University of Zurich[UZH]

BACHELOR THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

Design and Implementation of an MTD Strategy Selection Agent using IoT Platform Metrics

Nicolas Huber
Zürich, Urdorf
Student ID: 16-936-205

Supervisor: Dr. Alberto Huertas Celdrán, Jan von der Assen
Date of Submission: September 14, 2022

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

Bachelor Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
URL: http://www.csg.uzh.ch/

Abstract

IoT-Technologien haben in den letzten Jahren einen kontinuierlichen Aufschwung erlebt, der zu unzähligen Arten von vernetzten Geräten geführt hat. Mit dem Aufblühen dieser Technologien und insbesondere dem Einsatz von IoT-Sensoren hat sich das tägliche Leben in vielerlei Hinsicht verbessert und neue Möglichkeiten eröffnet, wie etwa die IoT-Crowdsensing-Plattform ElectroSense. Die Tatsache, dass diese ressourcenbeschränkten Geräte anfällig für Cyberangriffe sind, wirft die Frage auf, wie ein geeignetes und zuverlässiges Sicherheitssystem bereitgestellt werden kann. Die Anwendung des MTD-Paradigmas legt den Einsatz spezifischer Verteidigungsmechanismen nahe, um den Schaden durch bösartige Software zu mindern. Für den eigentlichen Entscheidungsprozess, d. h. unter welchen Umständen eine entsprechende Gegenmaßnahme ausgelöst wird, werden oft komplexe Technologien wie maschinelles Lernen, Spieltheorie oder evolutionäre Algorithmen auf der Grundlage von systemfremden Metriken eingesetzt. Diese Arbeit leistet einen Beitrag zur Forschung, indem sie einen MTD Strategy Selection Agent *StraSelA* vorschlägt, der seine Entscheidungen anhand eines einfachen Regelwerks, auch Policy genannt, auf der Basis von Systemmetriken trifft. Zu diesem Zweck wurden sieben Schadprogramme aus den Kategorien Command and Control, Ransomware und Rootkit sowie ein ElectroSense-Fernsensor auf einem Raspberry Pie 4 untersucht. Die zugrundeliegenden Metriken wurden zunächst durch eine systematische Literaturrecherche über Malware definiert. Anschließend wurde das Geräteverhalten aufgezeichnet und mittels Datenanalyse der einzelnen Metriken wurden spezifische Regeln für die Ausführung von Abwehrmaßnahmen erstellt und experimentell feinjustiert. Die Architektur von *StraSelA* wurde definiert und anschließend so implementiert, dass die Software zusammen mit der synthetisierten Policy-Datenbank auf dem IoT-Gerät auftretende Malware erstens erkennen und zweitens eine geeignete Gegenmaßnahme auslösen kann. Nach Auswertung verschiedener Metriken wie Erkennungsrate und Overhead wurde gezeigt, dass es möglich ist, ein System mit einem einfachen und ressourceneffizienten Entscheidungsprozess auf der Grundlage von Systemmetriken vor Angreifern zu schützen. Somit wird der Mehrwert des einfachen Strategieauswahlalgorithmus von *StraSelA* bestätigt. Abschließend sind alle Datensätze, die während der Policy-Synthese und der Evaluation verwendet wurden, für weitere Forschungen frei verfügbar.

IoT technologies have experienced a continuous upswing in recent years, resulting in countless types of networked devices. With the flourishing of these technologies and especially the use of IoT sensors, daily life has been improved in many ways and brought new opportunities, such as the IoT crowdsensing platform ElectroSense. The fact that these resource-constraint devices are vulnerable to cyberattacks raises the question of how to

provide a suitable and reliable security system. Using the MTD paradigm suggests deploying specific defense mechanisms to mitigate malicious software damage. For the actual decision process, i.e., under which circumstances a corresponding countermeasure is triggered, often complex technologies such as machine learning, game theory, or evolutionary algorithms are used based on non-system metrics. This work contributes to the research by proposing an MTD Strategy Selection Agent *StraSelA* that makes decisions using a simple set of rules, also called policy, based on system metrics. For this purpose, seven malware from Command and Control, Ransomware, and Rootkit, as well as an ElectroSense remote sensor running on a Raspberry Pie 4, were considered. The underlying metrics were initially defined in systematic literature research about malware. Subsequently, the device behavior was recorded, and specific rules for executing defense measures were created through data analysis of the individual metrics and then experimentally fine-tuned. The architecture of *StraSelA* was defined and then implemented so that the software, together with the synthesized policy database on the IoT device, can firstly detect any malware that occurs and secondly trigger a suitable countermeasure accordingly. After evaluating various metrics, such as detection rate and overhead, it was shown that it could protect a system from attackers with a simple and resource-efficient decision-making process based on system metrics. Thus, the added value coming from the simple strategy selection algorithm of *StraSelA* is confirmed. Finally, all data sets used during the policy synthesis and evaluation are freely available for further research.

Acknowledgments

First and foremost, I would like express my gratitude to my two supervisors Dr. Alberto Huertas Celdrán and Mr. Jan von der Assen. Thanks to their expertise, support and valuable inputs, I was able to successfully complete my thesis.

Furthermore, I would like to thank Prof. Dr. Burkhard Stiller who gave me the opportunity to do this bachelor thesis at the Communication Systems Group (CSG). This enabled me to gain exciting insights into the fields of activity of the CSG.

Finally, I would like to express my appreciation to close friends and my family who have supported me throughout this time.

Contents

Chapter 1

Introduction

Technological progress enhances the connections and the number of devices in our digital world. By 2025, it is estimated that around 55.7 billion devices will be interconnected worldwide, whereas about 75% of them will be connected to the Internet of Things (IoT) devices [1]. This growing number of IoT devices has an increasing impact on our dealing with technology [2], and these devices are present in various application areas [3].

Undoubtedly, these new devices offer numerous advantages. Unfortunately, IoT devices also provide an additional attack surface for cyberattacks [4]. Not only are IoT devices the targets, but they also have security issues and vulnerabilities [5]–[7]. At the same time, the number of attacks targeting such devices is also increasing [8]. The question arises of how IoT devices can be successfully protected.

In 2009 a novel method called Moving Target Defense (MTD) was proposed, which differs from conventional approaches: Instead of trying to build a perfectly secure system, the objective is to mutate the attack surface [9]. MTD was proposed to deal with the fact that static systems are vulnerable due to the attacker's asymmetric information advantage [9].

Research has shown that MTD is a promising approach to making IoT devices more secure [10]–[12]. Several studies have already proposed many different MTD techniques to mitigate attacks, introducing additional complexity for the attacker [10]–[13]. Since a whole spectrum of malware exists, each follows a unique attack behavior. It is crucial to match the defense mechanism to the attack pattern. Therefore, this thesis aims to develop a strategy selection agent for IoT devices to protect against multiple attack types. This agent includes a set of metrics and a policy. The following chapter outlines the motivation for such an agent, defines the scope, and describes the structure of this thesis.

1.1 Motivation

Using MTD to protect the IoT device has been proven successful [10]–[12]. Moreover, using an agent that consequently deploys the optimal MTD technique according to the attack type sounds promising. However, the following difficulties appear:

First, a set of metrics consisting of different system parameters that can be successfully used as indicators for a policy has to be defined. These metrics should reflect the behavior of various malware. Second, a working policy that triggers the optimal MTD technique according to the attack type has to be proposed. For this purpose, the rules of the policy must be adapted to the actual malware samples. This policy is part of the decision-making mechanism, which also includes a strategy selection agent that uses this policy, which forms the third point. The agent and its embedded policy are responsible for detecting a compromised system and consistently deploying the MTD technique that mitigates the malware. Finally, an adequate method to evaluate the agent's performance has to be outlined. It is probably advisable to consider several dimensions of the agent.

1.2 Description of Work

To contribute to the aforementioned challenges, this thesis aims to provide the design and implementation of an MTD strategy selection agent. In order to do that, the main contributions of this work are the following:

First, an in-depth analysis of the state-of-the-art MTD techniques and current malware types and a brief review of general IoT security. This will provide the necessary terms and fundamentals to understand this thesis. Second, the knowledge of the research phase will be processed to create a taxonomy of the considered malware to define a set of system metrics on the IoT device. This step is crucial for further procedure since it is the prerequisite for creating the monitoring script. The development of this script, as well as the necessary data collation and the accompanying data visualization scripts, constitute the third point. Fourth, the design and implementation of an MTD strategy selection agent and a policy. The agent and policy synthesis implementation is carried out in an expert-based, data-driven approach. Finally, the evaluation of the project in real life scenario, considering a Raspberry Pi 4 in the ElectroSense cluster.

1.3 Thesis Outline

The rest of this thesis is organized as follows: Chapter 2 conveys the necessary background knowledge regarding MTD techniques, the functionalities of the considered malware types, and IoT security. Furthermore, this chapter briefly explains some of the state-of-the-art MTD deployment mechanisms and introduces the ElectroSense service. Related work is captured in Chapter 3 in which different policy selection methods are investigated and compared. At the end of Chapter 3, Table 3.1 shows the differences between various approaches. Starting with introducing the concrete use case of this thesis, Chapter 4 also covers the high-level solution architecture and explains the methodology. Chapter 5 covers the implementation of the policy and its synthesis, as well as the details of the MTD strategy selection agent. A set of metrics is proposed to understand the former after an in-depth study of different malware types. The application of the MTD strategy selection agent and the proposed policy in other use cases with the MTDFramework that

provides a set of MTD solutions [14] is provided in Chapter 6. In addition to that, it also covers the evaluation of the agent. The final chapter, Chapter 7 summarizes the results, concludes, and provides an outlook on future work.

Chapter 2

Background

The following passage introduces IoT Security, and the final section briefly overviews the crowd-sensing platform ElectroSense. First, a brief summary of the suitable malware attack types and the consequences for the victim system is made. Second, an introduction to IoT security and the related challenges are provided. Third, the paradigm of MTD is explained, the differences to conventional security techniques are outlined, various MTD techniques for IoT devices are introduced as well as the different policy deployment mechanisms.

2.1 Malware

Command and Control (C&C) attacks are often launched by so-called botnets, an army of infected devices that perform malicious actions such as participating in distributed denial of service (DDoS) attacks, completing data theft, or distributing ransomware [15]. IoT devices can also be targeted explicitly by botnets, as the research from [8] has shown. The good news, however, is that there have also been studies that have found specific countermeasures, for instance, MTD techniques that mitigate the reconnaissance phase of an IoT botnet, i.e., the time when the botnet tries to spy on the victim in order to obtain important information [16]. An example of a botnet attack is the Mirai malware: Mirai launched 2016 a massive attack on the Domain Name System (DNS) service Dyn with a botnet DDoS from countless infected IoT devices that caused up to 1.5Tbps traffic [17].

Ransomware is digital extortion, where first, the sensitive and or valuable data of the victim is encrypted by the software, and then a ransom has to be paid to get a key to unlock the data [18]. Various studies estimate that the threat posed by ransomware to IoT devices, in particular, is quite real [18], [19]. Moreover, [20] showed that the industrial internet of things (IIoT), an extension of IoT, is notably vulnerable to ransomware attacks and even provides a proof of concept (POC) ransomware and a plethora of countermeasures.

A Rootkit hides and acts unseen and enables the attacker to get privileged access to the victim [21]–[24]. To gain this camouflage, rootkits hijack system commands "*ls*, *ps*"

to make the unsuspecting user believe that everything is in the usual order [24], [25]. Further, according to [24], [26] rootkits can be classified into at least, five categories based on their privilege level. The LoJax rootkit demonstrates that once a rootkit is installed, it can be used to download, receive and execute malicious commands from a C&C server [22]. Regarding the detection of rootkits, it remains to be mentioned that [25] found an alternative rootkit detection method for resource-limited IoT devices to the common non-deterministic ML-based approaches. Figure 2.1 illustrates the different functionalities per malware family that this thesis is considering.

Reconnaissance attacks aim to gather intelligence about the target system [8], [27], [28]. Also called scanning attacks, reconnaissance attacks can be performed before the attack to increase the chances of a successful attack [8], [27], [28]. Network Mapper (Nmap) [29] is a standard tool for performing reconnaissance attacks [28]. For this thesis, there is any specific reconnaissance attack considered. However, this paragraph was added anyway since the attack type is highly relevant for IoT devices: Many MTD studies analyze reconnaissance attacks in order to develop and evaluate MTD counter mechanisms [30]. [16], [31] investigated the MTD techniques to counteract the scanning phase of botnets and [32] did research on IP address (IP) randomization to mitigate reconnaissance attacks.

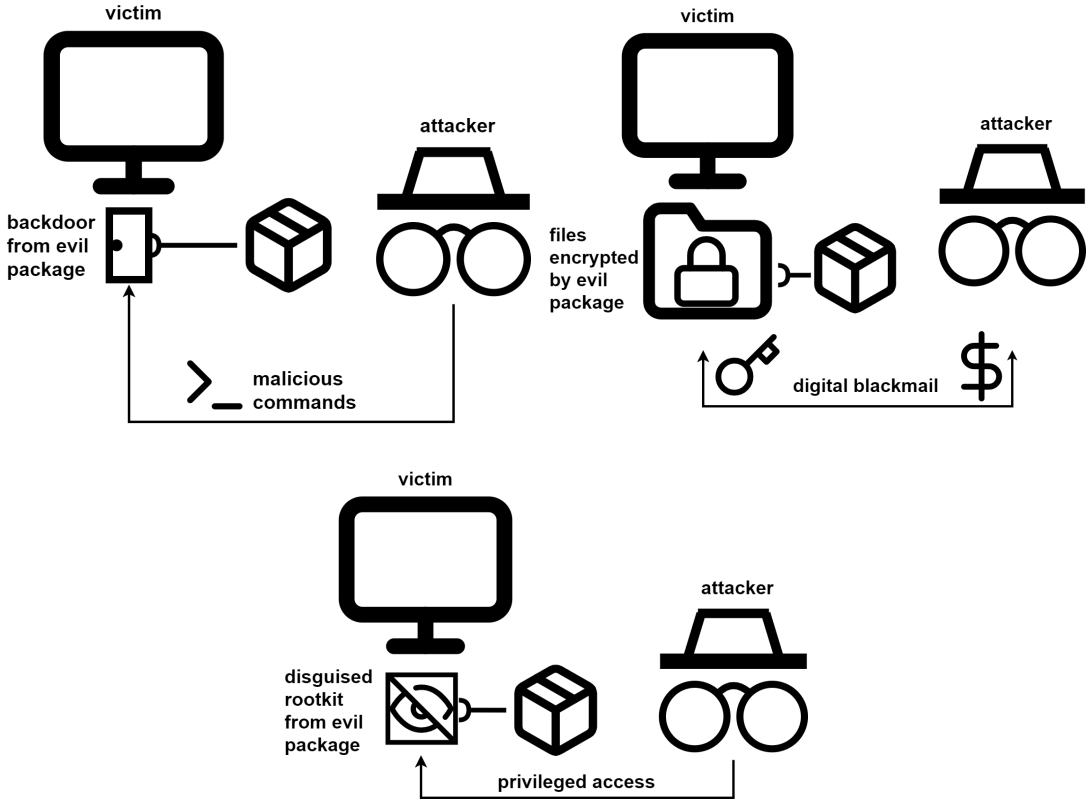

Figure 2.1: The functionality of each considered malware family

2.2 IoT Security

IoT security is associated with difficulties and challenges. First, during the development process of IoT devices, features like usability and resource efficiency are more prioritized than security aspects [33]. Second, existing security approaches and mechanisms for non-IoT devices cannot be simply transferred and applied to IoT domain [33], [34]. Finally, resource-constrained IoT devices may require specific lightweight security solutions [33], [35]: Cryptography solutions that need a lot of computational power are difficult to implement on IoT devices regarding the trade-off between security and performance [36].

2.3 Moving Target Defense

2.3.1 MTD Paradigm

One of the main issues in cybersecurity is asymmetric time advantage for the attacker that is compromising a static system [37], [38]. MTD mainly aims to change a defender system's attack surface to increase the attacker's uncertainty and effort [38]. MTD's paradigm of a constantly changing system surface is achieved by changing components, sometimes called moving parameters, of the system [2]. The specific setting and way the moving parameter is mutated represents an instance of the MTD paradigm and is called MTD technique [2]. Table 2.1 briefly illustrates the differences between traditional and MTD cybersecurity approaches.

Table 2.1: Comparison between traditional and MTD cybersecurity principles

Aspect	**Traditional Security**	**MTD**
Nature of the system	Static [37]	Constantly changing [30], [37]–[39]
Mode of operation	Find and eliminate threats [30]	Shifting attack surface [30], [38]
Method	Cycle orchestrated by a policy: Protection, Detection, Response and Recovery[38]	2 modes: Reactive and active defense [38]
Examples of protection mechanisms	Authentication, Cryptography, Firewall [38]	IP shuffling [7], [10]–[12], [40], [41], Port shuffling [42] Code diversification [43]
Goal	Build a seamless, perfectly safe system [30]	Increase effort for attackers [38]

According to [38], each MTD technique consists of three elements regarding the element, or as described above, moving parameter, that is, "moved": WHAT, HOW, and WHEN. WHAT to move specifies the moving parameter. HOW to move defines the procedures that mutate the moving parameter. WHEN to move determines when the mutation is triggered.

The taxonomy proposed by [30] concerning the WHAT characteristics of MTD techniques consists of Application-, Operating System-Host- (OS), Virtual Machine-Instance- (VM), VM Manager-, and Hardware-Layer. Examples of moving parameters are, for instance IP addresses [7], [10]–[12], [40], [41], [44]–[48], port numbers [37], [42] or program code [37], [43].

The HOW of existing MTD techniques was divided into three different categories by [39]: Shuffling, diversity, and redundancy. Shuffling exchanges the moving parameter according to a specific schema. Diversity introduces a new component with the same functionality but a different realization. Redundancy introduces one or more imitations, so-called "replicas", for a given component. A widespread example of shuffling is IP address shuffling, which was proposed several times [7], [10]–[12], [40], [41].

[30], [39], categorized MTD techniques regarding the WHEN based on their temporal aspect: reactive, proactive, and hybrid-based approaches. Reactive means that the MTD technique is triggered in response to a detected attack. Proactive means that a predefined time interval, which might be random, repeatedly triggers an MTD technique. Hybrid-based approaches use both techniques as mentioned earlier at the same time, for instance starting an MTD on a fixed time interval but also when an intruder was detected.

Table 2.2 summarizes the fundamental aspects and categories of an MTD defense system.

Table 2.2: Key aspects of MTD

WHAT	**HOW**	**WHEN**
Application Layer	Shuffling	Time-based
OS-Host	Diversity	Event-based
VM-Instance	Redundancy	Hybrid
Virtual Machine Manager		
Hardware		

2.3.2 MTD on IoT

The two surveys from [30] and [49] regarding MTD state that MTD is a valid option to increase the security level of IoT devices. [30] studied the current state-of-the-art MTD technologies and concluded that mostly shuffling- and diversity-based approaches are used for IoT devices. For this bachelor thesis, systemic research was conducted with the same approach as [2], whereas the search string 2.2was used. "iot" refers to the term "IoT" but with a slightly different spelling. The search was done and completed in March 2022. The following five sources were used: ACM, IEEE, Springer, Wiley, and ScienceDirect. Table 2.3 summarizes the search requests, and Table 2.4 represents the result. Table 2.5 was created as a byproduct of the non-systematical malware research phase that contains advanced MTD methods used on IoT devices.

