



**INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA**

**Área Departamental de Engenharia Eletrónica e Telecomunicações e de Computadores**

**Proactive Cybersecurity tailoring through deception  
techniques**

**LUÍS MARIA DE FIGUEIREDO CRUZ GUERRA**

**(Licenciado em Engenharia Informática e de Computadores)**

Dissertação de natureza científica para obtenção do grau de Mestre em Engenharia  
Informática e de Computadores

**Orientador:**

Professor Doutor Luís Carlos Gonçalves

**Júri:**

**Presidente:** Professor Doutor José Simão

**Vogais:** Professor Doutor Nuno Cruz  
Professor Doutor Luís Carlos Gonçalves

**Dezembro 2023**



# Acknowledgments

This thesis owes its feasibility to the invaluable guidance, patience, advice, and companionship of the people mentioned here.

First and foremost, I would like to express my profound gratitude to my thesis' advisor, Professor Luís Carlos Gonçalves. His constant support, priceless input, utmost respect for me and encouragement throughout this thesis were key aspects which not only enabled the completion of this thesis but also enriched my personal and professional growth. Thank you.

To my family, especially my parents, for always being there for me during all my journey and believing in me, even in my most difficult periods. Without them, nothing that I accomplished would be possible.

To my girlfriend Inês, for her unlimited support, kindness and understanding, never once ceasing to amaze me with her unwavering love.

Thank you João for accompanying me, almost daily, since this thesis' day one. Your presence and companionship were a valuable asset throughout this venture.

To all my hometown friends Luís, André, Alexandre, Pinto, Laura, Rita, Daniel, Tiago, and others, who I occasionally bailed on due to the responsibility of doing this thesis. Thank you for being a part of my life for so long and supporting me through everything in life.

To all my friends and colleagues which I met on Instituto Superior de Engenharia de Lisboa Pedro, Tiago, Daniel, Ana, Wilson, Miguel, José, and others, which made my college years better.

Finally, I would like to express my deepest appreciation for all the professors I have encountered in Instituto Superior de Engenharia de Lisboa which shaped my academic path. Every one of them contributed to this thesis and to my self-development.

I'm extremely thankful to each and every one of you, and to the dear readers, to whom I dedicate this thesis.

*“All warfare is based on deception.*

*Hence, when we are able to attack, we must seem unable,  
when using our forces, we must appear inactive;  
when we are near, we must make the enemy believe we are far away;  
when far away, we must make him believe we are near”*

*Sun Tzu*

# Abstract

A proactive approach to cybersecurity can supplement a reactive posture by helping businesses to handle security incidents in the early phases of an attack. Organizations can actively protect against the inherent asymmetry of cyber warfare by using proactive techniques such as cyber deception. The intentional deployment of misleading artifacts to construct an infrastructure that allows real-time investigation of an attacker's patterns and approaches without compromising the organization's principal network is what cyber deception entails. This method can reveal previously undiscovered vulnerabilities, referred to as zero-day vulnerabilities, without interfering with routine corporate activities. Furthermore, it enables enterprises to collect vital information about the attacker that would otherwise be difficult to access. However, putting such concepts into practice in real-world circumstances involves major problems.

This study proposes an architecture for a deceptive system, culminating in an implementation that deploys and dynamically customizes a deception grid using Software-Defined Networking (SDN) and network virtualization techniques. The deception grid is a network of virtual assets with a topology and specifications that are pre-planned to coincide with a deception strategy. The system can trace and evaluate the attacker's activity by continuously monitoring the artifacts within the deception grid. Real-time refinement of the deception plan may necessitate changes to the grid's topology and artifacts, which can be assisted by software-defined networking's dynamic modification capabilities.

Organizations can maximize their deception capabilities by merging these processes with advanced cyber-attack detection and classification components. The effectiveness of the given solution is assessed using numerous use cases that demonstrate its utility.

## *Keywords*

Cyber Deception; Deception Grid; Proactive Cybersecurity; Software-Defined Networking; Cyber Warfare; Network Virtualization.

# Resumo

Uma abordagem proativa à cibersegurança pode complementar uma postura reativa ajudando as empresas a lidar com incidentes de segurança em fases iniciais. As organizações podem proteger-se ativamente contra a assimetria inerente à guerra cibernética através do uso de técnicas proativas, como por exemplo a ciber *deception*. A implantação intencional de artefactos enganosos para construir uma infraestrutura que permite a investigação em tempo real dos padrões e abordagens de um atacante sem comprometer a rede principal da organização é o propósito da *deception* cibernética. Esta metodologia pode revelar vulnerabilidades por descobrir, conhecidas como vulnerabilidades de dia-zero, sem interferir com as atividades de rotina da organização. Além disso, permite às empresas a extração de informações vitais sobre o atacante que, de outra forma, seriam difíceis de adquirir. No entanto, colocar estes conceitos em prática em circunstâncias reais constitui problemas de grande ordem.

Este estudo propõe uma arquitetura para um sistema informático de *deception*, que culmina numa implementação que implanta e adapta dinamicamente uma rede enganosa através do uso de técnicas de redes definidas por *software* e de virtualização de rede. A rede ilusora é uma rede de ativos virtuais com uma topologia e especificações pré-planeadas, coincidentes com uma estratégia de *deception*. O sistema pode rastrear e avaliar a atividade do atacante através da monitorização contínua dos artefactos da rede. O refinamento em tempo real do plano de *deception* pode exigir alterações na topologia e nos artefactos da rede, possíveis devido às capacidades de modificação dinâmica das redes definidas por *software*.

As organizações podem maximizar as suas capacidades de *deception* ao combinar estes processos com componentes avançados de deteção e classificação de ataques informáticos. A eficácia da solução proposta é avaliada usando vários casos de estudo que demonstram a sua utilidade.

## ***Palavras-Chave***

Ilusão cibernética; Rede de *Deception*; Cibersegurança Proativa; Redes Definidas por *Software*; Guerra Cibernética; Virtualização de Redes.

# Content

<b>Acknowledgments</b>	<b>ii</b>
<b>Abstract</b>	<b>iv</b>
<b>Resumo</b>	<b>v</b>
<b>List of Figures</b>	<b>ix</b>
<b>List of Tables</b>	<b>xii</b>
<b>List of Listings</b>	<b>xiii</b>
<b>Acronyms</b>	<b>xiv</b>
<b>1. Introduction</b>	<b>1</b>
1.1. Context	2
1.2. Work Objectives	3
1.3. Contributions	4
1.4. Document Structure	4
<b>2. Background and Related Work</b>	<b>6</b>
2.1. Defining Deception	7
2.2. Deception Taxonomy and Techniques	10
2.2.1. Deception by Method	13
2.2.2. Deception by Commission-Omission	17
2.2.3. Deception Based on Level of Sophistication	18
2.3. Deception in Practice	19
2.4. Deception Technology in Computer Security	26
2.4.1. Virtualization and Containerization	34
2.4.2. Software Defined Networking	35
2.5. Honeypots and Honeynets	36
2.6. Deception Grids	44
2.7. Cyber Intelligence and Deception Planning	47
<b>3. Proposed Solution</b>	<b>54</b>
3.1. System Architecture	55
3.1.1. Base Architecture (Research Deployment)	56
3.1.2. Extended Architecture (Production Deployment)	69
3.2. Proposed System and Deception Techniques	74
<b>4. System Implementation</b>	<b>76</b>
4.1. System Components	77

## Proactive Cybersecurity tailoring through deception techniques

4.1.1.	Sophisticated Detection and Auto Attack Analyst	82
4.1.2.	Attack Info Adapter	83
4.1.3.	Virtual Asset Repository	86
4.1.4.	Virtual Asset Network and SDN Controller	88
4.1.5.	Deception Generator	93
4.1.6.	Static Deception Planner	95
4.1.7.	Asset Monitoring	99
4.1.8.	Intelligence	100
4.2.	Deception Plans	103
4.3.	Threat Modeling	106
4.3.1.	Threat Model	107
4.3.2.	Hardening and Resilience Concerns	109
<b>5.</b>	<b>System Validation</b>	<b>111</b>
5.1.	Test Environment	112
5.2.	Use Case Evaluation	112
5.2.1.	Production Deployment	113
5.2.2.	Research Deployment	118
5.2.3.	System Information Retrieval	120
5.2.4.	Deception Grid's Log Observability	122
5.2.5.	Deception Strategy Adjustments	124
<b>6.</b>	<b>Conclusions and Future Work</b>	<b>129</b>
6.1.	Conclusions	130
6.2.	Future Work	132
<b>7.</b>	<b>References</b>	<b>133</b>





