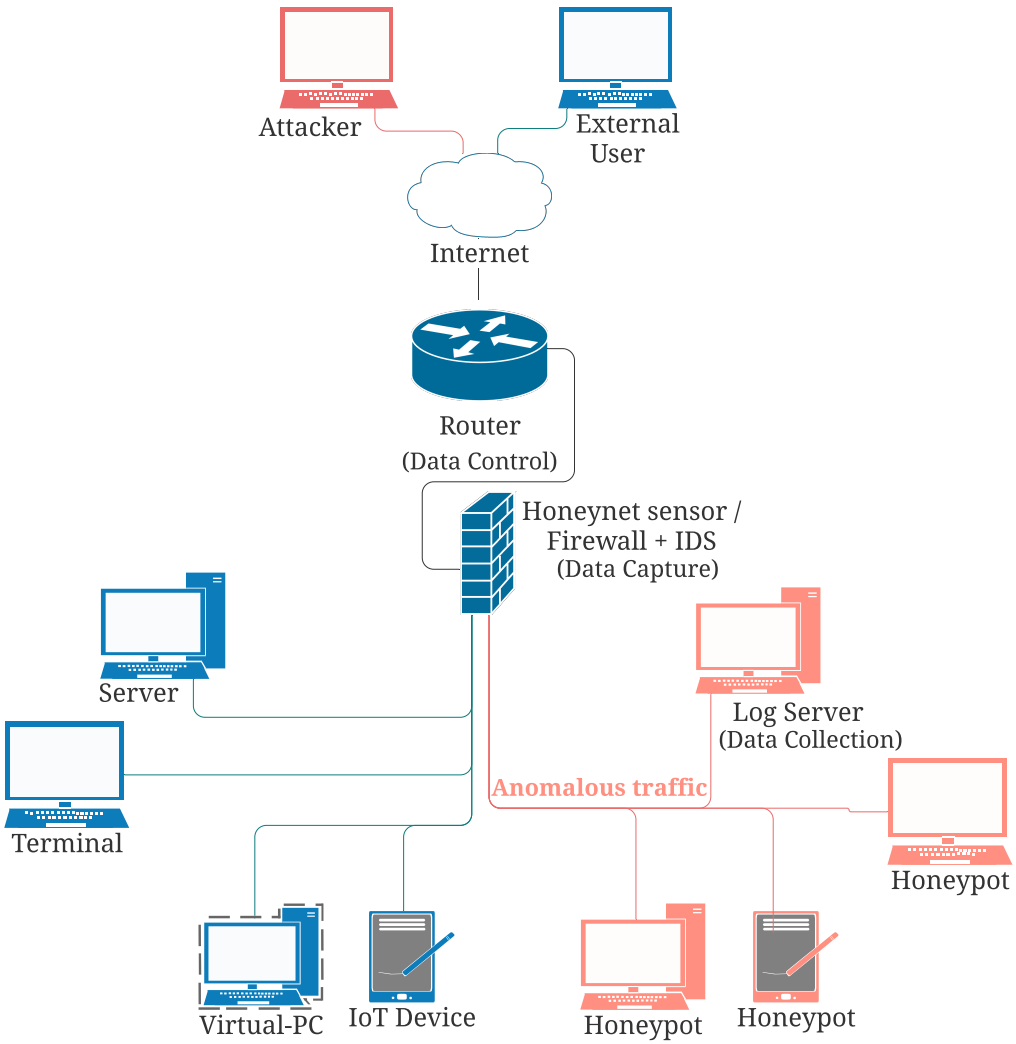

Figure 2.1: Basic honeynet architecture.

works. 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 [AT15], the IMUNES emulator/simulator [Zec03], the GridLab-D power

distribution simulator [Gri20], the SoftGrid smart grid security toolkit [GMC16], the PowerWorld simulator [Pow20], and the Mininet emulator [LHM10] 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.

2.4 Classification Methodology

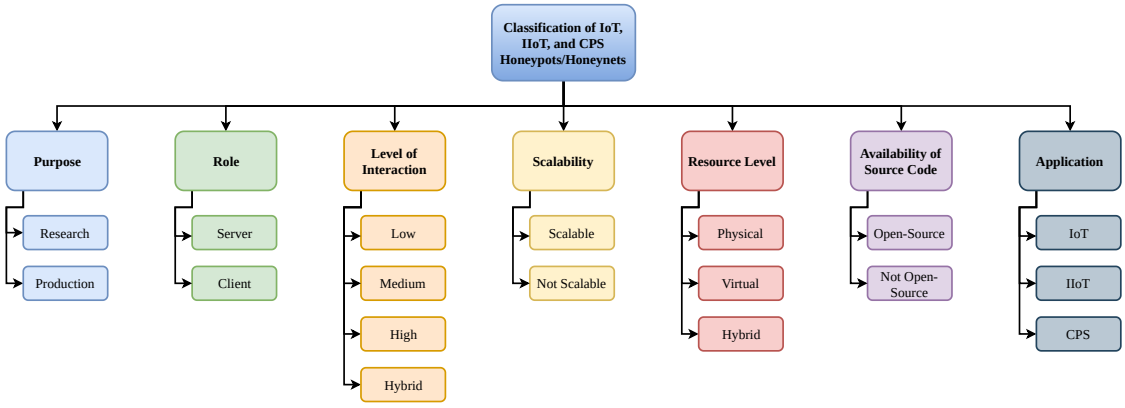

Figure 2.2: Classification categories of honeypots and honeynets for IoT, IIoT, and CPS in which some of the items in the categorization build upon [FDFV18, CPM15, ZKF19, RRM+18]. Details of works corresponding to each category are tabulated in Tables I, II, and III.

Honeypots and honeynets can be classified in various ways. In order to classify the honeypots and honeynets for IoT, IIoT, and CPS in this survey, we build upon prior surveys [FDFV18, CPM15, ZKF19, RRM+18]. 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. 2.2. 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 [Spi01].

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 [FDFV18].

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 [RRM+18].

High interaction honeypots provide much more interaction, not only emulating services but also allowing access to an operating system [RRM+18]. 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 [RRM+18]. 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 2.3 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 [DSI+19].

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 [FDFV18]. 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.

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

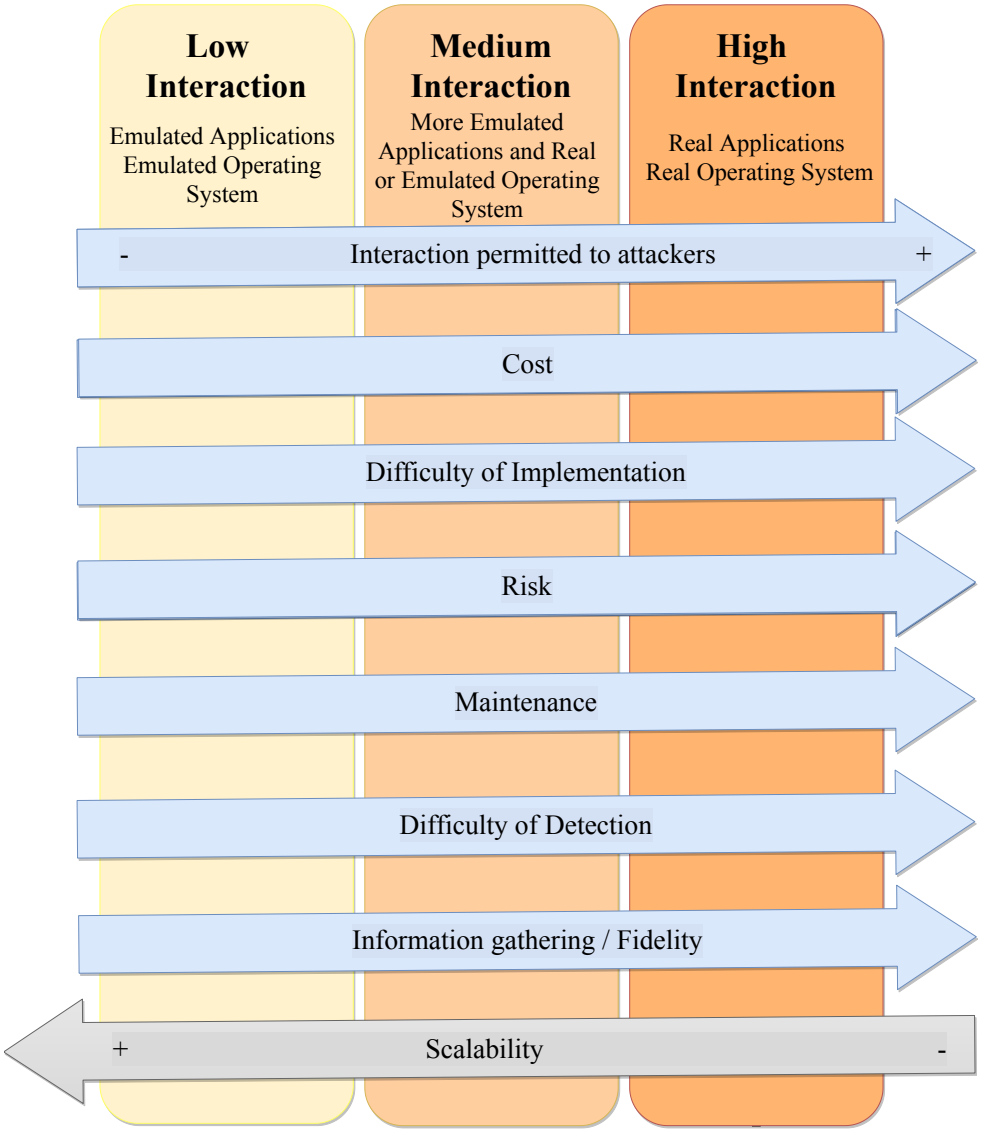

Figure 2.3: Characteristics by level of interaction.

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 [FDF15].

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 study for the classification by application purposes.

2.5 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.

2.5.1 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 [Pau12]. Table 2.1 provides a list of the considered general IoT honeypots.

Honeyd: HoneyD [Pro07a] 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 [Sta19] 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.

Table 2.1: List of General IoT Honeypots

Honeypot	Interaction Level	Simulated Services
HoneyD [Pro07a]	Low	FTP, SMTP, Telnet, IIS, POP
Dionaea [Din]	Medium	Black hole, EPMAP, FTP, HTTP, Memcache, MongoDB, MQTT, MySQL, Nfq, PPTP, SIP, SMB, TFTP, UPnP
Kippo [Kip16]	Medium	SSH
Cowrie [Cow19]	Medium/High	SSH, Telnet, SFTP, SCP
HoneyPy [foo13]	Low/Medium	Created as required
AHA [Wag11]	Low/High	SSH
AHA with Rootkit Detection [Pau12]	Medium	SSH
RASSH [PB14]	Medium	SSH
QRASSH [PIB18]	Medium	SSH

Dionaea: Dionaea [Din] 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.) [Dio15]. 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 [FDFV18]. Metognon et al. [MS18] used Dionaea in their IoT honeypot research. Kaur and Pateriya [KP18] 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 [Kip16] 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 [DSM17b]. Dowling et al. [DSM17b] modified Kippo

in order to implement a ZigBee IoT honeypot. Kippo is chosen because of the high number of attacks that SSH receives. Pauna [PB14] used Kippo for the creation of Reinforced Adaptive SSH (RASSH) honeypot.

Adaptive Honeypot Alternative (AHA): Wagener [Wag11] 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) [G.W18]. While Wagener does not implement his honeypot in an IoT environment, his honeypot serves as the basis for Pauna's works [Pau12, PB14, PIB18, PBPC19]. 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 [Pau12] 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 [PB14, PIB18, PBPC19].

RASSH: In 2014, Pauna et al. presented an adaptive honeypot, RASSH [PB14], 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 [PIB18] [PBPC19], which led to the creation of IRASSH-T [Pau18a] self-adaptive IoT honeypot.

Cowrie: Cowrie [Cow19] 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 et al. [MS18], IRASSH-T [PBPC19], ML-Enhanced Cowrie [SBH19], and Lingenfelter et al. [LVS20].

HoneyPy: HoneyPy [foo13] 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 et al. [MS18] used HoneyPy in their IoT honeypot research.

QRASSH: In 2018, Pauna et al. [PIB18] proposed another SSH honeypot, namely Q Reinforced Adaptive SSH (QRASSH) [Pau18b] 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 [PBPC19] and led to the creation of IoT Reinforced Adaptive SSH (IRASSH-T) [Pau18a] honeypot.

Metongnon and Sadre: Metongnon and Sadre [MS18] 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 [Cow19], HoneyPy [foo13], and Dionaea [Dio15] 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.

2.5.2 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 2.2 provides a list of the considered IoT honeypots that perform full IoT device emulation.

Table 2.2: List of IoT Honeypots for Full Device Emulation

Honeypot	Interaction Level	Emulated Devices
FIRMADYNE [CEWB16b]	High	COTS network-enabled IoT devices
ThingPot [WSK18]	Medium	Philips Hue, Belkin, Wemo, Tplink
ML-Enhanced ThingPot [VJ19]	Medium	General IoT devices
IoTCandyJar [LXJ+17]	Intelligent	General IoT devices
Chameleon [Zho19]	Hybrid	Any real IoT device
Honware [VC19]	High	CPE devices

FIRMADYNE: Chen et al. [CEWB16b] presented FIRMADYNE [CEWB16a], 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. [WSK18] proposed ThingPot [Wan17], 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 [TP] to stay anonymous. Vishwakarma and Jain [VJ19] 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. [LXJ+17] 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 [Zho19] 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 [Sho] fingerprinting tool. The honeypots simulated by Chameleon were not fingerprinted while all the traditional honeypots were.

Honware: Vetterl and Clayton [VC19] 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 2.3: List of IoT Honeypots that Focus on Specific Attacks

Target Attack(s)	Honeypots	Interaction Level
Telnet	IoTPOT [PSY+16]	Hybrid
	MTPot [Cym], Semic and Mrdovic [SM17]	Low
	Phype [Phy19]	Medium
SSH and Telnet	Shrivastava et al. [SBH19], IRASSH-T [Pau18a], Lingenfelter et al. [LVS20]	Medium
Telnet, SSH, HTTP, and CPE WAN Management	Krishnaprasad [Kri17]	Hybrid
Man-in-the-Middle	Oza et al. [OKKT19]	High
D/DoS	Anirudh et al. [ATN17], Vishwakarma and Jain [VJ19]	Medium
	Tambe et al. [TAS+19], Molina et al. [MBSC20]	High
Fileless attacks	HoneyCloud [DLL+19]	High
SSH on Zigbee networks	Dowling et al. [DSM17b]	Medium
UPnP	U-Pot [Hak]	Medium
Attacks on Authentication	HioTPot [GKK+18]	Not identified
Reconnaissance	HoneyIo4 [A.G17]	Low
Attacks on home networks	Pot2DPI [MCB17]	Medium
Attacks on device characteristics	Siphon [GTB+17]	High
	Metongnon and Sadre [MS18]	Low/Medium
	Zhang et al. [ZZZ+19]	Hybrid

2.5.3 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 2.3 provides a list of the considered IoT honeypots by their target attack types.

Only Telnet Attacks: IoTPOT [PSY+16] is a hybrid honeypot proposed by Pa et al. [PSY+15] 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 [Cym] is a low-interaction, unscalable, virtual IoT honeypot that was designed specifically for Mirai attacks. According to Evron [Evr16], 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.

Semic and Mrdovic [SM17] 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 [Phy19] 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 [Phy19], 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. [SBH19] 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. [PBPC19] 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. [LVS20] 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 [Kri17] used IoTPOT [PSY+15] 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 [Lab14] event-driven networking engine, and employs Logstash [Ela20a] to collect log data. The log data is pushed to Elasticsearch [Ela17] for storage and Kibana [Ela20b] 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. [OKKT19] 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. [ATN17] 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 [WSK18], Vishwakarma and Jain [VJ19] 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. [TAS+19] 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 [Sho]. The authors also proposed two live traffic analysis methods for the detection of large scale attacks.

Molina et al. [MBSC20] 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. [DLL+19] 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. [DSM17b] 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 [Hak] 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 [GKK+18] 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 [A.G17] 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. [MCB17] 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 re-mapping between the router and IoT devices. HoneyD [Pro07b] 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 [GTB+17] 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. [ZZZ+19] 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.

2.6 Taxonomy of Honeypots and Honeynets for Internet of Things

Honeypots and honeynets proposed for IoT are listed in Table 2.4 and the tools, implementation and attack type details of the corresponding honeypots and honeynets are also outlined in Table 2.5. 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.

2.6.1 Development of Research Over Time

Research of honeypots specifically created for IoT begins in 2015 with the creation of IoTPOT [PSY+15]. Previous research included in this survey was originally created for general application and later built upon for IoT applications. As shown in Figure 2.4, about half of IoT honeypot models have a form of inheritance with each other, where a honeypot is built based on another. Cowrie [Cow19] 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 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 [LXJ+17] [A.G17] [GTB+17] [MCB17] [Kri17] [Evr16] [SM17]. 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.

2.6.2 Common Characteristics

All of the honeypot/honeynet models surveyed were created for research purposes, except for HoneyIo4 [A.G17] and the IoT honeynet presented by Molina [MBSC20], which are production, and HIoTPot [GKK+18] 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 [Phy19], which has both a client

and a server role. In addition, all of the open-source models were written in Python programming language.

2.6.3 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 study, 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.

2.6.4 Resource Level

The great majority of available research on IoT honeynets has been carried out with virtual resources rather than physical resources. Only Siphon [GTB+17], Stafira [Sta19], and Scalable VPN forwarded Honeypots [RRM+18] were carried out with physical resources.

2.6.5 Scalability

Most of the honeypot and honeynet research was carried out using scalable honeypot systems, except for Dionaea [Dio15] and HoneyIo4 [A.G17], 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.

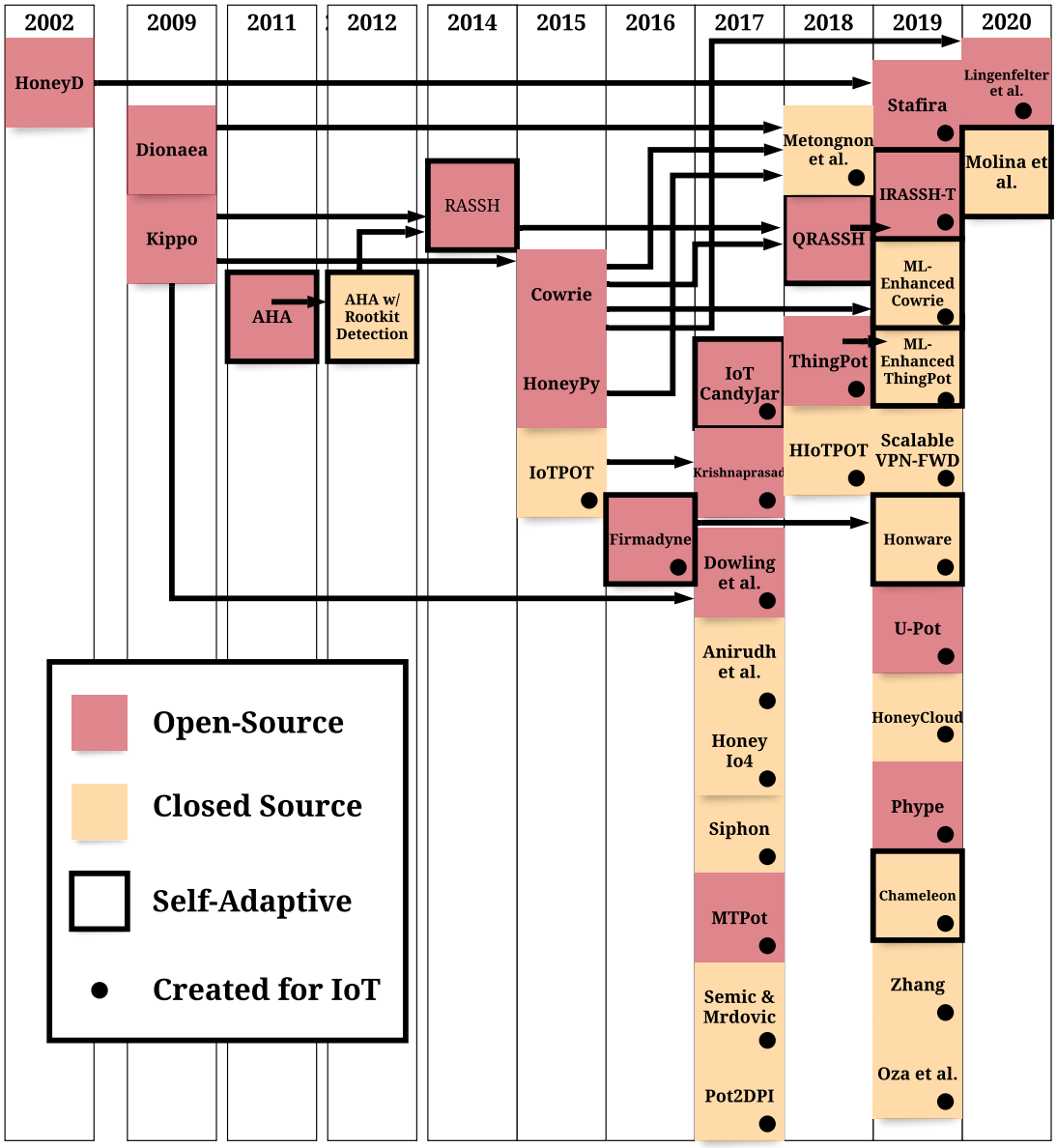

Figure 2.4: Evolution of inheritance for the IoT honeypot and honeynet models and research.