("moving target defense" OR "moving target defense")
AND
("internet of things" OR "iot")

Figure 2.2: Search query for the systematic MTD research

Table 2.3: Search results

Date	Platform	Filter	Results
6.3.2022	ACM	research article	29
6.3.2022	IEEE	conferences, journals	31
6.3.2022	Springer	article	24
6.3.2022	Wiley	journals	15
6.3.2022	ScienceDirect	review articles research articles	95

Table 2.4: Systematic IoT MTD Research

Solution	Year	Attack types	Technique	How	When
[12]	2017	RA, DoS	S	6HOP (Port hopping + IP shuffling)	Regular time interval
[41]	2019	RA	S	IP shuffling	Dynamic (lease time)
[50]	2019	RA, DDoS	S, D	Honeypots (Gateway diversification)	Regular time interval
[17]	2019	DDoS	S, R	IP shuffling SDN-based honeypots	Random time interval
[51]	2019	Ransomware	S	Changing file extensions	Only once
[52]	2019	DnS Attacks	S	Port hopping	Dynamic
[48]	2021	RA, DDoS	S	IP shuffling	Dynamic

RA=Reconnaissance Attack, S=Shuffling, D=Diversity, R=Redundancy

2.3.3 MTD Deployment Mechanisms

This section focuses on deployment mechanisms in MTD. The main task of a defender system using MTD is to achieve the best possible chance to the attack surface through an innovative and deliberate strategy. According to [30], three types of techniques can be used to solve this crucial MTD selection problem: Game theoretic, genetic algorithms, and machine learning (ML) approaches. [54] proposed a game theoretical using a Stackelberg game between defender and attacker. The defender takes the role of a leader and

Table 2.5: Non-systematic IoT MTD research during the malware research phase

Solution	Year	Attack types	Technique	How	When
[10]	2014	DoS, MitM	S	MT6D: IP shuffling	regular time interval
[27]	2019	RA	S, D, R	Network Function Virtualization	hybrid: regular time interval packet threshold
[31]	2021	DDOS	S, R	IP shuffling Replica server shuffling	regular high-frequency interval
[32]	2015	RA	S	Random Host Address Mutation: IP, MAC address and domain name shuffling	frequently
[16]	2019	RA	S, R, D	IP shuffling Replica application server	regular time interval
[53]	2019	RA	S, R	Replica application server Proxy Shuffling	Random time interval
[7]	2017	RA	S	μMT6D: IP shuffling	not specified

RA=Reconnaissance Attack, MitM=Man-in-the-middle attack
S=Shuffling, D=Diversity, R=Redundancy

offers an IP shuffling policy, whereas the follower, the attacker, determines the scan rate. Performing a simulation, they showed an equilibrium for both leader and follower and that an optimal policy for the IP randomization can be derived. In 2014, [55] proposed a genetic algorithm that generated many different system combinations, which are then implemented and assessed to find the best configuration. ML algorithms can also be used, as [56] demonstrated: They solved Bellman Equations to solve a Markov Decision Process for the optimal policy deployment.

2.4 ElectroSense

ElectroSense [57] is used as the real-world use-case in this thesis. ElectroSense is a crowd-sensing service that analyzes spectrum data from a cluster of IoT devices. The aggregated data is accessible to everybody in real-time available on their web page and via application programming interface (API). ElectroSense recommends a setup consisting of Raspberry Pis sensors equipped with an antenna and a software-defined radio (SRD) dongle. From a technical perspective, the Raspberry Pie's ElectroSense client is an ideal way to investigate cyber security.

Chapter 3

Related Work

The following chapter is dedicated to existing research on MTD selection methods. First, the current state-of-the-art policy selection methods are briefly discussed and analyzed regarding the metrics used, platform, malware examined, and evaluation model. The second part summarizes the studies and compares them to identify similarities and differences.

3.1 MTD Selection Methodologies

This section elaborates on some current policy selection methods, i.e., the process that determines which point in time which MTD technique will be deployed. In order to justify the development of an MTD strategy selection agent, the current modern approaches are analyzed. Based on these fundamentals, the aim is to identify and adapt suitable mechanics to the use case of this thesis.

[58] did research on the signal game and proposed an MTD policy selection approach that takes attack and defense metrics, a set of strategies for both parties, and calculates the optimal defense strategy based on an equilibrium solution method. In this signal game, the defender acts as the sender and the attack represents the receiver, which means the attacker reacts to the generated signal of the defender. The available MTD techniques are IP and Port shuffling and sophisticated service architecture mutation. Further, they considered two scanning attacks strategies: code injection and remote buffer overflow. For evaluation purposes, they simulate a network topology where a server is attacked, and no IoT is present.

In their work, [59] discussed a policy selection method that uses differential game combined with the Markov Decision Process (MDP) to counterbalance both time continuity and stochastic under the use of the strategy sets and metrics for attacker and defender. Instead of limiting themselves to one attack, they examined various attack and defense strategies such as a Structured Query Language (SQL) injection or IP hopping. Based on their intensity, the attack strategies were divided into the groups low, medium, and high. In contrast, the defense strategies were grouped by their taxonomy by data, network, software, and platform. They used MATLAB to simulate a network structure that,

among other things, included a business network, a connection network, and a network defense facility but no IoT devices to evaluate their proposed decision algorithm.

Existing game-theoretic strategy selection approaches have been extended by [60]: They combined MDP with game theory to decide on the defense strategy with a novel multi-phase model. The result was Markov game MTD (*MG-MTD*), a method for finding the optimal MTD strategy. The decision is based on a set of defense strategies that include a hopping element (WHAT), hopping strategy (HOW), and hopping period (WHEN). Furthermore, the authors had attack strategies that define general attack operations such as scanning or information theft, revenue sets for both attacker and defender that represent the cost for the operations, a network transition table, and a criterion function to evaluate the considered strategies. By solving a nonlinear problem, *MG-MTD* can be evaluated to find the optimal defense strategy eventually. To assess their work, they conducted a Simulation case study with a standard network topology similar to [61], [62] that includes two servers, one database, and one client but no IoT device.

Another game theoretic approach was described and developed by [63]. Markov robust game, which combines MDP with robust game. The resulting MTD strategy selection method Markov robust game model (*MRG-MTD*) has some similarities to *MG-MTD* proposed by [60]: (*MRG-MTD*) Also evaluates the optimal defense strategy by solving a nonlinear problem and requires a set of defense and attack strategies and the corresponding revenue sets, a network transition table and an objective criterion function. They could successfully validate their results by doing a well-known network topology simulation that does not consider any IoT devices [61].

Based on the idea of a genetic algorithm [64] proposed *Joint Defense*. The *Joint Defense* strategy selection approach entails changing three elements of the defender system by using a genetic algorithm operating on chromosomes, representing different defense strategies that consist of genes. In contrast, each gene describes a single mutation parameter of the defense system. In contrast to most of the other research, they focused on selecting an optimal strategy under multiple attacks, showing that the proposed genetic algorithm, given an attack and defense metric, can find numerous mutation parameters of the system to mitigate the attack. The algorithm evaluates different metrics for the defender and attacker, such as cost and payoff, as well as the defense efficiency. Like others, they used simulation to evaluate their findings on a desktop computer with MATLAB.

FlipIt, a new game theoretical approach used as a decision-making algorithm during the policy selection process to trigger the optimal MTD strategy, was introduced by [65]. *FlipIt* uses network parameters as well as a set of attack and defense strategies. The collection of attack strategies covers common attack patterns such as scripting (conservative attack strategy) and transmitted data manipulation (aggressive attack strategy). The set of MTD strategies includes techniques that change the attack- and the exploration surface. The threat came from highly skilled advanced persistent threat (ATP) attacks. For the evaluation, the software defined network (SDN) simulation test platform Mininet and Ryu controller was used to emulate servers with 2.6GHz CPU, 8GB RAM, and a 500GB hard disk, which is far beyond IoT capacities.

In contrast to the other studies in this section, [27] developed a proof-of-concept MTD framework to mitigate DDoS, more specifically, the reconnaissance phase of Crossfire

attacks, which belong to the group of flooding attacks, that uses a simple algorithm for decision making. A Crossfire attack aims to impair, or at worst completely cut off, the connection to chosen servers [66]. The proposed algorithm triggers advanced SDN-based MTD methods through the threshold-based evaluation of the system metric number of packages from the *traceroute* command or expired timeout timer. Subsequently, one of three MTD techniques is randomly selected and deployed. They used Mininet to simulate a virtual network on a single, commercially available computer for the evaluation.

[67] have developed a policy selection method that evaluates a system metric, a CPU threshold, to control the deployment of an MTD defense similar to [27] technique to defend against DoS attacks by thwarting the reconnaissance phase. Thereby, shuffling, diversity, and redundancy based MTD methods were triggered as soon as a the system exceeds 40% CPU utilization to mitigate incoming attacks. They used Amazon Web Service (AWS) Elastic Compute Cloud (EC2) instances as servers and proxies for simulations to validate their work.

Table 3.1: Comparison of different MTD policy selection methods

Solution	**MW**	**IoT**	**Policy Selection Method**	**Parameters**	**System Metrics**	**EM**
[60], 2017	V	No	Markov game	Strategy Sets Revenue Sets Transition table Criterion function	None	S
[64], 2019	V, M	No	*Joint Defense*	Attack Metrics Defense Metrics	None	S
[58], 2020	V	No	Signal game	Attack Metrics Defense Metrics Strategy Effectiveness	None	S
[59], 2020	V	No	Dynamic Markov Differential Game	Attack Metrics Defense Metrics Strategy Sets	None	S
[63], 2019	V	No	Markov robust game (*MRG-MTD*)	Strategy Sets Revenue Sets Transition table Criterion function	None	S
[65], 2021	ATP	No	Differential Game	Strategy Sets	None	S
[27], 2021	Crossfire attack	No	Simple Algorithm	Timeout timer Network packages	Packages	S
[67], 2021	DoS	No	Simple Algorithm	CPU	CPU	S

MW=Malware, EM=Evaluation Method
V=Various, M=Multiple, S=Simulation

3.2 Discussion

Table 3.1 compares the discussed approaches. To sum everything up, the following key findings are remarkable:

Generally, most papers use a game theoretic approach to select the optimal MTD technique. These sophisticated approaches often make their decision-based set sets of strate-

gies or attack and defense metrics, but they do not consider any system metrics. These studies have successfully shown that using the previously mentioned strategies or metrics, an MTD strategy selection method can be implemented. In contrast, only two studies used a simple, threshold-based policy selection algorithm that requires a system metric. Overall, only limited research uses a system threshold to trigger the appropriate MTD technique.

The research work studied only used simulation for evaluation. In addition, the focus of these studies is seldom the actual implementation of such a strategy selection agent but more on the selection process and the associated non-linear optimization problem. As far as I can tell, little focus has been placed on the concrete development, implementation, and experimental evaluation of MTD policy selection methods.

Most research considers various malware behavior but not concrete malware examples for the environment. In addition to that, the studied papers do not consider the IoT domain: To the best of my knowledge, there are no studies that focus on the policy selection method for the Raspberry Pie.

Chapter 4

Scenario and Solution

The following passage describes the use case this thesis is considering and proposes a solution for the MTD Strategy Selection Agent *MTD StraSelA*. First, this paper describes the scenario concerning the ElectroSense service, and a requirements analysis is presented. Second, the solution's architecture is introduced together with a brief description of the individual components. Finally, the methodology is explained and this project's milestones are described.

4.1 Scenario

The concrete IoT device this thesis considers is the Raspberry Pi 4 Model B. It has a 64-bit ARM architecture-based Broadcom BCM2711 Quad-core processor running at 1.5 GHz and a RAM capacity of 4 GB. In addition, it is equipped with Gigabit Ethernet and a 2.4 GHz / 5.0 GHz WLAN module, both ensuring excellent network connectivity. A Bluetooth module opens up further possibilities, whereas USB ports and an HDMI port complete the entire equipment, which makes the Raspberry Pi a very versatile IoT device.

ElectroSense [57] is a service that uses these small heroes: It provides a Raspberry Pi and an antenna set to participants that can measure electromagnetic waves in their environment. This means the ElectroSense cluster is a network of IoT devices that collect data in a private setting of volunteers. The crux is that ElectroSense does not influence the handling of each user and their Raspberry Pi: If one Pi was successfully contaminated, this could potentially infect other devices in the cluster or affect the transmission to the ElectroSense central server.

Services like ElectroSense are fully aware of potential cyberattacks. The question is, how can they identify and mitigate potential attacks to protect their cluster? As already pointed out in Chapter 2.3.2, is it possible to use MTD? A possible inventory of requirements is shown in table 4.1.

The MTD Strategy Selection Agent *MTD StraSelA* is proposed to implement these requirements. The selected strategy is a so-called MTD technique. In this thesis, the

Table 4.1: Requirements

Number	User Story
1	As an ElectroSense operator, I want to minimize the potential threats posed by malware.
2	As an ElectroSense operator, I want to maximize the quality of service.
3	As an ElectroSense operator, I want to have a solution for all devices in the cluster.
4	As an ElectroSense operator, I want to have a solution that does not require any user actions.

following terminology is used: The term MTD technique refers to the actual MTD relevant procedure that the system executes in order to counter malware, such as IP shuffling. The author from [68] proposed in his Bachelor thesis at the University of Zurich (UZH) a framework called *MTDFramework* [14] that consists of multiple MTD techniques. Since one of the main contributions of this thesis is a strategy selection agent, the actual MTD techniques [14] are provided by the *MTDFramework* from [68].

4.2 Architecture

The *MTD StraSelA* consists of three components, which are indicated in Figure 4.1. The Observer component monitors the system metrics and supplies these data to the Policy component in real-time. The Policy component evaluates this data by checking all rules if the current snapshot of system metrics matches a known attack pattern. If the policy detects an attack, the Deployer component triggers the according MTD technique to mitigate the incoming attack. Furthermore, Table 4.2 summarizes each component's core function.

Table 4.2: Agent Components

Component	Function
Observer	Measures defined a set of metrics in real-time Provides information to the Policy component
Policy	Monitors metrics data Triggers Deployer component, if applicable
Deployer	Deploys the required MTD technique

4.3 Methodology

As pointed out in Chapter 2.3.3, the critical component of an MTD selection agent is a set of rules called policy. To define such a set of policies, two key questions can be asked:

- Which *system metrics* reflect the incoming attack in the most reliant way?

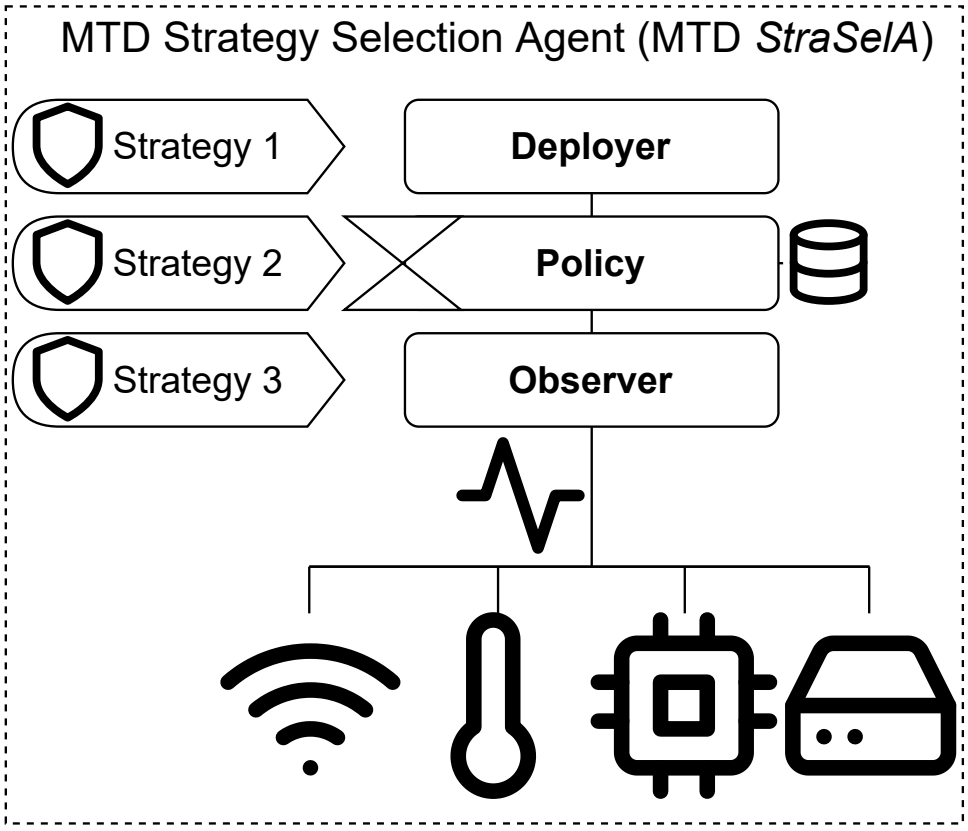

Figure 4.1: The architecture of the MTD strategy selection agent

- Which numerical values, also called *thresholds*, do these metrics have to *either exceed or fall short* to be reliably assigned to a specific attack pattern?

This means we first need to define a set of metrics and then subsequently determine thresholds. Malware behaves differently, so an in-depth malware analysis can shed light on these different patterns. This research investigates the relation between influenced system metrics and attacking malware type for each type considered in this thesis. This thesis considers the following malware: "The Tick" [69], "backdoor" [70], "httpBackdoor" [71], and "BASHLITE" [72] from the C&C category, "BEURK" [73], and "bdvl" [74] as rootkit examples, and finally "Ransomware-PoC" [75].

After obtaining the set of relevant system metrics, a monitoring script can be developed. The script's purpose is to monitor the set of metrics on the Raspberry Pi. Employing the monitoring script, data sets for healthy (non-infected), infected, and cleansed (cured behavior) of the Raspberry Pi can be systematically recorded. This step aims to generate a data set for each type of considered malware plus corresponding MTD, as well as for regular, non-infected behavior like a fingerprint.

The next step includes the actual policy synthesis. To achieve this, data analysis methods are applied. The crucial delivery of this stage is the actual policy, the agent can consult that during the decision process. For this thesis, we consider only simple rules (*e.g.*, $IF \rightarrow THEN$).

The integral part of this thesis is an MTD strategy selection agent *MTD StraSelA*, which is the aim of this next step. Along with the actual implementation of *MTD StraSelA* that uses the gained findings, the agents use the MTD techniques proposed [14] in his MTDFramework.

Finally, an evaluation of the *MTD StraSelA* can be done to assess the quality and performance of the policy. The purpose of this last step is to analyze the findings critically. Table 4.3 briefly summarizes the milestones of this thesis.

Table 4.3: Workflow

Deliverable	**Milestone description**	**Method**
Set of Metrics	Provide a set of system metrics	Research on malware
Data sets	Collect data sets for different behavior	Develop a monitoring script Conduct data acquisition through experiments
Policy	Synthesize a policy	Data-analysis Expert-based policy creation and fine-tuning
Selection agent	Develop strategy selection agent	Develop an MTD strategy selection agent
Evaluation	Evaluate the findings	Conduct data acquisition through experiments Measure performance by data analysis

Chapter 5

Implementation

This chapter elaborates on the process that led to the creation of the set of system metrics and the MTD Strategy Selection Agent *MTD StraSelA*. First, the policy synthesis, starting with the malware research process, is explained in more detail. This step leads to a general understanding of the malicious software and affected system metrics on the victim system. Next, a monitoring script is provided that observes a basic set of system metrics and generated data sets, representing different behavior of the various malware types. The subsequent data analysis was essential to, on the one hand, identify specific system metrics as possible candidates and, on the other hand, to design a system for the rules of the policies. With all this preliminary work, the policy synthesis process can provide policies consisting of the final set of metrics and thresholds. The last section of this chapter finally explains the structure and mode of operation of the MTD Strategy Selection Agent *MTD StraSelA*.