# List of Figures

<b>Figure 1</b> - Deception’s subsidiary concepts .....	11
<b>Figure 2</b> - Interrelationship of truth, lies and deception .....	11
<b>Figure 3</b> - Taxonomy by method.....	13
<b>Figure 4</b> - CIA Triad .....	27
<b>Figure 5</b> - The OSI Model.....	28
<b>Figure 6</b> - Common security attacks in the OSI layer model .....	28
<b>Figure 7</b> - The seven layers of cybersecurity .....	29
<b>Figure 8</b> - Deployment types.....	34
<b>Figure 9</b> - SDN architecture .....	36
<b>Figure 10</b> - Honeypot classification.....	37
<b>Figure 11</b> - Low-Interaction vs High-Interaction Honeypots.....	38
<b>Figure 12</b> - Honeypots Threat Modelling.....	39
<b>Figure 13</b> - Overview of counter measures against honeypots.....	40
<b>Figure 14</b> - Deception based technologies vs typical security mechanisms. ....	41
<b>Figure 15</b> - Generation III HoneyNet architecture .....	42
<b>Figure 16</b> - HoneyNet classification scheme.....	43
<b>Figure 17</b> - Example of a typical deception system.....	45
<b>Figure 18</b> - Communication cycle. ....	47
<b>Figure 19</b> - The deception planning loop.....	48
<b>Figure 20</b> - Revised deception process .....	49
<b>Figure 21</b> - Deception strategy development and execution .....	50
<b>Figure 22</b> - From raw data to Intelligence.....	51
<b>Figure 23</b> - System’s base architecture applicability. ....	57
<b>Figure 24</b> - Example of deception Grid(s) public IP address management. ....	59
<b>Figure 25</b> - Active Deception System base architecture .....	61
<b>Figure 26</b> - Intelligence component as a gateway.....	62
<b>Figure 27</b> - Overview of research deployment via Intelligence's API .....	63
<b>Figure 28</b> - Deception Generator's responsibility within the deceptive system.....	64
<b>Figure 29</b> - Asset provisioning through Virtual Asset Repository.....	65
<b>Figure 30</b> - The role of Asset Monitoring component within the deceptive process .....	68
<b>Figure 31</b> - System’s extended architecture applicability .....	69
<b>Figure 32</b> - Detection and enforcing mechanisms prompting the attacker to the deception grid.....	70
<b>Figure 33</b> - Active Deception System extended architecture .....	71
<b>Figure 34</b> - Attack Info Adapter responsibility as a broker.....	72
<b>Figure 35</b> - Parallel dispatchment of attack information through Attack Info Adapter.....	73
<b>Figure 36</b> - Proposed system's implementation with an example of a deception grid.....	78

## Proactive Cybersecurity tailoring through deception techniques

<b>Figure 37</b> - MITRE ATT&CK classification .....	80
<b>Figure 38</b> - CAPEC attack classification. ....	81
<b>Figure 39</b> - Simulation of Sophisticated Detection and Auto Attack Analyst.....	83
<b>Figure 40</b> - Attack Info Adapter implementation and associated mechanisms.....	83
<b>Figure 41</b> - Attack Info Adapter's method flow.....	85
<b>Figure 42</b> - Virtual Asset Network and SDN implementation through Containernet and ONOS, respectively.....	91
<b>Figure 43</b> - Host in the Containernet network and associated network interfaces .....	92
<b>Figure 44</b> - Deception Generator's responsibilities and interaction with Static Deception Planner. ....	94
<b>Figure 45</b> - Deception playbook selection process. ....	95
<b>Figure 46</b> - Static Deception Planner and associated mechanisms .....	97
<b>Figure 47</b> - Intelligence's implementational mechanisms .....	103
<b>Figure 48</b> - Proposed system's threat model STRIDE diagram. ....	107
<b>Figure 49</b> - Proposed system's code vulnerability assessment by Snyk .....	108
<b>Figure 50</b> - Proposed system's code vulnerability assessment by CodeQL.....	109
<b>Figure 51</b> - Simulated attacker performing NMAP network discovery on ONOS IP address.....	113
<b>Figure 52</b> - Cyber-attack launching simulation via ICMP ping.....	113
<b>Figure 53</b> - IP address extraction from detected ping .....	114
<b>Figure 54</b> - Cyber-attack classification procedures after IP address identification. ....	114
<b>Figure 55</b> - Attack information parsing and dispatching to Intelligence and Deception Generator. ....	114
<b>Figure 56</b> - Intelligence's reception of the detected cyber-attack information.....	115
<b>Figure 57</b> - Deception Generator's reception of the detected cyber-attack information.....	115
<b>Figure 58</b> - Resulting deception grid in the context of the ongoing cyber-attack .....	116
<b>Figure 59</b> - Command, traffic policy and flow injection on deception grid hosts .....	116
<b>Figure 60</b> - Link connection testing of network devices and hosts of the deception grid. ....	116
<b>Figure 61</b> - Service testing after traffic policy and flow implantation .....	117
<b>Figure 62</b> - Resulting containers that form the hosts on the deception grid. ....	117
<b>Figure 63</b> - Service discovery by the attacker after deception grid deployment.....	117
<b>Figure 64</b> - Deception grid web page service.....	117
<b>Figure 65</b> - Deception grid web API service.....	118
<b>Figure 66</b> - Execution of research deployment procedures .....	118
<b>Figure 67</b> - Intelligence recognizing the research deployment request .....	118
<b>Figure 68</b> - Deception Generator receiving attack parameters to begin deployment procedures.....	119
<b>Figure 69</b> - Deception grids resultant of both deployments .....	119
<b>Figure 70</b> - Resulting containers that form the hosts on both deception grids.....	120
<b>Figure 71</b> - Deployment request denied by Deception Generator.....	120
<b>Figure 72</b> - Notification to Intelligence of the deployment blocking .....	120
<b>Figure 73</b> - Fetching the list of recorded production deployments .....	121
<b>Figure 74</b> - Fetching the list of recorded research deployments .....	121
<b>Figure 75</b> - Deception grids topology retrieval .....	121

## Proactive Cybersecurity tailoring through deception techniques

<b>Figure 76</b> - Kali virtual machine prepared to receive logs, simulating a SIEM.....	122
<b>Figure 77</b> - Asset Monitoring receiving ONOS SDN logs and forwarding them to Intelligence .....	122
<b>Figure 78</b> - Intelligence receiving ONOS SDN logs and dispatching them to the SIEM.....	123
<b>Figure 79</b> - Simulated SIEM receiving and presenting ONOS SDN logs sent by Intelligence .....	123
<b>Figure 80</b> - Asset Monitoring aggregating host logs and forwarding them to Intelligence .....	123
<b>Figure 81</b> - Intelligence receiving and dispatching host logs to the SIEM.....	124
<b>Figure 82</b> - Simulated SIEM receiving and presenting host logs sent by Intelligence.....	124
<b>Figure 83</b> - HTTP request to dynamically add a host to a deception grid. ....	125
<b>Figure 84</b> - Resulting deception grid after dynamic allocation of new host.....	125
<b>Figure 85</b> - HTTP request to dynamically remove a host of a deception grid without postmortem ....	126
<b>Figure 86</b> - Resulting deception grid after dynamic deallocation of the web API host.....	126
<b>Figure 87</b> - Web API unavailable after dynamic host removal .....	126
<b>Figure 88</b> - HTTP request to dynamically remove a host of a deception grid with postmortem .....	127
<b>Figure 89</b> - Resulting deception grid after dynamic deallocation of the web page host.....	127
<b>Figure 90</b> - Deception Generator exporting a container's filesystem for postmortem forensics .....	127
<b>Figure 91</b> - Exported filesystem of dynamically removed host with postmortem .....	128
<b>Figure 92</b> - Deception grid's web page host service unavailability after its dynamic deallocation.....	128

# List of Tables

<b>Table 1</b> - IPS vs. IDS properties .....	30
<b>Table 2</b> - Firewall vs. IDS/IPS properties .....	30
<b>Table 3</b> - Summary of Rowe's assessment of deceptive types in information-system attack .....	53
<b>Table 4</b> - Deceptive system and examples of deception.....	75
<b>Table 5</b> - Intelligence's API operations. ....	101

# List of Listings

<b>Listing 1</b> - Sample of type_of_attack.json parsing attack type external tool specific information (key) to system-acknowledged data (value).....	84
<b>Listing 2</b> - Sample of attack_details.json parsing attack details external tool specific information (key) to system-acknowledged data (value).....	84
<b>Listing 3</b> - Parallel process creation for respective socket connection. ....	85
<b>Listing 4</b> - Example of Dockerfile modification to enable deceptive host logging. ....	87
<b>Listing 5</b> - Example of a Python script that sends logs from a container to Asset Monitoring. ....	87
<b>Listing 6</b> - Associated frontend code that provides simplified logging. ....	88
<b>Listing 7</b> - Example of Dockerfile modification to fit Containernet's requirements .....	90
<b>Listing 8</b> - Docker command to run Containernet in the context of the proposed deceptive system. ....	90
<b>Listing 9</b> - Host definition for a given deception strategy .....	97
<b>Listing 10</b> - Adding hosts to the network.....	98
<b>Listing 11</b> - Configuring network policies via ONOS SDN REST API (intents) .....	98
<b>Listing 12</b> - Asset Monitoring capability of extracting and dispatching network (SDN) and host log..	100
<b>Listing 13</b> - Research deployment enabled by Intelligence's REST API .....	102
<b>Listing 14</b> - Example of host command assignment for a given deceptive strategy .....	105
<b>Listing 15</b> - Example of SDN controller assignment for a given deceptive strategy .....	105
<b>Listing 16</b> - Example of host assignment for a given deceptive strategy .....	106