2.6.6 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.

2.6.7 Simulated Services

The most commonly simulated services in the research coincide with the top three most attacked protocols identified by Metongnon and Sadre [MS18]: 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 [Sym19]. 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 [LXJ+17], Chameleon [Zho19], Firmadyne [CEWB16b], Honware [VC19], and ML-enhanced ThingPot [VJ19]. However, of these, only IoTCandyJar and Firmadyne are open-source.

2.6.8 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.

Table 2.4: Classification of IoT Honeypots and Honeynets

Work	Year	Interaction level	Scalability	Resource level	Simulated services	Role	Application
HoneyD [Pro07b]	2002	Low	✓	Virtual	FTP, SMTP, Telnet,IIS, POP	Server	General
Dionaea [Dio15]	2009	Medium	X	Virtual	Black hole, EPMAP, FTP, HTTP, Memache, Mirror, MongoDB, MQTT, MSSQL, MySQL, nfq, PPTP, SIP, SMB, TFTP, UPnP	Server	General
Kippo [Kip16]	2009	Medium	✓	Virtual	SSH	Server	General
Adaptive Honeypot Alternative [Wag11]	2011	Low and High	✓	Virtual	SSH	Server	General
AHA with Rootkit Detection [Pau12]	2012	Medium	✓	Virtual	SSH	Server	General
RASSH [PB14]	2014	Medium	✓	Virtual	SSH	Server	General
Cowrie [Cow19]	2015	Medium/Hig	✓	Virtual	SSH, Telnet, SFTP, SCP	Server	General
HoneyPy [foo13]	2015	Low/Medium	✓	Virtual	Created as required	Server	General
IoTPOT [PSY+15]	2015	Hybrid	✓	Virtual	Telnet	Server	IoT
Firmadyne [CEWB16b]	2016	High	✓	Virtual	Full device emulation	Server	IoT
Dowling et al. [DSM17b]	2017	Medium	✓	Virtual	Zigbee, SSH, HTTP	Server	IoT
IoT CandyJar [LXJ+17]	2017	Intelligent	✓	Virtual	Full device emulation	Server	IoT
Krishnaprasad [Kri17]	2017	Hybrid	✓	Virtual	Telnet, SSH, HTTP, CWMP	Server	IoT

Table 2.4: Classification of IoT Honeypots and Honeynets

Work	Year	Interaction level	Scalability	Resource level	Simulated services	Role	Application
Anirudh et al. [ATN17]	2017	Medium/Hig	✓	Virtual	Not identified	Server	IoT
HoneyIo4 Production Honeypot [A.G17]	2017	Low	X	Virtual	SNMP, SSH, SMTP, DNS, HTTP	Server	IoT
Siphon [GTB+17]	2017	High	✓	Physical	HTTP, Telnet, SSH, RTSP	Server	IoT
MTPot [Evr16]	2017	Low	X	Virtual	Telnet	Server	IoT
Semic and Mrdovic [SM17]	2017	Low	✓	Virtual	Telnet	Server	IoT
Pot2DPI [MCB17]	2017	Medium	✓	Virtual	Telnet, UPnP	Server	Smart Home
Metongnon et al. [MS18]	2018	Low/Medium	✓	Virtual	SSH, Telnet, EPMAP, FTP, HTTP, Memcache, MQTT, MSSQL, MySQL, PPTP, SIP, SMB, UPnP, TFTP, TR-069.1,TR-069.2, CoAP	Server	IoT
QRASSH [PIB18]	2018	Medium	✓	Virtual	SSH	Server	General
ThingPot et al. [WSK18]	2018	Medium	✓	Virtual	Full device emulation	Server	Smart Home
HIoTPOT [GKK+18]	2018	Not identified	✓	Virtual	Not identified	Server	IoT
Stafira [Sta19]	2019	Low	✓	Physical	TCP/IP, HTTP	Server	Smart Home
IRASSH-T [PBPC19]	2019	Medium	✓	Virtual	SSH	Server	IoT
ML enhanced Cowrie [SBH19]	2019	Medium	✓	Virtual	SSH, Telnet	Server	IoT

Table 2.4: Classification of IoT Honeypots and Honeynets

Work	Year	Interaction level	Scalability	Resource level	Simulated services	Role	Application
ML enhanced ThingPot [VJ19]	2019	Medium	✓	Virtual	Full device emulation	Server	IoT
Scalable VPN-forwarded Honeypots [TAS+19]	2019	High	✓	Physical	HTTP, TFTP, Telnet, others not specified	Server	IoT
Zhang [ZZZ+19]	2019	Hybrid	✓	Physical/Virtu	UPnP, SOAP	Server	IoT
U-Pot [Hak]	2019	Medium	✓	Virtual	UPnP	Server	IoT
HoneyCloud [DLL+19]	2019	High	✓	Physical/Virtu	SSH, Telnet, SMB, HTTP, HTTPS, RDP, MySQL, SQL Server	Server	Smart Home
Phype [Phy19]	2019	Medium	✓	Virtual	Telnet	Server	IoT
Oza et al. [OKKT19]	2019	High	✓	Virtual	Not identified	Server	IoT
Honware [VC19]	2019	High	✓	Virtual	Full device emulation	Server	IoT
Chameleon [Zho19]	2019	Hybrid	✓	Virtual	Full device emulation	Server	IoT
Lingenfelter et al. [LVS20]	2020	Medium	✓	Virtual	SSH, Telnet, SMTP, HTTP	Server	IoT
Molina et al. [MBSC20]	2020	High	✓	Virtual	Not identified	Server	IoT

Table 2.5: Tools, Implementation and Attack Types of Honeypots and Honeynets for IoT

Work	Tools	Simulated services	Attack Types	Data Analyzed	Study Length
HoneyD [Pro07b]	N/A	FTP, SMTP, Telnet, IIS, POP	N/A	N/A	N/A
Dionaea [Dio15]	N/A	Black TTP, Memache, Mirror, MongoDB, MQTT, MSSQL, MySQL, Nfq, PPTP, SIP, SMB, TFTP, UPnP	N/A	N/A	N/A
Kippo [Kip16]	N/A	SSH	N/A	N/A	N/A
Adaptive Honeypot Alternative [V	AHA Daemon	SSH	SSH-brute force	User/passwords, TTY buffer, TCP/UDP packets	8 hours
AHA with Rootkit Detection [Pau12]	AHA Daemon, Kernel rootkit Kbeast, Argos	SSH	Rootkit malware	Keystroke logging, Rootkit malware	7 days
RASSH [PB14]	Pybrain RL, SARSA, Markov	SSH	SSH attack	Logs, commands offering downloading	N/A
Cowrie [Cow19]	N/A	SSH, Telnet, SFTP, SCP	N/A	N/A	N/A
HoneyPy [foo13]	N/A	Created as required	N/A	N/A	N/A

Table 2.5: Tools, Implementation and Attack Types of Honeypots and Honeynets for IoT

Work	Tools	Simulated services	Attack Types	Data Analyzed	Study Length
IoTPOT [PSY+15]	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 [CEWB16b]	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 [LXJ+17]	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
Krishnaprasa [Kri17]	Twisted, ELK Stack, Docker,	Telnet, SSH, HTTP, CWMP	Brute-force attack, Hajime, ZmEu attacks	Attack types and characteristics	7 days

Table 2.5: Tools, Implementation and Attack Types of Honeypots and Honeynets for IoT

Work	Tools	Simulated services	Attack Types	Data Analyzed	Study Length
Anirudh et al. [ATN17]	IDS, logs	N/A	DoS attacks	IP Address, MAC Address	N/A
HoneyIo4 Production Honeypot [A.G17]	Shodan, Nmap, Wireshark, Scapy, VM.	SNMP, SSH, SMTP, DNS, HTTP	Reconnaissance attacks	TCP, UDP and ICMP packets	N/A
Siphon [GTB+17]	Shodan, Tcpdump, Nmap	HTTP, Telnet, SSH, RTSP	Brute-force login attempts	TCP connections per wormhole, services consulted, access gained, movements statistics	60 days
MTPot [Evr16]	N/A	Telnet	N/A	Incoming connections on any port using telnet	N/A
Semic and Mrdovic [SM17]	N/A	Telnet	Telnet attack	Protocols, IP addresses, logs	N/A
Pot2DPI [MCB17]	N/A	Telnet, UPnP	Mirai and Persirai attacks protocols, ports scans	Packet traces, attack signatures, protocols, ports	N/A

Table 2.5: Tools, Implementation and Attack Types of Honeypots and Honeynets for IoT

Work	Tools	Simulated services	Attack Types	Data Analyzed	Study Length
Metongnon et al. [MS18]	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 [PIB18]	Deep Q-learning, Keras with Theano backend, Nmap	SSH	SSH attack	Commands(downloadi hacking, linux)	N/A
ThingPot et al. [WSK18]	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 [Sta19]	Nmap, Wireshark, VMWare Workstation	TCP/IP, HTTP	Only user testing	Access time, HTML code, network headers and Nmap scan	N/A
IRASSH-T [PBPC19]	Apprenticeship Learning	SSH	SSH attack	N/A	N/A

Table 2.5: Tools, Implementation and Attack Types of Honeypots and Honeynets for IoT

Work	Tools	Simulated services	Attack Types	Data Analyzed	Study Length
ML enhanced Cowrie [SBH19]	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 [VJ19]	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 [TAS+19]	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 [ZZZ+19]	Tc, own script	UPnP, SOAP	UPnP	Protocols, packets per port, timestamp, inject behaviors	7 days
U-Pot [Hak]	Shodan, Zmap, U-Pot	UPnP	N/A	N/A	N/A

Table 2.5: Tools, Implementation and Attack Types of Honeypots and Honeynets for IoT

Work	Tools	Simulated services	Attack Types	Data Analyzed	Study Length
HoneyCloud [DLL+19]	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 [Phy19]	Phype Telnet	Telnet	N/A	N/A	N/A
Chameleon [Zho19]	Nmap, Shodan	N/A	Reconnaissance attacks	IP whitelist, received requests	N/A
Honware [VC19]	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. [OKKT19]	OAuth2, MySQL, QEMU	N/A	Man in the Middle attacks	MAC address, Unauthorized access	N/A
Lingenfelter [LVS20]	Filebeat, ELK stack, Logstash, VirusTotal	SSH, Telnet	IoT botnet malware	Packets per port, System logs, IPs, Brute-force scan, file hash	40 days

2.6.9 Most Commonly Used Tools

The three tools that were most commonly used in the honeypot research studies included in this survey are Shodan [Sho20], Nmap [Nma20], and MASSCAN [Gra19]. Shodan [Sho20] 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 [Nma20] 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 [Gra19] 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.

2.6.10 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.

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

2.7.1 Honeypots and Honeynets for Industrial Control Systems

In this subsection, we give brief overview of honeypots and honeynets ICS. Table **??** 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 [PF04]

in 2004. SCADA HoneyNet is based on the Honeyd [Pro07a] 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 [Pet06, Bon11]. 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 [PF04] that aims to monitor and control the honeypot's traffic and attacker interactions.

Wade [Wad11] 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 [RVH+20]. It is an open-source low-interaction honeypot that was developed under the Honeynet Project [Hon20] 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.

CryPLH: Buza et al. [BJM+14] 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

Table 2.6: List of Smart Grid Honeypots and Honeynets

Honeypots	Interaction Level	Simulated Services
CryPLH [BJM+14]	Low	HTTP(S), SNMP, Step7 ISO-TSAP
SHaPe [KG15]	Low	IEC 61850 MMS, HTTP, FTP, SMB
GridPot [RLB15]	Hybrid	IEC 61850 GOOSE/MMS, Modbus, HTTP
Scott [Sco14]	Low	Modbus/TCP, HTTP, SNMP
Mashima et al. [MCGT17]	Low	IEC 60870-5-104, IEC 61850, SSH
Pliatsios et al. [PSL+19]	Low	Modbus/TCP
Mashima et al. [MLC19]	Low	TCP port listener on IEC 61850 MMS, S7comm, Modbus/TCP, Niagara Fox, EtherNet/IP, IEC 60870-5-104, DNP3, BACnet

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 [KG15], 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. [RLB15] 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 sim-

ulation. In their demonstration, namely GridPot, the authors utilized GridLab-D simulator [Gri20] 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.

Kendrick and Rucker [KR19] 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 [Sco14] 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. [MCGT17] 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 [LHM10] virtual hosts and SoftGrid [GMC16] IED simulations; and simulation of smart grid components (e.g., circuit breakers, transformers, etc.) via POWERWORLD [Pow20] power

simulator. The proposed honeynet is highly scalable and resistant to fingerprinting against Shodan and attacker tools such as Nmap.

Hyun [Hyu18] 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.

Pliatsios et al. [PSL+19] proposed a honeypot system for Smart-Grid 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. [MLC19] 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.

2.7.2 Honeypots and Honeynets for Water Systems

In this category, we give brief overview of the honeypots and honeynets for water systems. Table 2.7 provides a list of the water system honeypots.

Wilhoit [Wil13b, Wil13a] 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. [AAT16] proposed a virtual high interaction honeypot for ICS that is based on the MiniCPS ICS simulation framework [AT15]. 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

Table 2.7: List of ICS Honeypots for Water Systems

Honeypots	Interaction Level	Simulated Services
Wilhoit [Wil13b]	Hybrid	Modbus/TCP, HTTP, FTP
Antonioli et al. [AAT16]	High	EtherNet/IP, SSH, Telnet, VPN
Murillo et al. [MCG+18]	Low	EtherNet/IP
Petre and Korodi [PK19]	Medium	Modbus
MimePot [BCP19]	High	Modbus/TCP

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. [MCG+18]. The presented virtual testbed environment which uses MiniCPS [AT15] 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 [PK19] 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 [OPC20] 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. [BCP19] 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.

2.7.3 Honeypots and Honeynets for Gas Pipelines

Wilhoit and Hilt developed a low-interaction virtual honeypot, namely GasPot [WH15], 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. [ZGQA19] 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.

2.7.4 Honeypots and Honeynets for Building Automation Systems

Litchfield et al. [LFR+16] 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.

2.7.5 Honeypots and Honeynets for IIoT

Ammar and AlSharif [AA18] 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 [DW20a] 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.

2.8 Taxonomy of Honeypots and Honeynets for IIoT and CPS

Honeypots and honeynets proposed for IIoT and CPS are listed in Table 2.8 and the tools, implementation, and attack type details of the corresponding honeypots and honeynets are also outlined in Table 2.9. 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.

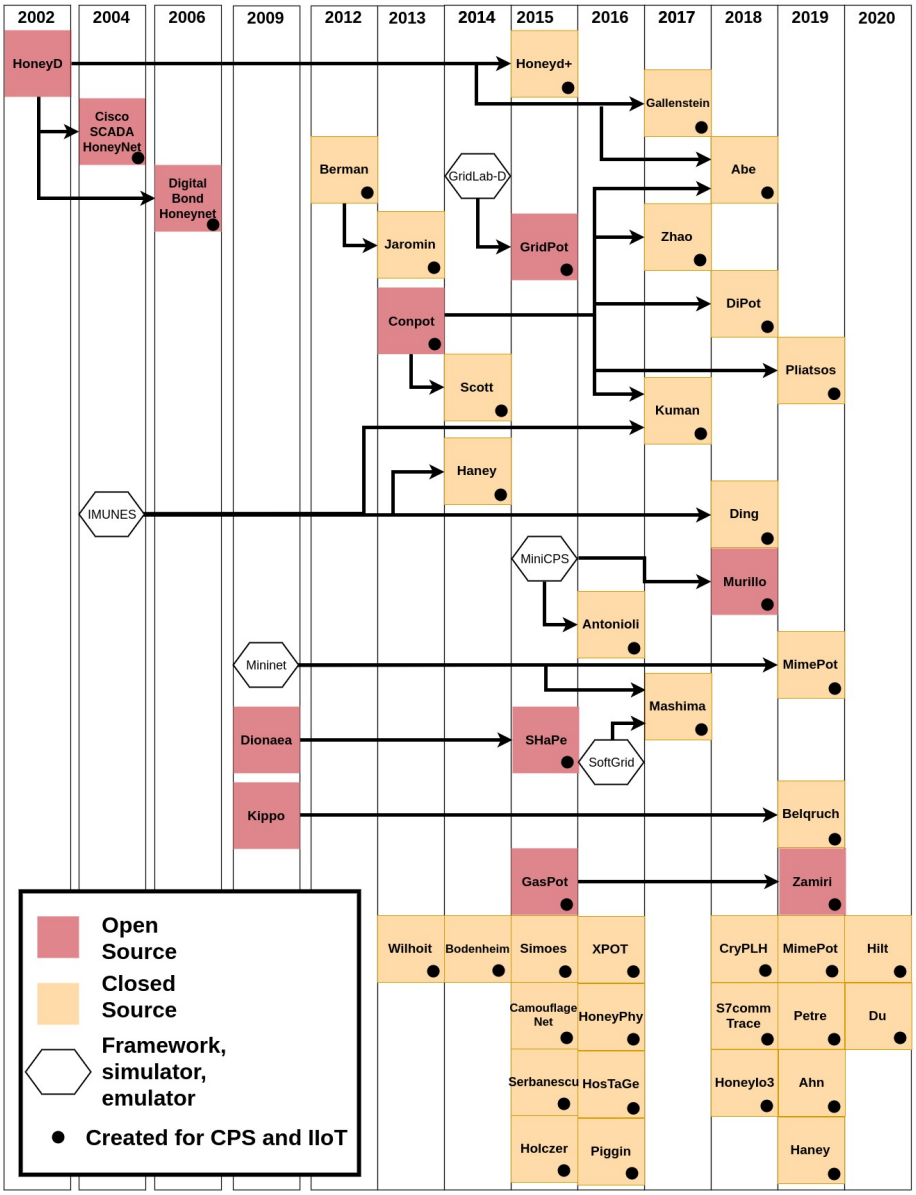

Figure 2.5: Evolution of inheritance for the honeypots and honeynets of IIoT and CPS.

2.8.1 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 2.5. As shown in the figure, honeypots and honeynets for IIoT and

CPS started with the SCADA HoneyNet [PF04] project of Pothamsetty and Franz from Cisco Systems in 2004. This project was followed by Digital Bond's SCADA Honeynet [Pet06] 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 [Ber12] in 2012 was the first study. His thesis was followed by another thesis conducted by Jaromin [Jar13] the following year. It is interesting to note that both studies were performed in the US 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 [RVH+20] started and Wilhoit from Trend Micro Research published the white paper of their low interaction ICS honeypots [Wil13b]. After these works, honeypot and honeynet research and practice in IIoT and CPS gained a momentum.

As shown in Figure 2.5, 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 [RVH+20] 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 [Hon20], 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 2.5 shows, Mininet and IMUNES emulators, GridLab-D, Soft-

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

2.8.2 Common Characteristics

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

In terms of purpose of the honeypots, we can see that the majority of the honeypots and honeynets outlined in Table 2.8 and Table 2.9 have research purposes. The only studies which have production purposes are Antonioli et al. [AAT16], Piggin et al. [PB16], and Scott [Sco14]. 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 [SCGM13, Sco14]. In addition to this, industrial devices typically have real-time constraints with guaranteed response times [HFB15]. 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).

Considering the roles of honeypots, we see that the overwhelming majority of the proposals have server roles. The honeypots and honeynets that have compo-

nents which act like clients are Haney et al. [Han19], Pliatsos et al. [PSL+19], and MimePot [BCP19].

Linux is by far the most popular operating system environment choice of honeypot and honeynet developers. Apart from Linux, we see that only Haney et al. [HP14] 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.

2.8.3 Level of Interaction

Honeypots and honeynets proposed for IIoT and CPS environments exhibit all possible interaction levels. In this respect, as Table 2.8 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. How-

ever, 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).

2.8.4 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 ([WRD+15, Gal17, MCGT17] and [GLA+17]) highlighted, components of an industrial system such as PLCs have high costs in the order of tens of thousands of dollars.

2.8.5 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 2.4, 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. [MCGT17] 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.

2.8.6 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).

2.8.7 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. [AAT16] used equation of continuity from hydraulics and Bernoulli's principle for the trajectories (for drain orefice), Murillo et al. [MCG+18] utilized a nonlinear model with Luenberger observer, and Bernieri et al. [BCP19] employed linear time invariant model for plant simulation. GridPot [RLB15] made use of Newton-Raphson power flow solver for electrical grid process. Lastly, for building automation systems, Litchfield et al. [LFR+16] considered Newton's Law of Cooling for the building process model.