5.1 Policy Synthesis

5.1.1 Malware Research

Considering the given malware, an in-depth study regarding its functionality was conducted: In a non-systematical approach, the behavior of each malware type and especially the influence on the victim system was investigated. The same five sources as in Chapter 2.3.2 were used: ACM, IEEE, ScienceDirect, Springer, and Wiley. This intermediate step aims to systematize the impact of malware on the victim's respective system metrics and record it in a table. Various research reports were consulted, and information was obtained on which system parameters are specifically affected during a malware infection. The result of this stage was the generation of Table 5.1 from concentrated knowledge. The table indicates if the attack pattern of a specific type of malware impacts a system metric or not to find a way to detect the malware eventually effectively. For instance, it is visible that C&C malware does not mainly cause Disk I\O activity such that this system metric is not particularly affected by it (No, not affected). On the other hand,

ransomware will naturally generate a lot of Disk I\O activity. This means the Disk I\O represents a valuable indicator for detecting ransomware (Yes, affected).

Table 5.1: Malware-Metric indicator table

	Observation				
	Processes	**Disk I\O**	**Network connections**	**Network I\O**	**CPU activities**
Attack Type			Affected?		
C&C [76], [77]	Y	N	Y	Y	Y
Rootkit [21]–[24], [26], [78]	Y	Y	Y	Y	P
Ransomware [18]–[20]	Y	Y	N	N	Y
Reconnaissance [8], [16], [27], [28], [31], [32], [53]	N	N	Y	Y	N

Y=Yes, N=No, P=Probably affected

5.1.2 Monitoring Script

After defining which metrics are possible candidates for policy rules, a data collection had to be conducted first to perform data analysis for the actual policy creation eventually. This data collection included data sets showing healthy, infected, and cleansed behavior that had to be recorded. For this purpose, a monitoring script that records a predefined group of metrics for a certain amount of time was created. The versatile system monitoring tool *Dstat* [79] was a perfect choice for this since *Dstat* allows to adjust the monitoring parameters by adding flags to the command. Command 5.1 was used by the monitoring script to obtain 33 system metrics from 8 categories (CPU, Memory, File system, Disk, Network, Transmission Control Protocol (TCP) Connections, Sockets, System, and Processes). The following two tables give the interested reader even more background: Table 5.2 explains each flag of the *Dstat* command, and Table 5.3 lists all proposed system metrics and provides a brief explanation for each.

Listing 5.1: *Dstat* command with flags

```
1    dstat -t --cpu --mem --fs -d --disk-tps -n --tcp --socket
         -y -p -N eth0 --output \$filename \$delay \
       $observations
```

With the completion of the monitoring script, data collection could begin. This stage resulted in 8 data sets: one with healthy behavior and 7 with malicious behavior. The procedure below was followed:

1. Connect to Raspberry Pi (Server) via Secure Shell (SSH)
2. Stop all redundant SSH connections
3. Start *MTD StraSelA*
4. Start the monitoring script

5. Wait for 2 minutes
6. Execute the Malware
7. Wait for 2 minutes (C&C and Rootkit) or 10 seconds (Ransomware)
8. Trigger the MTD technique
9. Wait for 2 minutes
10. Download the generated comma-separated values (CSV) file via SSH

Table 5.2: *Dstat* command: Flags and corresponding metrics

Flag	Description	Metrics	Metric Type
-t	Time/date output		
–cpu	Enable CPU stats	*usr, sys, idl, wai, hiq, siq*	CPU
–mem	Enable Memory stats	*used, buff, cach, free*	Memory
–fs	Enable file system stats	*files, innodes*	File system
-d	Enable I/O transaction stats	*read, writ*	Disk
–disk-tps	Enables I/O transactions stats	*reads, writs*	Disk
-n	Enable Network stats	*resv, send*	Network
–tcp	Enable TCP Connection stats	*lis, act, syn, tim, clo*	TCP Connection
–socket	Enable Socket stats	*tot, udp, raw, frg*	Sockets
-y	Enable Process stats	*int, csw*	Processes
-p	Enable Process statsq	*run, blk, new*	Processes
-N eth0	Specifies the network interface		
-output	Write output to file		
$filename	Specifies the filename		
$delay	Specifies the delay between two observations		
$observations	Specifies the number of observations		

Table 5.3: *Dstat* command: Metric description

Metric	Description	Unit	Category	Proc Path
usr	CPU usage by user processes	%	CPU	/proc/stat
sys	CPU usage by system processes	%	CPU	/proc/stat
idl	Idle CPU usage	%	CPU	/proc/stat
wai	Number of waiting processes	#	CPU	/proc/stat
hiq	Number of hard interrupts	#	CPU	/proc/stat
siq	Number of soft interrupts	#	CPU	/proc/stat
used	Amount of used memory	Bytes	Memory	/proc/meminfo
buff	Amount of buffered memory	Bytes	Memory	/proc/meminfo
cach	Amount of cached memory	Bytes	Memory	/proc/meminfo
free	Amount of free memory	Bytes	Memory	/proc/meminfo
files	Number of allocated file handles	#	File system	/proc/sys/fs/file-nr
inodes	Number of used file handles	#	File system	/proc/sys/fs/file-nr
read	Amount of read bytes on disk	Bytes	Disk	/proc/diskstats
writ	Amount of written bytes on disk	Bytes	Disk	/proc/diskstats
reads	Number of read operations on disk	#	Disk	/proc/diskstats
writs	Number of read operations on disk	#	Disk	/proc/diskstats
recv	Amount of received bytes on eth0	Bytes	Network	/proc/net/dev
send	Amount of received bytes on eth0	Bytes	Network	/proc/net/dev
lis	Number of TCP connections with status "listening"	#	TCP connections	/proc/net/tcp
act	Number of TCP connections with status "established" (active)	#	TCP connections	/proc/net/tcp
syn	Number of TCP connections with status "syn_sent", "syn_receive" or "last_ack"	#	TCP connections	/proc/net/tcp
tim	Number of TCP connections with status "waiting"	#	TCP connections	/proc/net/tcp
clo	Number of TCP connections with status "fin-wait1/2", "close/_wait" or "closing" (closed)	#	TCP connections	/proc/net/tcp
tot	Number of total sockets	#	Sockets	/proc/net/sockstat
tcp	Number of TCP sockets	#	Sockets	/proc/net/sockstat
udp	Number of UDP sockets	#	Sockets	/proc/net/sockstat
raw	Number of RAW (using no protocol) sockets	#	Sockets	/proc/net/sockstat
frg	Number of FRAG sockets	#	Sockets	/proc/net/sockstat
int	Number of interrupts	#	Processes	/proc/stat
csw	Number of context switches	#	Processes	/proc/stat
run	Number of processes with status "running"	#	Processes	/proc/stat
blk	Number of processes with status "blocked"	#	Processes	/proc/stat
new	Number of processes with status "new"	#	Processes	/proc/stat

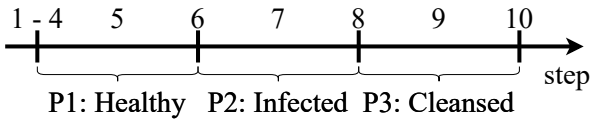

Figure 5.1: Phases during observation

This process was done several times during July and August 2022. Figure 5.1 visualizes the timeline, steps, and corresponding phases during the monitoring period. The final data sets in Table 5.4 form the basis for the subsequent policy creation process.

Table 5.4: Data records as the basis for policy creation

No.	Name	Type
1	00 healthy	Healthy
2	01 httpBackdoor	Command and Control
3	02 backdoor	Command and Control
4	03 The Tick	Command and Control
5	04 BASHLITE	Command and Control
6	05 Ransomware-PoC	Ransomware
7	06 BEURK	Rootkit
8	07 bdvl	Rootkit

5.1.3 Data Analysis and Visualization

The next step consists of data analysis. To gain knowledge from the raw protocols with system metrics entries obtained in the former paragraph, multiple data visualization scripts in the form of Jupyter notebooks were written. First, the CSV data was loaded, preprocessed, and saved as a data frame. Preprocessing entails formatting the timestamp, removing the first line since it represents the average of the current uptime, and removing outliers. Second, the actual visualization in the form of a time series was performed: Each graph plots the system metric value (y-axis) concerning the time (x-axis). The whole duration (x-axis range) amounts to six minutes, totaling 360 data points (1 observation per second). The entire time frame is divided into three phases: The healthy phase at the beginning, the infected phase in the middle, and cleansed phase at the end. Every malware data set splits these three phases equally into two minutes, except for the ransomware, where we have 2 minutes healthy, 10 seconds infected and then 5 minutes and 50 seconds cleansed. This is because the ransomware encryption mechanism is a one-time operation, so we need to shorten the second time frame to detect some anomalies during its operation phase. Since most of the considered data show natural fluctuation, different aggregation methods have been used to see trends and thresholds. In every subplot, there are vertical markers in red and green that either highlight the malware's activation or the MTD technique's execution. The top plot is a visual representation (blue graph) of the metric for the time. The middle plot shows a rolling, non-overlapping average for 10 data points in black that shows stair tread characteristics and, the average over the entire phase in magenta. The bottom plot only shows the average per phase in magenta.

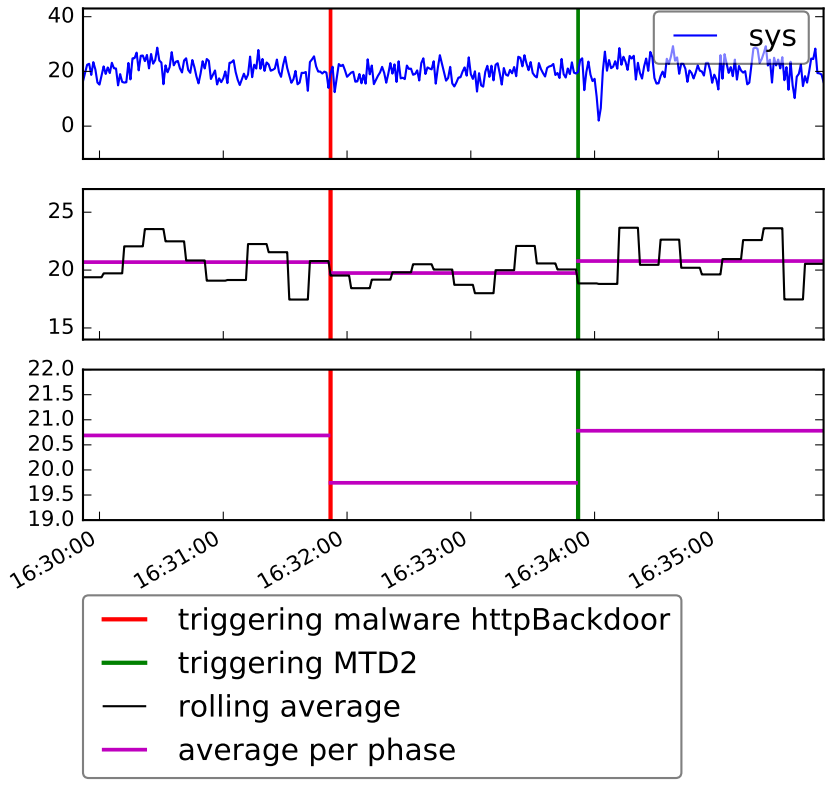

Figure 5.2: Example: Time series of the *sys* metric

A complete example of the system metric *sys* for the C&C malware "httpBackdoor" is shown in Figure 5.2. The main objective of this visualization was to prepare the policy synthesis process and to develop a general understanding of system metrics data.

5.1.4 Structure and Rules of the Policy

After the insights of Section 5.1.1 and Section 5.1.3, the final step of the policy synthesis could be implemented. The object of this synthesis is a defined set of simple $IF \rightarrow THEN$ rules that define the conditions when an MTD has to be deployed. That means the characteristics of these rules are the following:

$$IF \text{ metric } M \text{ has } CONDITION \text{ to threshold } T$$
$$THEN \text{ trigger MTD } N$$

A policy corresponding to one MTD might consist of multiple rules since certain malware types require multiple metrics to be proven. The CSV file contains a set of six $IF \rightarrow THEN$ rules that form four distinct policies (Policy I to IV) for each MTD (MTD1, MTD2, and MTD3). In the example, Policy I, which finally triggers MTD1, contains only one rule. Policy II, also starting MTD1, has two rules for the metrics *recv* and *send*. As soon as a policy consists of several rules, the question arises about how many restrictions must apply to be sufficient for malware detection. At this moment,

the determination of this ratio is not yet relevant for policy creation but only in agent development. Consequently, this threshold ratio is only set in Chapter 5.2.

Table 5.5: Structure of the policies

IF			THEN	
Metric	**Relation**	**Threshold**	**MTD**	**Policy**
idl	<=	50	MTD1	Policy I
recv	<=	950	MTD1	Policy II
send	<=	6000	MTD1	
idl	<=	60	MTD2	Policy III
sys	>=	25	MTD2	
usr	>=	7	MTD3	Policy IV

One such $IF \rightarrow THEN$ rule can now be represented by different columns in a CSV file. An example of the structure of these CSV files is shown in the Table 5.6.

Table 5.6: CSV Rules

Metric	**Relation**	**Threshold**	**MTD**
idl	<=	50	MTD1
recv	<=	950	MTD1
send	<=	6000	MTD1
idl	<=	60	MTD2
sys	>=	25	MTD2
usr	>=	7	MTD3

5.1.5 Policy Synthesis

To create a good rule for a policy concerning one single metric, the course of this metric has to fulfill two criteria: First, when triggering the malware, the metric has to differ from the non-infected behavior systematically. This guarantees that the metric is an indicator for this specific malware type. Second, the metric should take on its old value when triggering the MTD. This proves that the MTD was successfully deployed and could mitigate the malware. If both criteria are applicable, we have found a possible candidate metric to build a rule for a policy.

If the considered metric shows the same behavior after the execution of the MTD, this can indicate either that the MTD was unsuccessful or that the metric has changed its value in the longer term. Regarding the latter, this concludes that this metric cannot be a candidate because even if the MTD does neutralize the attack, the metric does not provide the required information that the system acts under normal behavior again. It can therefore happen that the MTD is triggered once again, unaware that the attack was

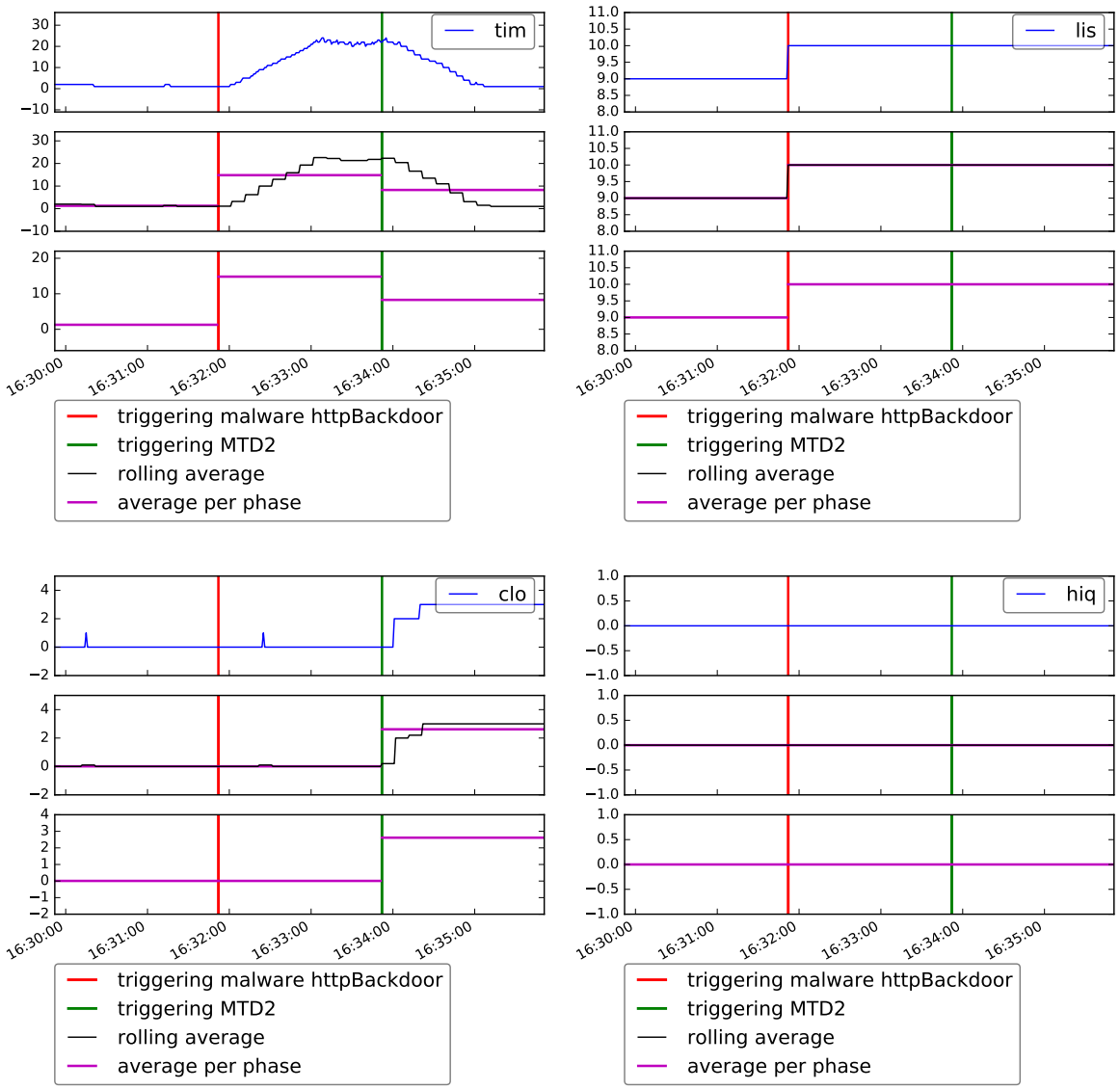

Figure 5.3: Case distinction regarding the metrics *tim*, *lis*, *clo* and *hiq*

already mitigated but self-detecting the influence of the MTD. This can lead to an infinite loop, which must be avoided.

Further, it is also possible that the malware does not influence the considered metric. If that is the case, the metric cannot be considered an indicator for apparent reasons. Table 5.7 summarizes these findings. An example of the metrics *tim*, *lis*, *clo* and *hiq* is shown in Figure 5.3. *tim* as a possible candidate metric, whereas *lis*, *clo* and *hiq* cannot be metric candidates.

To create these $IF \rightarrow THEN$ rules, first, it had to be determined which metrics are worth considering, given malware. According to Table 5.1, each malware type has a specific attack pattern that influences different system parameters. For this purpose, a new Table 5.8 is proposed, based on the gained insights of Table 5.1 and 5.7. Table 5.8 maps each considered system metric to a malware type and indicates if this metric is

potential evidence regarding that malware. With the help of the acquired knowledge from Table 5.8 and the data sets, the actual policy synthesis can now be done by performing data analysis to find rules that consist of metric, relation, threshold, and MTD.