# Acronyms

**ACL** – Access Control List

**API** – Application Programming Interface

**APT** – Advanced Persistent Threat

**ARP** – Address Resolution Protocol

**CIA** – Confidentiality, Integrity, and Availability

**CLI** – Command-Line Interface

**CSIRT** – Computer Security Incident Response Teams

**CTI** – Cyber Threat Intelligence

**CVE** – Common Vulnerabilities and Exposures

**DDoS** – Distributed Denial-of-Service

**DMZ** – Demilitarized Zone

**DoS** – Denial of Service

**InfoSec** – Information Security

**IoC** – Indicators of Compromise

**IoT** – Internet of Things

**ICMP** – Internet Control Message Protocol

**IDS** – Intrusion Detection System

**IP** – Internet Protocol

**IPS** – Intrusion Prevention System

**IT** – Information Technology

**JSON** – JavaScript Object Notation

**(K)ASLR** – (Kernel) Address Space Layout Randomization

**LLM** – Large Language Model

**MILDEC** – Military Deception

**MTD** – Moving Target Defense

**NGFW** – Next-Generation Firewall

**ONOS** – Open Network Operating System

**OS** – Operating System

**OSI** – Open System Interconnection

**OSINT** – Open-Source Intelligence

**REST** – Representational State Transfer

**SCADA** – Supervisory Control And Data Acquisition

**SDN** – Software-Defined Networking

## **Proactive Cybersecurity tailoring through deception techniques**

**SIEM** – Security Information and Event Management

**SOC** – Security Operations Center

**SWOT** – Strengths, Weaknesses, Opportunities, and Threats

**TCP** – Transmission Control Protocol

**TTP** – Tactics, Techniques, and Procedures

**UDP** – User Datagram Protocol

**UTM** – Unified Threat Management

**VM** – Virtual Machine

**WAN** – Wide-Area Network

**WSGI** – Web Server Gateway Interface

**WWI** – World War One

**WWII** – World War Two

**YARA** – Yet Another Recursive Acronym



# 1. Introduction

Cyber deception can be a pivotal instrument to enhance computer defensive capabilities for organizations. This technique works beyond traditional detect-then-prevent approaches (that are limited to known attacks) and enables a proactive posture regarding cybersecurity incidents. Through a calculated and deliberate use of deception techniques in parallel with technologies that provide its materialization, cyber deception can achieve its highest sophistication potential. Nevertheless, implementing this type of infrastructures is technically demanding and their design most often lacks a comprehensive and complete architecture that responds to all required necessities. In order to address and possibly revert the asymmetry present in cybersecurity efforts, the need to possess these kinds of systems is imperative.

## 1.1. Context

Proactive cybersecurity practices rely on adopting a preemptive posture and a set of appropriate measures with the objective of acting before a security-related incident occurs. This posture should complement the typical reactive approach already widely adopted by organizations. Including both panoramas in an enterprise's cyber defensive spectrum diminishes even more a perpetrator's attack success, consequently hardening the technological infrastructure. Examples of proactive practices within computer security landscape are penetration testing, threat modelling, and the adoption of deception technology in cyber warfare. An organization should embrace and apply these techniques and procedures to their cyber defensive strategy. The referred proactive techniques are not mutually exclusive and should be employed simultaneously in conjunction with the reactive security measures in-place. This work explores cyber deceptive concepts as a way of tackling the cyber asymmetry problem.

The asymmetry phenomenon in cybersecurity represents the difference between the defender's immense effort to safeguard every attack surface within its organization and the attacker's endeavor, which requires only the identification of one weakness to be able to exploit and possibly compromise a system. A growth in number of (reactive) defensive mechanisms is observed, accompanied by the increase on their level of sophistication. However, over the years, more attacks are being successfully performed [1], which may indicate that a change in posture must take place. To battle those statistics, a proactive approach to cybersecurity tailored through deception techniques should be considered to complement the already established and implemented reactive defensive measures.

Deception technology has already a considerable rate of adoption by organizations as its global market in 2022 reached 2.6 billion U.S. dollars with expectance to reach 7 billion U.S. dollars in 2030 [2]. Amongst the cybersecurity landscape, deception technology is many times narrowed to honeypots. Honeypots are simply an example of deceptive technologies. Other examples are honeynets and deception grids/systems. Honeypots and honeynets have limited capabilities when deceiving sophisticated attackers. Limitations acknowledged to these types of solutions include its rapid uncovering by the attacker, the restricted deception flexibility, and the insufficient capture of attack information. The evolution of deception technology, regarding its sophistication, is culminating on deceptive systems that aim to tailor the attacker's behavior through deception techniques.

This document describes the development of the present work, whose motivations are to present and discuss an architectural approach to deception technology with deception theoretical concepts as background. This approach aims to better understand what deceptive and infrastructural needs are in place, to further culminate on a deceptive system that, in all perspectives, tries to achieve excelling cyber deception capabilities.

## **1.2. Work Objectives**

The main objective of this work is to design a system that can meet the deceptive requirements that enable the tailoring of the attacker's behavior to defender's advantage. To leverage the inherent benefits of adopting deception technology into an organization's cyber defensive processes, a deceptive system is proposed that addresses cyber asymmetry, defends the IT infrastructure, empowers the discovery of zero-day vulnerabilities, and gathers vital information from attacker's Tactics, Techniques and Procedures (TTP) that ultimately contribute to the refinement of the defensive capabilities of an organization. This system should be dynamic and refined deceptively to successfully manipulate the attacker's course of actions. The objectives of this work are listed below:

- Theoretical study of deception.
- Conceptual approach to cyber deception and analysis of deception technology.
- Design a system architecture with cyber deceptive capabilities.
  1. Identify system core components.
  2. Identify external tools that may assist the system's functionalities.
  3. Define and segregate responsibilities between the components.
- Implementation of system-core functionalities.
- Use case scenario conception to correctly test the solution.
- Validate the correct and advantageous use of the system.
- Identification of system improvements and future work.

## 1.3. Contributions

This work offers significant contributions to the field of Cybersecurity, specifically, as well as to the broader domains of Computer Science and Engineering, and they include:

- The present dissertation which extensively describes the scope of this work and discusses important aspects concerning deception and cyber deception, from background to implementation and validation.
- The elaboration of a complementary document, in the form of a Springer LNCS paper/article, which was published and presented on INForum 2023 (Portuguese National Conference on Informatics). Furthermore, it was awarded the “Best Student Paper” of its category (“Comunicações e Redes de Computadores”) and the “Best Student Paper” overall.
- The entirety of the system’s source code is made publicly available at: <https://github.com/LuisGuerraa/DeceptiveSystem>, promoting further development and understanding.

## 1.4. Document Structure

This document is divided into six chapters and apart from the present introductory chapter, the remainder of this document is structured as follows: Chapter 2 provides background on theoretical concepts fundamental to the understanding of deception and deception technology in cybersecurity. Additionally, this chapter presents related work describing solutions whose objectives align with the proposed work’s. Chapter 3 presents the proposed system’s architecture and discusses each component’s responsibilities and goals within the system. Conceptual mechanisms are also presented to enable the proposed system’s completeness. Additionally, a correlation between deception techniques and the system is elaborated. Chapter 4 addresses the implementation of the proposed system and the inherent technological options that compose every part of the system. Furthermore, the depiction of how deceptive plans were materialized is presented. Threat modelling processes were also elaborated concerning the proposed system and its implementation. Chapter 5 validates the

## **Proactive Cybersecurity tailoring through deception techniques**

proposed system through use case evaluation which proves the system's worth and cyber defensive utility. Finally, Chapter 6 is centered on the conclusions that summarily explain the work outcomes that were accomplished, its underlying benefits, and addresses future work.

## 2. Background and Related Work

This chapter aims to outline the background and related work of this thesis and is organized as follows: Section 2.1 seeks to define deception, where various published definitions are presented and discussed, and a common ground is sought. In Section 2.2 deceptive techniques and taxonomies are presented and clarified, focusing on the more pertinent ones. Numerous examples of deception use cases are extensively described in Section 2.3, as envisioning its gains in practical cases is found to be of vital importance. Section 2.4 clarifies the motive and in what manner can deception be used in cybersecurity to defender's advantage, by introducing the concepts and explaining the technological pillars that enable deception. Section 2.5 provides an overview of deception technology examples and its particularities, specifically focusing on honeypots and honeynets. Section 2.6 discusses the state of the art and available solutions concerning deception grids/systems, where implementational details are crucial to discuss. Finally, Section 2.7 presents the correlation between Intelligence and deception planning, acquainting each concept and its importance.

## 2.1. Defining Deception

The word “deception” comes from the Old French *déception* (13c., *deception*) or directly from Late Latin *deceptionem* (nominative *deceptio*) and is documented in the English dictionary as “*the act of hiding the truth, especially to get an advantage*” [3]. Even though the referred definition is incomplete and broad, it is in fact a reasonably plausible starting point to acquire a better understanding of the concept. Throughout the years, many attempts were made to define clearly what deception is. Before any efforts were even made to get to a “correct” definition of deception, this concept has been applied throughout time, especially in the context of warfare, as seen on “The Art of War” (which dates back to the 5<sup>th</sup> century B.C.) [4], and on personal relationships where deception has been observed in children of two to three years old [5]. It is believed that a good strategy to start conceiving a complete and coherent definition of this concept is to dive into those two areas.

The definition of deception used by the U.S. Army doctrine since 1969 is as follows: “*Activity designed to mislead an enemy by manipulation, distortion, or falsification of evidence to induce him to react in a manner prejudicial to his interest*” [6]. According to James D. Monroe, this definition is not sufficient because the term “induces” is an imprecise way to describe what deception does. Secondly, the same author claims that the most precise and efficient deceptions rely on a strong foundation of truth to reinforce and support the falsehoods. This idea contradicts the principle of the stated definition as this reinforces that deception is done only in the context of falsehoods [7].

Daniel and Herbig in “Propositions on Military Deception” define deception as being “*the deliberate misrepresentation of reality done to gain a competitive advantage*” but like the previously presented definition, it is limited to falsehoods and lies, leaves out other deceptive tactics like selective truth, and doesn’t make any effort to refer that deception is targeted, as it’s deemed to be incomplete. An important point can still be withdrawn out of this definition: the concept of gaining “*a competitive advantage*” [8].

Diving into the personal relation and psychology fields, “Psychology Today” which is an American media organization with a focus on psychology and human behavior defines deception as “*the act - big or small, cruel or kind - of encouraging people to believe information that is not true. Lying is a common form of deception - stating something known to be untrue with the intent to deceive*” [9]. It is stated that this definition suffers from the same lack of clarity and coherency as the above mentioned although admitting that lying is only a form

deception. In “Illusions of reality: A history of deception in social psychology” [10], James H. Korn makes a very simple approach to the definition of deception, as is “*to cause to accept as true that which is false*”. Korn uses this stripped definition in order to consider as many forms of deception as possible, but it does not help us to reach a concrete definition since it is only sufficiently applicable on usual cases.