Table 2.8: Classification of Honeypots and Honeynets for IIoT and CPS

Work	Year	Interaction Level	Scalability	Resource level	Simulated services	Role	Application
CISCO [PF04]	2004	Low	✓	Virtual	Modbus/TCP, Telnet, HTTP, FTP	Server	ICS
Digital Bond [Pet06]	2006	Low	✓	Virtual	Modbus/TCP, Telnet, HTTP, FTP, SNMP	Server	ICS
Conpot [RVH	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 [ZQ17]	2017	Medium	✓	Virtual	S7comm, Modbus, SNMP, HTTP	Server	ICS
DiPot [CLLL	2018	Low	✓	Virtual	HTTP, Modbus, Kamstrup, SNMP, IMPI, BACnet, Guardian AST, S7comm	Server	ICS
CamouflageN	2015	Low	✓	Virtual	N/A	Server	ICS
XPOT [LKA	2016	Medium	✓	Virtual	S7comm, SNMP	Server	ICS
HosTaGe [VS	2016	Low	✓	Virtual	Modbus, S7comm, HTTPS, FTP, MySQL, SIP, SSH, SNMP, HTTP, Telnet, SMB and SMT	Server	ICS
S7CommTrac	2018	Medium	✓	Virtual	S7comm	Server	ICS
Disso et al. [DJB13]	2013	Hybrid	Limited	Hybrid	N/A	Server	ICS
Honeyd+ [W	2015	High	Limited	Hybrid	EtherNet/IP, HTTP	Server	ICS
Gallenstein [Gal17]	2017	Low	✓	Virtual	EtherNet/IP. ISO-TSAP, HTTP	Server	ICS
Abe et al. [ATUH18]	2018	Low	✓	Virtual	Modbus, S7comm, BACNet, IPMI, Guardian AST, HTTP, SNMP	Server	ICS

Table 2.8: Classification of Honeypots and Honeynets for IIoT and CPS

Work	Year	Interaction Level	Scalability	Resource level	Simulated services	Role	Application
Haney et al. [HP14]	2014	Low	✓	Virtual	Modbus/TCP, Telnet, SSH, HTTP(S)	Server	ICS
Kuman et al. [KGM17]	2017	Low	✓	Virtual	Modbus/TCP	Server	ICS
Ding et al. [DZD18]	2018	Medium	✓	Virtual	S7comm, SNMP	Server	ICS
Bodenheim [Bod14]	2014	High	Limited	Physical	HTTP, EtherNet/IP, SNMP	Server	ICS
Piggin et al. [PB16]	2016	High	X	Physical	SSH, HTTP, RDP	Server	ICS
Haney [Han19]	2019	High	✓	Hybrid	Modbus/TCP, SSH, Telnet, SNMP, HTTP	Client, Server	ICS
Hilt et al. [HMP+20]	2020	High	X	Hybrid	S7comm, Omron FINS, EtherNet/IP, VNC	Server	ICS
Berman [Ber12]	2012	Low	Limited	Virtual	Modbus/TCP	Server	ICS
Jaromin [Jar13]	2013	Low	Limited	Virtual	Modbus/TCP, HAP, HTTP	Server	ICS
Holczer et al. [HFB15]	2015	High	✓	Virtual	S7comm, SNMP, HTTP(S)	Server	ICS
Serbanescu et al. [SOY15b]	2015	Low	✓	Virtual	DNP3, IEC-104, Modbus, ICCP, SNMP, TFTP, XMPP	Server	ICS
Simões [SCPM15]	2015	Low	✓	Virtual	Modbus, SNMP, FTP	Server	ICS
Ahn et al. [ALK19]	2019	Low	✓	Virtual	Modbus	Server	ICS

Table 2.8: Classification of Honeypots and Honeynets for IIoT and CPS

Work	Year	Interaction Level	Scalability	Resource level	Simulated services	Role	Application
Belqruch et al. [BM19]	2019	Medium	✓	Virtual	SSH	Server	ICS
SHaPe [KG15]	2015	Low	✓	Virtual	IEC 61850 MMS, HTTP, FTP, SMB	Server	Smart Grid
GridPot [RLB15]	2015	Hybrid	✓	Virtual	IEC 61850 GOOSE/MMS, Modbus, HTTP	Server	Smart Grid
Scott [Sco14]	2014	Low	✓	Virtual	Modbus/TCP, HTTP, SNMP	Server	Smart Grid
Mashima et al. [MCGT17]	2017	Medium / High	✓	Virtual	IEC 60870-5-104, IEC 61850, SSH	Server	Smart Grid
CryPLH [BJ	2018	Low	✓	Virtual	HTTP(S), SNMP, Step7 ISO-TSAP	Server	Smart Grid
Pliatsios et al. [PSL+19]	2019	Low	Limited	Hybrid	Modbus/TCP	Client, Server	Smart Grid
Mashima et al. [MLC19]	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. [MCG+18]	2018	Low	✓	Virtual	EtherNet/IP	Server	Water System
Petre et al. [PK19]	2019	Medium	✓	Virtual	Modbus	Server	Water System
Wilhoit [Wil	2013	Hybrid	Limited	Hybrid	Modbus/TCP, HTTP, FTP	Server	Water System

Table 2.8: Classification of Honeypots and Honeynets for IIoT and CPS

Work	Year	Interaction Level	Scalability	Resource level	Simulated services	Role	Application
Antonioli et al. [AAT16]	2016	High	✓	Virtual	EtherNet/IP, SSH, Telnet, VPN	Server	Water System
MimePot [BC	2019	High	✓	Virtual	Modbus/TCP	Client, Server	Water System
GasPot [WH15]	2015	Low	✓	Virtual	N/A	Server	Gas System
Zamiri et al. [ZGQA19]	2019	Medium	✓	Virtual	Veeder-Root ATG	Server	Gas System
HoneyPhy [LFR+16]	2016	Hybrid	X	Hybrid	DNP3	Server	Building Auto.
HoneyIo3 [A	2018	Hybrid	✓	Hybrid	IPMI, S7comm, Kamstrup, SNMP, HTTP(S), Ntopng, SSH	Server	IIoT
Du and Wang [DW20	2020	Hybrid	✓	Virtual	Not identified	Server	IIoT

Table 2.9: Summary of Tools, Implementation, and Attack Types of Honeypots and Honeynets for IIoT and CPS

Work	Tools	Ports	Attack Types	Data Analyzed	Study Length
CISCO [PF04]	N/A	Modbus/TCP (502), Telnet (23), HTTP (80), FTP (21)	N/A	N/A	N/A
Digital Bond [Pet06]	Sebek, Argus, Walleye, Snort IDS	Modbus/TCP (502), SNMP (161), Telnet (23), HTTP (80), FTP (21)	N/A	N/A	N/A
Conpot [RVH+20]	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 [ZQ17]	Flask framework, Wireshark	N/A	Traffic from 244 IP addresses from 34 countries	Types, sources, requests from IPs	43 days
DiPot [CLLL18]	N/A	N/A	Modbus and Kamstrup scan, Modbus over-length packets	Access sequences to protocols and their IPs	6 months
Camouflage [NMM+15]	Nmap, Kali Linux	N/A	Scanning	N/A	N/A
XPOT [LKAR16]	Nmap, nfqueue	N/A	N/A	S7comm handshakes and queries	1 month

Table 2.9: Summary of Tools, Implementation, and Attack Types of Honeypots and Honeynets for IIoT and CPS

Work	Tools	Ports	Attack Types	Data Analyzed	Study Length
HosTaGe [VSCM16]	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 [XCX18]	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. [DJB13]	N/A	N/A	N/A	Link latency, network traffic counting and connection limiting, background network traffic	N/A
Honeyd+ [WRD+15]	Nmap, Zenmap, Wget	EtherNet/IP (44818, 2222), HTTP (80)	Scanning	Fingerprints of Honeyd+ hosts, error rates and protocol data rates	N/A

Table 2.9: Summary of Tools, Implementation, and Attack Types of Honeypots and Honeynets for IIoT and CPS

Work	Tools	Ports	Attack Types	Data Analyzed	Study Length
Gallenstein [Gal17]	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. [ATUH18]	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. [HP14]	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. [KGM17]	OSSEC host-based IDS, PLCScan, Shodan, iptables	Modbus/TCP (502)	Port scans on Modbus and HTTP protocols	Conpot logs	2 weeks
Ding et al. [DZD18]	Nmap, snmpwalk, STEP7 software, PLCscan	S7comm (102)	Scanning	Scanning result	N/A

Table 2.9: Summary of Tools, Implementation, and Attack Types of Honeypots and Honeynets for IIoT and CPS

Work	Tools	Ports	Attack Types	Data Analyzed	Study Length
Bodenheim [Bod14]	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. [PB16]	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 [Han19]	SecurityOnion, iptables, Snort-Sam, 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

Table 2.9: Summary of Tools, Implementation, and Attack Types of Honeypots and Honeynets for IIoT and CPS

Work	Tools	Ports	Attack Types	Data Analyzed	Study Length
Hilt et al. [HMP+20]	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 [Ber12]	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 [Jar13]	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. [HFB15]	Step7, Szilu SSL, MiniWeb, iptables	S7comm (102), HTTP (80), HTTPS (443), SNMP (161)	Pings, port scans, SSH scans	Attack origins, logs	50 days

Table 2.9: Summary of Tools, Implementation, and Attack Types of Honeypots and Honeynets for IIoT and CPS

Work	Tools	Ports	Attack Types	Data Analyzed	Study Length
Serbanescu et al. [SOY15b]	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 [SCPM15]	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. [ALK19]	N/A	Modbus (502)	ARP poisoning	N/A	N/A
Belqruch et al. [BM19]	Kali Linux	SSH (22)	SSH brute force	Username-password combinations, password attempts	N/A
SHaPe [KG15]	libiec61850	N/A	N/A	TCP connection information (connection ID, source and destination IPs and ports), Dionaea logs	N/A

Table 2.9: Summary of Tools, Implementation, and Attack Types of Honeypots and Honeynets for IIoT and CPS

Work	Tools	Ports	Attack Types	Data Analyzed	Study Length
GridPot [RLB15]	ETSY Skyline project anomaly detection modules, GridLab-D, hpfeeds logging	N/A	IED switching attack	Physics impact of the attack	N/A
Scott [Sco14]	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
Mashima et al. [MCGT17]	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 [BJM+14]	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. [PSL+19	Wireshark, Tshark	N/A	N/A	N/A	N/A

Table 2.9: Summary of Tools, Implementation, and Attack Types of Honeypots and Honeynets for IIoT and CPS

Work	Tools	Ports	Attack Types	Data Analyzed	Study Length
Mashima et al. [MLC19]	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. [MCG+18]	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. [PK19]	Node-RED, Softing OPC UA Client, SQLite	N/A	Unauthorized access	Database entries	N/A
Wilhoit [W Wil13a]	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

Table 2.9: Summary of Tools, Implementation, and Attack Types of Honeypots and Honeynets for IIoT and CPS

Work	Tools	Ports	Attack Types	Data Analyzed	Study Length
Antonioli et al. [AAT16]	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 [	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 [WH15]	N/A	N/A	Reconnaissance, DDoS	Connection attempts, commands, attack origins	N/A
Zamiri et al. [ZGQA19]	Nmap	Veeder-Root ATG (10001)	N/A	N/A	N/A
HoneyPhy [LFR+16]	OpenDNP3 library, LabVIEW	N/A	N/A	Heating and cooling curve from both physical system and the process model	N/A

Table 2.9: Summary of Tools, Implementation, and Attack Types of Honeypots and Honeynets for IIoT and CPS

Work	Tools	Ports	Attack Types	Data Analyzed	Study Length
Du and Wang [DW20a]	SDN testbed	N/A	DDoS attacks, SYN Flood attack, FTP flow	Protocols, packets per port, packet characteristics	N/A
HoneyIo3 [AA18]	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

2.8.8 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 2.8 while ports that are exposed for such protocols in the honeypots are outlined in Table 2.9. 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 [HP14,SOY15b,ADM16,HLLL17, ABF18,PK19,PSL+19] 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 [AAT16, Sco14] and also it enables the remote configuration of industrial components such as PLCs [HFB15, VSCM16]. For these reasons, it is stated as the target of scanning activities performed by malicious entities [KR19]. SNMP on the other hand is used for monitoring and management purposes in industrial environments [HFB15, WRD+15].

2.8.9 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 [PF04] source code is still available. However, the last shared version was in 2015. Unfortunately, Digital Bond's

SCADA Honeynet [Pet06] 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 et al. [ZGQA19] 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 [RLB15] and SHaPe [KG15] are still available, but their last update was in 2015. The last update for GasPot [WH15] was in 2016, and honeypot-like testbed of Murillo et al. [MCG+18] was maintained in 2018.

2.8.10 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 indetify 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.

2.8.11 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, ma-

licious 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.

2.9 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.

2.9.1 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 [SCGM13, HFB15]. 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 [Sco14]. 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 [Bod14,HFB15] 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 [SOY15a, MLC19] and three IoT studies [ZZZ+19, DLL+19, VC19] 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 [HMP+20, WH15] and three IoT studies [G.W18, GTB+17, TAS+19] use public IP addresses which is the better option. In addition to this, Guarnizo [GTB+17] 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 2.4. 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 2.6 and Section 2.8. 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. [DLL+19] found that approximately 92.1% of malware-based attacks target multiple IoT device architectures and emphasized the need for a virtual IoT hon-

eypot 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. [DLL+19] 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 [GTB+17] 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 [Sco14] 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. [SHH+19] 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. [ZGQA19] 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 [Sco14]. 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 [GTB+17] 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 [Sco14] advised to be familiar with laws before deployment of honeypots since honeypots are interpreted as entrapment by jurisdictions in some places. Haney [Han19] 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 [SA15] 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. [DLL+19] 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.

2.9.2 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. How-

ever, 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 [NSRU20, NSBU20, NSRU19], 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 [OALS21], cryptocurrency mining attacks [NAB+21, TAS+21], and attacks to enterprise IoT systems [RBAU19, PBAU20, RBA+20, PBA+21], 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 [Fie20] industrial protocols. In addition, Enterprise IoT environments can employ various proprietary communication protocols that rely on security through obscurity [PBA+21]. 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 [BDC+21]. 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 [BAR+20, AUB18], 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 [Cen21], 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. [ANSB19].

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) [The20a]. 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 ([Wag11, PB14, PIB18, LXJ+17, SBH19, PBPC19, LVS20, WSK18]) employed ML for IoT honeypots/honeynets, we see that only one study [CLLL18] 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. [FPZ19] 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.

2.10 Conclusion

In this chapter, 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.

CHAPTER 3

S-POT: A SMART HONEYPOT FRAMEWORK WITH DYNAMIC RULE CONFIGURATION FOR SDN

3.1 Introduction

The digital transformation has been converting all aspects of life in recent years. The ever-growing number of Internet of Things (IoT) devices has exacerbated demand on traditional networks, making it increasingly complex to manage and scale. Enterprise networks in this regard are becoming increasingly heterogeneous where enterprise devices/services and IoT devices coexist. SDN emerged in response to these needs, transforming networking infrastructure, moving the brain from network devices to a centralized software controller [LSFF16]. SDN has been playing a crucial role in providing high performance networking around the globe during the Covid-19 pandemic [The20b]. More and more enterprise networks are moving to SDN [The20b], and the market size of SDN is expected to reach $59 billion by 2023 [Mar]. Simultaneously, cyber threats are evolving and increasing in quantity and impact. The cost of cybercrime is expected to reach $10.5 trillion in 2025 [Mor].

Although SDN has its benefits to ease security operations (e.g., isolation, segmentation, attack mitigation, etc.) in enterprise networks, it can also pose new threat vectors [The20b]. Intrusion Detection and Protection Systems (IDPS) have been widely used to protect SDN from attacks. However, IDPS solutions have fundamentally limited signature rules, which can leave SDN-powered enterprise networks vulnerable to new threats (zero day attacks). While some IDPSs offer anomaly-based detection, they can have false positives (FP) and false negatives (FN). Machine Learning (ML) has become a valuable tool to overcome these limitations. However, ML-based solutions can also struggle to discriminate benign traffic from

malicious traffic, and suffer from FN. In addition, although ML models can be retrained to cope with zero-day attacks, it may not be possible for every enterprise to have the necessary human resources that have the technical know-how to perform retraining. In parallel with IDPS and ML-based solutions, honeypots have been widely used to understand evolving cyber threats and develop effective defenses. Honeypots are mostly used for research purposes and they aim to lure attackers. In addition, they generally receive only malicious traffic [Wat07, FACU21]. Although there exist various studies on the use of IDPS, ML, and honeypots with research purposes, including various combinations of these tools implemented together in SDN [WW19, DW20b, KHT+17, AHM18, KA20, NZD+16, ELDJ19, SCPA19], to the best of our knowledge, *no study considered to benefit from honeypots for production purposes in improving the security of an SDN-based network.*

In this study, we propose S-Pot, an open-source smart honeypot framework that integrates the use of IDPS and ML for securing SDN-based enterprise networks through dynamic rules configuration. S-Pot benefits from honeypots that can simulate both enterprise services and IoT devices in gathering attack information. It employs an IDPS and ML classifiers to detect and learn from the attacks in honeypots. *Unlike research honeypots, it utilizes the obtained attack information from the honeypots for production purposes for the security of SDN networks, dynamically creates new rules for the attacks, and shares them with the SDN-based enterprise network.* Since honeypots (hence S-Pot) generally only receive malicious traffic, S-Pot can benefit from the identified malicious traffic of honeypots and greatly reduce FPs on the real network [Wat07]. Moreover, it can also detect new attacks that target the enterprise network by means of ML classifiers, and thus improve FP performance of the defense solutions in the enterprise network.

We implemented S-Pot and evaluated its performance in detecting attacks using various ML classifiers. Our evaluations show that S-Pot can detect attacks with 97% accuracy with J48 algorithm. In addition, we did another analysis in evaluating the performance of S-Pot in generating new attack signatures and dynamically configuring the rules of a realistic enterprise SDN testbed network. Our analysis demonstrates that, S-Pot can efficiently generate new rules and dynamically configure rules of the realistic testbed network to block attacks. Our evaluations show that S-Pot can improve the security of an enterprise SDN network compared to the case without S-Pot.

Contributions: The contributions of S-Pot are as follows:

- We propose a fully open-source smart honeypot framework that benefits from enterprise and IoT honeypots to dynamically generate new IDPS and SDN flow table rules for securing SDN-based hybrid enterprise networks.
- With S-Pot, we demonstrate how enterprises can benefit from honeypots for production purposes to secure their networks against both known and zero-day attacks.

Organization: This chapter is organized as follows. Section 3.2 identifies the related work. Section 3.3 provides background information on honeypots, SDN, and IDPS. Section 3.4 defines the problem scope and the threat model. Section 3.5 describes S-Pot. Section 3.6 details the implementation of S-Pot, data collection and processing, and evaluates the performance of it. Finally, Section 3.7 concludes the chapter.

3.2 Related Work

Several studies exist in the literature on securing SDN networks with the use of honeypots, IDPS, or ML tools. The use of SDN is presented by Wang and Wu [WW19] as a necessity for improving honeynet topology. Sultana et al. [SCPA19] surveyed ML methods using SDN and IDS. Valdovinos et al. [VPDCB21] and Swami et al. [SDR19] focused on SDN vulnerabilities and DDoS attack detection and mitigation for SDN, yet in this case, both ML and honeypot tools are not included. [MMS19], [XYH+19], and [ZLZ+19] focused on research with ML and SDN. Du and Wang [DW20b] aimed to be the first study to focus on DDoS attacks on honeypots-based SDN, particularly in Industrial IoT. Molina et al. [MBSC20] presented a high interaction IoT honeynet using SDN to protect the devices from DDoS botnet attacks. Kyung et al. [KHT+17] presented an SDN-based honeynet architecture to improve detection of fingerprinting attacks and avoid malware propagation with the application of SDN controller. Azab et al. [AHM18] introduced a smart gateway that isolates the northbound and southbound interfaces, and works as an IDS to protect the SDN controller from being compromised. In addition to the mentioned studies, various studies exist that use IDPSs in SDN. Sarica and Angin [KA20] introduced an SDN-dataset for intrusion detection in IoT. Other studies focused on attack detection in SDN using ML, such as Nanda et al. [NZD+16] and Elsayed et al. [ELDJ19], but do not use IDPS or provide dynamic rules configuration.

Differences from the existing work: Despite the wealth of research in SDN security, to the best of our knowledge, *none of the aforementioned studies have benefited from honeypots for production purposes for the security of SDN-based enterprise networks*. Also, unlike prior work, S-Pot integrates the use of honeypots, IDPS, and ML for securing an SDN network. In addition, S-Pot can detect new

attacks targeting SDN-based networks, create new rules for the detected attacks, and dynamically configure the SDN-based network.

3.3 Background

3.3.1 Software-Defined Networking

SDN allows for creating a more reliable, secure, and flexible network by adding the capacity of a centralized network controller to program and manage the network [PP17]. SDN reduces the traditional network's limitations by separating the network into three layers: application, control, and infrastructure layers. These are also referred to as application, control, and data planes. The application layer contains the network applications, such as IDPS and firewalls. The SDN controller software is the control layer, which handles the flow of all traffic in the network and enforces policies that can be established by the network administrators. The physical switches in the network compose the infrastructure layer. Communication between the layers is carried out through northbound and southbound Application Programming Interfaces (APIs). Northbound APIs enable communication between the application layer and control layer, whereas Southbound APIs enable communication between the control layer and the infrastructure layer. Although various protocols can be used with SDN to communicate between the controller and the network devices in the infrastructure layer, Open Flow Protocol (OFP) is the most widely used [AHM18]. SDN provides capabilities such as traffic analysis, dynamic rules updating, a global view of the network, and logically centralized network control [XYH+19]. While SDN provides control over the network that can be combined with other tools to implement strong security mechanisms, on its own, SDN

lacks adequate security features making it vulnerable [TQF+17]. Varadharajan et al. [VKTH19] categorize the threats in SDN, pointing to how the controller and networking devices such as switches, can be affected by multiple attacks, including denial-of-service (DoS) and distributed DoS (DDoS) attacks.