Table 5.7: Metrics candidates and their behavior after triggering malware and MTD

MTD \ Malware	Change	Stable
Change	PC	NC
Stable	NC	NC

PC=possible candidate, NC=non-candidate

Table 5.8: Considered metrics per malware

		Malware Type			
Metric Type	**Metric**	**C&C**	**RO**	**RA**	**RE**
CPU	*usr*	P	P	P	D
	sys	P	P	P	D
	idl	P	P	P	D
	wai	P	P	P	D
	hiq	P	P	P	D
	siq	P	P	P	D
Memory	*used*	P	P	P	D
	buff	P	P	P	D
	cach	P	P	P	D
	free	P	P	P	D
File system	*files*	P	P	P	D
	inodes	P	P	P	D
Disk	*read*	P	P	P	D
	writ	P	P	P	D
	reads	P	P	P	D
	writs	P	P	P	D
Network	*recv*	P	P	D	P
	send	P	P	D	P
TCP Connections	*lis*	P	P	D	P
	act	P	P	D	P
	syn	P	P	D	P
	tim	P	P	D	P
	clo	P	P	D	P
Sockets	*tot*	P	P	D	P
	tcp	P	P	D	P
	udp	P	P	D	P
	raw	P	P	D	P
	frg	P	P	D	P
Processes	*int*	P	P	P	D
	csw	P	P	P	D
Processes	*run*	P	P	P	D
	blk	P	P	P	D
	new	P	P	P	D

RO=Rootkit, RA=Ransomware, RE=Reconnaissance
P=Potential Indicator, D=Doubtful Indicator

In the final step during the policy creation process, the plots for the malware "Ransomware-PoC", "BEURK" and "httpBackdoor" were analyzed and compared in an expert-based approach to find possible candidates 5.7. After an intensive study of the different plots

and experimental fine-tuning of each policy, the following policy database was proposed 5.9. Therefore, the set of metrics this thesis is offering includes all metrics that are used in the policy database, thus *idl*, *sys*, *usr*, *writ*, *writs*, *new* and *recv*. Of the initial 33 system metrics, seven were used in policies, which corresponds to approximately 21%.

Table 5.9: Policy database

Metric	**Condition**	**Threshold**	**MTD**
idl	<=	50	MTD1
sys	>=	24	MTD1
usr	>=	10	MTD1
writ	>=	200000	MTD1
writs	>=	20	MTD1
tim	>=	6	MTD2
new	>=	17	MTD3
recv	>=	2500	MTD4

5.2 Selection Agent

The proposed strategy selection agent works as follows: Observing the complete set of system metrics elaborated in the Table 5.8, the MTD Strategy Selection Agent *MTD StraSelA* waits 10 seconds and calculates the average of the last ten observations. To do so, the observer component must do some data cleaning and preprocessing.

The observer component provides a system metrics vector of length 33 to the policy component. Equipped with the policy database, the policy component can then iterate over the input vector and check if a corresponding rule exists for the actual metric in the database. If such a rule exists, the rule consisting of metric, sign, threshold, and MTD technique can be evaluated: The rule either applies or not, such that a corresponding value will be incremented in the MTD indicator. Each evaluation is recorded by the observer component in the *observer.log* file. In the end, the indicator contains two integers corresponding to the number of positive (rule evaluated as accurate) and negative (rule considered as false) occurrences for each policy.

The policy component makes the actual decision on which MTD should be deployed: By calculating the percentage of positive concerning total occurrences (rules evaluated as true plus rules considered as false), it determines the policy with the highest ratio and prepares the MTD command for the deployer component. The corresponding MTD is deployed if this ratio is more significant or equal to 0.6.

The deployer component is activated in such a case, and the MTD command is executed. Furthermore, the deployer component keeps a log file called *deployer.log* with different timestamps, one for triggering the MTD and one as soon as the MTD has successfully deployed. After the execution, the deployer component waits for 60 seconds. This is crucial since the system needs some time to recover to its old healthy condition. If there is no maintenance or too short a period, the observer could detect its own MTD mechanisms

and what to avoid. A complete sequence for the detection cycle is illustrated in Figure 5.4.

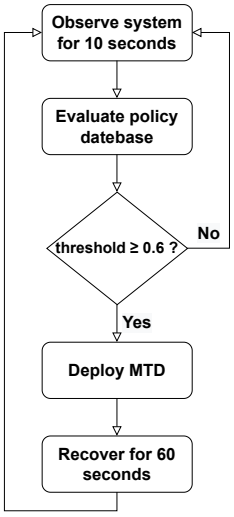

Figure 5.4: Complete detection cycle of MTD *MTD StraSelA*

Chapter 6

Evaluation Scenario and Experiments

The following evaluation part of this thesis has four parts. First, *MTD StraSelA* is tested for each malware individually to evaluate the detection time and the defensive behavior. Second, the detection rate of *MTD StraSelA* is analyzed. Third, *MTD StraSelA* is exposed to a mix of three malware to simulate attack situations in the real-world. Finally, the overhead of *MTD StraSelA* is evaluated.

The general setup for the individual evaluation was the following: *MTD StraSelA* was installed on the system and used *policy-db.csv*, the policy database. Five seconds before the actual measurements, the *MTD StraSelA* was started. For each malware, a fresh image was used. Each malware was triggered after two minutes, and potential downloads were started after a further 30 seconds. With the monitoring script, a complete sequence of 6 minutes, which corresponds to 360 data points, was recorded. For each malware, the victim, a Raspberry Pi armed with *MTD StraSelA*, is regarded as the client, whereas the attacker, a desktop computer running Ubuntu 20.04, is represented as the server.

When triggering "httpBackdoor", the Python attack script *httpBackdoor_attack _script.py* was launched on the attacker server after the *httpBackdoor.py* was started on the client. The attacker sends commands to the server at a random time interval. Regarding "The Tick" and "backdoor", the server can start an arbitrary download on the client using a simple command. This can be any malicious software in the real-world, but for this testbed, the Python binary 3.10.6 was used. "BASHLITE" was slightly modified, such that the server sends a "PING" not every 60 but every 10 seconds and that the client logs every incoming "PONG" into a file. Both rootkits, "BEURK" and "bdvl", were executed by a specific sequence of commands on the client. Using "bdvl", the server could start compromising the system by opening a backdoor or other malicious actions. The "Ransomware-PoC" encrypts the folder "sample-data" that has a size of 386 MB on /root/ that contains various folders with files: 21.4 MB audio files, 136 KB code files, 7.11 MB document files, 62.8 MB images, 24.5 MB video files, and 269.7 MB zip archives. To start "Ransomware-PoC" Command 6.1 was executed.

Listing 6.1: "Ransomware-PoC" command

```
1    python3 main.py -p "/root/sample-data" -e
```

During the mixed evaluation, the attacker script *evaluation_attack _script.py* was launched on the Raspberry Pi that triggered a specific sequence of three concrete malware: First, the rootkit "BEURK", then "Ransomware-PoC" and finally "httpBackdoor" from the C&C family. A whole sequence of 10 minutes, which is equal to 600 data points, was recorded with the monitoring script. The attacker script did trigger the first malware after 60 seconds, but from then on, no predefined timestamps were used. If the attacker script can perform a successful attack, meaning no interruption, it will wait for 60 seconds before it launches the next attack. If the script gets interrupted, hence the attack is inhibited; the hand will wait 60 seconds and then go to the next attack.

The interrelationship of malware and associated countermeasures implemented as MTD techniques from the MTDFramework [14] can be explained as follows: MTD 1, corresponding to *CreateDummyFiles.py*, mitigates a ransomware attack by generating honey files in a specific directory that acts as a dummy file such that sensitive files are less likely to get encrypted, and it also seeks the corresponding ransomware process and terminates it by calling *KillProcess.py*. After the ransomware attack, a script is executed to list the folder contents to see which files are encrypted and unharmed. MTD 1 is connected to the *idl*, *sys*, *usr*, *writ* and *writs* metrics. To counter C&C attacks, MTD 2 implemented in *ChangeIpAddress.py* undertakes an IP shuffle. Shuffling the IP results in a lost connection for the C&C server. MTD 4 also performs the exact same IP shuffle by executing *ChangeIpAddress.py*, but they do not have the same policy: MTD 2 corresponds to the *tim* metric, whereas MTD 4 includes the *recv* metric. MTD 3 is the countermeasure for both rootkits, "BEURK" and "bdvl". Triggering MTD 3 corresponds to calling *RemoveRootkit.py*, executes a command that replaces a compromised ld.so.preload file. Table 6.1 summarizes these interrelationships.

Table 6.1: Malware and MTD mitigation

MTD	Metrics	Malware	Description	Command
1	*idl*, *sys*, *usr*, *writ*, *writs*	R-P	Create dummy files Kill ransomware process	python3 CreateDummyFiles.py –path=/root/sample-data –numberOfDummyFiles=30 –numberOfDummyFilesPerSubdirectory=15 –size=10 –extension=pdf
2	*tim*	hB, B	IP shuffling	python3 ChangeIpAddress.py
3	*new*	be,bd	Replace ld.so.preload	python3 RemoveRootkit.py
4	*recv*	tt, ba, B	IP shuffling	python3 ChangeIpAddress.py

tt="The Tick", ba="backdoor", hB="httpBackdoor", B="BASHLITE", R-P="Ransomware-PoC", be="BEURK", bd="bdvl"

The detection rate evolution was conducted as follows: *MTD StraSelA* was slightly modified such that the deployer component triggered no MTD strategy since we are only interested in the pure detection rate. The history length was adjusted to two, and observation took 200 seconds. For both "backdoor" and "The Tick", the connection was established five seconds before the download was started at the trigger point one second before *MTD StraSelA* was started, whereas the other malware examples were also

triggered in that second. Additionally, the monitoring script was started also 5 seconds before *MTD StraSelA* was activated. As for the individual evaluation, the *policy-db.csv* was used and for each malware, a freshly installed system was used. For the download, not the Python binary 3.10.6 was used but the Ubuntu 22.04 image, due to the reason that the larger download takes optimally the whole 200 seconds to finish.

6.1 Individual Evaluation

6.1.1 Case Study C&C

For "httpBackdoor", *MTD StraSelA* detects the attack after 26 seconds (green line) and deploys the correct MTD that changes the IP address. The deployment took 8.94 seconds. Regarding the metric *tim*, approximately 60 seconds after the finished deployment, the system has conformed to normal behavior. Moreover, it can be shown that the metric *recv* is not influenced by the malware but by the MTD. Both of metrics are visualized in Figure 6.1.

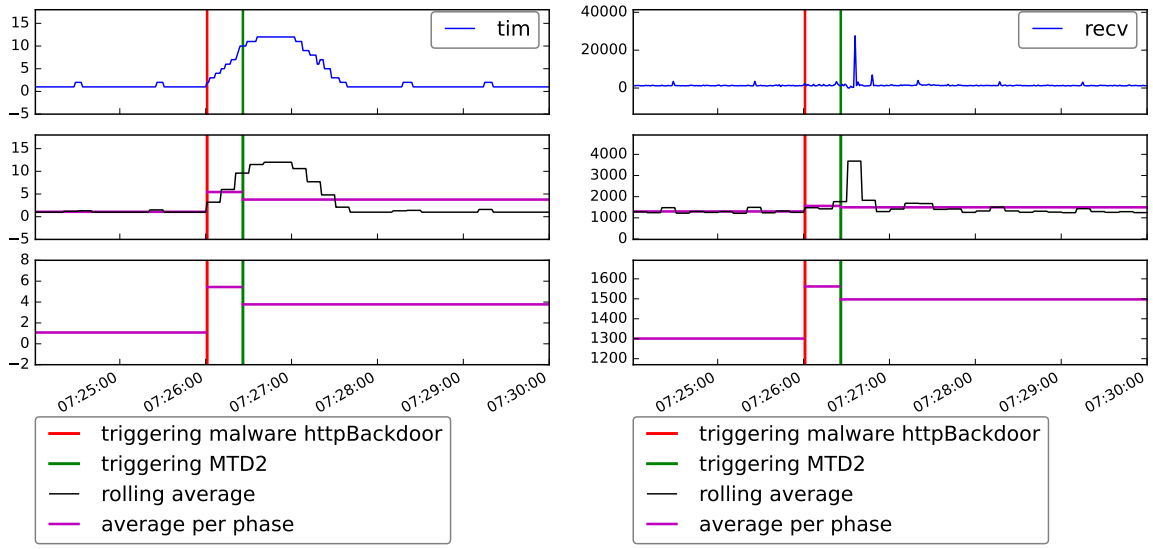

Figure 6.1: Analysis of the metrics *tim* and *recv* under the influence of "httpBackdoor"

Looking at *MTD StraSelA* under the attack of "The Tick", *MTD StraSelA* overreacts and triggers an MTD twice before the actual attack was launched. First, the metric *tim* triggers MTD 2 at 07:48:20 which means 20 seconds after starting the observation. This first deployment took 10.33 seconds. At 07:49:54, MTD 4 was launched and deployed in 8.82 seconds, because *recv* exceeded the critical threshold of 2500 (5678). After triggering the malware at 07:50:00 and initiating the download at 07:50:30, the correct MTD 4 was triggered at 07:52:34 again due to the threshold corresponding to the metric recv, which means it took roughly 2 minutes to detect the download. Executing MTD 4 for the second time took 9.01 seconds. Figure 6.2 illustrates these statements.

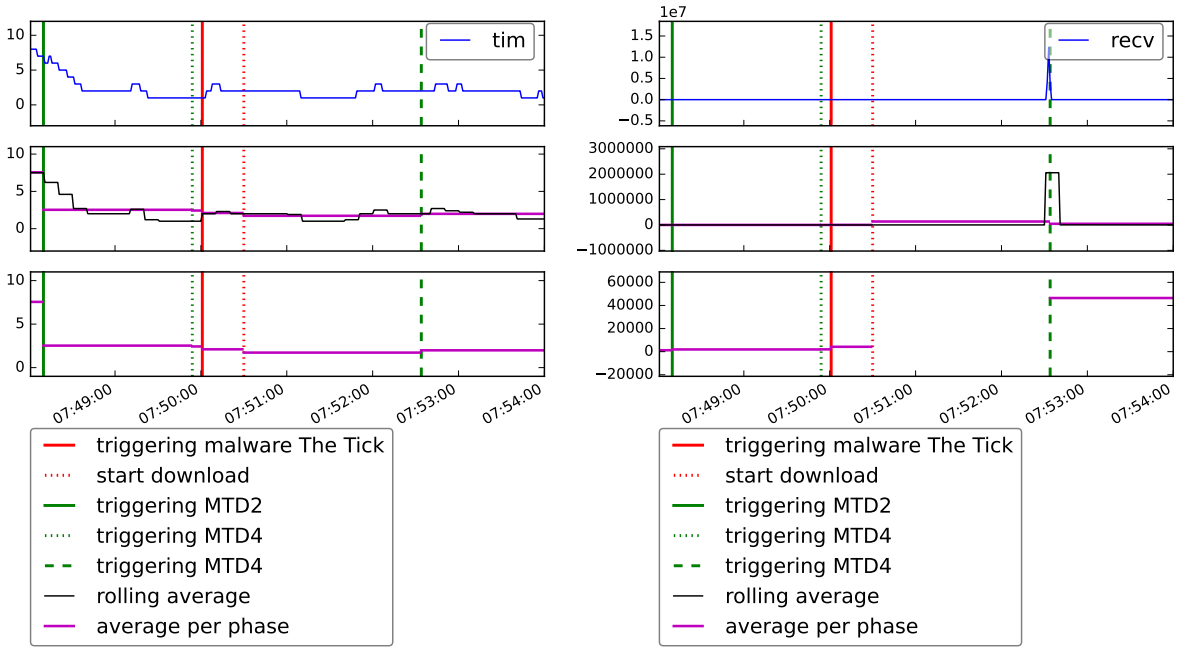

Figure 6.2: Analysis of the metrics *tim* and *recv* under the influence of "The Tick"

The third C&C malware, "backdoor", was correctly detected by *MTD StraSelA*. When the download was started at 08:10:30, after 8 seconds the deployment of MTD 4 was ordered. It took 9.02 seconds to execute MTD 4 and 6.3 shows that *recv* normalized quickly.

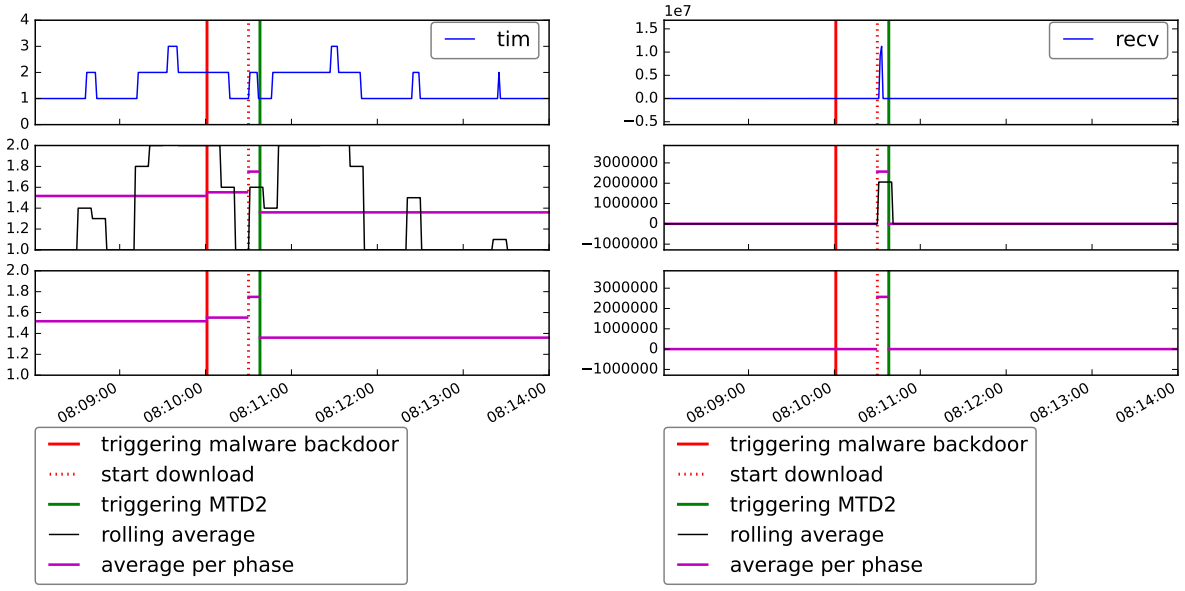

Figure 6.3: Analysis of the metrics *tim* and *recv* under the influence of "backdoor"

Unfortunately, *MTD StraSelA* was not able to detect "BASHLITE", the last C&C software. Hence, we just have a healthy and an infected phase, but no cleansed phase, which is demonstrated in Figure 6.4. There is little to no difference around the malware trigger point (solid red line). No significant change could be detected in either the *recv* nor *tim* metrics.