A common point between these definitions is that the deception practitioner must be consciously and intentionally utilizing deception. That is only true when the deceiver or both deceiver and target are human or human-controlled. Considering biological and nature-based deception, those entities do not possess conscious awareness and therefore do not exercise deception on a fully intentional manner. Examples of this will be depicted in Section 2.3.

Considering the existing definitions of deception and taking into account their flaws and upsides, Monroe crafted the following definition: “*Deception is the deliberate misleading of a target into taking actions prejudicial to the target’s interests by manipulating the target’s decision-making processes through the communication of true, manipulated, distorted, and/or falsified information*” [11]. This definition aggregates the simple aspects of the presented elementary definitions and combines them in a comprehensive and complete manner.

However, deception may lead and involve the deceived party to take no action. That means that deceived party’s inaction can still be self-harming, thus critical to a successful deception and should also be considered. Additionally, as discussed previously, referring to animal realm, deception is used unconsciously by its participants. A more refined definition of deception can then be crafted by adapting the original as follows: Deception is the deliberate or unconscious misleading of a target into taking actions or inactions prejudicial to the target’s interests by manipulating the target’s decision-making processes through the communication of true, manipulated, distorted, and/or falsified information or evidence. This definition will be used as a guideline for this thesis.

The same author claims that a “*successful deception is more than creating a perception; successful deception is about the target taking action*” [11], or taking no action (as discussed):

$$\text{Successful Deception} = \text{Creating a Perception} + \text{Target (In)Action}$$

This claim is found to be pertinent as the establishment of the perception is not enough to achieve a successful deception, as it involves the target to interact or not in a self-damaging manner. Nevertheless, it is considered that it lacks clarity on the type of action. Thus, this work introduces the concept of two types of action: an action pre-deception, where an adversary



## Proactive Cybersecurity tailoring through deception techniques

shows a certain behavior without any perception created or perceived; and an action post-deception which is different than the pre-deception and occurs after the deception action has been performed, leading to a different behavior, thus tailored behavior.

This perspective allows the introduction of successful deception, extending all the previous concepts where:

$$\text{Successful Deception} = \text{Creating a Perception} + \text{Target (In)Action}_{\text{Post Deception}},$$

where  $\text{Target (In)Action}_{\text{Post Deception}} \neq \text{Target (In)Action}_{\text{Pre Deception}}$

and also, unsuccessful deception:

$$\text{Unsuccessful Deception} = \text{Creating a Perception} + \text{Target (In)Action}_{\text{Post Deception}},$$

where  $\text{Target (In)Action}_{\text{Post Deception}} = \text{Target (In)Action}_{\text{Pre Deception}}$

These two concepts of pre and post deception action are of utmost importance since it allows a defender to self-evaluate the effectiveness of its deception techniques. While the action or inaction of the deceived party after deception implantation remains equal to the pre-deception action or inaction, it means that there was no behavior change and that probably the deception technique was not the most appropriate, thus allowing the defensive party to change its techniques until a behavior change is produced and the deceived party changes its attack pattern, i.e., its post deception target action:

1. Deploy Deception Technique
2. Get Target (In)Action Post-Deception
3. While (Target (In)Action Post-Deception equals Target (In)Action Pre-Deception)  
{  
    Behavior Unchanged.  
    Deception Unsuccessful  
    Deploy new Deception Technique  
}
4. Behavior Changed
5. Deception Successful

## 2.2. Deception Taxonomy and Techniques

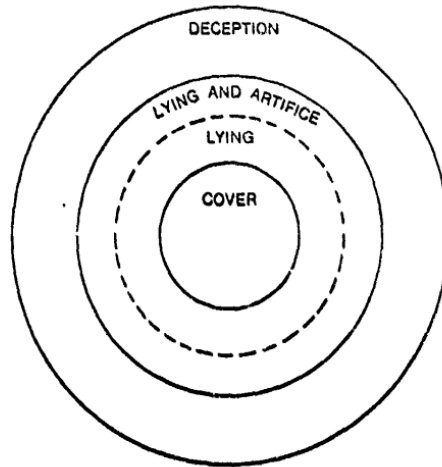
Taxonomy is a system for naming, organizing, classifying, and categorizing concepts into groups and types regarding a set of characteristics. This branch of science helps researchers understand the various existing types of concepts in a specific landscape by comprehending the whole picture through its parts. The need to possess a taxonomy in deception is not an exception, thus classifying the multiple forms of deception is of paramount importance.

In J. Bowyer Bell and Barton Whaley's "Cheating and Deception", a simple and straightforward approach is developed to the theory of deception. This methodology categorizes all types of deception in two substantial sets: "showing the false" (simulation) and "hiding the real" (dissimulation) [12]. "Showing the false" is subdivided in mimicking, inventing, and decoying. While "hiding the real" include techniques such as masking, repacking, and dazzling. Masking is defined by the authors as *"to eliminate an old pattern or blend it with a background pattern"*. Repacking's objective is *"to modify an old pattern by matching another"*. The concept of dazzling aims *"to blur an old pattern, reducing its certainty"*. Within the simulation subset, mimicking plans *"to recreate an old pattern, imitating it"*. Inventing is simply *"to create a new pattern"* and finally decoying means *"to give an additional, alternative pattern, increasing its certainty"* [12].

Another paper, "Two Taxonomies of Deception for Attacks on Information Systems" refers the book "Victory and Deceit" [13], by James F. Dunnigan and Albert A. Nofi, advocating that deception techniques can be militarily divided as follows. The concept of concealment is *"hiding your forces"*, camouflage is achieved by *"hiding your troops and movements from the enemy by artificial means"*, false and planted information (or simply *"disinformation"*) is accomplished by *"letting the enemy get his hands on information that will hurt him"*, lies are used *"when communicating with the enemy"*, displays are *"techniques to make the enemy see what isn't there"*, ruses are defined as *"tricks, such as displays that use enemy equipment and procedures"*, demonstrations correspond to *"making a move with your forces that implies imminent action, but is not followed through"*, feints are *"like a demonstration, but you actually make an attack"* and ultimately, insights involve to *"deceive the opponent by outthinking him"* [14].

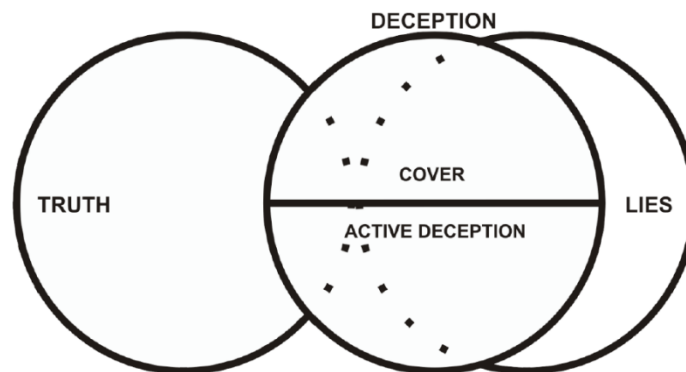
One important consideration to extract from the previously stated work is that deception and lies are different concepts. Many constructs mistakenly compare lies and deception as to being equals or synonyms, when actually lies are just a technique, amongst many others, that

can be utilized in a deceptive process. An example of this is present on Daniel and Herbig's deception model, where lies are presented as contained in by deception, as shown on *Figure 1*.



**Figure 1** - Deception's subsidiary concepts, extracted from [8].

From this approach, it is stated that lies are a part of a bigger scheme which is deception, but as discussed in Section 2.1, an effective deception is one where the factor of truth is highly present. Monroe confirms in "Deception: Theory and Practice" that "*while lies can be used in deception, not all deceptions are lies*" and the "*judicious use of the truth can be far more supportive of deception than outright falsehood*" [6]. The same author proposes other way to look at the relationship between lies, deception and truth present in *Figure 2*.



**Figure 2** - Interrelationship of truth, lies and deception, extracted from [15].

Here, lies and deception are not obligatorily subsumed to each other, instead, lies and truth are a part of the deceptive panorama, with both having varied significance according to the deception technique used.

Additionally, deception has a special correlation with uncertainty. This concept pretends to prove that there are no bullet proof plans or a flawless understanding of any given situation. Uncertainty is a variable that none of the deception participants can control, as its dependent on countless situation-specific particularities. One simple aspect that emphasizes this idea is the maxim inability for the deceiver to read the adversary's mind and vice-versa. Thus, you cannot one-hundred percent predict what is going to happen. Military entities name this phenomenon as "*fog of war*" as it is a very common and a nearly unavoidable reality during military warfare. In "Propositions on Military Deception", Daniel and Herbig seize the idea of "*fog of war*" to categorize two types of deceptions: A-Type and M-Type. A-type deceptions seek to thicken the "*fog of war*" with the objective of increasing target's uncertainty, leading him to be unsure on what to believe. One important goal that may be achieved using A-Type deception is to delay adversary's decision-making hoping to learn more about him or to flee from a confrontational situation. Monroe also points another important objective that is "*to cause the target to spread their forces in an effort to cover every potential outcome, thus affording the deceiver the opportunity to achieve relative superiority at the point of decision*". M-Type, as Daniel and Herbig put it, "*(...) reduce ambiguity by building up the attractiveness of one wrong alternative*". The goal with M-Type deception is to build up a seemingly good situation for the target (although being bogus) and thus obscuring the true situation. The adversary upon identifying that situation, will deficiently concentrate its resources while leaving more room for the deceiver to maneuver [8].