3.3.2 Honeypots and Honeynets

A honeypot is a decoy that is used to lure attackers and deceive them into thinking they have accessed a real system and to observe and learn from their actions by gathering data about their interaction with the honeypot [FACU21]. The gathered data can be used to develop countermeasures against attacks [FDFV18]. Honeypots can vary greatly in the level of interaction that they allow the attacker, ranging from low interaction honeypots that emulate particular services, to high interaction honeypots that emulate entire operating systems (OS). They can be used in a wide variety of applications, for research or production purposes, and can be implemented with physical or virtual resources. Two or more honeypots implemented on a system form a honeynet [FACU21]. Since there are generally no licit causes for which to interact with a honeypot, any traffic is usually malicious, which allows honeypots to greatly reduce the amount of FP in comparison with anomaly-based IDSs [Wat07].

3.3.3 Intrusion Detection and Protection Systems

Intrusion Detection and Intrusion Protection Systems identify potential threats in a network using signature-based and/or anomaly-based methods. For signature-based detection, the systems capture traffic in a network and compare the packets to a database of known threats. For anomaly-based detection, the systems identify deviations from normal traffic in the network. An IDS detects and alerts of malicious

traffic in a network but does not take any action. On the other hand, an IPS not only detects but also executes an action such as restricting access to attackers based on a set of rules. Snort IDPS [Cisb], maintained by Cisco Systems, is one of the most widely used open-source IDPS that can be used with diverse OSs.

3.4 Problem Scope and Threat Model

3.4.1 Problem Scope

We consider an SDN enterprise network containing both enterprise systems and IoT devices for smart buildings. The network has a signature-based IDPS that can detect the known attacks but fails to detect new attacks. To detect new attacks, the network can employ an ML-based anomaly detection system. However, although it may improve the security, it can struggle to discriminate benign traffic from the malicious, thus suffer from FP. In addition, although retraining the ML model is possible, not every enterprise may have the necessary human resources that have the technical know-how to perform this task. For these reasons, despite the security measures, the network can be vulnerable to zero-day attacks that can go undetected, and also suffer from the blocked benign traffic. To improve security of the SDN enterprise network, in this work, we propose S-Pot that benefits from enterprise and IoT honeypots, IDPS, and ML classifiers in detecting attacks targeting the network, and improves security of the network by generating new attack rules and dynamically configuring the rules. S-Pot is deployed in a demilitarized zone (DMZ) under the same domain as the real network. It publicly exposes honeypots to the Internet and aims to protect the real SDN network. S-Pot identifies key features from the gathered attack data and creates a new rule in the IDPS in the S-Pot

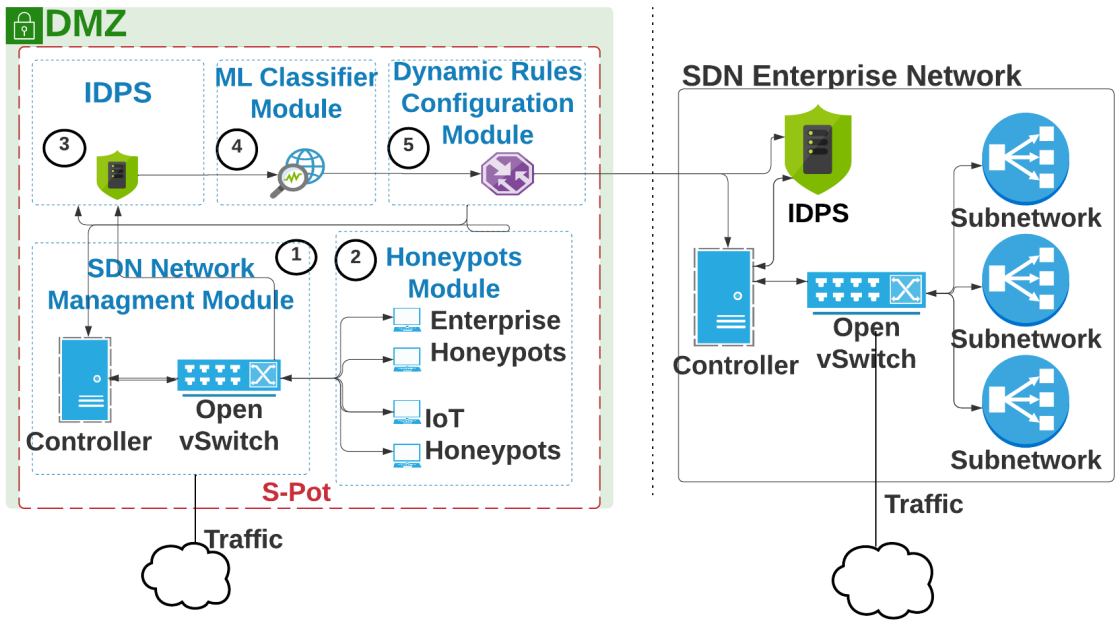

Figure 3.1: Architecture of S-Pot.

framework that instructs the controller of S-Pot to drop a packet if it exhibits the same features. This rule is passed over to the IDPS on the real enterprise network, which dynamically instructs the controller on the real network to do the same. This improves the security of the real SDN enterprise network by updating the controller rules to defend against zero-day attacks. When an attacker attempts to send a similar attack to a device in the real network, the controller drops the packet, protecting the real SDN network from the attack.

3.4.2 Threat Model

This work considers DoS, DDoS, scanning, trojan malware, and zero-day attacks targeting SDN-based enterprise networks. DoS, DDoS, and malware attacks have been identified amongst the most common types of cyber attacks [Cisc], while scanning is used in the reconnaissance phase of the attacks. In addition, the protocols

targeted by these attacks are commonly used in enterprise and IoT environments and have been identified among the most targeted protocols for attacks in the dark web [MS18].

3.5 S-Pot Framework

3.5.1 Overview

Figure 3.1 presents a general overview of the proposed S-Pot framework. S-Pot consists of the following components: (1) The *SDN Network Management module* that includes of a virtual switch for connecting devices to the network and a controller for centralized network monitoring, managing the flow of all traffic in the network. (2) The *Honeypot module* is comprised of virtual honeypots to lure attackers. (3) The *IDPS module* captures and logs data, detects potential threats, and distributed new rules. (4) The *ML Classifier module* is where data preprocessing, feature extraction, and classification are carried out. It also feeds the output to the Dynamic Rules Configuration module. (5) Finally, in the *Dynamic Rules Configuration module* the new rules are created, distributed, and integrated into the S-Pot network and the real SDN network. This is where the IDPS and controller rules are dynamically changed with the acquired knowledge from S-Pot. S-Pot utilizes SDN to better manage its components. While S-Pot could be implemented without an SDN platform, SDN provides greater features than the use of an IDPS on its own, to be able to dynamically implement new rules in a network based on the learned information. The modules of S-Pot are explained in detail in the following subsection.

3.5.2 S-Pot Modules

SDN Network Management Module

The controller is the brain of the network. The Open Daylight (ODL) controller Oxygen version [Thea] was chosen for the proposed framework. The ODL controller offers open-source flexibility, as well as open protocols, centralized network monitoring, and programmable control actions for S-Pot. At the application layer, ODL allows REST API calls to the controller to push down rules to the infrastructure layer. The Open vSwitch (OVS) [Theb] connects all the devices to the network. S-Pot is able to mirror all honeypot traffic to the IDPS using the OVS.

Honeypots Module

The production honeypots are the tools used to attract attackers to our network. S-Pot is composed of virtual machines (VM) running on Virtual Box. This framework allows for building high interaction virtual honeypots that can simulate both enterprise services and IoT devices with open-source and closed-source OSs and provides easy scalability.

IDPS Module

All traffic goes through the IDPS module. The IDPS captures the data packets, normalizes them, and checks the packets against its ruleset database. The processed information and results are sent to output plugins of the IDPS, where options for output can be selected.

ML Classifier Module

The ML Classifier module obtains the network traffic log of the IDPS module. The first step is *Data Cleaning*. Here, the data is cleaned before proceeding with feature selection. The next step is *Feature Selection*, where selected features are combined to identify the type of attacks. Finally, pre-trained multi-class ML *classifiers* examine the unknown traffic and detect the attacks. For S-Pot, the following classification algorithms were selected: Random Forest, SMO, BayesNet, and J48. These algorithms were selected to include a variety of approaches: BayesNet is probability based, J48 is decision tree based, Random-Forest combines multiple decision trees, and SMO is used in support vector machine (SVM) implementation. We would like to highlight that although we employ classical ML algorithms in this study, S-Pot does not have any limitation to employ more advanced ML algorithms such as deep learning. The ML Classifier outputs results to the Dynamic Rules Configuration Module.

Dynamic Rules Configuration Module

This module gathers the alert log data from the ML classifications from the S-Pot framework, parses out the data such as source IP, destination IP, Ethernet type, protocol, source and destination ports, type of service, etc., and generates IDPS and SDN rules using these features, which are passed to the SDN flow tables of both S-Pot and the real SDN network. The new rules are also passed to the IDPS on both the S-Pot network and the real SDN network. In the case of new attacks classified as unknown by the classifier, in addition to generating new rules accordingly, the module sends a notification with the log data to security administrators, so that they may review the logs. The security administrators can use this information to

develop new rules and manually change the IDPS and SDN rules, which can be imperative for defense against zero day attacks.

3.6 Performance Evaluation

In this section, we explain the implementation and performance evaluation of the S-Pot framework.

3.6.1 Implementation of S-Pot

SDN Network Management Module Implementation: Open Daylight controller and Open vSwitch are selected for the SDN Network Management module. The ODL controller manages the flow tables in the OVS, if a packet matches a rule in the OVS flow table, then the rule is applied (e.g. drop packets). Otherwise, the OVS checks with the controller for new rules before forwarding the packets. The OVS interconnects all the modules of our framework. All traffic associated with the honeypots is compared against the OVS rules tables to check if they comply with previously recorded rules and is also forwarded to the IDPS module to be processed and logged. If attack traffic is detected in the IDPS module, a request for a new rule is sent to the Dynamic Rules Configuration module to generate a new rule for the ODL controller via the RESTCONF API, and the OVS flow table is updated to apply the new rule (e.g. drop packets from source).

Honeypots Module Implementation: We employ two enterprise honeypots where each is implemented in a VM with different OSs and services to attract a greater variety of attack types. Debian 10 was selected due to its common use in enterprise servers, and Windows 10 was selected due to their wide use as clients in enterprise environments. Two IoT honeypots are also implemented. IoT Candy-

```
drop tcp any any -> 192.168.100.4 80 (flags: S; msg:"Flood Attack
Detected.";flow:to_server, established; classtype: attempted-dos;
detection_filter: track by_dst, count 18 , seconds 1 ; sid:
10000007; rev:1;)
```

Figure 3.2: An example Snort rule to block packets of SYN flood.

jar [LXJ+17] and Thingpot [Wan17] open-source honeypots were selected because they were both created for IoT, provide full-device emulation, and are scalable. Furthermore, ThingPot was created specifically for DDoS attacks for IoT, and IoT Candyjar applies machine learning to automatically learn the behaviors of IoT devices from the Internet. Telnet, Domain Name System (DNS), Dynamic Host Configuration Protocol (DHCP), SSH, HTTP(S), FTP, and SMB protocols were selected because they are commonly used in enterprise and IoT environments and have been identified among the most targeted protocols for attacks in the darkweb [MS18].

IDPS Module Implementation: For the IDPS module, we selected Snort [Cisb] because it is one of the most widely used open-source IDPSs. We used ODL controller and the OVS to connect the honeypots and forward all the traffic to the Snort IDPS. Figure 3.2 demonstrates an example of a Snort IDPS rule. This rule blocks packets from any IP address using any port that is targeting the same destination (e.g., 192.168.100.4) with more than 18 SYN packets within a second. This rule could vary based on the ML outputs.

ML Classifier Module Implementation: We utilized Weka [Gro] for use in our ML-Classifier module. Weka is an open-source tool for ML applications that provides various algorithms. Details with the data collection and training are given in Section 3.6.2.

Dynamic Rules Configuration Module Implementation: For this module, we used a script to generate the IDPS and SDN rules and to pass newly created rules from it. The script gathers the alert log data from the ML Classifier module, parses

out data from the alert (source and destination IPs, Ethernet type, protocol, source and destination ports, type of service, etc.), and generates rules using these features, which are added to Snort and the SDN flow tables of both the S-Pot framework and the emulated SDN network. The controller is queried by the OVS every 30 seconds for new registered rules on both networks as well. One advantage when generating new dynamic rules for the SDN controller is that the filter values are similar to the ones used for Snort. This facilitates creating filters for the rule. It is also possible to combine multiple conditions. Rules are pushed to the SDN controller in XML or JSON format. Listing 3.1 demonstrates an example SDN rule generated by S-Pot to be passed to the SDN controller to drop packets that match the source IP address of x.x.x.x/30 and port 22222, destination IP address of y.y.y.y/24 and port 8080, Ethernet source and destination addresses.

Listing 3.1: A sample SDN rule generated by S-Pot for the enterprise SDN controller to drop packets from x.x.x.x/30 to y.y.y.y/24

```
{
  "flow2": [
      {
          "id": "2",
          "table_id": 2,
          "flow-name":"flow2",
          "strict": 2,
          "match": {
              "ipv4-source": "x.x.x.x/30",
              "ipv4-destination": "y.y.y.y/24",
              "ip-match": {
                  "ip-dscp": 2,
                  "ip-protocol": 6,
                  "ip-ecn": 2},
              "in-port": "0",
              "tcp-source-port": 22222,
              "tcp-destination-port": 8080,
              "ethernet-match": {
                  "ethernet-type": {
                      "type": 2048},
                  "ethernet-source": {
                      "address": "00:02:b2:f2:c3:05"},
                  "ethernet-destination": {
                      "address": "00:02:c4:a6:b1:03"}}},
          "cookie": 3,
          "instructions": {
              "instruction": [
                  {
                      "order": 0,
                      "apply-actions": {
                          "action": [
                              {
                                  "order": 0,
                                  "drop-action": {}
                              }]}}}]}}
```

3.6.2 Data Collection and Processing

To test the performance of S-Pot in classifying attacks, we prepared a dataset with TCP SYN Flood DoS, PUSH and ACK Flood DoS, UDP Flood DDoS, scanning, trojan malware, and unknown attacks. In this study, unknown attacks represent zero day attacks and are all the logged packages which did not fit into the other attack classifications that the classifier came across during the training. These were generated using pcaps. Next, we used libpcap [lib] to capture network traffic for eight consecutive hours and performed variations of the selected attacks. For this, we used hping3, LOIC, Nmap, and pcaps, respectively. We stored the captured network traffic into JSON files and converted them into CSV format for use in Weka. We chose the following features based on the values of each feature, the statistical relationship, and the redundancy of the features to produce a more accurate model [NG20]: *inter-arrival time (IAT), packet direction (dir), destination address and port (dst_ap), destination MAC address of Ethernet (eth_dst), source MAC address of Ethernet (eth_src), packet length (pkt_len), protocol type (proto), source address and port (src_ap), capture time (timestamp), transmission control protocol flags (tcp_flags), transmission control protocol window (tcp_win)*, and *time to live (ttl).*

In the *Data Cleaning* step, the collected data was cleaned by extracting only the selected features and forming the feature vectors. We eliminated the features that are not relevant to the classification of the attacks (e.g., rule revision, generator id, eth_type, or where columns have missing information (e.g., class, service). In the *Feature Selection* step, we combined the selected features such as seconds, the source IP address, and the packet length or seconds and packet number to identify the type of attack. Seconds, which represent the inter-arrival time between each continuous packet, have been determined to be helpful to identify different types of attacks in

Table 3.1: Performance Evaluation Results of S-Pot.

Model	TPR	FPR	Prec.	Recall	F1	ROC Area	Class
RF	0.971	0.02	0.965	0.971	0.968	0.995	Scanning
	0.945	0.016	0.925	0.945	0.935	0.975	DoS2
	0.947	0.001	0.973	0.947	0.96	0.976	Trojan
	0.95	0.011	0.963	0.95	0.957	0.996	DDoS
	0.996	0.004	0.979	0.996	0.987	0.998	DoS
	0.881	0.003	0.95	0.881	0.914	0.986	Unknown
	0.958	0.013	0.959	0.958	0.958	0.991	Avg.
J48	0.988	0.014	0.975	0.988	0.981	0.986	Scanning
	0.931	0.007	0.968	0.931	0.949	0.965	DoS2
	0.951	0	1	0.951	0.975	0.965	Trojan
	0.969	0.012	0.961	0.969	0.965	0.987	DDoS
	0.991	0.005	0.97	0.991	0.981	0.994	DoS
	0.936	0.003	0.962	0.936	0.949	0.987	Unknown
	0.97	0.01	0.97	0.97	0.969	0.983	Avg.
BayesNet	0.942	0.011	0.979	0.942	0.96	0.994	Scanning
	0.927	0.017	0.921	0.927	0.924	0.986	DoS2
	0.976	0.004	0.87	0.976	0.92	0.976	Trojan
	0.935	0.018	0.94	0.935	0.937	0.99	DDoS
	0.996	0.004	0.974	0.996	0.985	0.997	DoS
	0.927	0.013	0.835	0.927	0.878	0.992	Unknown
	0.945	0.013	0.947	0.945	0.945	0.991	Avg.
SMO	0.968	0.022	0.961	0.968	0.964	0.985	Scanning
	0.934	0.017	0.922	0.934	0.928	0.96	DoS2
	0.953	0	1	0.953	0.976	0.979	Trojan
	0.943	0.013	0.958	0.943	0.95	0.978	DDoS
	0.996	0.004	0.974	0.996	0.985	0.996	DoS
	0.899	0.004	0.942	0.899	0.92	0.995	Unknown
	0.955	0.015	0.955	0.955	0.955	0.981	Avg.

networks [OCD16]. We began by labeling the known network traffic generated each as a different class. Class categories are presented as DoS, DoS2, DDoS, scanning, trojan malware, and unknown, where DoS represents TCP SYN Flood DoS attacks, DoS2 represents PUSH and ACK Flood DoS attacks, and DDoS represents UDP Flood DDoS attacks. DoS attacks were from a real static IP address flooding the target, while DoS2 was a spoofed IP address to simulate a trusted source. To identify these attacks, the selected features were used in *Classifiers* component of the ML Classifier module to build ML models using Random Forest, SMO, BayesNet, and J48 algorithms. Feature combinations were selected for the identification of each attack type. DoS, DoS2, and DDoS attacks were identified through feature combinations of identical IAT, src_ap, dir, and packet length, consecutive values of timestamp, pktnum, and src_ap, and identical values of pktlen, and tcp_win. When the traffic is benign, it was observed that these features change uniformly. Also, while maintaining the same dest_ap, the src_ap port number increases by one when it is under DoS attacks. For the scanning class, dst_ap, TTL, eth_src, and tcp_flags were considered to detect active scanning. For the Trojan malware, Snort detected the packets as malicious based on direction flowing to the server, dest_ap, pkt_len repeated in groups of three, and proto targeting HTTP. Finally, *Classifier* was trained using a multi-class classification of labeled signatures from known attack types by applying a supervised approach.

3.6.3 S-Pot Classification Accuracy

In this section, we evaluate the accuracy of S-Pot in detecting the attacks. Table 3.1 presents the results considering several accuracy metrics such as True Positive Rate (TPR), False Positive Rate (FPR), Precision, Recall, F1, and ROC Area. The

BayesNet model showed the best performance for identifying scanning, achieving 97.9% precision. For identifying trojan malware attacks J48 and SMO showed the best performance with 100% precision. The metrics proved the most accurate in regards to DoS attacks, with RF achieving 97.9% precision. In regards to DDoS attacks, RF demonstrated the best performance with 96.3% precision, whereas J48 proved to have the highest precision in regards to DoS2 attacks with 96.8% as well as unknown attacks with 96.2%. J48 provided the best performance overall, with the highest average true-positive rate of 97%, lowest average false-positive rate of 1%, and highest average precision of 97%. The obtained results show that the J48 model is the most suitable classifier to be used in the ML Classifier module of S-Pot. Figure 3.3 displays the confusion matrix obtained as a result of the classification of different attack types and the J48 algorithm.

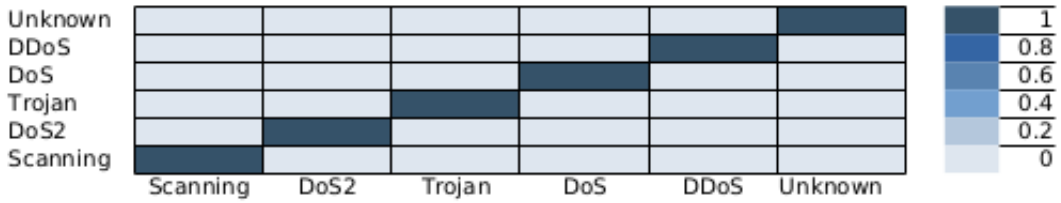

Figure 3.3: Confusion Matrix generated from the result of the classification of different types of attacks using J48 algorithm.

3.6.4 Performance Evaluation of the SDN Enterprise Network with S-Pot vs. without S-Pot

To compare the security of an enterprise network with and without S-Pot, we built an enterprise SDN testbed network that includes the components tabulated in Table 3.2. In this evaluation, we simulated TCP SYN Flood DoS, PUSH and ACK

Flood DoS, UDP Flood DDoS, scanning, and trojan malware attacks using the aforementioned tools and applied each type of attack 10 times.

Table 3.2: Components of the SDN Testbed Network.