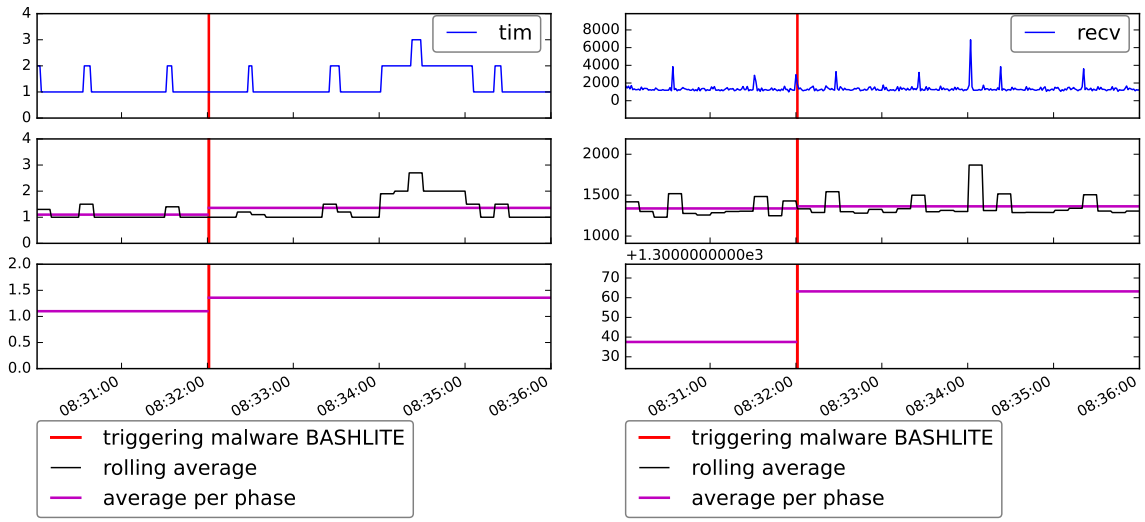

Figure 6.4: Analysis of the metrics *tim* and *recv* under the influence of "BASHLITE"

6.1.2 Case Study Ransomware

MTD StraSelA was able to detect the ransomware after 26 seconds. The deployment of MTD 1 took 143.3 seconds. After 10 further seconds, the behavior has normalized. A total of 39 files have been encrypted, which is 42.4% of all files. No honey files were encrypted. Regarding the volume, 68.1% have been encrypted, which is illustrated in Figure 6.5. That means, *MTD StraSelA* was able to protect and save about one-third of the data volume. The Figure 6.6 shows the behavior of the metrics that can determine a ransomware attack.

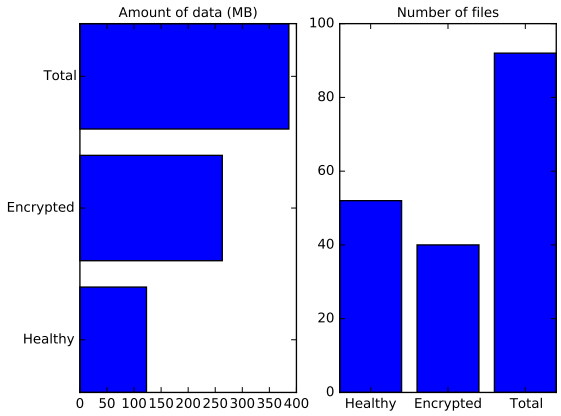

Figure 6.5: Damage analysis of "Ransomware-PoC" during the individual evaluation

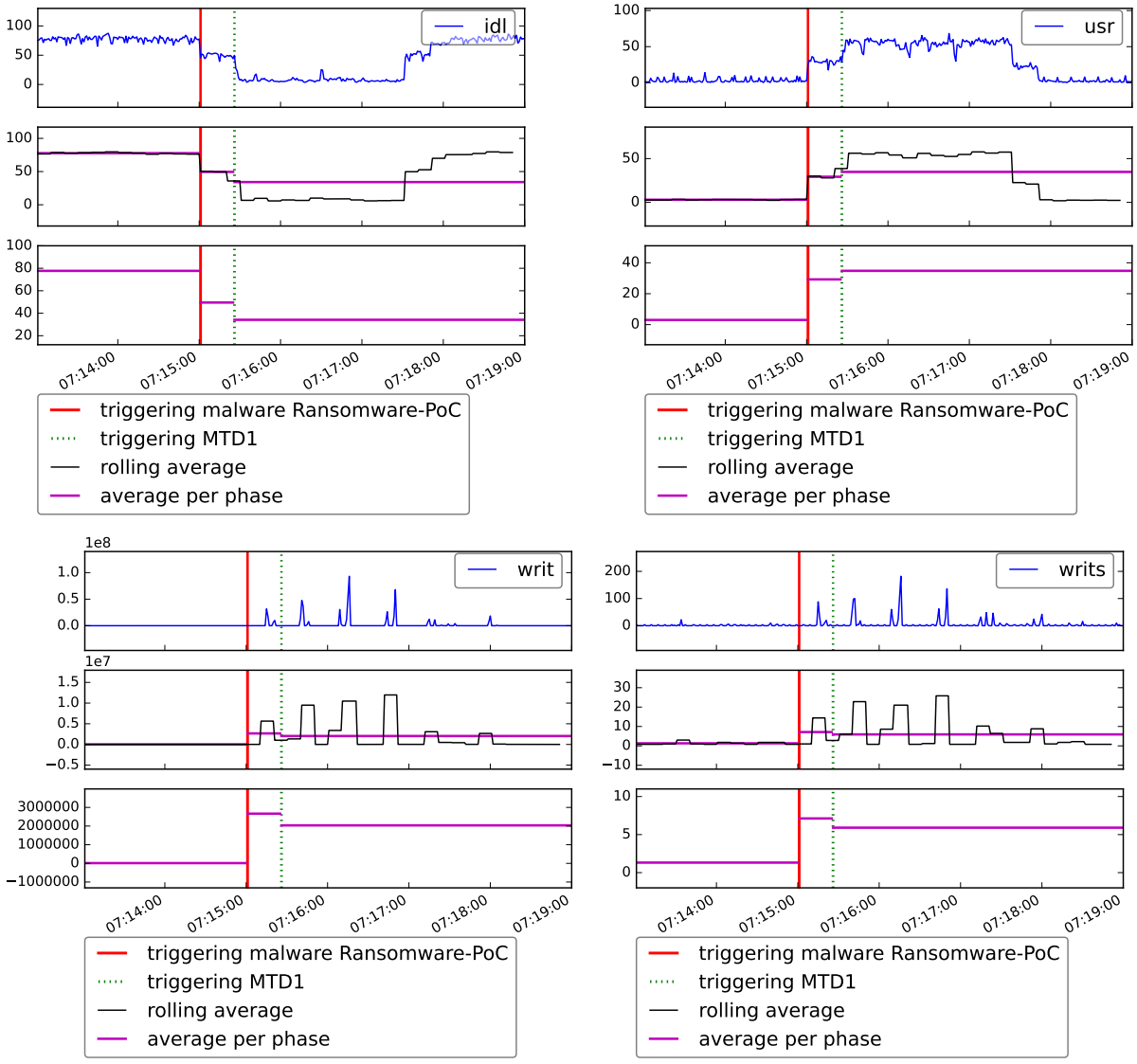

Figure 6.6: Analysis of the metrics *idl*, *usr* *writ* and *writs* under the influence of "Ransomware-PoC"

6.1.3 Case Study Rootkit

Facing ”BEURK”, *MTD StraSelA* could successfully detect and trigger the corresponding MTD 3. After 4 seconds, the MTD was deployed and this operation took 0.3 seconds. The System could recover within 30 seconds, which can be seen in Figure 6.7. ”bdvl” has behaved similarly to ”BEURK”: MTD 3 was triggered 4 seconds after the rootkit was launched, and the execution of the MTD took 0.25 seconds. At the time of the execution, the criteria for triggering MTD 4 were also full-filled, but since only one MTD can be triggered at a time, MTD 4 was not triggered. The execution hierarchy was given by the names of the MTDs.

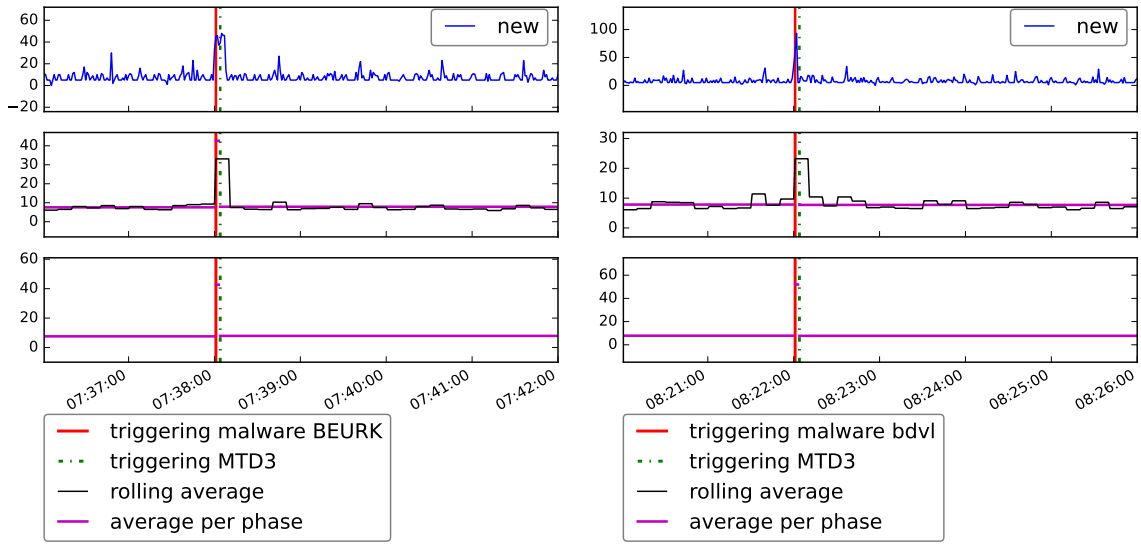

Figure 6.7: Analysis of the metrics *new* under the influence of ”BEURK” and ”bdvl”

Table 6.2 lists the relevant times regarding detection and deployment. For ”The Tick” and ”backdoor”, the detection time relates to the download trigger point and the number inside the braces refers to the malware trigger point. The detection time ranges from 4 seconds up to 124 seconds, and the deployment time from 0.25 seconds up to 143.78 seconds. Since our detection windows amount to 10 seconds, we can estimate the number of iterations based on the detection time: ”BEURK”, ”backdoor” and ”bdvl” can be detected in one iteration (< 10 seconds), ”Ransomware-PoC” and ”httpBackoor” need about two to three iterations (< 30 seconds) and ”The Tick” was detected in around 13 iterations (< 130 seconds).

Table 6.2: Detection time

	Trigger point				Time interval	
	Malware	**Download**	**MTD**	**MTD**	**Detection (s)**	**Deployment (s)**
"Ransomware-PoC"	07:15:00	-	07:15:26	1	26	143.78
"httpBackdoor"	07:26:00	-	07:26:26	2	26	8.94
"BEURK"	07:38:00	-	07:38:04	3	4	0.3
"The Tick"	07:52:00	07:50:30	07:52:34	4	124 (154)	9.01
"backdoor"	08:10:00	08:10:30	08:10:38	4	8 (38)	9.02
"bdvl"	08:22:00	-	08:22:04	3	4	0.25s
"BASHLITE"	08:32:00	-	-	-	-	-

6.2 Detection Rate

The results of this detection rate analysis are shown in Table 6.5, which lists the different detection rates for the corresponding policies. Each policy has four subgroups: True Positive Rate (TPR), False Positive Rate (FPR), True Negative Rate (TNR), and False Negative Rate (FNR). A detailed explanation is provided in Table 6.4. The conducted data is summarized in Table 6.3.

Table 6.3: Detection Analysis: Raw data

				Actions					
Name	**MTD**	**Start**	**End**	**1**	**2**	**3**	**4**	**-**	**Total**
Healthy	-	08:54:00	08:57:20	0	0	2	0	81	83
"Ransomware-PoC"	1	09:13:00	09:16:20	20	0	1	1	59	79
"httpBackdoor"	2	09:24:00	09:27:20	0	81	2	3	2	84
"BEURK"	3	09:31:00	09:34:20	0	0	3	1	79	83
"backdoor"	4	10:26:00	10:29:20	65	0	3	78	0	78
"BASHLITE"	2, 4	10:34:00	10:37:20	0	0	2	1	80	83
"The Tick"	4	10:43:00	10:46:20	62	0	1	74	0	74
"bdvl"	3	10:55:00	10:58:20	1	0	4	2	69	75

Table 6.4: Detection Analysis: Description of different evaluation Metrics

Metric	**Description**	**Definition**
True Positive Rate (TPR)	Ratio of correct execution of MTD_i in case of the actual occurrence of $malware_i$	$\frac{TP}{TOTAL}$
False Negative Rate (FNR)	Ratio of incorrect non-execution of MTD_i in case of the actual occurrence of $malware_i$	$\frac{FN}{TOTAL} = 1 - TPR$
False Positive Rate (FPR)	Ratio of incorrect execution of MTD_i in case of the absence of the $malware_i$	$\frac{FP}{TOTAL}$
True Negative Rate (TNR)	Ratio of correct non-execution of MTD_i in case of the absence of $malware_i$	$\frac{TN}{TOTAL} = 1 - FPR$

Policy 1 does have a 25.42% TPR for "Ransomware-PoC" and does have a low ("bdvl") to perfect (healthy, "httpBackdoor", "BEURK", and "BASHLITE") TNR except for the malware "backdoor" and "The Tick": For both, policy 1 has a TNR of only 16.67%.

With TPRs of 96.43% for "httpBackdoor" but a 0% for "BASHLITE", Policy 2 has evidently clear strengths and weaknesses. Remarkably, it is the only Policy that has a perfect TNR score for all malware subjects and the non-infect healthy behavior.

Considering Policy 3, the performance for "BEURK" and "bdvl" is pretty low with a TPR of 3.61% and 5.33% respectively. The TNRs for Policy 3 are relatively high and range from 96.15% ("backdoor") to 98.73% ("Ransomware-PoC").

Lastly, Policy 4 has a perfect TPR for "backdoor" and "The Tick" but was majority incapable to detect "BASHLITE" (only 1.2% TPR). Looking at the TNR, Policy 4 does obtain achieve reasonably high values, namely from 96.43% ("httpBackdoor") up to perfect for the healthy subject.

Table 6.5: Detection Rate

Malware \ Policy		P1	P2	P3	P4		P1	P2	P3	P4
healthy	TPR (%)					TNR (%)	100	100	97.59	100
	FNR (%)					FPR (%)	0	0	2.41	0
"Ransomware-PoC"	TPR (%)	25.32				TNR (%)		100	98.73	98.73
	FNR (%)	74.68				FPR (%)		0	1.27	1.27
"httpBackdoor"	TPR (%)		96.43			TNR (%)	100		97.62	96.43
	FNR (%)		3.57			FPR (%)	0		2.38	3.57
"BEURK"	TPR (%)			3.61		TNR (%)	100	100		98.80
	FNR (%)			96.39		FPR (%)	0	0		1.20
"bdvl"	TPR (%)			5.33		TNR (%)	98.67	100		97.33
	FNR (%)			94.67		FPR (%)	1.33	0		2.67
"backdoor"	TPR (%)				100	TNR (%)	16.67	100	96.15	
	FNR (%)				0	FPR (%)	83.33	0	3.85	
"The Tick"	TPR (%)				100	TNR (%)	16.22	100	98.65	
	FNR (%)				0	FPR (%)	83.78	0	1.35	
"BASHLITE"	TPR (%)		0		1.20	TNR (%)	100		97.59	
	FNR (%)		100		98.80	FPR (%)	0		2.41	

TPR=True Positive Rate, FPR=False Positive Rate
TNR=True Negative Rate, FNR=False Negative Rate

6.3 Mixed Evaluation

For the mixed evaluation, every malware was correctly identified, and the corresponding MTD could be deployed successfully. The state of the system can be divided into 10 phases which are defined by 9 key moments: From 08:43:00 to 08:44:00 we have the healthy phase (1), where correctly no MTD was triggered. At 08:44:00, the first malware, "BEURK", was executed. From 08:44:00 to 08:44:07, we have the detection phase of "BEURK" (2). At 08:44:07, the system triggers MTD 3, which took 0.33 seconds. Hence, this short time frame of 0.33 seconds is the deployment phase of MTD 1 (3). The attacker script *evaluation_attack _script.py* pauses now for 60 seconds, such that we have the cleansed phase from "BEURK" (4).

At 08:45:07, the attacker script launched the ransomware, so we have a detection phase of "Ransomware-PoC" from 08:45:07 to 08:45:18 (5). *MTD StraSelA* triggers at 08:45:18 MTD 1 to mitigate the ransomware attack, and it finishes its actions at 08:47:42, what the end of the deployment phase of MTD 1 (6) means. The next attack, namely "httpBackdoor" is launched at 08:48:23, which gives us the cleansed phase from "Ransomware-PoC" (7) from 08:47:42 to 08:48:23. The detection phase of "httpBackdoor" (8) starts from 08:48:23 and ends at 08:48:53. At this moment, MTD 2 is deployed, leading to a deployment phase of MTD 2 (9) from 08:48:53 to 08:49:02. From 08:49:02 to the end of the observation (08:53:00), the final cleansed phase for "httpBackdoor" (10) took place. Table 6.6 summarizes the findings of this paragraph.

Table 6.6: Timeline mixed attacks

No	Status	Phase	Start	End	Duration (s)	Description
1	healthy	healthy	08:43:00	08:44:00	60	pause
2	infected	detection b	08:44:00	08:44:07	7	trigger r
3	infected	deployment b	08:44:07	08:44:07	0.33	MTD 3
4	healthy	cleansed b	08:44:07	08:45:07	60	pause
5	infected	detection r	08:45:07	08:45:18	11	trigger r
6	infected	deployment r	08:45:18	08:47:42	143.44	MTD 1
7	healthy	cleansed r	08:47:42	08:48:23	41	pause
8	infected	detection h	08:48:23	08:48:53	30	trigger h
9	infected	deployment h	08:48:53	08:49:02	9.07	MTD 2
10	healthy	cleansed h	08:49:02	08:53:00	238	pause

b="BEURK", r="Ransomware-PoC", h="httpBackdoor"

Figure 6.8 is a visualization of the four metrics *idl*, *usr*, *writ* and *writs*. The metrics *idl* and *usr* significantly change in the detection phase of "Ransomware-PoC", which is indicated by the second red-colored segment. As soon as the deployment phase, depicted in green color, starts, the difference becomes even clearer. The metrics *writ* and *writs* now deviate as well strongly from their previous behavior. This is clearly due to the activated MTD 1, which corresponds to the second green-colored segment. As soon as the deployment phase of MTD 1 is finished, the device status returns to normal.