Monroe depicts four types of taxonomies within the several existing in the field of deception which are: taxonomy by method, sophistication, effect, and commission-omission. Taxonomy considering deception type of method concentrates on its mode or type of technique. Taxonomy by sophistication "*categorizes deceptions by the degree to which the deception adapts or does not adapt to changing circumstances*". Taxonomy that takes into account the deceptive effect breaks down what deception does and finally taxonomy by commission-omission is based "*on whether the deception causes the target to acquire a false belief or contributes to the target continuing a false belief*" [16].

For the purpose of this thesis, three of the four taxonomies described by Monroe are taken into consideration: taxonomy by method, commission-omission and by level of sophistication. These selected taxonomies are important because they factor Bell and Whaley's

proposed taxonomy while also considering Rowe’s vision (author of [14]) on the categorization of deceptive techniques. This also closely matches to existing U.S. Army doctrine deception concepts. Monroe’s proposed effect-based taxonomy is based on Bell and Whaley’s previously stated categorization and is not considered due to its similar definitions and redundant aspects with his own taxonomy by method, only differing in the taxonomy purpose itself. It is acknowledged that the three considered taxonomies, plus the one proposed by Daniel and Herbig, regarding “*fog of war*” manipulation, are important to help select the most appropriate deceptive technique for a given situation. With a clear categorization of the various techniques, the process of selection of a deception technique becomes unambiguous and more prone to success.

### 2.2.1. Deception by Method

Concerning, firstly, Taxonomy by Method, Monroe subdivides it into two different planes: Active Deception and Cover, as shown on *Figure 3*. In the depiction of each category and technique of deception, the terms items and indicators will be used in the clarification of the concepts. The term item refers to an artifact meant to be kept away from the adversary and used in the deception process, and indicators are a set of information (true or false) that the target will collect in each situation according to his perception.

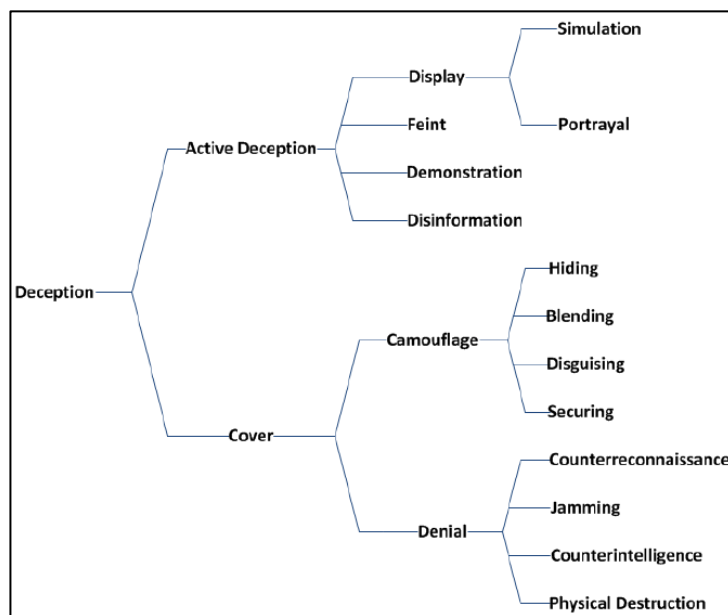


Figure 3 - Taxonomy by method, extracted from [16].

Active Deception (also specified as “*simulation*” or “*showing the false*”) comprises the actions “*to convey deceptive indicators to the target*” while Cover (also mentioned as “*hiding the true*” or “*dissimulation*”) is “*a set of actions designed to prevent the target access to the indicators necessary for constructing a correct perception of the situation and environment, and thus necessary for proper decision-making*” [16]. This clearly separates both areas, as active deception diligently transmits deceptive indicators to manipulate target’s perceptions and make him take conscious actions with that (deceptive) indicators in consideration. Contrary to active deception, cover neutralizes the target’s attempts to get an accurate understanding of the situation he is in. This diminishes the possibility of a sensible judgement of a particular situation and its surroundings, which lead to liabilities on the decision-making. Liabilities that the deceiver can and should exploit to its own advantage.

## 1. Active Deception

Active deception is divided in four categories: Displays, Feints, Demonstrations and Disinformation. These techniques can be depicted as follows.

### a. Displays

Displays are “*static depictions of activities, forces or equipment for the purpose of deceiving the target’s collection apparatus*”. Within this category, there exists two types of displays: simulation and portrayals. Simulations “*use decoys*” (Monroe uses this term to refer to Bell and Whaley’s decoy category) to “*create a dummy force or capability*”. Decoys are dummy items or representations used to replicate real items (e.g., equipment, hardware, people, etc.) [16]. On the other hand, portrayals utilize true and genuine items to provide a replica of something that does not actually exist or to present an image of items of a different type than they really are.

### **b. Feints**

Feints are “*operations designed to deceive the target into reacting as if the feint is an actual decisive operation*”. The concept here is that a feint is a series of events that imitate a known or familiar procedure (in a given context) to the target, in order to appear that a crucial operation is taking place that affects the outcome of a situation, while being totally bogus. This aspect compels the target to take some kind of action that in the end harms him. For example, to distract the target from the actual decisive action that is happening, or to trick the target into prematurely committing to his actions. Monroe also notes that “*feints differ from demonstrations in that some manner of contact with the target is sought*” [16].

### **c. Demonstrations**

Demonstrations consist of “*deceptive shows of force where actual engagement with the target is not sought*” [16]. The core objective of demonstrations and associated processes is to avoid confrontation with the target. This is accomplished by “*putting on a show*” of muscular forces and strength (that the deceiver does, or does not, possess) to intimidate and discourage the adversary into not taking any hostile behaviors. This can be helpful when the deceiver, in any particular moment, cannot match the enemy forces or solely does not want to engage in any type of confrontation. Sun Tzu in the “Art of War”, addresses this type of tactic stating that “*If you are strong, appear weak. But if you are weak, appear strong*” and “*If a battle cannot be won do not fight it*” [4].

### **d. Disinformation**

Disinformation is characterized by exposing to the target “*assets of false, modified, or selectively true information with the intent to deceive*”. Disinformation has no particular form as it can be any type of communication, physical or digital proof, or other type of planted evidence [16]. This procedure involves that the target gets hold of the adulterated information (indicators) and have some level of belief in its veracity. The level of belief is variable and dependent on the content itself and on the way that the information came to the possession of the adversary e.g., from a source trusted or untrusted by the target.

## 2. Cover

Cover is divided in two categories: Camouflage and Denial. Both categories can be described as follows.

### a. Camouflage

Camouflage seeks to *“prevent indicators from being detected by the target’s collection assets”*. Within this category, four broad methods can be found: hiding, blending, disguising, and securing. Regarding the hiding technique, it’s simply the action of physically concealing an item “behind” a barrier e.g., vault or a bunker. Denote that the barrier itself can be hidden, blended, or disguised (these correspond to the other techniques of Camouflage), but the barrier can also be visible. Blending means that *“the item is concealed by means that merge the item with the background”*. This can be done by adapting or changing the item to closely match the environment and not be noticed. In disguising, *“the item is concealed by making it look like something innocuous”*. This tactic aims to mask an item as something else that seems to not be harmful to the adversary or simply not important enough to be noticed. Finally, in securing, *“indicators are reduced via the use of operations security, information security and emissions control”* which points out that the use of secure procedures also leads to minimizing the leak of indicators [16].

### b. Denial

As opposed to camouflage, the techniques encompassed in denial aim to *“attack the channels indicators travel on”* to reach the adversary. *“Denial seeks to degrade target’s collection channels”* so that himself is neither able to receive indicators or is dependent on much smaller set of indicators that may not lead him to realize the real circumstances of the situation. Denial techniques englobe counter reconnaissance, jamming, counterintelligence and physical destruction of the adversary’s collection tools. Counter reconnaissance comprehends a set of measures taken to prevent enemy’s reconnaissance. Impeding the enemies’ attempts of reconnaissance will lead to less knowledge obtained from a situation by the target. Jamming is simply blocking the communication; counterintelligence is to



block the enemy's sources of information (different from counter reconnaissance) and by destroying the target's collection tools (physical destruction), the enemy's awareness may be completely skewed [16].

### 2.2.2. Deception by Commission-Omission

This taxonomy is taken into consideration due to the contribution of Chisholm and Feehan on "The Intent to Deceive" to the deception taxonomy landscape, referred also by Monroe. The authors outline eight types of deception divided into two categories [17]. "*Deceptions by commission are typified by the deceiver contributing causally to the belief of the target*" and "*Deceptions by omission are typified by the deceiver facilitating the target's maintenance of an existing belief*" [16]. Both approaches are distinguished by the level of involvement of the deceiver. Either the deceiver has an active part on the deception process and outcome (Commission), or the deceiver does not take engage actively with the target in his process of deceiving (Omission). The eight types are listed below, extracted from [16].

Deception by Commission:

- Deceiver contributes causally to target acquiring belief in the proposition.
- Deceiver contributes causally to target continuing to believe in the proposition.
- Deceiver contributes causally to target stopping belief in the negation of the proposition.
- Deceiver contributes causally to target not acquiring belief in the negation of the proposition.

Deception by Omission:

- Deceiver allows the target to acquire belief in the proposition.
- Deceiver allows the target to continue belief in the proposition.
- Deceiver allows the target to cease belief in the negation of the proposition.
- Deceiver allows the target to continue without the belief in the negation of the proposition.