Component	OS	Services
Enterprise Honeypot 1	Debian 10	SSH, HTTP/HTTPS, DNS
Enterprise Honeypot 2	Windows 10	SMB, FTP, SSH, DHCP
IoT Honeypot 1	Full device emulation	SSH, Telnet
IoT Honeypot 2	Full device emulation	SSH, Telnet
Snort IDPS	Ubuntu 20	IDPS
ODL	Ubuntu 18	Controller
OVS	Ubuntu 18	Virtual Switch

We began by identifying the number of attacks that were effectively blocked by the use of Snort IDPS alone in the SDN network, without S-Pot. Our analysis showed that the Snort IDPS detected and blocked the trojan malware, and flagged the TCP SYN Flood DoS, PUSH and ACK Flood DoS attacks. However, it did not block the TCP SYN Flood, and PUSH and ACK Flood attacks. A rule is required to be added to Snort to block these attacks. On the other hand, Snort did not detect the UDP Flood DDoS or the scanning attacks.

As the second step of our evaluation, we applied the same attacks (i.e., UDP Flood DDoS, TCP SYN Flood DoS, PUSH and ACK Flood DoS, and scanning) which were not detected by the IDPS of the testbed network on different honeypots of S-Pot, which in turn were analyzed by the IDPS module of S-Pot. Once again, the IDPS module of S-Pot was able to detect and block only the trojan malware. The TCP SYN Flood DoS and PUSH and ACK Flood DoS attacks were only detected

but not blocked, and the UDP Flood DDoS and scanning attacks were not detected. For the attacks that were not detected by the IDPS module, the ML Classifier module of S-Pot obtained the traffic logs from the IDPS, and detected the attacks with 100% accuracy. Following the detection process, new signatures for the detected attacks were generated by the Dynamic Rules Configuration module and fed to the IDPS of S-Pot successfully. In addition, the new rules were sent to the IDPS of the SDN testbed network. Once this process was completed, as the last step of the evaluation, we applied the same attacks against the hosts on the testbed network again. At this point, the IDPS of the network was able to detect all of the attacks, and the packets were dropped. Our evaluation with simulated attack instances shows that S-Pot can detect new attacks and generate new rules to block attacks with 40% improvement over legacy systems without S-Pot, based on the applied attacks, thus effectively improving the security of an SDN network. It is important to note that while our analysis only focused on S-Pot effectiveness against these five types of attacks, this is expandable to all kinds of attacks as new attack data is captured by the honeypot module and the ML Classifier module is also extendable.

3.7 Conclusion

In this chapter, we proposed S-Pot, a novel smart honeypot framework that aims to improve security of SDN networks. S-Pot benefits from honeypots that can simulate both enterprise services and IoT devices in gathering attack information and employs an IDPS and ML classifiers to detect and learn from the attacks in honeypots. Unlike research honeypots, it utilizes the obtained attack information from the honeypots for production purposes for the security of SDN-based networks, dynamically creates new rules in real-time for the attacks, and shares them with the

SDN-based network. We implemented S-Pot and evaluated the attack detection performance of it with respect to various ML algorithms. Our results showed that J48 algorithm provides the best accuracy for S-Pot, with 97% precision. We also created a realistic enterprise SDN testbed network and tested the security of it with and without S-Pot. Our evaluations demonstrated that S-Pot can improve the security of the SDN network and can effectively add a newly created rule to the flow tables on both the S-Pot and the realistic enterprise testbed network. These new rules can effectively block attacks based on the acquired knowledge from our S-Pot framework, improving network security.

CHAPTER 4

FORENSIC ANALYSIS OF CRYPTOJACKING IN HOST-BASED DOCKER CONTAINERS USING HONEYPOTS

4.1 Introduction

Blockchain-based cryptocurrencies have transformed financial transactions. There are currently over 18,000 types of cryptocurrencies, and the cryptocurrency market value was more than 2 trillion as of April 2022 [Coi], having reached an all-time high of 3 trillion in November 2021 [Lau21]. Cryptocurrency is particular in that investors can either purchase cryptocurrency, or carry out mining operations to generate new coins. The lucrative potential of cryptomining has led to the exploitation of this methodology through cryptojacking, by which a cybercriminal performs unauthorized cryptocurrency mining using the victim's computational power and resources for their own financial gain. This is also something which insiders can seek to make a profit from, with employees that have legitimate access to computational resources abusing their privileges [Mil21].

While cybercriminals perform cryptojacking exploiting a wide range of devices (e.g., personal computers, IoT devices [TAU22]), the researchers spotted an increasing trend towards targeting devices with greater processing power through host-based cryptojacking [TAU$^+$21, IBM, C.P]. For host-based cryptojacking, since the client does not go to the attacker as they do in the case of in-browser cryptojacking, the attackers need to find the way to deploy and install the malicious mining script on the victim's device. The greater the computational power, the greater and faster the possible profit yield from mining. This usually also means a greater number of connections and processes, which expands the attack surface [TAU$^+$21]. Some examples of targeted powerful devices are servers [Fox21], enterprise cloud in-

frastructures [Hac20, Dom20], and Docker engines [Sas21] to maximize profit using host-based cryptojacking.

Docker engines are widely deployed in today's enterprise environment, from universities to big companies worldwide providing services to their employees, making host-based Docker containers a prime target for cryptojacking. In fact, Docker has become one of the top three most popular development platforms [Sta20]. Despite the trend toward host-based cryptojacking and the wide use of Docker, the current literature has not explored the cryptojacking malware targeting Docker containers and its detection methods.

At the same time, honeypots and honeynets have become very important tools for security researchers in the past years to observe and analyze how attackers perform attacks and find out their behaviors [FDFV18]. Honeypots and honeynets can provide actionable intelligence on the attackers to continuously learn from attacks and develop effective defense mechanisms [FDFV18]. A honeypot is a decoy that is used to lure attackers and deceive them into thinking they have accessed a real system and to observe and learn from their actions by gathering data about their interaction with the honeypot [FACU21]. Research honeypots have been a very active field of research during the last decade. However, there have been few research studies on the use of honeypot data for developing security solutions for production purposes [FACU21], and to the best of our knowledge, no research studies have considered the use of production honeypots for detection and mitigation of cryptojacking. Given the current global impact and scope of cryptocurrency and cryptojacking, this is an important research area to consider.

In this study, we conduct a forensic analysis of cryptojacking in host-based Docker containers using honeypots to collect host resource and network data. We identify that monitoring the resource usage of a Docker host can reveal potentially

unauthorized cryptomining. For example, increased temperature of a Docker host, when combined with increased CPU and RAM loads which can be attributed to a particular container and it can be a strong indicator of an intrusion or possible cryptomining. When further combined with network data that includes the identification of Stratum protocol, keywords in DNS requests, and the use of the container's ephemeral ports, these are notable indicators to alert of cryptojacking. We identify countermeasures to secure Docker containers to avoid the such attacks and vulnerabilities that can lead to cryptojacking. We also propose an approach for monitoring host-based Docker containers and alerting system administrators when the indicators point to a cryptojacking attack.

Contributions: The contributions are as follows:

- We identify key indicators for the detection of cryptomining in a host-based Docker engine through forensic analysis using honeypots. Our analysis shows that both host resource usage and network traffic-based features can be used to detect unauthorized cryptomining.
- We also propose an approach for monitoring host-based Docker containers for the detection of cryptojacking.
- We present countermeasures for securing Docker containers to to prevent cryptojacking attacks.

Organization: This chapter is organized as follows: Section 4.2 provides related work. Section 4.3 provides background information. Section 4.4 describes the problem scope and threat model. Section 4.5 describes our methodology. Section 4.6 provides a forensic analysis of the collected data. Section 4.7 provides measures for securing host-based Docker containers. Section 4.8 presents an approach to

monitor host-based Docker containers. Section 4.9 proposes conclusions and future work.

4.2 Related Work

There are currently very limited studies on detection mechanisms for host-based cryptojacking [ASKS19, MPB+20, Tan20, GPLC19, LEBM20]. A few recent studies present detection mechanisms which can be used for both browser and host-based cryptojacking. The studies in [MPB+20, Tan20] focus on CPU as a key factor for detection while study in [MPB+20] uses Hardware Performance Counters (HPC). [ASKS19] focuses on providing a lightweight model for detecting cryptojacking malware in low power devices such as Internet of Things (IoT) and Cyber Physical System (CPS) devices. [GPLC19] focuses on a combination of hardware events, software events, and hardware cache events. However, only two studies [DHD+20, CROD21] focus solely on host-based cryptojacking. These two studies focus on the use of opcode sequences, system call invocations, and network traffic. However, none of the aforementioned studies focused on the container implementations. There is only one study [KKH+21] in the current literature that specifically targets detecting cryptomining in containers. This study concentrates on monitoring Linux-kernel system calls with a machine learning based system of anomalous pods in a Kubernetes cluster. Finally, [SK18] and [BG21] focus on docker-based honeypot systems; however, they do not focus on cryptojacking threats.

Differences from the existing work: To the best of our knowledge, this is the first study investigating host-based cryptojacking targeting Docker containers focusing on both device resource usage and network traffic features. Moreover, it is also the first study utilizing a honeypot system for the cryptojacking detection.

4.3 Background

4.3.1 Host-based Cryptojacking

Cryptojacking refers to the act of performing unauthorized cryptocurrency mining using a victim's computational power and resources for financial gain. While in-browser cryptojacking is carried out through a web script, host-based crytojacking is carried out on the host system. To accomplish this, cybercriminals fraudulently embed malware in legitimate third-party applications [CV20], target identified Common Vulnerabilities and Exposures (CVEs) as in the case of Mikrotik routers [Tre18] or CVE-2019-2725 for the delivery of Monero cryptominer [VTG19], exploit poor security [Qui21], install the malicious files through a web page, pop-up window, or email attachment [SRG18], or use social engineering techniques [McD21]. Insiders can also seek to make a profit mining using the resources of the company abusing their privileges [Mil21].

4.3.2 Stratum Protocol

Stratum is an application layer protocol that was created for pooled mining, in which miners pool their computing resources together, each fulfilling different tasks and reporting it to the mining pool server. When the server sends a valid output to the cryptocurrency network, it receives the reward, takes a commission and distributes the remaining portion according to the tasks each carried out [RL21].

4.3.3 Docker Containers

Docker is an open source platform for running, developing and distributing applications [BRBA17]. It is lightweight, fast, highly portable, and allows for dynamic

management of workloads [Inca]. The Docker platform is container-based, with containers holding everything that is needed to run an application. This is particularly useful for workflow, as this allows for sharing containers while reliably ensuring that everyone receives the same container without alteration, in a way that can be easily worked on [Inca].

4.3.4 Honeypots and Honeynets

A honeypot is a decoy that can be used as a first line of defense to lure attackers and deceive them into thinking they have accessed a real system. This allows the owner of the attacked system to observe and learn from attacker actions by gathering data about their interaction with the honeypot [FACU21]. The gathered data can be used to develop countermeasures against attacks [FDFV18]. Honeypots can vary greatly in the level of interaction that they allow the attacker, ranging from low interaction honeypots that emulate particular services, to high interaction honeypots that emulate entire operating systems (OS). They can be used in a wide variety of applications, for research or production purposes, and can be implemented with physical or virtual resources. Two or more honeypots implemented on a system form a honeynet or a honeypot system [FACU21]. Since there are generally no licit causes for which to interact with a honeypot, any traffic is usually malicious, which allows honeypots to greatly reduce the amount of false positives in comparison with anomaly-based Intrusion Detection Systems [Wat07].

4.4 Problem Scope and Threat Model

In this section, we present the problem scope as well as the threat model considered for our study.

4.4.1 Problem Scope

Cryptojacking is a stealth attack which can go unnoticed for a long time. However, it can produce significant losses for victims, reducing computational efficiency of the host's system and generating operational costs. We consider a host-based Docker engine with diverse Docker containers used for enterprise purposes. Resources can be vulnerable to cryptojacking, whether it is an external cybercriminal which finds its way into the system, or an employee who abuses their access privileges to the computational power of their employer's system and uses it for their personal financial gain at their employer's expense. It is worth noting that developers tend to be targets of attack, as they have high access and even administrative privileges to carry out their work. However, while they are very knowledgeable technically, that does not always translate into high security awareness, as their focus is on getting their job done. Also, sometimes as part of their work, they need to carry out operations that require turning off security controls to avoid false positives. These factors can increase the vulnerability of containers [Che].

4.4.2 Threat Model

In this work, we consider cryptojacking attacks targeting host-based docker containers, which are commonly used in enterprise environments. Docker has become one of the top three most popular development platforms [Sta20] and cryptojacking attacks have been identified among the most common types of cyberattacks on Docker containers [Sas21]. Furthermore, there is an increasing trend of cybercriminals targeting devices with greater processing power through host-based cryptojacking [TAU+21, IBM, C.P], in order to produce greater financial gain in a shorter time

frame. We consider both outsider and insider threats, as insiders can also seek to make a profit mining using the resources of the company [Mil21].

4.5 Methodology

In this section, we describe the honeypot system deployment, and host resources and network data collection processes.

4.5.1 Honeypot System Deployment

The host is an Intel Core i7-10700K Processor with 16 Cores and 32 GB RAM. We set up a high-interaction honeypot system isolated in a demilitarized zone (DMZ) with ten Docker containers: four of these used Redis container images, four used Nginx container images, and two used Ubuntu container images. Of those using Ubuntu images, one was used to run Prometheus and one was used to run Cadvisor. One of each of the Redis and Nginx containers was used to understand the baseline behavior, and the other three of each were tested with three scripts mining Monero on Cudominer, Minexmr, and Xmrpool.eu mining pools. Redis and Nginx were selected because they have been identified as the two most widely used container images [Dat21]. We selected to use Monero, as it is the most popular coin used for cryptojacking due to its strong anonymity and its efficient mining properties on CPUs [RL21]. The containers were allowed access to the 16 cores with unlimited swap and memory usage. On the baseline containers, a script that simulates 11,500 requests per second was run on the Nginx container and a Redis-benchmark utility that simulates 1000 user requests per second was run on the Redis container. Figure 4.1 presents a general overview of the honeypot system.

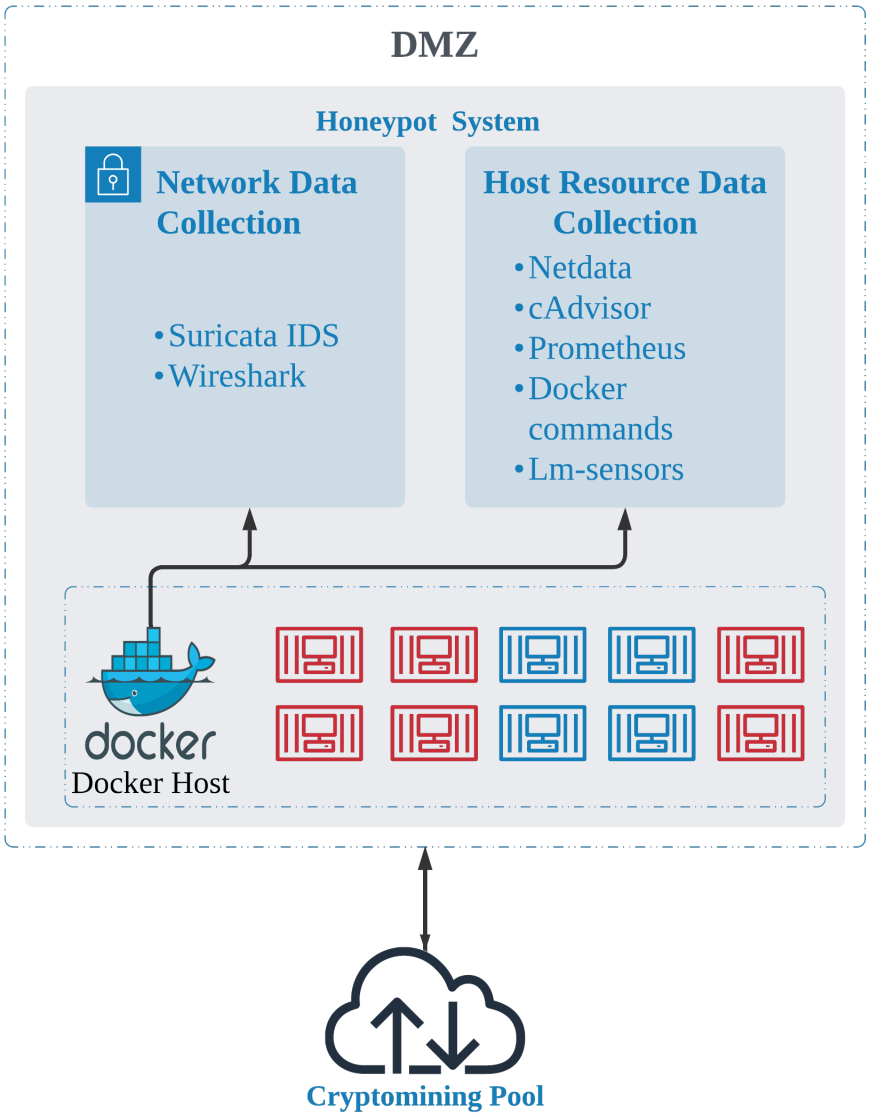

Figure 4.1: Honeypot system overview.

We used version 3.2 of the docker-compose tool [Incb] to define and execute the containers in our honeypots. The containers are run with a simple command docker-compose up, and if needed they all can be down with a single command docker-compose down. Compose is configured in a predefined YAML file, where we input the services that will be running in the docker host, and the configuration they will be using such as image, container name, ports, network, volumes, commands, etc.

In the docker-compose command, other files such as Dockerfile and prometheus.yml are invoked to participate in the set up process.

Mining activity was collected for a total of eight hours. The averages of the collected data for each container is presented in Table 4.1. We stopped the mining activities and the data collection process from time to time to cool the host device since the CPU temperature passed the safe threshold of 82°Celsius.

4.5.2 Host Resource Data Collection

We run the open-source tool cAdvisor (Container Advisor) [Goo] in a separate container using docker-compose to collect metrics on the usage and performance of the running containers and the Docker host. We then allow the Prometheus container to dig into the cAdvisor metrics and filter them using the Prometheus [Pro] expression browser. For each container and for the host, the cAdvisor metrics provide the isolation parameters, CPU usage including usage per Core and top container demand, Memory consumption, network metrics such as throughput and errors, filesystem usage, and the host /system.slice. cAdvisor can be accessed from the docker host through their web UI at http://localhost:8080. Although cAdvisor provides this interface, the Prometheus expression increases the metrics analysis. Prometheus [Pro] is an open-source tool that scrapes metrics collected by the cAdvisor container. It's part of the docker-compose deployment. Prometheus allows filtering of the metrics by providing expressions such as *container_cpu_usage_seconds_total, container_tasks_state, container_memory_usage_bytes, container_fs_writes_total,* etc, that will scrape cAdvisor metrics approximately every three seconds for a duration of 122ms approximately using the labels *instance="cadvisor:8080" and job="cadvisor"*. Prometheus can be accessed from the docker host through their web interface at

http://localhost:9090. Netdata [Incd] is a web application that provides key metrics, useful charts, and alarms to monitor the system and its components and graphically visualize the metrics in real time. Lm-sensors [lms] was used to monitor the CPU temperature of the Docker host, and nvme-cli [Exp] was used to monitor Nvme ssd temperature.

4.5.3 Network Data Collection

Wireshark version 3.2.3 [Wir] open source network protocol analyzer was used to log the network data. Wireshark allows to filter by protocol, and identify the cryptomining protocols such as Stratum. It also collects the exchanges between the cryptojacked container and the pools, timestamp, and network information such as IP address, packet length, protocol, and ports. However, it does not allow data parsing. For this purpose, Suricata version 6.0.5 [Fou] open source intrusion detection and intrusion prevention system (IDPS) was also used to log all network traffic, including timestamp, event type, IP addresses, packet length, source and destination port, DNS information, etc.

4.6 Data Analysis

In this section, we analyze the collected data and provide notable findings for the detection of unauthorized cryptomining.

4.6.1 Host Resource Data Analysis

The data collected from the baseline container running Redis for 1000 user requests demonstrated an average of 16% CPU consumption, 5% RAM, 7Mb disk I/O, and

Table 4.1: Host Resource Data Collection

Tests	Crypto	CPU	CPU Peaks	RAM	Disk I/O	iowait	Hash rate	Cryptojacking	CPU Temp.	Nvme Temp
Redis	N/A	16.00%	N/A	5.00%	7Mb	10.00%	N/A	Baseline	42.25°C	36.9°C
Redis – Test 1	Monero	65.00%	75.00%	28.50%	7Mb	13.00%	5.70Kb/s	Script1	69°C	42°C
Redis – Test 2	Monero	74.00%	90.00%	26.30%	7Mb	14.00%	4.1 kh/s	Script2	71°C	42°C
Redis – Test 3	Monero	75.00%	92.00%	27.40%	7Mb	17.00%	5.2 kh/s	Script3	70°C	42°C
Nginx	N/A	15.00%	N/A	1.00%	6Mb	10.00%	N/A	Baseline	42°C	37°C
Nginx – Test 1	Monero	70.00%	85.00%	30.50%	6Mb	12.00%	4.6 kh/s	Script1	69.5°C	41°C
Nginx – Test 2	Monero	82.00%	97.00%	32.00%	6Mb	16.00%	4.8 kh/s	Script2	69°C	40°C
Nginx – Test 3	Monero	75.00%	97.7%	27.50%	6Mb	15.00%	5.2 kh/s	Script3	72°C	41°C

10% iowait. We also observed that the temperature on average of the CPU cores and the nvme were 42.25°Celsius and 36.9°Celsius respectively. In our comparison tests running mining scripts, there was a dramatic increase of more than 4 times the average container CPU compared to the baseline, with a peak approximately every 2 minutes where the container CPU consumption increases to more than 5 times the baseline average for a period of approximately one minute. During our observations, CPU never goes lower than 55% in the cryptojacked containers, which is more than 3 times the average CPU consumption of the baseline. The RAM consumption of the containers also goes up, positioning it at an average of 27.4% which is more than 5 times the baseline average. We did not observe distinctive rate changes in the Disk I/O metrics while mining, keeping an average of 7Mbs. However, there was an increase in the average overall iowait to 14.66%. The temperature on average of the CPU cores was 70°Celsius, which is a significant increase of 65.68%, and that of the nvme was 42°Celsius.