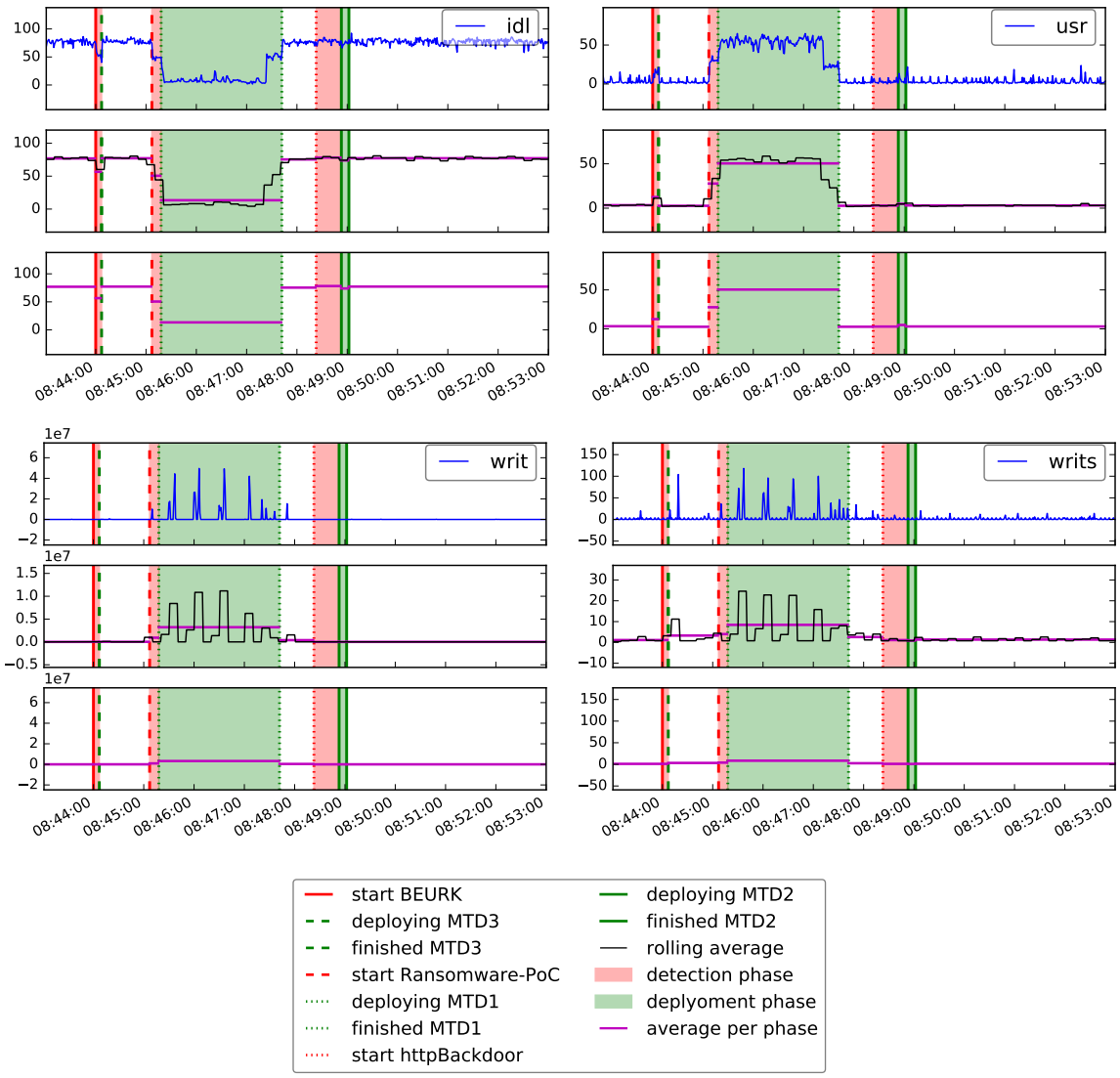

Figure 6.8: Mixed evaluation: Behavior of the CPU relevant metrics *idl* and *usr*, as well as the disk relevant metrics *writ* and *writs*

Looking at Figure 6.9, *MTD StraSelA's* behavior regarding "httpBackdoor" and "BEURK" can be understood: During the detection phase of "httpBackgroound", the last red-colored segment, the metrics *tim* records a significant increase. The system detects the malware and initiates the deployment phase of MTD 2, the last green segment. Shortly after that, the behavior normalizes. The metric *new* changes abruptly and also the deployment phase of MTD 3 is almost instantaneous, such that the first green and red segments are almost not visible.

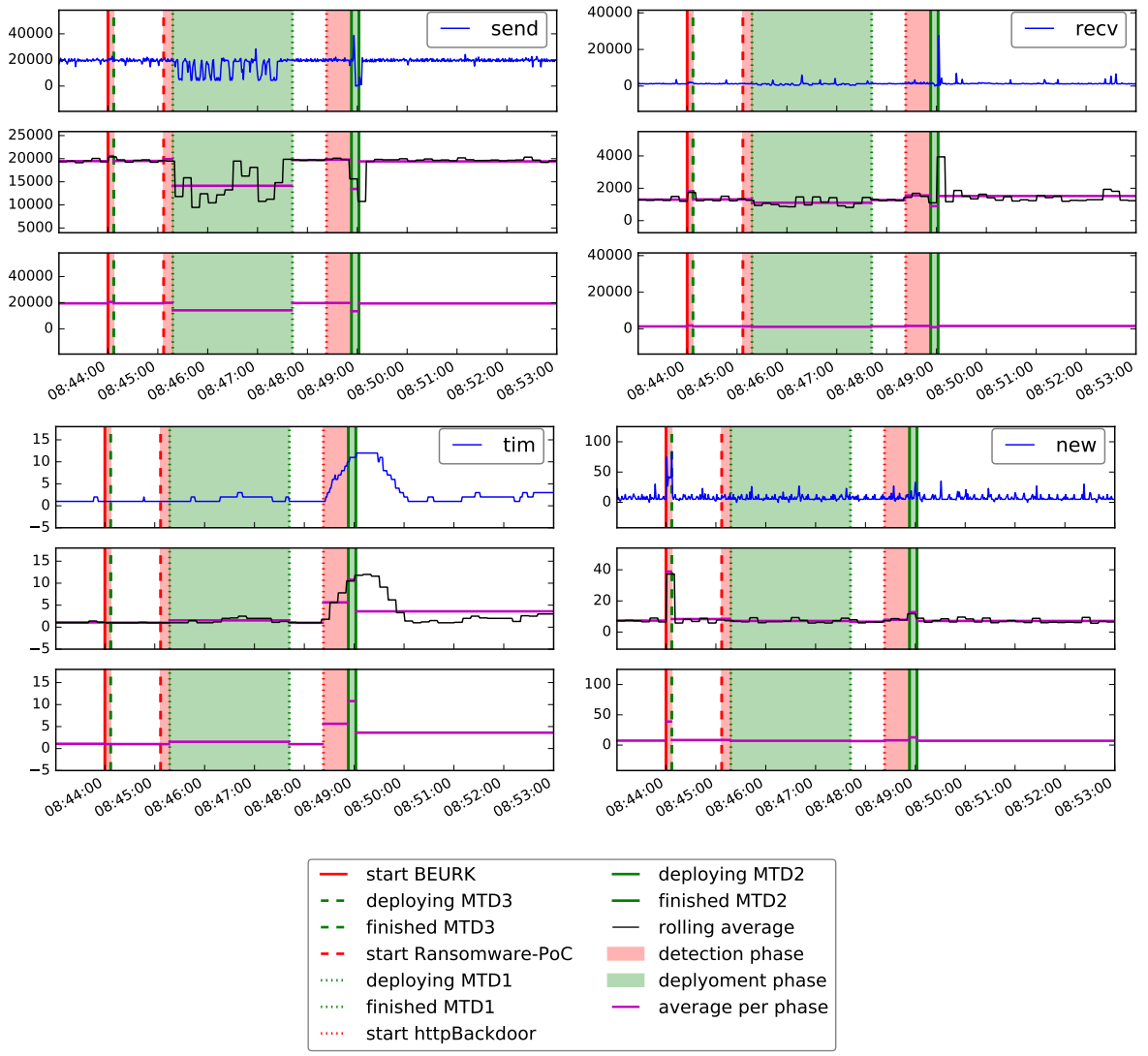

Figure 6.9: Mixed evaluation: Behavior of the network relevant metrics *send* and *recv*, as well as the TCP connections relevant metrics *tim* processes related and *new*

"Ransomware-PoC" encrypted 38 regular and 0 honey files, which means 41.3% of all the total files have been encrypted. The volume of these files amounts to 64.8%. An illustration of this result can be found in figure 6.10. In the mixed evaluation, *MTD StraSelA* was also able to secure approximately one-third of the data volume.

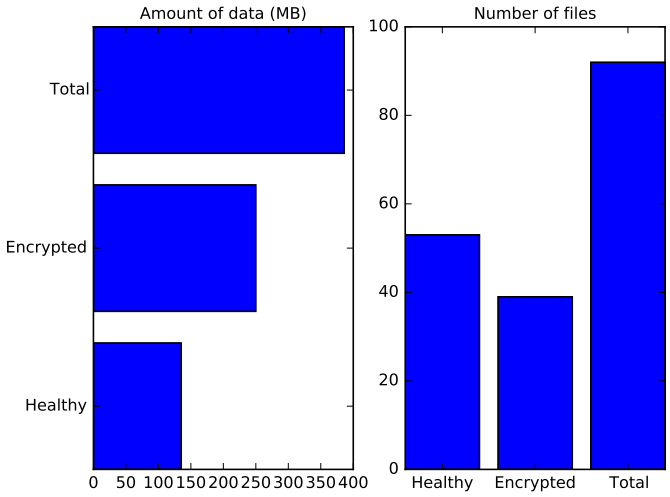

Figure 6.10: Damage analysis of "Ransomware-PoC" during the mixed evaluation

6.4 Overhead Evaluation

The last part of the evaluation is dedicated to an overhead analysis: Is *MTD StraSelA* resource-efficient? To answer this, two data sets at 10 minutes each were collected, one with no running *MTD StraSelA* and one with active *MTD StraSelA*. No malware was triggered during that period since only the impact of the agent was of interest. It should also be noted that in that particular evaluation, *MTD StraSelA* did not trigger any MTD; this branch was simply commented out to make sure that only the overhead of the agent was measured and not the influence of any MTD technique deployment. Both Figures, Figure 6.11 and Figure 6.12, visualize the metrics *usr, idl*, and *run* in the aforementioned conditions. This data was aggregated and the average value for multiple evaluation metrics was compared: Regarding CPU-relevant metrics, *MTD StraSelA* caused an increase of 16% in the *usr* activity. This seems to be a lot since the *sys* activity only decreased by 6.6% and the *idl* activity marginally increased by 1.2%. A comparison of the memory values makes it clear that *MTD StraSelA* needs about 73.77 MB of RAM, resulting in a 2.08% decrease. Lastly, the comparison showed that running *MTD StraSelA* resulted in 3.57 active processes on average at a time, compared to 3.19 on average for the base system. The increase is therefore 14.9%. Table 6.7 summarizes the findings.

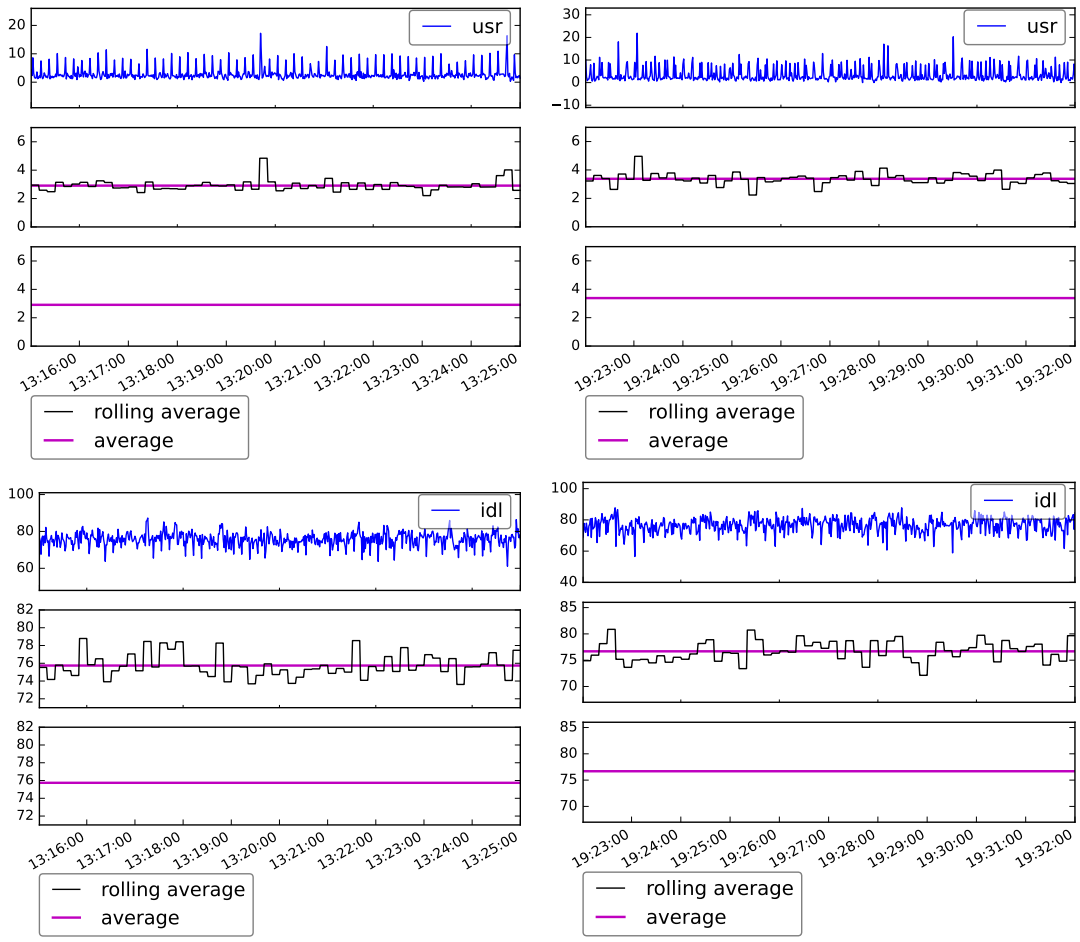

Figure 6.11: Overhead analysis regarding the *usr* and *idl* metrics: System without *MTD StraSelA* on the left, running *MTD StraSelA* on the right

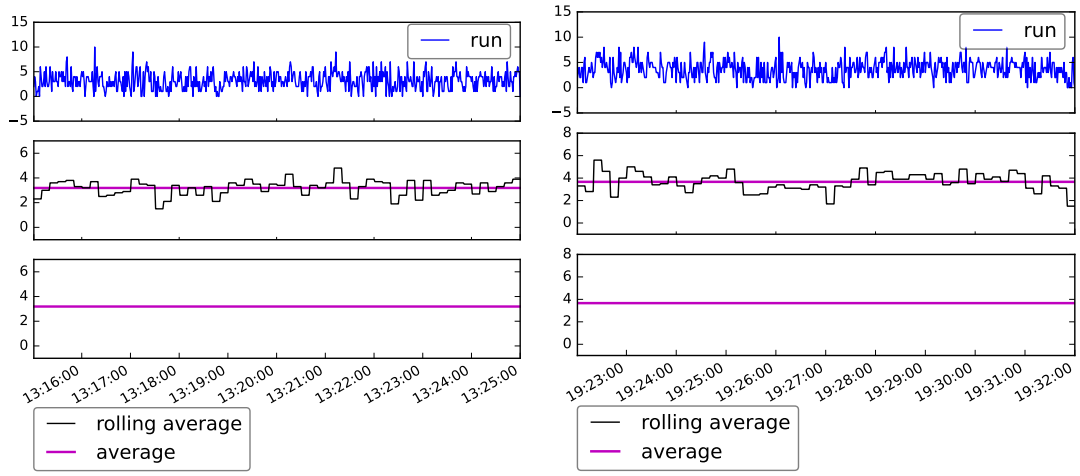

Figure 6.12: Overhead analysis regarding the *run* metric: System without running *MTD StraSelA* on the left, running *MTD StraSelA* on the right

Table 6.7: *MTD StraSelA* Overhead Analysis

Metric	Normal avg	***MTD StraSelA* avg**	Absolute change	Relative change (%)
usr	2.91%	3.4%	0.9%	16.0
sys	21.29%	19.9%	1.6%	-6.6
idl	75.74%	76.7%	-2.6%	-1.2
free	3539.35 MB	3465.58 MB	-73.77 MB	-2.08
run	3.19	3.67	0.48	14.9

6.5 Discussion

The individual evaluation suggests that in general, it is possible to detect malware based on a system metrics and mitigate the attacks using MTD. The most vivid example is probably "Ransomware-PoC", where almost 30% of the data could be secured. However, with the reference to "BASHLITE", there exists malware that are much harder to detect. More specifically, no set of metrics could be provided for "BASHLITE" such that the attack could be recognized and neutralized. That means everything depends on the set of metrics: If the impact, more precisely the system metric and the concrete change in form of an exact threshold, of a malware type on the victim system can be accurately converted into a fingerprint in the form of a collection of system parameters, then it is possible to detect the malware using *MTD StraSelA* and eventually mitigate it. A classification of these suitable metrics is provided in Chapter 5.1.5.

Another challenge is based on the fact, that certain malware behaves mostly statically: For "The Tick" and "backdoor", the Policy 4 was successful, but only because a large download was triggered. It is highly questionable whether a real attacker would actually cause such a large load. Both of the rootkits, "BEURK" and "bdvl" were only launched, and no further actions were taken by the malware. The actual installation was detectable and could be mitigated by *MTD StraSelA*, but there is no policy provided for its active but sleeping behavior.

The temporal aspect also harbors some inadequacies: First, *MTD StraSelA* rests for 60 seconds, whenever an MTD was deployed. The agent needs this time, to recover and to restore its usual behavior. During this time, the system is highly vulnerable because the detection is paused. Further, the history length of 10 is a time frame, where a lot of damage can already be done. In addition, Table 6.2 shows that besides "BEURK" and "bdvl", all *MTD StraSelA* needs more than one iteration to detect malicious behavior.

From the mixed evaluation it can be concluded that *MTD StraSelA* is technically able to deal with multiple attacks in a sequence: It was able to identify correctly and mitigate the attacks. Nevertheless, the same aforementioned reservations apply. Next, an investigation of the overhead of *MTD StraSelA* shows that the actual agent, without taking into account any deployments of MTD behaves in a resource-efficient manner: *MTD StraSelA* runs a the cost of 2.6% CPU load and 61 MB of RAM.

Last but no least, the detection rate analysis is admittedly sobering: With a TPR of just around 25%, Policy 1 is anything else as but perfect. The same also applies for Policy 3. That raises the question of why *MTD StraSelA* had few to no problems during the

previous evaluations concerning Policy 1 regarding ”Ransomware-PoC” and Policy 3 with respect to ”BEURK” and ”bdvl”. This can be justified by the fact that the detection rate evaluation considers and tests policy regarding malware over the entire period, while the individual and mixed evaluation focus on the first single appearance. Therefore, it is possible that there may exist even better policies, with different metrics and thresholds for policy 1 and policy 3. If you exclude ”BASHLITE” for once, policies 2 and 4 offer a very good, even almost perfect performance, with regard to both TPR and TNR. It is striking that these are two policies with exclusively network metrics.

Chapter 7

Summary, Conclusions, and Future Work

The last chapter of this thesis recaps the different stations of the project, proposes the main findings, and provides ideas for further research and projects. First, a complete review summarizes the thesis and the concrete procedure followed. Second, with regard to the evaluation, the key insights are formed, and a conclusion is derived. Finally, suggestions for future projects and investigations are formulated.