### 2.2.3. Deception Based on Level of Sophistication

In “Unweaving the Web: Deception and Adaptation in Future Urban Operations” by Scott Gerwer and Russel W. Glenn, a new manner of categorizing deceptions was used. The authors believe that when it comes to classifying a type of deception, its level of sophistication should be acknowledged [18]. The level of sophistication characterizes that “*any given deception is a spectrum ranging from static and context-insensitive to tailored and premeditated*”. It is discussed that even though the scale is indeed a spectrum (has countless values in the interval), it is still feasible to mark milestones along that same spectrum with the objective of classifying the type of deception where sophistication is determined by the degree to which the deception takes in count the variables of a situation. The authors assert that the level of sophistication varies between static, dynamic, adaptive, and premeditative. Static deceptions are considered the least sophisticated and “*are in place regardless of state, activity, or the histories of either the deceiver or target*” [18]. That being, independently of the nuances and the events that occur in the meanwhile within the environment, the deception process never changes. This type of deception can be mistakenly understood as useless but can effectively discard minor targets. “*Dynamic deceptions are those that activate under specific circumstances. The ruse itself and the trigger do not change over time, nor do they vary much by circumstance or adversary*” [18]. Deceptions within the dynamic category are activated by static triggers and the outcome of the deception in each circumstance never changes. “*Adaptive deceptions are triggered like dynamic deceptions, but either the trigger or the ruse itself can be modified with experience. This category covers deception improved through trial and error*” [18]. And ultimately, “*Premeditative deceptions are designed and implemented based on experience, knowledge of friendly capabilities and vulnerabilities, and, moreover, observations about the target’s sensors and search strategies*” [18]. This genre of deception is particularly interesting as the deception assembling process depends on the target’s characteristics and TTP directly, and as consequence, the same deception plan would not probably work for two distinct targets.

## 2.3. Deception in Practice

Regarding real world scenarios where deception is applied consciously or unconsciously, there are multiple areas that can be disclosed where deception practitioners can be found. But first, why should deception be used?

A single and concrete answer to that question cannot be outlined, as deception is a volatile instrument that is dependent on the nuances of specific situations and each desired outcome. Hereupon, deception is a situation-dependent weapon that generally is used where it is necessary to mislead someone in order to achieve a particular goal (or outcome). Some common reasons include protect oneself or other from harm, gain some strategic advantage in a conflict scenario, extract benefits from a negotiation or to obtain information and resources that would be otherwise difficult or impossible to acquire. Some of these scenarios were inferred and described in Section 2.2, where deception taxonomies were discussed.

In light of that, it is prominent to dive into different areas where deception is applied on a daily basis. The natural and biological world is an excellent place to start investigating because many species of plants and animals have been using and perfecting deception techniques throughout millions of years of evolution with the objective of evolving, thriving, and surviving the environment. As psychologist Harriet Lerner puts it, “*Deception and 'con games' are a way of life in all species and throughout nature. Organisms that do not improve their ability to deceive - and to detect deception - are less apt to survive*” [19]. One example of this is *Lamium album* (known commonly as “dead nettle”) which evolved to look just like a stinging nettle. Despite looking particularly the same, dead nettle does not possess the painful sting while taking advantage of the same benefits as the stinging nettle. By looking similar, dead nettle induces fear and apprehension to potential predators safeguarding itself. This technique can be mapped into being a “demonstration” within the previously discussed taxonomy by method (Section 2.2.1). Another example present in the animal kingdom is the *Thaumoctopus mimicus* or “mimic octopus”, which can mimic up to fifteen species of marine organisms as a primary defense mechanism. Analyzing the same taxonomy, the tactic of disguising is being used by the octopus. Probably the most recognizable example of a biological deception practitioner is the *Melanocetus johnsonii* – known as anglerfish – that withholds a long filament above its head which emits light. Wiggling it makes a resemblance to a prey animal with the objective of luring other predators close, which is a clear example of a display, more specifically a simulation (or decoy). The prominent objective of the use of deception in

the natural world is to survive and it is duly noted that these creatures are unconsciously using deception to thrive in contrary to human-involved deception [20].

Journalism is the *“collection, preparation, and distribution of news and related commentary and feature materials through such print and electronic media”* as newspapers, magazines, social media, radio, television etc. (*per* Britannica [21]). There is no need to proclaim the importance of this field of work, as it is inevitably present in every human’s daily routine whether you turn the television on, log in to social networks or power the car radio, news are always being bombarded into a human’s brain with the objective of informing on current major topics of the world. Ideally, every news piece and journalist should follow a code of conduct and ethics of journalism, thus, obeying five core principles: truth and accuracy, independence, fairness and impartiality, humanity, and accountability [22]. A journalist should strive *“to ensure that information disseminated is honestly conveyed, accurate and fair”* and needs to resist *“threats or any other inducements to influence, distort or suppress information...”* [23]. Unfortunately, that is not always the case as some news are strategically issued with some agenda behind them, using techniques present in the process of deception. For instance, journalists may use slightly distorted or even false information in their reports to create a sensational or attention-grabbing story. This leads to a misrepresentation of the truth, even if there is some truth behind it, which makes it more believable and harder to distinguish from actual reality. This aspect has been debated in Section 2.1, as truth is a fundamental part of an exceptionally well executed deception. This is an example of “disinformation” as it uses false or selective truth in the deceptive process. Another case of deception in journalism is when a journalist pretends to be someone else in order to gain access to information by deceiving his sources. This is referred to as “undercover journalism”. In this scenario, the use of deception is not on the news piece itself but on the method of acquiring the information that will culminate on the journalistic artifact. At last, the most perceptible example are “fake news” as they have become a major issue in the last years [24]. “Fake news” are the *“deliberate making up of news stories to fool or entertain”* [24], and are considered a significant problem as it undermines the credibility of all media, unlike the previous stated examples where only the journalist or the paper he represents is affected. In contrary to sensational news, fake news use disinformation deceptive technique with no fundament of truth, even if some of them are sensational, as it is an important factor for the same to spread.

Alongside journalism, the sphere of advertising floods us every day and it is present everywhere we look. Advertising consists of the *“techniques and practices used to bring products, services, opinions, or causes to public notice for the purpose of persuading the public*

*to respond in a certain way toward what is advertised” (per Britannica [25]). It is a form of communication used to persuade an audience (intended future consumers) into taking some action and is typically present on all media platforms and through physical artifacts such as billboards and flyers. The importance of this lucrative field cannot be denied as “it is calculated that between 2018 and 2022 global advertising spending will increase by more than 160 billion U.S. dollars, reaching close to 790 billion by the end of that period” [26]. This domain is not an exception when it comes to the use of deception techniques. Many advertisements conceal or emit false statements in order to make people confident and trust the product/services’ safeness and reliability with the final objective of purchasing it. Making exaggerated claims about a product’s effectiveness or benefits using deceptive imagery and misrepresentation of the reality of the product are examples of deception. An advertising strategy that falsely claims that its product is endorsed by a well-known expert to convince consumers to acquire it is other example of deception that relates a believable figure to a product to induce (false) trust in costumers. Comparing a news piece with a product and the advertising of a product with the spread of the news piece, the deceptive techniques that empower the consuming of news are similar to the ones that enable the persuasion of the consumer into buying a product, as it involves disinformation.*

Tom Donohue, the former President and CEO of the United States Chamber of Commerce affirms that *“Business is the engine of progress. It creates jobs, spurs innovation and drives growth”* and Børge Brende, a Norwegian politician, diplomat, and current president of the World Economic Forum defines business as *“the driving force of the global economy and a key driver of social progress and environmental sustainability”*. Business depends mainly on negotiating in order to expand. This process is done between two or more entities with preferably both or all parties involved taking financial or reputational advantage out of the deal. Deception in business can refer to any type of dishonest or fraudulent behavior used to gain advantage over other competitors or customers. Deception can be used within many angles on how business is conducted, two examples are through its product(s) and in their negotiations with other parties. Firstly, in the products, it is fair to refer that the advertising sector (a part of the business plan) and its deceptive techniques, discussed antecedently, are included in business deception. Other example of this is when a company publishes fake reviews or testimonials to boost its reputation. Furthermore, concerning the negotiation process, a negotiator who pretends to have more negotiating power or influence than they actually do is a clear example of a demonstration and simulation deceptive techniques. Using emotional appeals or flattery to manipulate other party’s decision-making, hiding important information or making false claims

about the terms of a deal to convince the other party to agree to some terms, and delaying/stalling strategies to wear down the other part into agreeing are other deceptive examples that include misinformation and displays.

Law enforcement refers to the practices and procedures used by governments to maintain social control and safety, prevent crimes, and enforce laws (created and trialed by the judicial system and applied in the correctional system). This is typically done through the police department and other institutional entities. To achieve this purpose, law enforcement practitioners often need to resort to deceptive techniques. One way that deception is practiced is when some types of police investigations require the use of undercover agent(s) [27], this was already discussed in the previous example of “undercover journalism” as both share the same principle. An alternative example is when a suspect of a crime is already under police custody and during the interrogatory phase, the detectives try to get the confession out of the perpetrator through deception techniques (such as disinformation or feints) that confuse and consequently steer the interrogee to make contradictory statements to prove his guiltiness [28].