Next, we observe the data for the baseline container running Nginx. Here, the simulated 11,500 requests per second demonstrated an average of 15% CPU consumption, 1% RAM, 6Mb disk I/O, and 10% iowait. The temperature on average of the CPU cores and the nvme were 42°Celsius and 37°Celsius, respectively. During mining activity, there was a dramatic increase of more than 5 times the average CPU in comparison to the baseline, and peaks can be observed approximately every

2 minutes where the CPU increases to more than 6 times the baseline average for approximately one minute. 60% was the lowest CPU consumption measured during mining, which is more than three times the average CPU consumption of the baseline. RAM consumption increased to 30%. This is 30 times the baseline average. As with the Redis containers, Disk I/O metrics did not see a significant change. The average overall iowait increased to 14.33%. The temperature on average of the CPU cores saw a significant increase of 66.67% to 70°Celsius, and that of the nvme rose to 41.66°Celsius.

While increased CPU and RAM loads, as well as increased temperature only provide hints of a possible problem such as hardware failure in a regular system, when this is identified in combination and on a Docker host where increased CPU and RAM load can be identified as coming from a particular container, this is a much clearer indication that the issue is likely an intrusion or insider mining.

4.6.2 Network Data Analysis

In all our tests, we first identify the TCP 3-way handshake between the container and the cryptomining pool server. Once communication between the cryptojacked container and the server is established and validated, the data flow begins. The server assigns the container the task to work, and once this task is solved, the client will send the results to the pool server.

We analyzed the captured network traffic from Wireshark and Suricata to find indicators or patterns that could be identified as possible cryptojacking activity. Cudominer and Minexmr communications use TCP, while Xmrpool.eu was encrypted using TLS.

The use of Stratum protocol, which uses plain TCP socket and JSON-RPC messages, and presence of certain keywords such as monero, pool, coin, xmr, mine, hash, and cudo identified in DNS requests under domain name are the two most direct ways to detect cryptojacking with network data. These can be seen in Figure 4.4. However, the hackers could apply obfuscation techniques such as use of a proxy to go unnoticed. When encrypted with Transport Layer Security (TLS), the packets by which the container receives jobs, sends the solutions, and the pool acknowledges the solution as "Application Data". Notably, Suricata flagged the Stratum protocol as a potential corporate privacy violation, specifically an "ET POLICY cryptocurrency miner checkin". This can be seen in Figure 4.2.

While use of the container's ephemeral ports by the mining pools is not guaranteed, this could be used as a notable indicator to alert that further review of the network traffic is needed.


```
{"timestamp":"2022-05-02T02:49:38.433205-0400","flow_id":685929882857629,"in_i
face":"wlo1","event_type":"alert","src_ip":"192.168.1.114","src_port":55270,"d
est_ip":"147.135.37.204","dest_port":30010,"proto":"TCP","community_id":"1:ep2
ZnRQqSWZIZK+drAcY1RylmFQ=","alert":{"action":"allowed","gid":1,"signature_id":
2024792,"rev":4,"signature":"ET POLICY Cryptocurrency Miner Checkin","category
":"Potential Corporate Privacy Violation","severity":1,"metadata":{"affected_p
roduct":["Windows_XP_Vista_7_8_10_Server_32_64_Bit"],"attack_target":["Client_
Endpoint"],"created_at":["2017_10_02"],"deployment":["Perimeter"],"former_cate
gory":["POLICY"],"signature_severity":["Minor"],"updated_at":["2018_06_15"]}},
"flow":{"pkts_toserver":3,"pkts_toclient":1,"bytes_toserver":720,"bytes_toclie
nt":74,"start":"2022-05-02T02:49:38.340125-0400"}}
```


Figure 4.2: Suricata detection alert.

We also identified the following patterns:

- The cryptojacked containers began receiving jobs without having requested them.
- When they are not encrypted, all the packets have the ACK flag that confirms receipt of packets.

- When they are not encrypted, all the packets have the PSH flag which tells the container to process packets even if the buffer is not full.
- Mining sessions are lengthy processes. During these mining sessions, ACK and PSH are the only flags that are used. FIN, RST, and SYN flags were only identified near the start or end of communications.
- All container ports used by the pools were ephemeral ports greater than 30,000.
- Between the communications for assigning jobs, sending solutions, and acknowledging receipt, there are frequent 68- byte ACK flags to maintain communication.
- For all three scripts used, similar package lengths were noted for new jobs sent from the pool to the container, solutions sent to the pool, and acknowledging receipt of the solution. For Cudominer, these lengths were consistently 403, 240 or 241, and 132 or 133, respectively. For Minexmr they were 439, 154-262, and 153-162. A screenshot of a portion of the Minxmr packet information can be seen in Figure 4.3. For Xmrpool.eu they were 439, 261, and 153.

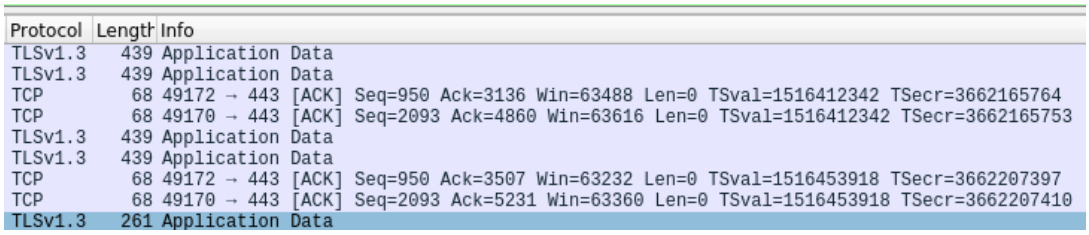

Figure 4.3: Minexmr packet information in Wireshark

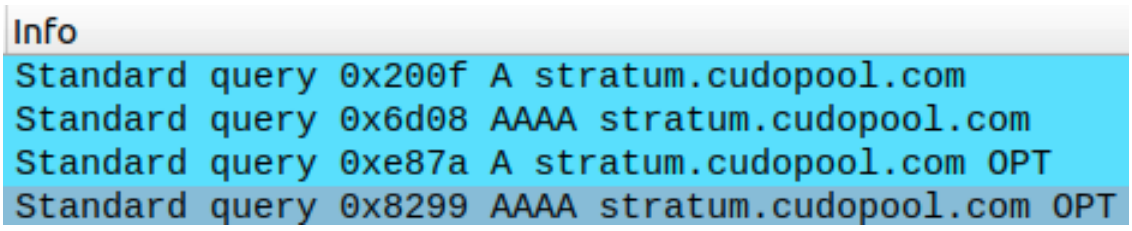

Figure 4.4: Detection of the Stratum protocol in Wireshark

4.7 Docker Container Security

While the focus of our analysis is on detecting unauthorized cryptomining in a Docker host once it is in action, the first line of defense against cryptojacking threats to docker containers is to take appropriate measures to follow best practices for Docker security. This is particularly important for host-based cryptojacking, since cybercriminals need to find vulnerabilities so they can bring the malicious script to their victim.

4.7.1 Stay Up to Date

Keeping software up to date is a key factor to mitigate the possibility of vulnerabilities in software which can be exploited by cryptojackers. It is also useful to run new software through the CVE (Common Vulnerabilities & Exposures) database [CVE].

4.7.2 Resource Isolation and Management

Namespaces isolate the processes in a container so they cannot see or affect the host system or another container. Control groups provide for resource management, making sure that each container has a fair amount of disk I/O resources, memory, and CPU. This helps to protect the containers as well as the host from denial of service attacks [Inca].

4.7.3 Whitelisting/Blacklisting Rules in iptables

When appropriate, whitelisting rules should be added to the DOCKER-USER iptables to allow only specific IP addresses or networks to access the containers and only expose certain ports [Inca]. Alternatively, blacklisting should be used to block

connections to known cryptomining pools, remote access, or certain ports. For example, pool.*, *pool.com, *pool.org, and *xmr.* block Monero pools that are used for cryptomining.

4.7.4 Principles of Least Privilege for Kernel Capabilities

Instead of deploying containers on the system having root privileges, assigning containers the minimum privileges required for their processes using an allowlist is recommended [Inca]. This helps prevent privilege-escalation attacks and will make it much more difficult for a cybercriminal to reach the host even if they are able to gain access to a container.

4.7.5 Image Authentication

The Docker Engine should be configured to only run images with Docker Content Trust signature verification in order to ensure the integrity and publisher of images, as these can be sources through which cryptojackers gain access to a system [Inca]. In addition to this, another method to mitigate the risk of falling victim to cryptojacking malware through an malicious file is to scan new images on VirusTotal [Vir] and ClamAV [Cisa]. Docker also provides a tool for scanning images for vulnerabilities [Incc].

4.8 Monitoring Host-based Docker Containers

Cryptojacking can be challenging to detect than other malware attacks that take control of the victim's device or make it unavailable because its' goal is precisely to take advantage of the victim's computational resources for as long as possible in

order to generate income. Constantly evolving threats and insider cryptojacking, where employees abuse employer resources for their own gain, create additional challenges. However, there are steps that can be taken to monitor host-based docker containers to identify attacks.

Open source tools can be used to monitor host-based docker containers and trigger alerts for system administrators when certain indicators are present. From the Docker host, a script can run periodically to collect and log the temperature of the host. At the same time, the Docker stats can collect and log the overall CPU and RAM consumption for each container and from the docker inspection we can get the network information of each container. If the temperature field goes more than 10°Celsius over the average temperature on the CPU core and CPU and RAM in a particular container are more than three times its average consumption, we correlate the IP address using the inspection command. An alert can be sent to the system administrator with the identified parameters for a deeper inspection in the Suricata [Fou] and Wireshark [Wir] logs for the container where they could observe the data to search for further indicators to confirm cryptojacking activity. A DNS record and the IP of the container can also be correlated. Netdata [Incd] can trigger automatic notifications by email and to the monitor channels such as Discord and Slack, providing metrics of high consumption in specific containers.

4.9 Conclusion

In this chapter, we conducted a forensic analysis using honeypots in order to identify key indicators for the detection of cryptomining in host-based Docker containers. We present key measures for securing host-based Docker containers and an approach for monitoring them for the detection of cryptojacking, based on the indicators identified

in our forensic analysis. As future work, we are planning to develop a cryptojacking detection framework for host-based Docker containers using honeypots and machine learning.

CHAPTER 5

CONCLUDING REMARKS AND FUTURE WORK

In this thesis, we explored honeypot-based security enhancements for information systems. First, we provided a comprehensive survey on honeypot and honeynet models that were proposed for IoT, IIoT, and CPS environments over the period 2002-2020. Our classification in this work improves the existing works [FDFV18, CPM15, ZKF19, RRM+18] by identifying some of the recurring key characteristics of the surveyed works. Specifically, we provided a taxonomy of 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. We also considered the simulated services, the inheritance relationships between the honeypots and honeynets, the platforms they were built on, and the programming languages they used. 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 discuss the key design factors for honeypots and honeynets for IoT, IIoT, and CPS applications.

Next, we proposed a smart honeypot framework based on open-source resources, that integrates the use of IDPS and ML for securing SDN-based networks through dynamic rules configuration. Unlike research honeypots, it is proposed to obtain attack information from the honeypots for production purposes for the security of SDN-based networks, dynamically creating new rules in real-time for the attacks, and sharing them with the SDN-based network. The framework is implemented on an enterprise SDN testbed network, and its' performance was tested in detecting attacks using various ML classifiers, as well as its' ability to effectively generate new rules and dynamically configure the rules of the SDN-based network. Our

performance evaluation of S-Pot in detecting attacks using various ML classifiers shows that it can detect attacks with 97% accuracy using J48 algorithm. In addition, we evaluated the effectiveness of S-Pot in improving the security of an enterprise SDN testbed network. Our results demonstrate that S-Pot can improve the security of the SDN networks by blocking attacks with 40% improvement over legacy systems without S-Pot, effectively generating rules, and dynamically configuring the network.

Finally, we carried out a forensic analysis to identify indicators for the detection of unauthorized cryptomining using honeypots, presented measures for securing host-based Docker containers, and proposed an approach for monitoring host-based Docker containers for cryptojacking detection. Our results revealed that host temperature, combined with container resource usage, Stratum protocol, keywords in DNS requests, and the use of the container's ephemeral ports are notable indicators of possible unauthorized cryptomining.

We also identified areas or interest for future research. Open issues in the research regarding the use of honeypots for IoT, IIoT, and CPS include emerging domains or technologies such as wearable devices, medical devices, and smart city, unexplored protocols, emerging platforms, optimized deployment locations, insider attacks, machine learning, discrimination of benign decoy traffic, and honeypots for production purposes. Further development of S-Pot should consider the use of an ML-based IDPS or deep learning methods to enhance its effectiveness in intelligently detecting zero-day attacks. Future work regarding the detection of unauthorized cryptomining should expand on the proposed approach, including the development of a cryptojacking detection framework for host-based Docker containers using honeypots and machine learning.

BIBLIOGRAPHY

[AA18] Z. Ammar and A. AlSharif. Deployment of iot-based honeynet model. In *ICIT 2018: Proceedings of the 6th Int. Conference on Information Technology: IoT and Smart City*, pages 134–139, Dec 2018.

[AAT16] Daniele Antonioli, Anand Agrawal, and Nils Ole Tippenhauer. Towards high-interaction virtual ics honeypots-in-a-box. In *Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy*, page 13–22, 2016.

[ABF18] S. Almulla, E. Bou-Harb, and C. Fachkha. Cyber security threats targeting CPS systems: A novel approach using honeypot. In *SECURWARE 2018: The Twelfth International Conference on Emerging Security Information, Systems and Technologies*, pages 85–91, Dec 2018.

[ADM16] Thiago Alves, Rishabh Das, and Thomas Morris. Virtualization of industrial control system testbeds for cybersecurity. In *Proceedings of the 2nd Annual Industrial Control System Security Workshop*, page 10–14, New York, NY, USA, 2016. ACM.

[A.G17] A.Guerra Manzanares. *HoneyIo4: The construction of a virtual, low-interaction IoT Honeypot*. PhD thesis, Universitat Politècnica de Catalunya, 2017.

[AGMAA+20] Mohammed Ali Al-Garadi, Amr Mohamed, Abdulla Khalid Al-Ali, Xiaojiang Du, Ihsan Ali, and Mohsen Guizani. A survey of machine and deep learning methods for internet of things (iot) security. *IEEE Communications Surveys Tutorials*, 22(3):1646–1685, 2020.

[AHM18] M. Azab, A. Hamdy, and A. Mansour. Repoxy: Replication proxy for trustworthy sdn controller operation. In *17th IEEE Int. Conference On Trust, Security And Privacy In Computing And Communications*, 2018.

[ALK19] S. Ahn, T. Lee, and K. Kim. A study on improving security of ics through honeypot and arp spoofing. In *Int. Conference on Information and Communication Technology Convergence*, pages 964–967, Oct 2019.

[ANFL18] A. Acien, A. Nieto, G. Fernandez, and J. Lopez. A comprehensive methodology for deploying iot honeypots. In *TrustBus 2018*, volume 11033, pages 229–243, Sept 2018.

[ANSB19] Amir Afianian, Salman Niksefat, Babak Sadeghiyan, and David Baptiste. Malware dynamic analysis evasion techniques: A survey. *ACM Comput. Surv.*, 52(6), November 2019.

[ASKS19] Azuan Ahmad, Wan Shafiuddin, Mohd Nazri Kama, and Madihah Mohd Saudi. *A New Cryptojacking Malware Classifier Model Based on Dendritic Cell Algorithm*. Association for Computing Machinery, New York, NY, USA, 2019.

[AT15] Daniele Antonioli and Nils Ole Tippenhauer. Minicps: A toolkit for security research on cps networks. In *Proc. First ACM Workshop on Cyber-Physical Systems-Secur. and/or Privacy*, page 91–100, 2015.

[ATN17] M. Anirudh, S. A. Thileeban, and D. J. Nallathambi. Use of honeypots for mitigating dos attacks targeted on iot networks. In *2017 International Conference on Computer, Communication and Signal Processing (ICCCSP)*, pages 1–4, Jan 2017.

[ATUH18] Shingo Abe, Yohei Tanaka, Yukako Uchida, and Shinichi Horata. Developing deception network system with traceback honeypot in ics network. *SICE Journal of Control, Measurement, and System Integration*, 11(4):372–379, 2018.

[AUB18] H. Aksu, A. S. Uluagac, and E. Bentley. Identification of wearable devices with bluetooth. *IEEE Transactions on Sustainable Computing*, pages 1–1, 2018.

[BAR+20] L. Babun, H. Aksu, L. Ryan, K. Akkaya, E.S. Bentley, and A.S.Uluagac. Z-iot: Passive device-class fingerprinting of zigbee and z-wave iot devices. In *2020 IEEE Int. Conf. Commun. (ICC)*, pages 1–7. IEEE, 2020.

[BARM17] Borja Bordel, Ramón Alcarria, Tomás Robles, and Diego Martín. Cyber–physical systems: Extending pervasive sensing from control theory to the internet of things. *Pervasive Mobile Comput.*, 40:156 – 184, 2017.

[BCP19] G. Bernieri, M. Conti, and F. Pascucci. Mimepot: a model-based honeypot for industrial control networks. In *2019 IEEE Int. Conference on Systems, Man and Cybernetics (SMC)*, pages 433–438, Oct 2019.

[BDC+21] Leonardo Babun, Kyle Denney, Z. Berkay Celik, Patrick McDaniel, and A. Selcuk Uluagac. A survey on iot platforms: Communication, security, and privacy perspectives. *Computer Networks*, 2021.

[Ber12] Dustin J. Berman. Emulating Industrial Control System Devices using Gumstix Technology. Master's thesis, Air Force Institute of Technology Air University, June 2012.

[BG21] Jorge Buzzio-Garcia. Creation of a high-interaction honeypot system based-on docker containers. In *2021 Fifth World Conference on Smart Trends in Systems Security and Sustainability (WorldS4)*, pages 146–151, 2021.

[BJM+14] Dániel István Buza, Ferenc Juhász, György Miru, Márk Félegyházi, and Tamás Holczer. Cryplh: Protecting smart energy systems from targeted attacks with a plc honeypot. In *Smart Grid Security*, pages 181–192, Cham, 2014. Springer International Publishing.

[BM19] A. Belqruch and A. Maach. Scada security using ssh honeypot. In *2019 Proceedings of the 2nd International Conference on Networking, Information Systems & Security*, pages 1–5, Mar 2019.

[Bod14] Roland C. Bodenheim. Impact of the Shodan Computer Search Engine on Internet-facing Industrial Control System Devices. Master's thesis, Air Force Institute of Technology Air University, March 2014.

[Bon11] Digital Bond. Digital Bond SCADA Honeynet. https://web.archive.org/web/20111215085656/http://www.digitalbond.com/tools/scada-honeynet/, 2011. [Online; accessed 2-May-2020].

[BÖS20] Ismail Butun, Patrik Österberg, and Houbing Song. Security of the internet of things: Vulnerabilities, attacks, and countermeasures. *IEEE Communications Surveys Tutorials*, 22(1):616–644, 2020.

[BRBA17] Babak Bashari Rad, Harrison Bhatti, and Mohammad Ahmadi. An introduction to docker and analysis of its

performance. *IJCSNS International Journal of Computer Science and Network Security*, 17(3):228–234, url = "http://paper.ijcsns.org/07_book/201703/20170327.pdf", 2017.

[Cen21] Center for Internet Security. The SolarWinds Cyber-Attack: What You Need to Know. https://www.cisecurity.org/solarwinds/, 2021. [Online; accessed 26-March-2021].

[CEWB16a] D. Chen, M. Egeley, M. Woo, and D. Brumley. Firmadyne. https://github.com/firmadyne/firmadyne, 2016. [Online; accessed 30-Apr-2020].

[CEWB16b] D. Chen, M. Egeley, M. Woo, and D. Brumley. Towards automated dynamic analysis for linux-based embedded firmware. In *2016 NDSS*, pages 21–24. Internet Society, Feb. 2016.

[Che] Cherny, M. and Dulce, S. Docker Overview. [Online; accessed 12-April-2022].

[Cisa] Cisco. ClamAV. https://www.clamav.net/. [Online; accessed 10-April-2022].

[Cisb] Cisco. Snort - network intrusion detection and prevention system. https://www.snort.org. [Online; accessed 17-May-2021].

[Cisc] Cisco. What are the most common cyber attacks? https://www.cisco.com/c/en/us/products/security/common-cyberattacks.html. [Online; accessed 2-Jun-2021].