This thesis aims to design and implement an MTD selection agent using IoT platform metrics to select from existing MTD techniques. In order to do so, the following steps were undertaken:

First, the necessary background knowledge regarding MTD, malware, IoT security, and policy selection methods were obtained. Second, an initial set of metrics was created as a result of a phase of systematic research on the impact of malware on a victim system. Third, with the help of a self-created monitoring script, data were collected for the respective malware types, showing healthy, infected, and cleansed behavior in each case. Subsequently, with the help of data analysis, a set of policies consisting of rules that include a threshold and a metric could be adopted. At the same time, the architecture of the MTD strategy selection agent *MTD StraSelA* has been elaborated. In the next stage, the design of *MTD StraSelA* using this set of policies was determined and subsequently implemented. This resulted in the proof of concept MTD selection agent *MTD StraSelA*. After that, the final policy was experimentally fine-tuned. At the beginning of the evaluation phase, an individual assessment for each malware was carried out. After that, the performance of *MTD StraSelA* was investigated regarding a real-world attack with an attacker script, and the overhead was measured. Finally, the detection rate of the proposed policy was analyzed.

To define the bottom line of these findings, the following conclusions are drawn:

- First, it is generally possible to use system metrics to detect malware and mitigate them with MTD techniques. This thesis suggests a simple, resource-efficient MTD strategy selection agent called *MTD StraSelA* that deploys MTD techniques based on a policy database. These rules follow a simple $IF \rightarrow THEN$ schema.

- Second, by using an expert-based approach, namely with the implementation of data analysis and data visualization, it is possible to find system metrics that can be used in a policy $IF \rightarrow THEN$. With the help of experimental fine-tuning, definitive policies can then be created.
- Third, a potential metric can be classified as a possible candidate if it exhibits the following behavior: There is a significant change when triggering malware and mitigating the associated MTD.
- For an agent to detect behavior that deviates from normal behavior, it is crucial that the original state is assumed again after executing an MTD. The time window required for this and the time needed for the MTD deployment represent a vulnerability.
- Not every malware can be successfully detected with this approach: The well-known botnet "BASHLITE" could not be successfully detected.
- The evaluation points out the difference between detecting malware at its trigger point or detecting malware at any moment.

Finally, we can say that it is generally possible to use system metrics to detect malware. Still, everything stands and falls with the selection of metric, threshold, and the respective policy. The difficulty lies in finding policies that reliably respond to the malware's initial contact and the more or less static behavior.

7.1 Future Work

In general, the existing agent could be further optimized. It would therefore be a good idea to modify and examine the concrete parameters and their influence on the agent's performance. Objects of this study are, in particular, the number of aggregated observations needed to make an MTD deployment decision and the recovering period after an MTD deployment.

According to the motto, prevention is more affordable than treatment; it might be an exciting idea to not only focus on MTD that is purely reactive deployed but more proactive. The agent is reactive, so it might be helpful to undertake further investigations in this context: Since "BASHLITE" could not be detected and mitigated, using a proactive or even hybrid agent might represent a valid approach. For this, further experiments would be necessary to determine the cost-benefit ratio.

Bibliography

[1] M. De Saulles, *Internet of Things Statistics - Information Matters*, https://informationmatters.net/internet-of-things-statistics/, 2019, [Online; accessed Sep 14 2022].

[2] R. E. Navas, H. Sandaker, F. Cuppens, N. Cuppens, L. Toutain, and G. Z. Papadopoulos, „IANVS: A Moving Target Defense Framework for a Resilient Internet of Things“, *Proceedings - IEEE Symposium on Computers and Communications*, vol. 2020-July, Jul. 2020, ISSN: 15301346. DOI: 10.1109/ISCC50000.2020.9219728.

[3] D. Chen, J. Cong, S. Gurumani, W.-m. Hwu, K. Rupnow, and Z. Zhang, „Platform choices and design demands for IoT platforms: cost, power, and performance tradeoffs“, *IET Cyber-Physical Systems: Theory & Applications*, vol. 1, no. 1, pp. 70–77, Dec. 2016, ISSN: 2398-3396. DOI: 10.1049/IET-CPS.2016.0020.

[4] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, „DDoS in the IoT: Mirai and other botnets“, *Computer*, vol. 50, no. 7, pp. 80–84, 2017, ISSN: 00189162. DOI: 10.1109/MC.2017.201.

[5] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose, „SoK: Security Evaluation of Home-Based IoT Deployments“, *Proceedings - IEEE Symposium on Security and Privacy*, vol. 2019-May, pp. 1362–1380, May 2019, ISSN: 10816011. DOI: 10.1109/SP.2019.00013.

[6] R. Gurunath, M. Agarwal, A. Nandi, and D. Samanta, „An overview: Security issue in IoT network“, *Proceedings of the International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud), I-SMAC 2018*, pp. 104–107, Feb. 2019. DOI: 10.1109/I-SMAC.2018.8653728.

[7] K. Zeitz, M. Cantrell, R. Marchany, and J. Tront, „Designing a Micro-Moving Target IPv6 Defense for the Internet of Things“, *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*, 2017. DOI: 10.1145/3054977.

[8] J. L. Leevy, J. Hancock, T. M. Khoshgoftaar, and N. Seliya, „IoT Reconnaissance Attack Classification with Random Undersampling and Ensemble Feature Selection“, *Proceedings - 2021 IEEE 7th International Conference on Collaboration and Internet Computing, CIC 2021*, pp. 41–49, 2021. DOI: 10.1109/CIC52973.2021.00016.

[9] A. K. Ghosh, D. Pendarakis, and W. Sanders, „National cyber leap year summit 2009 co-chairs' report", Tech. Rep., Sep. 2009. [Online]. Available: https://www.nitrd.gov/nitrdgroups/images/b/bd/National_Cyber_Leap_Year_Summit_2009_CoChairs_Report.pdf, [Online; accessed Sep 14 2022].

[10] M. Sherburne, R. Marchany, and J. Tront, „Implementing Moving Target IPv6 Defense to secure 6LoWPAN in the internet of things and smart grid", *ACM International Conference Proceeding Series*, pp. 37–40, 2014. DOI: 10.1145/2602087.2602107.

[11] F. Nizzi, T. Pecorella, F. Esposito, L. Pierucci, and R. Fantacci, „IoT security via address shuffling: The easy way", *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3764–3774, Apr. 2019, ISSN: 23274662. DOI: 10.1109/JIOT.2019.2892003.

[12] J. Aljosha, U. Johanna, M. Georg, V. G. Artemios, and W. Edgar, „Lightweight Address Hopping for Defending the IPv6 IoT", *Proceedings of the 12th International Conference on Availability, Reliability and Security*, vol. 10, 2017. DOI: 10.1145/3098954.

[13] V. Casola, A. De Benedictis, and M. Albanese, „A multi-layer moving target defense approach for protecting resource-constrained distributed devices", *Advances in Intelligent Systems and Computing*, vol. 263, pp. 299–324, 2014, ISSN: 21945357. DOI: 10.1007/978-3-319-04717-1{_}14.

[14] J. Cedeño, *MTDFramework*, https://github.com/CortexVacua/MTDFramework, 2022, [Online; accessed Sep 14 2022].

[15] G. Vormayr, T. Zseby, and J. Fabini, „Botnet Communication Patterns", *IEEE Communications Surveys and Tutorials*, vol. 19, no. 4, pp. 2768–2796, Oct. 2017, ISSN: 1553877X. DOI: 10.1109/COMST.2017.2749442.

[16] D. Fleck, A. Stavrou, G. Kesidis, N. Nasiriani, Y. Shan, and T. Konstantopoulos, „Moving-Target Defense Against Botnet Reconnaissance and an Adversarial Coupon-Collection Model", *DSC 2018 - 2018 IEEE Conference on Dependable and Secure Computing*, Jan. 2019. DOI: 10.1109/DESEC.2018.8625162.

[17] X. Luo, Q. Yan, M. Wang, and W. Huang, „Using MTD and SDN-based Honeypots to Defend DDoS Attacks in IoT", *2019 Computing, Communications and IoT Applications, ComComAp 2019*, pp. 392–395, Oct. 2019. DOI: 10.1109/COMCOMAP46287.2019.9018775.

[18] S. R. Zahra and M. Ahsan Chishti, „RansomWare and internet of things: A new security nightmare", *Proceedings of the 9th International Conference On Cloud Computing, Data Science and Engineering, Confluence 2019*, pp. 551–555, Jan. 2019. DOI: 10.1109/CONFLUENCE.2019.8776926.

[19] T. B. Alshammari and A. S. Alanazi, „Security Threats against the Internet of Things at Home", *3rd International Conference on Electrical, Communication and Computer Engineering, ICECCE 2021*, Jun. 2021. DOI: 10.1109/ICECCE52056.2021.9514258.

[20] M. Al-Hawawreh, F. D. Hartog, and E. Sitnikova, „Targeted Ransomware: A New Cyber Threat to Edge System of Brownfield Industrial Internet of Things", *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 7137–7151, Aug. 2019, ISSN: 23274662. DOI: 10.1109/JIOT.2019.2914390.

[21] Q. Hua and Y. Zhang, „Detecting Malware and Rootkit via Memory Forensics“, *Proceedings - 2015 International Conference on Computer Science and Mechanical Automation, CSMA 2015*, pp. 92–96, Jan. 2016. DOI: 10.1109/CSMA.2015.25.

[22] I. Kuzminykh and M. Yevdokymenko, „Analysis of Security of Rootkit Detection Methods“, *2019 IEEE International Conference on Advanced Trends in Information Theory, ATIT 2019 - Proceedings*, pp. 196–199, Dec. 2019. DOI: 10.1109/ATIT49449.2019.9030428.

[23] H. Liu, D. Zhang, G. Wei, and J. Zhong, „Detecting malicious rootkit web pages in high-interaction client honeypots“, *Proceedings 2010 IEEE International Conference on Information Theory and Information Security, ICITIS 2010*, pp. 544–547, 2010. DOI: 10.1109/ICITIS.2010.5689538.

[24] J. Wang, „A rule-based approach for rootkit detection“, *ICIME 2010 - 2010 2nd IEEE International Conference on Information Management and Engineering*, vol. 3, pp. 405–408, 2010. DOI: 10.1109/ICIME.2010.5478178.

[25] X. Jiang, M. Lora, and S. Chattopadhyay, „Efficient and trusted detection of rootkit in IoT devices via offline profiling and online monitoring“, *Proceedings of the ACM Great Lakes Symposium on VLSI, GLSVLSI*, pp. 433–438, Sep. 2020. DOI: 10.1145/3386263.3406939.

[26] Z. Tian, B. Wang, Z. Zhou, and H. Zhang, „The research on rootkit for information system classified protection“, *2011 International Conference on Computer Science and Service System, CSSS 2011 - Proceedings*, pp. 890–893, 2011. DOI: 10.1109/CSSS.2011.5974667.

[27] A. Aydeger, N. Saputro, and K. Akkaya, „Utilizing NFV for Effective Moving Target Defense Against Link Flooding Reconnaissance Attacks“, *Proceedings - IEEE Military Communications Conference MILCOM*, vol. 2019-October, pp. 946–951, Jan. 2019. DOI: 10.1109/MILCOM.2018.8599803.

[28] S. Liao, C. Zhou, Y. Zhao, *et al.*, „A Comprehensive Detection Approach of Nmap: Principles, Rules and Experiments“, *Proceedings - 2020 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2020*, pp. 64–71, Oct. 2020. DOI: 10.1109/CYBERC49757.2020.00020.

[29] G. Lyon, *Nmap*. [Online]. Available: https://nmap.org/.

[30] J. H. Cho, D. P. Sharma, H. Alavizadeh, *et al.*, „Toward Proactive, Adaptive Defense: A Survey on Moving Target Defense“, *IEEE Communications Surveys and Tutorials*, vol. 22, no. 1, pp. 709–745, Jan. 2020, ISSN: 1553877X. DOI: 10.1109/COMST.2019.2963791.

[31] N. Bandi, H. Tajbakhsh, and M. Analoui, „FastMove: Fast IP switching Moving Target Defense to mitigate DDOS Attacks“, *2021 IEEE Conference on Dependable and Secure Computing, DSC 2021*, Jan. 2021. DOI: 10.1109/DSC49826.2021.9346278.

[32] J. H. Jafarian, E. Al-Shaer, and Q. Duan, „An Effective Address Mutation Approach for Disrupting Reconnaissance Attacks“, *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2562–2577, Dec. 2015, ISSN: 15566013. DOI: 10.1109/TIFS.2015.2467358.

[33] W. Iqbal, H. Abbas, M. Daneshmand, B. Rauf, and Y. A. Bangash, „An In-Depth Analysis of IoT Security Requirements, Challenges, and Their Countermeasures via Software-Defined Security“, *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10 250–10 276, Oct. 2020, ISSN: 23274662. DOI: 10.1109/JIOT.2020.2997651.

[34] N. M. Karie, N. M. Sahri, W. Yang, C. Valli, and V. R. Kebande, „A Review of Security Standards and Frameworks for IoT-Based Smart Environments“, *IEEE Access*, vol. 9, pp. 121 975–121 995, 2021, ISSN: 21693536. DOI: 10.1109/ACCESS.2021.3109886.

[35] Y. Zheng, A. Pal, S. Abuadbba, S. R. Pokhrel, S. Nepal, and H. Janicke, „Towards IoT Security Automation and Orchestration“, *Proceedings - 2020 2nd IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications, TPS-ISA 2020*, pp. 55–63, Oct. 2020. DOI: 10.1109/TPS-ISA50397.2020.00018.

[36] D. E. Kouicem, A. Bouabdallah, and H. Lakhlef, „Internet of things security: A top-down survey“, *Computer Networks*, vol. 141, pp. 199–221, Aug. 2018, ISSN: 1389-1286. DOI: 10.1016/J.COMNET.2018.03.012.

[37] R. Zhuang, S. A. DeLoach, and X. Ou, „Towards a theory of moving target defense“, *Proceedings of the ACM Conference on Computer and Communications Security*, vol. 2014-November, no. November, pp. 31–40, Nov. 2014, ISSN: 15437221. DOI: 10.1145/2663474.2663479.

[38] G. l. Cai, B. s. Wang, W. Hu, and T. z. Wang, „Moving target defense: state of the art and characteristics“, *Frontiers of Information Technology and Electronic Engineering*, vol. 17, no. 11, pp. 1122–1153, Nov. 2016, ISSN: 20959230. DOI: 10.1631/FITEE.1601321/TABLES/7.

[39] J. B. Hong and D. S. Kim, „Assessing the Effectiveness of Moving Target Defenses Using Security Models“, *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 163–177, Mar. 2016, ISSN: 19410018. DOI: 10.1109/TDSC.2015.2443790.

[40] T. Preiss, M. Sherburne, R. Marchany, and J. Tront, „Implementing dynamic address changes in ContikiOS“, *International Conference on Information Society, i-Society 2014*, pp. 222–227, Jan. 2015. DOI: 10.1109/I-SOCIETY.2014.7009047.

[41] M. Ayrault, E. Borde, and U. Kühne, „Run or Hide? Both! A method based on IPv6 address switching to Escape while being hidden“, *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 47–56, Nov. 2019, ISSN: 15437221. DOI: 10.1145/3338468.3356827.

[42] Y. B. Luo, B. S. Wang, X. F. Wang, X. F. Hu, G. L. Cai, and H. Sun, „RPAH: Random port and address hopping for thwarting internal and external adversaries“, *Proceedings - 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2015*, vol. 1, pp. 263–270, Dec. 2015. DOI: 10.1109/TRUSTCOM.2015.383.

[43] K. Mahmood and D. M. Shila, „Moving target defense for Internet of Things using context aware code partitioning and code diversification“, *2016 IEEE 3rd World Forum on Internet of Things, WF-IoT 2016*, pp. 329–330, Feb. 2017. DOI: 10.1109/WF-IOT.2016.7845457.

[44] S. Antonatos, P. Akritidis, E. P. Markatos, and K. G. Anagnostakis, „Defending against hitlist worms using network address space randomization“, *Computer Networks*, vol. 51, no. 12, pp. 3471–3490, Aug. 2007, ISSN: 1389-1286. DOI: 10.1016/J.COMNET.2007.02.006.

[45] J. H. Jafarian, E. Al-Shaer, and Q. Duan, „OpenFlow Random Host Mutation: Transparent Moving Target Defense using Software Defined Networking“, *Proceedings of the first workshop on Hot topics in software defined networks - HotSDN '12*, 2012. DOI: 10.1145/2342441.

[46] D. Kewley, R. Fink, J. Lowry, and M. Dean, „Dynamic approaches to thwart adversary intelligence gathering“, *Proceedings - DARPA Information Survivability Conference and Exposition II, DISCEX 2001*, vol. 1, pp. 176–185, 2001. DOI: 10.1109/DISCEX.2001.932214.

[47] D. P. Sharma, D. S. Kim, S. Yoon, H. Lim, J. H. Cho, and T. J. Moore, „FRVM: Flexible Random Virtual IP Multiplexing in Software-Defined Networks“, *Proceedings - 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications and 12th IEEE International Conference on Big Data Science and Engineering, Trustcom/BigDataSE 2018*, pp. 579–587, Sep. 2018. DOI: 10.1109/TRUSTCOM/BIGDATASE.2018.00088.

[48] Y. Y. Chen, I. H. Liu, C. C. Wu, C. G. Liu, and J. S. Li, „Dynamic Interval Strategy for MT6D in IoT Systems“, *2021 IEEE International Conference on Electronic Communications, Internet of Things and Big Data, ICEIB 2021*, pp. 36–39, 2021. DOI: 10.1109/ICEIB53692.2021.9686425.

[49] R. E. Navas, F. Cuppens, N. Boulahia Cuppens, L. Toutain, and G. Z. Papadopoulos, „MTD, Where Art Thou? A Systematic Review of Moving Target Defense Techniques for IoT“, *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 7818–7832, May 2021, ISSN: 23274662. DOI: 10.1109/JIOT.2020.3040358.

[50] A. O. Hamada, M. Azab, and A. Mokhtar, „Honeypot-like Moving-target Defense for secure IoT Operation“, *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference, IEMCON 2018*, pp. 971–977, Jan. 2019. DOI: 10.1109/IEMCON.2018.8614925.

[51] S. Lee, H. K. Kim, and K. Kim, „Ransomware protection using the moving target defense perspective“, *Computers & Electrical Engineering*, vol. 78, pp. 288–299, Sep. 2019, ISSN: 0045-7906. DOI: 10.1016/J.COMPELECENG.2019.07.014.

[52] F. Song, Y. T. Zhou, Y. Wang, T. M. Zhao, I. You, and H. K. Zhang, „Smart collaborative distribution for privacy enhancement in moving target defense“, *Information Sciences*, vol. 479, pp. 593–606, Apr. 2019, ISSN: 0020-0255. DOI: 10.1016/J.INS.2018.06.002.

[53] G. Kesidis, Y. Shan, D. Fleck, A. Stavrou, and T. Konstantopoulos, „An adversarial coupon-collector model of asynchronous moving-target defense against botnet reconnaissance*“, *MALWARE 2018 - Proceedings of the 2018 13th International Conference on Malicious and Unwanted Software*, pp. 61–67, Mar. 2019. DOI: 10.1109/MALWARE.2018.8659359.

[54] A. Clark, K. Sun, L. Bushnell, and R. Poovendran, „A game-theoretic approach to IP address randomization in decoy-based cyber defense“, *International Conference on Decision and Game Theory for Security*, pp. 3–21, 2015, ISSN: 16113349. DOI: 10.1007/978-3-319-25594-1{_}1.