Politics is the process by which a group of people, usually a country, a community or a continent make decisions about their laws and policies that govern its society. Canadian American political scientist David Easton in “ The Political System: An Inquiry into the State of Political Science” states that politics is “*the authoritative allocation of values for a society*” [29], inferring that politics shape a country’s sociological principles and ideals. Numerous states (i.e., the political unit within which power and authority reside) and governments (or political system meaning a group of persons who direct the politic affairs of a state) exist around the world, where the government type can vary from state to state. Moreover, there are four major recognized types of political systems in place: democracy, monarchy, oligarchy, and authoritarianism/totalitarianism [30]. Despite the goal of this thesis not being to dissect each type of political system nor compare them, it states that in every one of those, deception is exercised with differing extents. Democracy is the form of government with which we are most familiarized, and is portrayed by Abraham Lincoln, the 16<sup>th</sup> president of the United States of America, as the “*government of the people, by the people, for the people*”. Ideally there is no room for any type of con, corruption, lies, frauds, or deceits in this political system, but like any other domain, it is not immune to deception. This concept can take many forms in a democracy, for example, misleading campaign promises which politicians never had the intention of keeping once they are in office with the objective of gaining votes. Politicians may try to use the media (through journalists as referred in previous examples) to present a biased perspective of events or hide conflicts of interest to avoid scrutiny. Parties may spread false

information to sway public opinion (e.g., against an adversary party). One pivotal sample of this, occurred during the 1988 Mexican general elections, where the PRI party (one of the contending parties) were not the favorite to win and tampered the polls, giving the party an immense advantage over the rest of the contenders. When those polls were communicated to the people through media sources, demotivation spread among opposition supporters who believed that there was nothing they could do to overturn the outcome of the elections and fail to turn up to the ballots. At the end of the day, the official results gave PRI the victory [31].

One of the most essential fields of study when it comes to the human being is psychology. This branch of science is presumed to exist since the ancient civilization of Egypt and Greece, mostly as a branch of philosophy and can be defined as a “*scientific study of the mind and behavior*” [32] where psychologists are “*actively involved in studying and understanding mental processes, brain functions, and behavior*” [33]. In summary, psychology helps people in many ways because it is capable of finding an answer and explain why people act the way they do, thus being so crucial, as none of other scientific areas can give explanations to those questions. As it happens with all other scientific subjects, research is a requirement for the evolution of the subject itself. Psychological research is the systematic investigation of behavior, cognition, and emotions to understand some psychological phenomena involving some collection and analysis of data using different research methods (e.g., experiments, surveys, and case studies), with the aim of testing hypothesis on subjects (patients) and generate knowledge about psychological processes. “*When it comes to the concept of research, there is no question that ethics is one of the most essential components there is. This is certainly true for all forms of psychological research*” [34]. The need for codes of conduct and ethics stipulations is of imperative nature. In spite of that fact, the same source states that “*Without question, when it comes to psychological or sociological experiments, there are going to be situations in which you do not want the subject to know everything*” [34]. Deception may be essential in certain situations concerning psychological research, even though it conflicts with some of the ethics. This debate will not be addressed here, instead, possible uses of deception in psychology research will be exhibited. Deception in psychology research is the practice of intentional misleading or withholding information from patients to test a hypothesis and/or measure certain reactions and responses. Concealing the true purpose of a study is one good example, as in some cases, researchers cannot disclose the full details to a patient to avoid influencing their behavior. Other great example is when researchers provide false information to manipulate certain variables and measure particular responses, for instance a study that uses “fake news” or fabricated social media posts to manipulate patient’s beliefs and attitudes.

*“Warfare is generally understood to be the controlled and systematic waging of armed conflict between sovereign nations or states, using military might and strategy, until one opponent is defeated on the field or sues for peace in the face of inevitable destruction and greater loss of human life”, per World History Encyclopedia [35].* The first recorded war dates back to 2700 B.C. between Sumer (the southernmost region of ancient Mesopotamia) and Elam (region in the Near East). Since then, many more wars were fought all around the globe, as it was one of the ways for some societies to prosper or survive. As stated in the previous sections, where deception definitions and taxonomies were discussed, military warfare is a stellar example of exercising deception in real world scenarios. Military Deception, commonly known in military operations as MILDEC, has been utilized as a core instrument in warfare from the existence of ancient civilizations such as the Egyptian, Greek, Macedonian, Chinese, and Roman, through Middle Ages and Renaissance periods, to the great World Wars (I and II) and latest war conflicts including Vietnam, Kosovo, and the contemporary Ukrainian-Russian conflict. It is impossible to display and choose all critical uses of deception in warfare contexts as they are numerous and open to subjectiveness on how vital they were to each conflict’s outcome.

According to a rare Egyptian papyrus (Papyrus Harris 500), an Egyptian army lead by Pharaoh Thutmose III and general Djehuty blockaded Caananite city (nowadays Jaffa) in around 1450 B.C. Several of Djehuty’s soldiers hid in baskets, those which were offered to the enemy to suggest the surrender and send tribute to the apparently winner of the war. Once inside the city gates, the hidden soldiers emerged opening city’s gates allowing other soldiers to enter the city and conquer it [36]. This type of deception tactic had already been described in a famous example, composed between the 9th and 6th centuries B.C. in poems. “The Odyssey” by Homer presented the “Trojan Horse”. This successfully executed deception by the Greeks, presumably in the 13th century B.C., unlocked a stalemate that lasted for several years. The Greeks built a hollow wooden horse which they advertised as an offer to the goddess Athena. They then departed from the Troy surrounding areas giving the perception that they left for Greece when instead, Greek soldiers had hidden within the horse and were brought inside city walls by the Trojans. By night they came out of concealment and opened the city gate which allowed the invasion to be successful [37]. A civilization that outstandingly used deceptive techniques and ruses to gain strategic advantages or control certain nuances during warfare conflicts were the Ancient Chinese. Many of old Chinese transcripts and books concerning warfare strategics were found and are now used as handbooks for many scenarios. Examples of this are “The Art of War” by Sun Tzu [4] and “The Thirty-Six Strategies” [38], a book that was found years later



after its conceiving and brought to light by many publishers with no identifiable original author. 16<sup>th</sup> stratagem of this book states that *“to catch something, first let it go”* [38]. This stratagem dictates that a cornered enemy will fight for their life, and you may risk an unnecessary defeat. Rather than pressing an exhausted enemy, do not obstruct its escape and he will be fooled that he still has a chance of escaping. *“His will to fight is thus dampened by his desire to escape”* [38] and in the end his freedom *“will be proved as a falsehood, the enemy’s morale will be defeated, and he will surrender without a fight”* [38]. This strategy is latent when in 506 B.C. an alliance of states led by Wu defeated Chu’s army in battle and its remaining troops retreated. At the time the king of Wu wanted to pursue and destroy the fleeing army but was advised by his brother to stand down. The king followed his brother’s advice and let the enemy cross the river, where they began to prepare garments and food to survive. Wu’s army then arrived on the opposite shore and found that Chu’s soldiers abandoned their camp and fled into the countryside. Wu’s army then followed the escaping troops and were able to easily conquer Chu’s capital [38]. In 341 B.C., the state of Qi under the command of general Sun Bin confronted Wei. Taking advantage of Wei’s forces perception in relation to Qi’s undermining and weaker capabilities, Sun Bin during the course of three nights lit one-hundred thousand campfires on the first, fifty thousand on the second and thirty thousand on the next [39]. The opposite side’s general Pang Juan (of Wei) believed, due to its previous assessments and perceptions, that Qi faced mass desertion problems and decided to attack not knowing that Sun Bin had prepared a well thought ambush and Wei ended up winning that confrontation. Campfires were many years later also used, during the “Battle of Long Island” in the American Revolution by Washington’s troops to evade British scouts and army. In that scenario, campfires were left blazing throughout the night to leave the impression that Continental Army’s troops were still in the site but had already retreat to safety [40]. Amidst the same revolution on the “Boston siege”, George Washington successfully used deception in order to nullify the larger and better-equipped opposite forces, the British Army. The Continental Army had several shortages of materials and supplies such as lack of gunpowder. To obscure that liability from the enemy, Washington ordered his troops to fill various gunpowder casks with sand and ship them to the depots [41]. This demonstration tricked British commanders into not searching for confrontation.

During both World Wars, various deceptive techniques and strategies were used in all types of terrains and situations, from land to sea. In October 1917, during WWI, British commander Edmund Allenby intended to attack the Ottomans in southern Palestine [42]. As a part of his deception plan, Allenby tried to convince the enemy into believing that his target

was Gaza, when instead it was Beersheba. Using a deceptive tactic now known in military as the “Haversack Ruse”, his officer intentionally dropped a knapsack containing false plans leading to an attack on Gaza that the Ottomans later recovered. This enabled the British to surprise the adversary and win the Battle of Beersheba on the 31<sup>st</sup> of October 1917. From land to sea, many decoys and disguises were used, from dummy horses to entire ships being disguised as civilian merchant ships. Concerning WWII, specifically on the North Africa front, a British army unit led by a magician and illusionist named Jasper Maskelyne prevented the devastation of the city of Alexandria, Egypt in 1941. His “trick” was to use lights to recreate a real-life size nighttime “portrayal” of the city while blacking out the actual city. This action combined with some planned explosions (“feints”), created the illusion of the portrayal was actually the real city leading the German planes to release their bombs on the fake city safeguarding the real [43]. Deception had once more proven its worth.

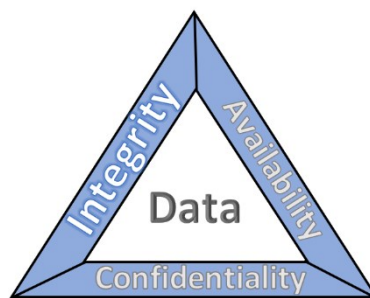
Ultimately, deception has always been a crucial strategic and tactical aspect of warfare. Many more examples of its use can be demonstrated, numerous historical references can be presented where deceptive ploys were designed to gain advantage over the enemy. On all the presented examples and others that can be found, it is demonstrated that the *“successful use of deception in warfare depends on secrecy, concealment, originality, and rapidity. The same deception can neither be used multiple times nor is appropriate in all circumstances. (...) A successful deception is always the result of an extremely well-orchestrated plan”* [20].