[CLLL18] Jianhong Cao, Wei Li, Jianjun Li, and Bo Li. Dipot: A distributed industrial honeypot system. In Meikang Qiu, editor, *Smart Computing and Communication*, pages 300–309, Cham, 2018. Springer International Publishing.

[CMAU17] Mehmet Hazar Cintuglu, Osama A. Mohammed, Kemal Akkaya, and A. Selcuk Uluagac. A survey on smart grid cyber-physical system testbeds. *IEEE Communications Surveys Tutorials*, 19(1):446–464, 2017.

[Coi] CoinGecko. Cryptocurrency Prices by Market Cap. https://www.coingecko.com/. [Online; accessed 10-April-2022].

[Cow19] Cowrie. Cowrie ssh and telnet honeypot. https://www.cowrie.org/, 2019. [Online; accessed 2-Apr-2020].

[C.P] C.P. Research. Cloud-based cryptojacking article . https://research.checkpoint.com/2020/the- 2020-cyber- security- report/. [Online; accessed 10-April-2022].

[CPM15] R. M. Campbell, K. Padayachee, and T. Masombuka. A survey of honeypot research: Trends and opportunities. In *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 208–212, 2015.

[CROD21] M. Caprolu, S. Raponi, G. Oligeri, and R. DiPietro. Cryptomining makes noise: Detecting cryptojacking via machine learning. *Computer Communications*, 171:126–139, 2021.

[CV20] R. Centeno and L. Victoria. Zoomed In: A Look into a Coinminer Bundled with Zoom Installer. TrendMicro, Apr. 3, 2020.

[CVE] CVE Authors. Security Vulnerability Datasource. [Online; accessed 6-Mayo-2022].

[Cym] Cymmetria. Mtpot. https://github.com/Cymmetria/MTPot. [Online; accessed 1-Apr-2020].

[Dat21] Datadog. 10 Trends in Real-World Container Use. Datadog, Oct. 2021.

[DHD+20] Hamid Darabian, Sajad Homayounoot, Ali Dehghantanha, Sattar Hashemi, Hadis Karimipour, Reza Meimandi Parizi, and Kim-Kwang Raymond Choo. Detecting cryptomining malware: a deep learning approach for static and dynamic analysis. *Journal of Grid Computing*, 18:293–303, 2020.

[Din] DinoTools. Dionaea. https://github.com/DinoTools/dionaea. [Online; accessed 2-Apr-2020].

[Dio15] Dionaea. Service. https://dionaea.readthedocs.io/en/latest/introduction.html, 2015. [Online; accessed 2-Apr-2020].

[DJB13] J. P. Disso, K. Jones, and S. Bailey. A plausible solution to scada security honeypot systems. In *8th Int. Conf. on Broadband and Wireless Comput., Comm. and Applications*, pages 443–448, 2013.

[DLL+19] F. Dang, Z. Li, Y. Liu, E. Zhai, Q. I. Chen, T. Xu, Y. Chen, and J. Yang. Understanding fileless attacks on linux-based iot devices with honeycloud. In *17th Annual International Conference on Mobile Systems, Applications, and Services*, pages 482–493, Nov 2019.

[Dom20] C. Doman. Team TNT: The First Crypto-Mining Worm to Steal AWS Credentials. Cado Security, Aug. 16, 2020.

[DSI+19] C. Dalamagkas, P. Sarigiannidis, D. Ioannidis, E. Iturbe, O. Nikolis, F. Ramos, E. Rios, A. Sarigiannidis, and D. Tzovaras. A survey on honeypots, honeynets and their applications on smart grid. In *2019 IEEE Conference on Network Softwarization (NetSoft)*, pages 93–100, June 2019.

[DSM17a] S. Dowling, M. Schukat, and H. Melvin. Data-centric framework for adaptive smart city honeynets. In *2017 Smart City Symposium Prague (SCSP)*, pages 1–7, 2017.

[DSM17b] S. Dowling, M. Schukat, and H. Melvin. A zigbee honeypot to assess iot cyberattack behaviour. In *2017 28th Irish Signals and Systems Conference (ISSC)*, pages 1–6, June 2017.

[DW20a] M. Du and K. Wang. An sdn-enabled pseudo-honeypot strategy for distributed denial of service attacks in industrial internet of things. *IEEE Transactions on Industrial Informatics*, 16(1):648–657, Jan 2020.

[DW20b] M. Du and K. Wang. An sdn-enabled pseudo-honeypot strategy for distributed denial of service attacks in industrial internet of things. *IEEE Tran. on Industrial Informatics*, 16(1):648–657, 2020.

[DZD18] Chenpeng Ding, Jiangtao Zhai, and Yuewei Dai. An improved ics honeypot based on snap7 and imunes. In *Cloud Computing and Security*, pages 303–313, Cham, 2018. Springer International Publishing.

[Ela17] Elastic. Elasticsearch 5.2.2. https://www.elastic.co/downloads/past-releases/elasticsearch-5-2-2/, 2017. [Online; accessed 9-Apr-2020].

[Ela20a] Elastic. Getting started with logstash. https://www.elastic.co/guide/en/logstash/current/getting-started-with-logstash.html, 2020. [Online; accessed 9-Apr-2020].

[Ela20b] Elastic. Kibana: Your window into the elastic stack. https://www.elastic.co/kibana, 2020. [Online; accessed 9-Apr-2020].

[ELDJ19] M. S. Elsayed, N. A. Le-Khac, S. Dev, and A. D. Jurcut. Machine-learning techniques for detecting attacks in sdn. In *IEEE 7th Int. Conf. on Computer Science and Network Technology*, pages 277–281, 2019.

[Evr16] G. Evron. Mirai open-source iot honeypot: New cymmetria research release. https://cymmetria.com/blog/mirai-open-source-iot-honeypot-new-cymmetria-research-release/, Nov. 2016. [Online; accessed 16-Apr-2020].

[Exp] NVM Express. NVMe-CLI. https://nvmexpress.org/open-source-nvme-management-utility-nvme-command-line-interface-nvme-cli/. [Online; accessed 10-April-2022].

[FACU21] Javier Franco, Ahmet Aris, Berk Canberk, and A. Selcuk Uluagac. A survey of honeypots and honeynets for internet of things, industrial internet of things, and cyber-physical systems. *IEEE Communications Surveys Tutorials*, 23(4):2351–2383, 2021.

[FDF15] W. Fan, Z. Du, and D. Fernández. Taxonomy of honeynet solutions. In *2015 SAI Intelligent Systems Conference (IntelliSys)*, pages 1002–1009, Nov 2015.

[FDFV18] W. Fan, Z. Du, D. Fernández, and V. A. Villagrá. Enabling an anatomic view to investigate honeypot systems: A survey. *IEEE Syst. J.*, 12(4):3906–3919, Dec 2018.

[FFV15] W. Fan, D. Fernández, and V. A. Villagrá. Technology independent honeynet description language. In *2015 3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, pages 303–311, Feb 2015.

[Fie20] FieldComm Group. HART Communication Protocol. https://fieldcommgroup.org/technologies/hart, 2020. [Online; accessed 14-May-2020].

[foo13] foospidy. Honeypy. https://github.com/foospidy/HoneyPy, 2013. [Online; accessed 30-Apr-2020].

[Fou] Open Information Security Foundation. Suricata. https://suricata.io/. [Online; accessed 10-April-2022].

[Fox21] W. Foxley. Crypto-Jacking Virus Infects 850,000 Servers, Hackers Run off With Millions. CoinDesk, Feb. 23, 2021.

[FPZ19] Pietro Ferretti, Marcello Pogliani, and Stefano Zanero. Characterizing background noise in ics traffic through a set of low interaction honeypots. In *Proceedings of the ACM Workshop on Cyber-Physical Systems Security & Privacy*, page 51–61, 2019.

[Gal17] Justin K. Gallenstein. Integration of the Network and Application Layers of Automatically-Configured Programmable Logic Controller Honeypots. Master's thesis, Air Force Institute of Technology Air University, March 2017.

[GBWG19] Christopher Greer, Martin Burns, David Wollman, and Edward Griffor. Cyber-physical systems and internet of things. Technical report, NIST, March 2019.

[GKK+18] U.D. Gandhi, P.M. Kumar, S. Kadu, R. Varatharajan, G. Manogaran, and R. Sundarasekar. Hiotpot: Surveillance on iot devices against recent threats. *Wireless Personal Communications*, 103(2):1179–1194, 2018.

[GLA+17] Benjamin Green, Anhtuan Lee, Rob Antrobus, Utz Roedig, David Hutchison, and Awais Rashid. Pains, gains and plcs: Ten lessons from building an industrial control systems testbed for security research. In *10th USENIX Workshop on Cyber Security Experimentation and Test*, Vancouver, BC, August 2017.

[GMC16] Prageeth Gunathilaka, Daisuke Mashima, and Binbin Chen. Softgrid: A software-based smart grid testbed for evaluating substation cybersecurity solutions. In *Proc. 2nd ACM Workshop on Cyber-Physical Syst. Secur. and Privacy*, page 113–124, 2016.

[GMSS15] Jorge Granjal, Edmundo Monteiro, and Jorge Sá Silva. Security for the internet of things: A survey of existing protocols and open re-

search issues. *IEEE Communications Surveys Tutorials*, 17(3):1294–1312, 2015.

[Goo] Google. CAdvisor. [Online; accessed 12-April-2022].

[GPLC19] Ankit Gangwal, Samuele Giuliano Piazzetta, Gianluca Lain, and Mauro Conti. Detecting covert cryptomining using hpc. *CoRR*, abs/1909.00268, 2019.

[Gra19] R.D. Graham. Masscan. https://github.com/robertdavidgraham/masscan/, 2019. [Online; accessed 14-May-2020].

[Gri20] GridLab-D Simulation Software. https://www.gridlabd.org/, 2020. [Online; accessed 7-April-2020].

[Gro] Machine Learning Group. Weka. https://www.cs.waikato.ac.nz/ml/weka/. [Online; accessed 10-May-2021].

[GTB+17] J. Guarnizo, A. Tambe, S.S. Bhunia, M. Ochoa, N.O. Tippenhauer, A. Shabtail, and Y. Elovici. Siphon: Towards scalable high-interaction physical honeypots. In *2017 Cyber Physical Systems Security Workshops (CPSS)*, pages 57–68, April 2017.

[G.W18] G.Wagener. Adaptive honeypot alternative (aha). http://git.quuxlabs.com/, 2018. [Online; accessed 23-Apr-2020].

[Hac20] R. Hackett. Tesla Hackers Hacked AWS Cloud Services to Mine Monero. Fortune, Oct. 19, 2020.

[Hak] M. A. Hakim. u-pot. https://github.com/azizulhakim/u-pot/. [Online; accessed 1-Apr-2020].

[Han19] Michael Haney. Leveraging cyber-physical system honeypots to enhance threat intelligence. In *Critical Infrastructure Protection XIII*, pages 209–233. Springer International Publishing, 2019.

[HFB15] Tamas Holczer, Mark Felegyhazi, and Levente Buttyan. The design and implementation of a plc honeypot for detecting cyber attacks against industrial control systems. In *Proc. Int. Conf. on Com-*

puter Security in a Nuclear World: Expert Discussion and Exchange. IAEA, 2015.

[HLLL17] A. Humayed, J. Lin, F. Li, and B. Luo. Cyber-physical systems security—a survey. *IEEE Internet of Things J.*, 4(6):1802–1831, Dec 2017.

[HMP+20] Stephen Hilt, Federico Maggi, Charles Perine, Lord Remorin, Martin Rösler, and Rainer Vosseler. Caught in the Act: Running a Realistic Factory Honeypot to Capture Real Threats. 2020.

[Hon01] Honeynet Project. Know your enemy: Honeynets. http://www.symantec.com/connect/articles/knowyour-enemy-honeynets, April 2001. [Online; accessed 2-Apr-2020].

[Hon20] Honeynet Project. https://www.honeynet.org/, 2020. [Online; accessed 2-May-2020].

[HP14] Michael Haney and Mauricio Papa. A framework for the design and deployment of a scada honeynet. In *Proceedings of the 9th Annual Cyber and Information Security Research Conference*, page 121–124, New York, NY, USA, 2014. ACM.

[Hyu18] Dahae Hyun. Collecting cyberattack data for industrial control systems using honeypots. Master's thesis, Naval Postgraduate School, March 2018.

[IBM] IBM Cloud Education. Machine learning. https://www.ibm.com/cloud/learn/machine-learning. [Online; accessed 10-May-2021].

[Inca] Docker Inc. Docker Security. https://docs.docker.com/engine/security/. [Online; accessed 10-April-2022].

[Incb] Docker Inc. Overview of Docker Compose. https://docs.docker.com/compose/. [Online; accessed 10-April-2022].

[Incc] Docker Inc. Vulnerability Scanning for Docker Local Images. https://docs.docker.com/engine/scan/. [Online; accessed 10-April-2022].

[Incd] Netdata Inc. Netdata. https://www.netdata.cloud/. [Online; accessed 10-April-2022].

[Jar13] Robert M. Jaromin. Emulation of Industrial Control Field Device Protocols. Master's thesis, Air Force Institute of Technology Air University, March 2013.

[KA20] A. Kaan Sarica and P. Angin. A novel sdn dataset for intrusion detection in iot networks. In *2020 16th International Conference on Network and Service Management (CNSM)*, pages 1–5, 2020.

[KG15] K. Kołtyś and R. Gajewski. Shape: A honeypot for electric power substation. *Journal of Telecommunications and Information Technology*, nr 4:37–43, 2015.

[KGM17] S. Kuman, S. Groš, and M. Mikuc. An experiment in using imunes and conpot to emulate honeypot control networks. In *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1262–1268, 2017.

[KHT+17] S. Kyung, W. Han, N. Tiwari, V. H. Dixit, L. Srinivas, Z. Zhao, A. Doupé, and G. Ahn. Honeyproxy: Design and implementation of next-generation honeynet via sdn. In *2017 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9, 2017.

[Kip16] Kippo. Kippo- ssh honeypot. https://github.com/desaster/kippo, 2016. [Online; accessed 2-Apr-2020].

[KKH+21] Rupesh Raj Karn, Prabhakar Kudva, Hai Huang, Sahil Suneja, and Ibrahim M. Elfadel. Cryptomining detection in container clouds using system calls and explainable machine learning. *IEEE Transactions on Parallel and Distributed Systems*, 32(3):674–691, 2021.

[KP18] B. Kaur and P. K. Pateriya. A survey on security concerns in internet of things. In *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 27–34, June 2018.

[KR19] Marian M. Kendrick and Zaki A. Rucker. Energy Grid Threat Analysis Using Honeypots. Master's thesis, Naval Postgraduate School, June 2019.

[Kri17] P. Krishnaprasad. *Capturing attacks on IoT deviceswith a multi-purpose IoThoneypot*. PhD thesis, Indian Institute of Technology Kanpur, 2017.

[KV17] P. Kumar and R.S. Verma. A review on recent advances & future trends of security in honeypot. *Int. J. of Adv. Res. Computer Science*, 8(3):1108–1113, Mar-Apr 2017.

[Lab14] Twisted Matrix Labs. Welcome to the twisted documentation. https://twistedmatrix.com/documents/current/, Sept. 2014. [Online; accessed 9-Apr-2020].

[Lau21] Y. Lau. Cryptocurrencies hit market cap of 3 trillion for the first time as Bitcoin and Ether reach record highs. Fortune, Nov. 9, 2021.

[LBAU17] Juan Lopez, Leonardo Babun, Hidayet Aksu, and Selcuk Uluagac. A survey on function and system call hooking approaches. *Journal of Hardware and Systems Security*, 1, 06 2017.

[LEBM20] Nada Lachtar, Abdulrahman Abu Elkhail, Anys Bacha, and Hafiz Malik. A cross-stack approach towards defending against cryptojacking. *IEEE Computer Architecture Letters*, 19(2):126–129, 2020.

[LFR+16] S. Litchfield, D. Formby, J. Rogers, S. Meliopoulos, and R. Beyah. Rethinking the honeypot for cyber-physical systems. *IEEE Internet Computing*, 20(5):9–17, Sep. 2016.

[LHM10] Bob Lantz, Brandon Heller, and Nick McKeown. A network in a laptop: Rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010.

[lib] Tcpdump & libpcap. Tcpdump & libpcap. hhttps://www.tcpdump.org/. [Online; accessed 17-May-2021].

[LKAR16] Stephan Lau, Johannes Klick, Stephan Arndt, and Volker Roth. Poster: Towards highly interactive honeypots for industrial control systems. In *Proc. 2016 ACM SIGSAC Conf. on Computer and Commun. Sec.*, CCS '16, page 1823–1825, 2016.

[lms] lm-sensors. https://github.com/lm-sensors/lm-sensors. [Online; accessed 10-April-2022].

[LSFF16] Felipe A. Lopes, Marcelo Santos, Robson Fidalgo, and Stenio Fernandes. A software engineering perspective on sdn programmability. *IEEE Communications Surveys Tutorials*, 18(2):1255–1272, 2016.

[LSOK21] Euijong Lee, Young-Duk Seo, Se-Ra Oh, and Young-Gab Kim. A survey on standards for interoperability and security in the internet of things. *IEEE Communications Surveys Tutorials*, 23(2):1020–1047, 2021.

[LVS20] B. Lingenfelter, I. Vakilinia, and S. Sengupta. Analyzing variation among iot botnets using medium interaction honeypots. In *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0761–0767, 2020.

[LXJ+17] T. Luo, Z. Xu, X. Jin, Y. Jia, and X. Ouyang. Iotcandyjar: Towards an intelligent-interaction honeypot for iot devices. In *Black Hat 2017*, 2017.

[MAL+19] Imran Makhdoom, Mehran Abolhasan, Justin Lipman, Ren Ping Liu, and Wei Ni. Anatomy of threats to the internet of things. *IEEE Communications Surveys Tutorials*, 21(2):1636–1675, 2019.

[Mar] Market Research Future. Software defined networking (sdn) market size usd 59 billion by 2023 growing at massive cagr of 42.41%. https://www.globenewswire.com/news-release/2019/04/04/1797303/0/en/Software-Defined-Networking-SDN-Market-Size-USD-59-Billion-by-2023-Growing-at-Massive-CAGR-of-42-41.html. [Online; accessed 12-June-2021].

[MBSC20] A. Molina Zarca, J. B. Bernabe, A. Skarmeta, and J. M. A. Calero. Virtual iot honeynets to mitigate cyberattacks in sdn/nfv-enabled iot networks. *IEEE Journal on Selected Areas in Communications*, 2020.

[MBVJ11] Abhishek Mairh, Debabrat Barik, Kanchan Verma, and Debasish Jena. Honeypot in network security: A survey. In *Proceedings of the 2011 International Conference on Communication, Computing & Security*, ICCCS '11, page 600–605, New York, NY, USA, 2011. Association for Computing Machinery.

[MCB17] V. Martin, Q. Cao, and T. Benson. Fending off iot-hunting attacks at home networks. In *Proceedings of the 2nd Workshop on Cloud-Assisted Networking*, pages 67–72. ACM, Dec. 2017.

[McD21] C. McDonald. Cryptojacking Malware Hid into Emails. Mailguard, Feb. 23, 2021.

[MCG+18] Andrés Felipe Murillo, Luis Francisco Cómbita, Andrea Calderón Gonzalez, Sandra Rueda, Alvaro A. Cardenas, and Nicanor Quijano. A virtual environment for industrial control systems: A nonlinear use-case in attack detection, identification, and response. In *Proceedings of the 4th Annual Industrial Control System Security Workshop*, page 25–32, New York, NY, USA, 2018. ACM.

[MCGT17] D. Mashima, B. Chen, P. Gunathilaka, and E. L. Tjiong. Towards a grid-wide, high-fidelity electrical substation honeynet. In *2017 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 89–95, Oct 2017.

[MCZ+19] 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.*, 6(5):8182–8201, Oct 2019.

[Mil21] A. Milano. Russian Scientists Arrested Crypto Mining Nuclear Lab. Coindesk, Feb. 23, 2021.

[MLC19] D. Mashima, Y. Li, and B. Chen. Who's scanning our smart grid? empirical study on honeypot data. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Dec 2019.

[MMS19] AysŞe Rumeysa Mohammed, Shady A. Mohammed, and Shervin Shirmohammadi. Machine learning and deep learning based traffic classification and prediction in software defined networking. In *2019 IEEE International Symposium on Measurements Networking (M N)*, pages 1–6, 2019.

[Mor] Steve Morgan. Cybercrime to cost the world $10.5 trillion annually by 2025. https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/. [Online; accessed 12-June-2021].

[MPB+20] Ganapathy Mani, Vikram Pasumarti, Bharat Bhargava, Faisal Tariq Vora, James MacDonald, Justin King, and Jason Kobes. Decrypto pro: Deep learning based cryptomining malware detection using performance counters. In *2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*, pages 109–118, 2020.

[MS18] L. Metongnon and R. Sadre. Beyond telnet: Prevalence of iot protocols in telescope and honeypot measurements. In *2018 WTMC*, pages 21–26, Aug. 2018.

[NAB+21] Faraz Naseem, Ahmet Aris, Leonardo Babun, Ege Tekiner, and Selcuk Uluagac. MINOS: A lightweight real-time cryptojacking detection system. In *28th Annual Network and Distributed System Security Symposium, NDSS, February 21-24, 2021*. The Internet Society, 2021.

[NBHC+19] Nataliia Neshenko, Elias Bou-Harb, Jorge Crichigno, Georges Kaddoum, and Nasir Ghani. Demystifying iot security: An exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations. *IEEE Communications Surveys Tutorials*, 21(3):2702–2733, 2019.