[55] B. Lucas, E. W. Fulp, D. J. John, and D. Cañas, „An initial framework for evolving computer configurations as a moving target defense“, *ACM International Conference Proceeding Series*, pp. 69–72, 2014. DOI: 10.1145/2602087.2602100.

[56] J. Zheng and A. S. Namin, „Enforcing optimal moving target defense policies“, *Proceedings - International Computer Software and Applications Conference*, vol. 1, pp. 753–759, Jul. 2019, ISSN: 07303157. DOI: 10.1109/COMPSAC.2019.00112.

[57] S. Rajendran, R. Calvo-Palomino, M. Fuchs, *et al.*, „Electrosense: Open and Big Spectrum Data“, *IEEE Communications Magazine*, vol. 56, no. 1, pp. 210–217, Jan. 2018, ISSN: 01636804. DOI: 10.1109/MCOM.2017.1700200.

[58] Y. Sun, W. Ji, J. Weng, B. Zhao, Y. Li, and X. Wu, „Selection of Optimal Strategy for Moving Target Defense Based on Signal Game“, *ACM International Conference Proceeding Series*, pp. 28–32, Dec. 2020. DOI: 10.1145/3444370.3444543.

[59] H. Zhang, J. Tan, X. Liu, and J. Wang, „Moving Target Defense Decision-Making Method: A Dynamic Markov Differential Game Model“, *MTD 2020 - Proceedings of the 7th ACM Workshop on Moving Target Defense*, pp. 21–29, Nov. 2020. DOI: 10.1145/3411496.3421222.

[60] C. Lei, D. H. Ma, and H. Q. Zhang, „Optimal strategy selection for moving target defense based on markov game“, *IEEE Access*, vol. 5, pp. 156–169, 2017, ISSN: 21693536. DOI: 10.1109/ACCESS.2016.2633983.

[61] H. Maleki, S. Valizadeh, W. Koch, A. Bestavros, and M. Van Dijk, „Markov modeling of moving target defense games“, *MTD 2016 - Proceedings of the 2016 ACM Workshop on Moving Target Defense, co-located with CCS 2016*, pp. 81–92, Oct. 2016. DOI: 10.1145/2995272.2995273.

[62] C. Lei, D. H. Ma, H. Q. Zhang, and L. M. Wang, „Moving target network defense effectiveness evaluation based on change-point detection“, *Mathematical Problems in Engineering*, vol. 2016, 2016, ISSN: 15635147. DOI: 10.1155/2016/6391502.

[63] J. l. Tan, C. Lei, H. q. Zhang, and Y. q. Cheng, „Optimal strategy selection approach to moving target defense based on Markov robust game“, *Computers & Security*, vol. 85, pp. 63–76, Aug. 2019, ISSN: 0167-4048. DOI: 10.1016/J.COSE.2019.04.013.

[64] H. Zhang, K. Zheng, X. Wang, S. Luo, and B. Wu, „Efficient strategy selection for moving target defense under multiple attacks“, *IEEE Access*, vol. 7, pp. 65 982–65 995, 2019, ISSN: 21693536. DOI: 10.1109/ACCESS.2019.2918319.

[65] J. Tan, H. Zhang, H. Zhang, H. Hu, C. Lei, and Z. Qin, „Optimal temporospatial strategy selection approach to moving target defense: A FlipIt differential game model“, *Computers & Security*, vol. 108, p. 102 342, Sep. 2021, ISSN: 0167-4048. DOI: 10.1016/J.COSE.2021.102342.

[66] M. S. Kang, S. B. Lee, and V. D. Gligor, „The crossfire attack“, *Proceedings - IEEE Symposium on Security and Privacy*, pp. 127–141, 2013, ISSN: 10816011. DOI: 10.1109/SP.2013.19.

[67] G. Rajakumaran and N. Venkataraman, „Performance assessment of hybrid MTD for DoS mitigation in public cloud“, *International Journal of Intelligent Networks*, vol. 2, pp. 140–147, Jan. 2021, ISSN: 2666-6030. DOI: 10.1016/J.IJIN.2021.09.003.

[68] J. Cedeno, *Mitigating cyberattacks affecting resource-constrained devices through moving target defense (mtd) mechanisms*, A. Huertas Celdran, J. von der Assen, and B. Stiller, Eds., 2022.

[69] nccgroup, *The Tick*, https://github.com/nccgroup/thetick, 2020, [Online; accessed Sep 14 2022].

[70] J. Koritar, *backdoor*, https://github.com/jakoritarleite/backdoor, 2020, [Online; accessed Sep 14 2022].

[71] SkryptKiddie, *httpBackdoor*, https://github.com/SkryptKiddie/httpBackdoor, 2020, [Online; accessed Sep 14 2022].

[72] HAMMERZEIT, *BASHLITE*, https://github.com/hammerzeit/BASHLITE, 2016, [Online; accessed Sep 14 2022].

[73] unix-thrust, *BEURK*, https://github.com/unix-thrust/beurk, 2017, [Online; accessed Sep 14 2022].

[74] Error996, *bedevil*, https://github.com/Error996/bdvl, 2022, [Online; accessed Sep 14 2022].

[75] jimmy-ly00, *Ransomware-PoC*, https://github.com/jimmy-ly00/Ransomware-PoC, 2021, [Online; accessed Sep 14 2022].

[76] D. Palmer, S. Fazzari, and S. Wartenberg, „Defense systems and IoT: Security issues in an era of distributed command and control“, *Proceedings of the ACM Great Lakes Symposium on VLSI, GLSVLSI*, vol. 18-20-May-2016, pp. 175–179, May 2016. DOI: 10.1145/2902961.2903038.

[77] E. D. Jovanovic and P. V. Vuletic, „Analysis and Characterization of IoT Malware Command and Control Communication“, *27th Telecommunications Forum, TELFOR 2019*, Nov. 2019. DOI: 10.1109/TELFOR48224.2019.8971194.

[78] W. J. Tsaur and Y. C. Chen, „Exploring rootkit detectors' vulnerabilities using a new windows hidden driver based rootkit“, *Proceedings - SocialCom 2010: 2nd IEEE International Conference on Social Computing, PASSAT 2010: 2nd IEEE International Conference on Privacy, Security, Risk and Trust*, pp. 842–848, 2010. DOI: 10.1109/SOCIALCOM.2010.127.

[79] D. Wieers, *Dstat*, 2022. [Online]. Available: http://dag.wiee.rs/home-made/dstat/, [Online; accessed Sep 14 2022].

[80] L. Cui, F. R. Yu, and Q. Yan, „When big data meets software-defined networking: SDN for big data and big data for SDN“, *IEEE Network*, vol. 30, no. 1, pp. 58–65, Jan. 2016, ISSN: 08908044. DOI: 10.1109/MNET.2016.7389832.

Abbreviations

API	Application Program Interface
AWS	Amazon Web Service
CPU	Central Processing Unit
CSG	Communication Systems Group
CSV	Comma-Separated Values
C&C	Command & Control
DDoS	Distributed Denial of Service
DNS	Domain Name System
DoS	Denial of Service
EC2	Elastic Compute Cloud
FNR	False Negative Rate
FPR	False Positive Rate
GB	Gigabyte
IIoT	Industrial Internet-of-Things
IoT	Internet-of-Things
IP	Internet Protocol
I/O	Input / Output
KB	Kilobyte
MB	Megabyte
MDP	Markov Decision Processes
ML	Machine Learning
MTD	Moving Target Defense
MT6D	Moving Target IPv6 Defense
OS	Operating System
POC	Proof of Concept
RAM	Random Access Memory
SDN	Software-Defined Networking
SDR	Software-Defined Radio
StraSelA	Strategy Selection Agent
Tbps	Terabit per second
TCP	Transmission Control Protocol
TNR	True Negative Rate
TPR	True Positive Rate
UDP	User Datagram Protocol
UZH	University of Zurich
VM	Virtual Machine

6LoWPAN Low-Powered Wireless Personal Area Networks

Glossary

Botnet A botnet is an army of infected devices, sometimes called zombies, that can be used to launch malicious actions such as DDoS attack (See **Distributed Denial of Service**) [15].

Crossfire The term Crossfire refers to a sophisticated botnet attack, that uses a small number of bots that cause low-intensity traffic on certain servers to impair.

Distributed Denial of Service Distributed Denial of Service refers to a malware attack in which many different devices attempt to access a server or service simultaneously, ultimately overloading it.

False Positive Rate The False Positive Rate defines the ratio of samples incorrectly identified as positive to positive samples.

Game Theory Game Theory is a field that deals with optimization, rational decisions and behavior. It can be applied in various domains.

Genetic Algorithm Genetic Algorithm generates multiple solutions for a optimization problem and evaluates their performance, so called fitness, to find the best possible in an iterative process.

IP Shuffling See **MTD technique**.

Mirai Mirai Was responsible for a very large DDoS attack in 2016 that was launched from countless IoT devices and targeted the DNS service Dyn [17].

MTD technique MTD technique designates a countermeasure against malware, which satisfies the MTD paradigm, e.g., performing IP shuffling.

Ransomware Ransomware is a type of malicious software, that encrypts your data in order to subsequently carry out digital blackmail.

Software Defined Network Software Defined Network refers to the configuration of a network by software, that introduces centralization [80].

True Negative Rate The True Negative Rate is a term used in statistics, which indicates the ratio of correctly identified negative samples with respect to negatives of the sample space.

List of Figures

List of Tables

Appendix A

Installation Guidelines

This section provides all the necessary information to successfully install and start the MTD strategy selection agent *MTD StraSelA*, the monitoring script, and the evaluation script used during the mixed evaluation. Finally, the malware setup explains the installation and process to launch the programs for each considered malware type. Furthermore, the following diagram shows the project structure and the most important folders and files:

```
MTDStrategySelectionAgent
├── agent
│   ├── MTD
│   ├── mtd_strategy_selection_agent.py
│   └── policy_db.csv
├── attacker
│   └── httpBackdoor_attack_script.py
├── monitoring-script
│   └── monitoring-script.sh
├── utils
└── visualizations
    ├── 01-policy-synthesis
    └── 02-evaluation
        ├── individual
        ├── mixed
        └── overhead
```

A.1 MTD Strategy Selection Agent

MTD StraSelA

Listing A.1: Install *MTD StraSelA*

```
# install git
apt-get install git

# navigate to the desired installation directory
# e.g. /root/
cd root/

# clone repository
git clone https://github.com/HuberNicolas/
   MTDStrategySelectionAgent

# install Dstat
apt-get install dstat -y

# install MTD dependencies
apt-get install arp-scan net-tools

# install the Python dependencies
pip3 install pyyaml numpy pandas psutil requests
   setproctitle

# if the first command don't work try
python3 -m pip install pyyaml numpy pandas psutil
   requests setproctitle
```

Listing A.2: Start *MTD StraSelA*

```
# navigate to the directory using the cd command
cd root/MTDStrategySelectionAgent/agent/

# run it with
python3 mtd_strategy_selection_agent.py
```

A.2 Monitoring script

Listing A.3: Start *monitoring_script.py*

```
# navigate to the directory using the cd command
cd root/MTDStrategySelectionAgent/monitoring-script/

```

```
4       # make it executable
5       sudo chmod +x monitoring-script.sh
6
7       # run it
8       ./monitoring-script.sh
```

A.3 Evaluation attack script

Listing A.4: Start *evaluation_attack_script.py*

```
1       # navigate to the directory using the cd command
2       cd root/MTDStrategySelectionAgent/attacker/
3
4       # install the following python dependencies
5       pip3 install requests
6
7       # make sure the malware is installed and has the correct
           configurations
8
9       # run it with
10      python3 evaluation_attack_script.py
```

A.4 Malware Setup

Regarding the malware setup, it is probably a good idea to create a single directory that contains all malware samples and install the text editor *nano*.

Listing A.5: Start *evaluation_attack_script.py*

```
1       cd root/
2       mkdir Malware
3       cd Malware/
4
5       # install nano
6       apt-get install nano
```

Afterward, each malware can be installed.

A.4.1 The Tick

Client
The installation of ”The Tick” requires some dependencies. After the installation, the

make command can be used. The result is an executable that starts the client. The command has flags for ADDR and PORT that specify the IP address and the port of the corresponding server.

Listing A.6: The Tick: Install and start client

```
# install dependencies
sudo apt-get install libcurl4-openssl-dev

# clone repository
git clone https://github.com/nccgroup/thetick

# ... navigate to the directory
# execute the following commands
cd thetick/src
make clean
make

# ./ticksvc [IP address] [port], e.g.,
cd ..
cd bin/
./ticksvc 192.168.1.5 6667
```

Server

The server of "The Tick" requires Python 2.7. A nice workaround was proposed in the wiki of MTDFramework [14]: First, install the program *curl* to afterwards install pip for Python 2.7. The missing packages for the server can then be installed.

Listing A.7: The Tick: Install and start server

```
# install curl
apt-get install curl

# get download pip for Python 2.7 and install it
curl https://bootstrap.pypa.io/pip/2.7/get-pip.py --
   output get-pip.py
sudo python2 get-pip.py

# install dependencies
pip2 install colorama
pip2 install argparse-color-formatter
pip2 install texttable

# clone repository
git clone https://github.com/nccgroup/thetick

# ... navigate to the directory
# start the server
python2 tick.py
```

```
19
20      # ... wait for a bot that connects
21
22      # select bot e.g., 0
23      use 0
24
25      # download malicious software via wget e.g.,
26      download https://www.python.org/ftp/python/3.10.6/Python
           -3.10.6.tar.xz malicious_software.zip
```

A.4.2 bdvl

Client
For this rootkit can be configured: After cloning the repository, the *setup.py* can be modified. I changed line 8 to set the password "password" instead of *None* and line 18 to port 1234.

Listing A.8: bdvl: Install and start client

```
1       # clone repository
2       git clone https://github.com/Error996/bdvl
3
4       # ... navigate to the directory
5       # configure setup.py
6       nano setup.py
7
8       # run the following commands
9       cd etc/
10      chmod +x auto.sh
11      cd ..
12      sh etc/depinstall.sh && make
13      etc/auto.sh build/super.b64
14      cd bdvl/
15      systemctl restart sshd
```

Server
The server needs some dependencies that can be easily installed with *apt-get*. Next, the connection can be established with the command below and the three parameters for the IP address, the port and the password.

Listing A.9: bdvl: Install and start server

```
1       # install dependencies
2       sudo apt-get install hping3
3       sudo apt-get install socat
4
5       # connect to client
```

```
    # sudo nc [IP address] 22 -p [port] : [password]
    sudo nc 192.168.1.5 22 -p 1234 #enter default password
    : password
```

A.4.3 BEURK

Client
For launching "BEURK", some dependencies are needed after the *clone* command. It is possible to make some changes in the config file. Finally, "BEURK" can be installed with a simple command.

Listing A.10: BEURK: Install and start client

```
    # clone repository
    git clone [https://github.com/unix-thrust/beurk](https://
       github.com/unix-thrust/beurk)

    # install dependencies
    apt-get install libpcap-dev libpam-dev libssl-dev
    apt-get update --fix-missing

    # ... navigate to the directory
    # configure beurk.conf
    nano beurk.conf

    # install
    make && make infect
```

A.4.4 backdoor

Client
First of all, in both *client.py* and *server.py*, the IP address of the designated server needs to be adjusted. Furthermore, the client of "backdoor" may need to be slightly modified: [14] suggested in his wiki to change, depending on the installed Python version, line 59 to:
s.send(str.encode("Welcome Dad \nI \`m " + str(Client.get_ip())+"\n \n"))
After the server recognizes a client, the server can use the *shell* command to open an "evil" shell to perform malicious actions.

Listing A.11: backdoor: Install and start client

```
    # clone repository
    git clone https://github.com/jakoritarleite/backdoor

    # ... navigate to the directory
```

```
5      # configure client.py
6      nano client.py
7
8      # start the client
9      sudo python3 client.py
```

Server

Listing A.12: backdoor: Install and start server

```
1      # clone repository
2      git clone https://github.com/jakoritarleite/backdoor
3
4      # ... navigate to the directory
5      # configure server.py
6      nano client.py
7
8      # start the server
9      sudo python3 server.py
10
11     # ... wait for a bot that connects
12     shell
13
14     # download malicious software via wget e.g.,
15     wget --no-check-certificate https://www.python.org/ftp/
          python/3.10.6/Python-3.10.6.tar.xz
```

A.4.5 BASHLITE

Client

After the cloning of "BASHLITE", [14] proposes to adjust line 72 to the IP address of the server and line 1879 of *client.c* to:
return "UNKN";
After these modifications, the client and server can be built using the *gcc* command. Finally, the binaries can be executed, whereas the server commands need one parameter for the port and one for the number of threads that are used for the process.

Listing A.13: BASHLITE: Install and start client

```
1      # clone repository
2      git clone https://github.com/hammerzeit/BASHLITE
3
4      # ... navigate to the directory
5      # configure client.c
6      nano client.c
7
```

```
8     # build
9     gcc -o client client.c
10
11    # start the client
12    ./client
```

Server

Listing A.14: BASHLITE: Install and start server

```
1     # clone repository
2     git clone https://github.com/hammerzeit/BASHLITE
3
4     # ... navigate to the directory
5     # build
6     gcc -pthread -o server server.c
7
8     # start the server
9     #./server [port] [threads]
10    ./server 6667 10
```

A.4.6 HttpBackdoor

Client

"HttpBackdoor" can also be modified to adjust the Port. To do so, line 6 of *httpbackdoor.py* can be changed. After launching *httpBackdoor.py*, the attacker script created for this thesis can be also started.

Listing A.15: HttpBackdoor: Install and start client

```
1     # clone repository
2     git clone https://github.com/SkryptKiddie/httpBackdoor
3
4     # ... navigate to the directory
5     # start the client
6     python3 httpBackdoor.py
```

Listing A.16: HttpBackdoor: Install and start server

```
1     # ... navigate to the directory
2     cd root/MTDStrategySelectionAgent/attacker/
3
4     # set the IP address and port
5     nano httpBackdoor_attack_script.py
6
7     # start the server
8     python3 httpBackdoor_attack_script.py
```

A.4.7 Ransomware-PoC

After downloading the "Ransomware-PoC" repository, files can be encrypted with a simple command that requires a parameter for the path.

Listing A.17: Ransomware-PoC: Install and execution

```
# clone repository
git clone https://github.com/jimmy-ly00/Ransomware-PoC

# ... navigate to the directory
# python3 main.py -p [path] -e
python3 main.py -p "/root/sample-data" -e
```

Appendix B

Contents of the zip File

This section lists all components of the zip file and gives a brief description.

BA_Nicolas_Huber.pdf The final report as a pdf file.

BA_Nicolas_Huber.zip The complete LaTeX source code of the final report. This includes all figures, tables, and listings.

Midterm_Nicolas_Huber.pptx The midterm presentation as a PowerPoint file.

MTDStrategySelectionAgent.zip This zip file contains the complete source code of the project, the policy database, and all visualizations, which include:

- MTD Strategy Selection Agent *MTD StraSelA*
- Policy database as a CSV file
- Evaluation attack script
- httpBackdoor attack script
- Monitoring script
- Policy synthesis visualization script
- Individual evaluation visualization script
- Mixed evaluation visualization script
- Overhead evaluation visualization script
- Helper scripts
- Visualization generated by the scripts as mentioned earlier