[NG20] Thomas Rincy N and Roopam Gupta. Feature selection techniques and its importance in machine learning: A survey. In *2020 IEEE International Students' Conference on Electrical, Electronics and Computer Science*, 2020.

[Nma20] Nmap. https://nmap.org/, 2020. [Online; accessed 14-May-2020].

[NMM+15] Hidemasa Naruoka, Masafumi Matsuta, Wataru Machii, Tomomi Aoyama, Masahito Koike, Ichiro Koshijima, and Yoshihiro Hashimoto. Ics honeypot system (camouflagenet) based on attacker's human factors. *Procedia Manufacturing*, 3:1074 – 1081, 2015. 6th Int. Conf. Applied Human Factors and Ergonomics.

[NSBU20] 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 *2020 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9, 2020.

[NSRU19] 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 *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pages 389–396, 2019.

[NSRU20] AKM Iqridar Newaz, Amit Kumar Sikder, Mohammad Ashiqur Rahman, and A. Selcuk Uluagac. A survey on security and privacy issues in modern healthcare systems: Attacks and defenses, 2020.

[NZD+16] S. Nanda, F. Zafari, C. DeCusatis, E. Wedaa, and B. Yang. Predicting network attack patterns in sdn using machine learning approach. In *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 167–172, 2016.

[OALS21] Harun Oz, Ahmet Aris, Albert Levi, and A. Selcuk Uluagac. A Survey on Ransomware: Evolution, Taxonomy, and Defense Solutions. *arXiv e-prints*, page arXiv:2102.06249, February 2021.

[OCD16] Opeyemi Osanaiye, Kim-Kwang Raymond Choo, and Mqhele Dlodlo. Change-point cloud ddos detection using packet inter-arrival time. In *2016 8th Computer Science and Electronic Engineering (CEEC)*, pages 204–209, 2016.

[OKK18] A. D. Oza, G. N. Kumar, and M. Khorajiya. Survey of snaring cyber attacks on iot devices with honeypots and honeynets. In *2018 3rd International Conference for Convergence in Technology (I2CT)*, pages 1–6, April 2018.

[OKKT19] A.D. Oza, G.N. Kumar, M. Khorajiya, and V. Tiwari. *Snaring Cyber Attacks on IoT Devices with Honeynet*. Springer Nature Singapore Pte Ltd., 2019.

[OPC20] OPC Unified Architecture. https://opcfoundation.org/about/opc-technologies/opc-ua/, 2020. [Online; accessed 2-May-2020].

[Pau12] A. Pauna. Improved self adaptive honeypots capable of detecting rootkit malware. In *2012 9th International Conference on Communications (COMM)*, pages 281–284, June 2012.

[Pau18a] A. Pauna. Irassh. https://github.com/adpauna/irassh/, 2018. [Online; accessed 23-Apr-2020].

[Pau18b] A. Pauna. Qrassh. https://github.com/adpauna/qrassh/, 2018. [Online; accessed 16-Apr-2020].

[PB14] A. Pauna and I. Bica. Rassh - reinforced adaptive ssh honeypot. In *2014 10th International Conference on Communications (COMM)*, pages 1–6, May 2014.

[PB16] R. Piggin and I. Buffey. Active defence using an operational technology honeypot. In *11th International Conference on System Safety and Cyber-Security (SSCS 2016)*, pages 1–6, 2016.

[PBA+21] Luis Puche Rondon, Leonardo Babun, Ahmet Aris, Kemal Akkaya, and A. Selcuk Uluagac. Survey on Enterprise Internet-of-Things Systems (E-IoT): A Security Perspective. *arXiv e-prints*, page arXiv:2102.10695, February 2021.

[PBAU20] L. C. PucheRondon, L. Babun, K. Akkaya, and A. S. Uluagac. Hdmi-watch: Smart intrusion detection system against hdmi attacks. *IEEE Transactions on Network Science and Engineering*, pages 1–1, 2020.

[PBPC19] A. Pauna, I. Bica, F. Pop, and A. Castiglione. On the rewards of self-adaptive iot honeypots. *Annals of Telecommunications*, 74:501–515, Jul 2019.

[Pet06] Dale Peterson. SCADA Honeywall: Use Your Own PLC As The Target. https://dale-peterson.com/2008/07/08/scada-honeywall-use-your-own-plc-as-the-target/, 2006. [Online; accessed 2-May-2020].

[PF04] Venkat Pothamsetty and Matthew Franz. SCADA HoneyNet Project: Building Honeypots for Industrial Networks. http://scadahoneynet.sourceforge.net/, 2004. [Online; accessed 2-May-2020].

[Phy19] Phype. Telnet iot honeypot. https://github.com/Phype/telnet-iot-honeypot, 2019. [Online; accessed 2-Apr-2020].

[PIB18] A. Pauna, A. Iacob, and I. Bica. Qrassh - a self-adaptive ssh honeypot driven by q-learning. In *2018 International Conference on Communications (COMM)*, pages 441–446, June 2018.

[PK19] C. Petre and A. Korodi. Honeypot inside an opc ua wrapper for water pumping stations. In *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*, pages 72–77, 2019.

[Pow20] Powerworld. PowerWorld Simulator. https://www.powerworld.com/, 2020. [Online; accessed 15-May-2020].

[PP17] H. Polat and O. Polat. The effects of dos attacks on odl and pox sdn controllers. In *2017 8th International Conference on Information Technology (ICIT)*, pages 554–558, 2017.

[Pro] Prometheus Authors. Prometheus. [Online; accessed 12-April-2022].

[Pro07a] N. Provos. Honeyd. https://github.com/DataSoft/Honeyd, 2007. [Online; accessed 2-Apr-2020].

[Pro07b] N. Provos. Honeyd frequently asked questions. http://www.honeyd.org/faq.php, May 2007. [Online; accessed 1-Apr-2020].

[PSL+19] 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 *2019 IEEE 24th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 1–6, Sep. 2019.

[PSY+15] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow. Iotpot: Analysing the rise of iot compromises. In *9th USENIX Workshop on Offensive Technologies (WOOT 15)*, Washington, D.C., Aug 2015.

[PSY+16] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow. Iotpot – analysing the rise of iot compromises. https://ipsr.ynu.ac.jp/iot/, June 2016. [Online; accessed 2-Apr-2020].

[Qui21] N. Quist. Watchdog: Exposing a cryptojacking campaign that's operated for two years. Palo Alto-Unit 42, Feb. 17, 2021.

[RBA+20] Luis Puche Rondon, Leonardo Babun, Ahmet Aris, Kemal Akkaya, and A. Selcuk Uluagac. Poisonivy: (in)secure practices of enterprise

iot systems in smart buildings. In *Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, BuildSys '20, page 130–139. ACM, 2020.

[RBAU19] Luis Puche Rondon, Leonardo Babun, Kemal Akkaya, and A. Selcuk Uluagac. Hdmi-walk: Attacking hdmi distribution networks via consumer electronic control protocol. In *Proceedings of the 35th Annual Computer Security Applications Conference*, ACSAC '19, page 650–659. ACM, 2019.

[RL21] N. Russo and P. Laskov. Detection of illicit cryptomining using network metadata. *Info. Security 2021*, 11, 2021.

[RLB15] Owen Redwood, Joshua Lawrence, and Mike Burmester. A symbolic honeynet framework for scada system threat intelligence. In *Critical Infrastructure Protection IX*, pages 103–118. Springer International Publishing, 2015.

[RRM+18] M. F. Razali, M. N. Razali, F. Z. Mansor, G. Muruti, and N. Jamil. Iot honeypot: A review from researcher's perspective. In *2018 IEEE Conference on Application, Information and Network Security (AINS)*, pages 93–98, Nov 2018.

[RVH+20] Lukas Rist, Johnny Vestergaard, Daniel Haslinger, Andrea De Pasquale, and John Smith. Conpot ICS/SCADA Honeypot. http://conpot.org/, 2020. [Online; accessed 2-May-2020].

[SA15] Pavol Sokol and Maroš Andrejko. Deploying honeypots and honeynets: Issues of liability. In *Computer Networks*, pages 92–101, Cham, 2015. Springer International Publishing.

[Sas21] A. Sasson. Docker Honeypot Reveals Cryptojacking as Most Common Cloud Threat. Palo Alto-Unit 42, May 27, 2021.

[SBH19] R.K. Shrivastava, B. Bashi, and C. Hota. Attack detection and forensics using honeypot in iot environment. In *International Conference on Distributed Computing and Internet Technology*, pages 402–409, Bhubaneswar, India, Jan 2019.

[SCGM13] 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 and Secur. (ECIW 2013)*, pages 263–270, 2013.

[Sco14] Charles Scott. Designing and Implementing a Honeypot for a SCADA Network. June 2014.

[SCPA19] Nasrin Sultana, Naveen Chilamkurti, Wei Peng, and Rabei Alhadad. Survey on sdn based network intrusion detection system using machine learning approaches. *P2P Networking and Applications*, 12, 2019.

[SCPM15] Paulo Simões, Tiago Cruz, Jorge Proença, and Edmundo Monteiro. *Specialized Honeypots for SCADA Systems*, pages 251–269. Springer International Publishing, 2015.

[SDR19] Rochak Swami, Mayank Dave, and Virender Ranga. Software-defined networking-based ddos defense mechanisms. *ACM Comput. Surv.*, 52, April 2019.

[SHH+19] O. Surnin, F. Hussain, R. Hussain, S. Ostrovskaya, A. Polovinkin, J. Lee, and X. Fernando. Probabilistic estimation of honeypot detection in internet of things environment. In *2019 International Conference on Computing, Networking and Communications (ICNC)*, pages 191–196, Feb 2019.

[SHL15] P. Sokol, M. Husak, and F. Lipták. Deploying honeypots and honeynets: Issue of privacy. In *2015 10th International Conference on Availability, Reliability and Security*, pages 397–403, 2015.

[Sho] Shodan. Honeyscore. https://honeyscore.shodan.io/. [Online; accessed 26-Jul-2020].

[Sho20] Shodan. https://www.shodan.io/, 2020. [Online; accessed 14-May-2020].

[SK18] D. Sever and T. Kisasondi. Efficiency and security of docker based honeypot systems. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1167–1173, 2018.

[SM17] H. Semic and S. Mrdovic. Iot honeypot: A multicomponent solution for handling manual and mirai-based attacks. In *2017 Telecommunication Forum (TELFOR)*, pages 1–4, 2017.

[SOY15a] A. V. Serbanescu, S. Obermeier, and D. Yu. A flexible architecture for industrial control system honeypots. In *12th Int. Joint Conference on e-Business and Telecommunications*, volume 04, pages 16–26, 2015.

[SOY15b] Alexandru Vlad Serbanescu, Sebastian Obermeier, and Der-Yeuan Yu. Ics threat analysis using a large-scale honeynet. In *Proceedings of the 3rd Int. Symposium for ICS & SCADA Cyber Security Research*, page 20–30, Swindon, GBR, 2015. BCS Learning & Development Ltd.

[SPA+21] Amit Kumar Sikder, Giuseppe Petracca, Hidayet Aksu, Trent Jaeger, and A. Selcuk Uluagac. A survey on sensor-based threats and attacks to smart devices and applications. *IEEE Communications Surveys Tutorials*, 23(2):1125–1159, 2021.

[Spi01] L. Spitzner. The value of honeypots, part one:definitions and values of honeypots. http://www.symantec.com/connect/articles/value-honeypots-part-onedefinitions-and-values-honeypots/, Oct 2001. [Online; accessed 14-Apr-2020].

[SRG18] K.G.R. Sharma, A. Reddy, and K. Goody. CVE-2017-10271 Used to Deliver CryptoMiners: An Overview of Techniques Used Post-Exploitation and Pre-Mining . Fireeye, Feb. 15, 2018.

[SSH+18] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund. Industrial internet of things: Challenges, opportunities, and directions. *IEEE Trans. on Ind. Inf.*, 14(11):4724–4734, 2018.

[Sta19] L. Stafira. *Examining Effectiveness of Web-Based Internet of Things Honeypots*. PhD thesis, Air Force Institute of Technology, 2019.

[Sta20] Stack Overflow. Stack Overflow Developer Survey, 2020. [Online; accessed 10-April-2022].

[Sur] O. Surnin. honeypot. https://gitlab.com/legik/honeypot. [Online; accessed 1-Apr-2020].

[Sym19] Symantec. Internet security threat report (istr) 2019. Technical report, Symantec, Feb 2019.

[Tan20] Dmitry Tanana. Behavior-based detection of cryptojacking malware. In *2020 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)*, pages 0543–0545, 2020.

[TAS+19] A. Tambe, Y.L. Aung, R. Sridaran, M. Ochoa an A. K. Jain, N.O. Tippenhauer, A. Shabtai, and Y. Elovici. Detection of threats to iot devices using scalable vpn-forwarded honeypots. In *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy (CODASPY)*, pages 85–96, Mar 2019.

[TAS+21] Ege Tekiner, Abbas Acar, A. Selcuk Uluagac, Engin Kirda, and Ali Aydin Selcuk. SoK: Cryptojacking Malware. *arXiv e-prints*, page arXiv:2103.03851, March 2021.

[TAU+21] Ege Tekiner, Abbas Acar, A. Selcuk Uluagac, Engin Kirda, and Ali Aydin Selcuk. Sok: Cryptojacking malware. In *2021 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 120–139, 2021.

[TAU22] Ege Tekiner, Abbas Acar, and A. Selcuk Uluagac. A lightweight iot cryptojacking detection mechanism in heterogeneous smart home networks. In *29th Annual Network and Distributed System Security Symposium, NDSS*, 2022.

[Thea] The Linux Foundation. Open Daylight. https://www.opendaylight.org. [Online; accessed 13-May-2021].

[Theb] The Linux Foundation. Open vSwitch. https://www.openvswitch.org. [Online; accessed 13-May-2021].

[The20a] The MITRE Corporation. Common Vulnerabilities and Exposures. https://cve.mitre.org/, 2020. [Online; accessed 17-May-2020].

[The20b] The President's National Security Telecommunications Advisory Committee. Nstac report to the president on software-defined networking, 2020.

[TP] Inc. Tor Project. Tor project. https://www.torproject.org/. [Online; accessed 26-Jul-2020].

[TQF+17] Dennis Tatang, Florian Quinkert, Joel Frank, Christian Röpke, and Thorsten Holz. Sdn-guard: Protecting sdn controllers against sdn rootkits. In *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 297–302, 2017.

[Tre18] TrendMicro. Over 200,000 MikroTik Routers Compromised in Cryptojacking Campaign. TrendMicro, Aug. 3, 2018.

[VC19] A. Vetterl and R. Clayton. Honware: A virtual honeypot framework for capturing cpe and iot zero days. In *2019 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–13, 2019.

[Vir] VirusTotal. VirusTotal. https://www.virustotal.com/gui/home/upload. [Online; accessed 10-April-2022].

[VJ19] R. Vishwakarma and A. K. Jain. A honeypot with machine learning based detection framework for defending iot based botnet ddos attacks. In *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 1019–1024, April 2019.

[VKTH19] Vijay Varadharajan, Kallol Karmakar, Uday Tupakula, and Michael Hitchens. A policy-based security architecture for software-defined networks. *IEEE Tran. on Information Forensics and Security*, 14(4), 2019.

[VPDCB21] Ismael Amezcua Valdovinos, Jesús Arturo Pérez-Díaz, Kim-Kwang Raymond Choo, and Juan Felipe Botero. Emerging ddos attack detection and mitigation strategies in software-defined networks: Taxonomy, challenges and future directions. *Journal of Network and Computer Applications*, 187:103093, 2021.

[VSCM16] E. Vasilomanolakis, S. Srinivasa, C. G. Cordero, and M. Mühlhäuser. Multi-stage attack detection and signature generation with ics honeypots. In *IEEE/IFIP Network Operations and Management Symposium*, pages 1227–1232, 2016.

[VTG19] B.G.M. Vicente, J. Triunfante, and B. Gelera. Cve-2019 Exploited, Used to Deliver Monero Miner. TrendMicro, June 20, 2019.

[Wad11] Susan Marie Wade. SCADA Honeynets: The attractiveness of honeypots as critical infrastructure security tools for the detection and analysis of advanced threats. Master's thesis, Iowa State University, 2011.

[Wag11] G. Wagener. *Self-Adaptive Honeypots Coercing and Assessing Attacker Behaviour*. PhD thesis, Institut National Polytechnique de Lorraine - INPL, 2011.

[Wan17] M. Wang. Thingpot. https://github.com/Mengmengada/ThingPot, 2017. [Online; accessed 14-May-2020].

[Wat07] David Watson. Honeynets: a tool for counterintelligence in online security. *Network Security*, 2007(1):4–8, 2007.

[WH15] Kyle Wilhoit and Stephen Hilt. The GasPot experiment : Unexamined perils in using gas-tank-monitoring systems. In *Black Hat USA*, 2015.

[Wil13a] Kyle Wilhoit. The SCADA That Didn't Cry Wolf Who's Really Attacking Your ICS Equipment? (Part 2). 2013.

[Wil13b] Kyle Wilhoit. Who's Really Attacking Your ICS Equipment? 2013.

[Wir] Wireshark. Wireshark. https://www.wireshark.org/. [Online; accessed 10-April-2022].

[WRD+15] Michael Winn, Mason Rice, Stephen Dunlap, Juan Lopez, and Barry Mullins. Constructing cost-effective and targetable industrial control system honeypots for production networks. *International J. of Critical Infrastructure Protection*, 10:47 – 58, 2015.

[WSK18] M. Wang, J. Santillan, and F. Kuipers. Thingpot: an interactive internet-of-things honeypot. *Computing Research Repository*, abs/1807.04114, Jul 2018.

[WW19] H. Wang and B. Wu. Sdn-based hybrid honeypot for attack capture. In *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pages 1602–1606, 2019.

[XCX18] Feng Xiao, Enhong Chen, and Qiang Xu. S7commtrace: A high interactive honeypot for industrial control system based on s7 protocol. In *Information and Communications Security*, pages 412–423, Cham, 2018. Springer International Publishing.

[XYH+19] Junfeng Xie, F. Richard Yu, Tao Huang, Renchao Xie, Jiang Liu, Chenmeng Wang, and Yunjie Liu. A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges. *IEEE Communications Surveys Tutorials*, 21(1):393–430, 2019.

[YKYZ21] Zhiyuan Yu, Zack Kaplan, Qiben Yan, and Ning Zhang. Security and privacy in the emerging cyber-physical world: A survey. *IEEE Communications Surveys Tutorials*, pages 1–1, 2021.

[Zec03] Marko Zec. Implementing a clonable network stack in the freebsd kernel. In *Proceedings of the FREENIX Track: USENIX Annual Technical Conference, June 9-14, San Antonio, Texas, USA*, pages 137–150, 2003.

[ZGQA19] Mohammad-Reza Zamiri-Gourabi, Ali Razmjoo Qalaei, and Babak Amin Azad. Gas what? i can see your gaspots. studying the fingerprintability of ics honeypots in the wild. In *Proceedings of the Fifth Annual Industrial Control System Security (ICSS) Workshop*, page 30–37. ACM, 2019.

[Zho19] Y. Zhou. Chameleon: Towards adaptive honeypot for internet of things. In *Proceedings of the ACM Turing Celebration Conference - China*, May 2019.

[ZKF19] L. Zobal, D. Kolář, and R. Fujdiak. Current state of honeypots and deception strategies in cybersecurity. In *2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pages 1–9, 2019.

[ZLZ+19] Yanling Zhao, Ye Li, Xinchang Zhang, Guanggang Geng, Wei Zhang, and Yanjie Sun. A survey of networking applications applying the software defined networking concept based on machine learning. *IEEE Access*, 7, 2019.

[ZQ17] C. Zhao and S. Qin. A research for high interactive honepot based on industrial service. In *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, pages 2935–2939, 2017.

[ZZZ+19] W. Zhang, B. Zhang, Y. Zhou, H. He, and Z. Ding. An iot honeynet based on multi-port honeypots for capturing iot attacks. *IEEE Internet of Things Journal*, pages 1–1, 2019.

VITA

JAVIER R. FRANCO

2020	B.S., Internet of Things Florida International University Miami, Florida
2022	M.S., Computer Engineering Florida International University Miami, Florida

PUBLICATIONS AND PRESENTATIONS

Javier Franco, Ahmet Aris, Berk Canberk, A. Selcuk Uluagac, "A Survey of Honeypots and Honeynets for Internet of Things, Industrial Internet of Things, and Cyber-Physical Systems", IEEE Communications Surveys & Tutorials, 23(4):2351-2383, 2021

Alvi Ataur Khalil, Javier Franco, Imtiaz Parvez, A. Selcuk Uluagac, Mohammad Ashiqur Rahman, "A Literature Review on Blockchain-enabled Security and Operation of Cyber-Physical Systems", IEEE Computer Software and Applications Conference (COMPSAC), June 2022

(Under Review) Javier Franco, Ahmet Aris, Leonardo Babun, A. Selcuk Uluagac, "S-Pot: A Smart Honeypot Framework with Dynamic Rule Configuration for SDN", IEEE Global Communications Conference (GLOBECOM), December 2022

(Under Review) Javier Franco, Abbas Acar, Ahmet Aris, A. Selcuk Uluagac, "Forensic Analysis of Cryptojacking in Host-based Docker Containers Using Honeypots", IEEE Global Communications Conference (GLOBECOM), December 2022