This thesis will now focus on how deception can be applied to cybersecurity landscape especially on a defensive point of view. The enumeration of the deceptive techniques applied in the different depicted areas was found of crucial importance as it holds a foundation to understand how deception can be used as a technology applied to computer security. This is especially noticeable in the case of warfare, where similar strategic and operational nuances are found to be in cyber warfare too, as *“Deception in the physical world is a ubiquitous phenomenon. Intuitively, it would seem that the same would be true in the virtual world”* [44].

## 2.4. Deception Technology in Computer Security

Computer security (or Cybersecurity), *per* Britannica English dictionary, is defined as *“the protection of computer systems and information from harm, theft, and unauthorized use”* [45]. Despite the fact that, this given definition is profoundly vague, it consists of a foundation

to better understand what computer security really is, as the clarification of the definition of this field is considered fundamental. Daniel Schatz, Rabih Bashroush, and Julie Wall published a paper in the “Journal of Digital Forensics Security and Law” where a more representative definition of cybersecurity is sought. These authors conclude that computer security is the *“approach and actions associated with security risk management processes followed by organizations and states to protect confidentiality, integrity and availability of data and assets used in cyber space. The concept includes guidelines, policies and collections of safeguards, technologies, tools and training to provide the best protection for the state of the cyber environment and its users”* [46]. This definition is found to be more accurate, complete and comprehensive since it refers the underlying processes that build the field of cyber security. Additionally, it mentions the CIA (Confidentiality, Integrity, and Availability) triad (*Figure 4*), consisting of the main purposes on why cybersecurity should be implemented. Confidentiality asserts the efforts to make sure that data is kept private and away from unauthorized people. Integrity is making sure that your data is trustworthy (authentic) and therefore kept free from tampering. Finally, availability consists of making the data useful, that being, even if the data has integrity and privacy assured, it is useless when unavailable, so there is a need for implementing availability mechanisms, namely against Denial of Service (DoS) cyber-attacks.



**Figure 4** - CIA Triad, extracted from [47].

As depicted in *Figure 4*, the CIA triad covers only Information Security (InfoSec), but other layers should be taken care of when securing organizational systems, namely the communication (network) scope. Thus, cybersecurity should be an area broad enough to englobe several layers of security from hardware through to personal, physical, and procedural security. Any layer that is not secured can compromise all the layers below it.

## Proactive Cybersecurity tailoring through deception techniques

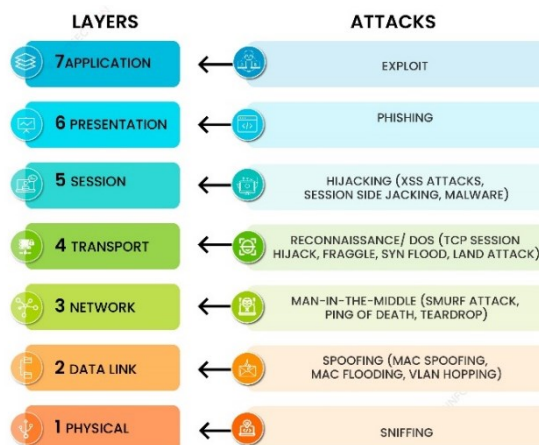
In many scenarios, cybersecurity practitioners rely on the Open System Interconnection (OSI) model to structure their cybersecurity approach into layers, as portrayed in *Figure 5*. This model is a theoretical framework that hierarchically partitions various functions (layers) of a computer system, specifically, seven types of functionalities of computer networks and its responsibilities. This layered approach has many inherent advantages by logically separating networking functions where network problems can be solved easily through divide-and-conquer methodology [48].

**OSI Model**

Content	Layer	Description
Data	7 - Application	Network process to application
Data	6 - Presentation	Data format conversions (encryption, compression, etc.)
Data	5 - Session	Interhost session communication
Segment	4 - Transport	End-to-end connection and control
Packet	3 - Network	Path Determination and Logical Addressing (IP)
Frame	2 - Data Link	Physical Addressing (MAC)
Bit	1 - Physical	Signal and Binary Transmission

**Figure 5** - The OSI Model, adapted from [48].

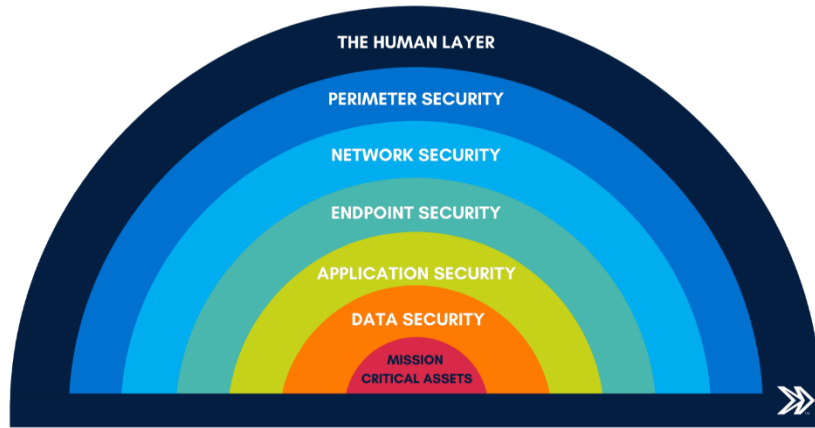
This approach is enough when only analyzing the network environment, as many types of cyber-attacks can be performed, targeting any of the seven layers of the OSI Model, as seen on *Figure 6*.



**Figure 6** - Common security attacks in the OSI layer model, extracted from [49].

## Proactive Cybersecurity tailoring through deception techniques

It is considered that using only the OSI model to craft defensive procedures can be an incomplete approach when facing the whole panorama of layer protection in cybersecurity. Defense in depth needs to be implemented to leverage multiple security measures to protect organization's assets as *"each layer offers additional protection so that if one layer is breached, the next layer of protection will be in place to prevent further exposure of data to unauthorized parties"* [50]. A multi layered approach should be adopted, as the one seen on *Figure 7*.



**Figure 7** - The seven layers of cybersecurity, extracted from [51].

Furthermore, other vital areas that compose cybersecurity must also be taken into account such as Disaster Recovery/Business Continuity planning, also belonging to other Information Technology (IT) departments [52], and the Threat Modelling [53] and Incident Response [54] teams. All these computer security domains contribute to a better defensive capability and consequent improved quality and resilience of an organization's infrastructure.

In order to secure a system, the security team in an organization must implement defensive measures and mechanisms in every layer. A set of mechanisms are available to cybersecurity practitioners depending on the layer they are trying to defend. Since the objective is not the extensive enumeration and discussion of those mechanisms, only a few of these instruments, considered important to this work, will be discussed. In terms of network defense, the most recurrent options to traffic security are Firewalls, Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS). An IDS is a network element that analyzes network traffic and searches for threat patterns or suspicious activity that are known for being malicious. If any malevolent traffic is detected, this element will send an alert to be then addressed by the security team. An IPS is essentially an IDS that adds the capability of blocking the traffic identified as malicious. A firewall is a device that monitors incoming and outgoing network

## Proactive Cybersecurity tailoring through deception techniques

traffic, which decides based on a set of rules if a given traffic is blocked or let in/out. It is commonly the first line of defense as its placed between the secure network, the Demilitarized Zone (DMZ) and the Internet, establishing barriers. Intrusion detection and prevention systems are typically placed behind firewalls. They can also be a part of the firewalls as a unique device such as Unified Threat Management (UTM) firewall or Next-Generation Firewall (NGFW) which combine the functionalities of the IDS/IPS with firewall's capabilities [55]. Particularities of IDS and IPS are compared in *Table 1*, while a comparison between IDS/IPS and firewalls is visible on *Table 2*.

**Table 1** - IPS vs. IDS properties, adapted from [55].

	<b>IPS</b>	<b>IDS</b>
Network Placement	In-line with network traffic.	Parallel (out-of-band) with the traffic.
Mode of operation	Active device. Block capabilities.	Passive device. Only detects.
Detection Mechanisms	Signature-based, rule-based, statistical anomaly detection etc.	Signature-based, rule-based, statistical anomaly detection etc.
Blocking options	Block packets at network level, reset connection, alert the administrator etc.	Non-blocking. Alerts the administrator, sends a reset connection request.
Hardware features	Must be high performance to perform Deep Packet Inspection and not slow down the traffic.	Does not need to be very high performance since it does not intervene in the traffic. However, in order to keep-up with traffic in real-time it must be able to handle the line bandwidth.

**Table 2** - Firewall vs. IDS/IPS properties, extracted from [55].

	<b>Firewall</b>	<b>IDS/IPS</b>
Network Placement	Usually placed at the front of the network to control traffic.	Behind the firewall either as in-line or out-of-band.
Main use case	Allows or blocks traffic between different network zones.	Dedicated to inspecting network packets to match them against signatures of known malicious attacks. Then, traffic is either blocked or an alarm is issued.
Detection Mechanisms	Usually works up to Layer 4 to allow or block IP address and ports.	Signature-based, rule-based, statistical anomaly detection etc.
Blocking options	Block or allow packets at the network level.	Detect attacks and either block traffic directly or send an alarm.
Hardware features	Usually has many physical network interfaces in order to segment the network into different security zones.	Must be high performance to perform Deep Packet Inspection and not slow down the traffic.

## Proactive Cybersecurity tailoring through deception techniques

Concerning the security of the computer software, its Operating System (OS) and applications running on it, an Antivirus is considered the most popular tool to defend attacks within that attack surface. An antivirus, succinctly, scans the OS and applications for malware and if a positive detection is found, it will remove it immediately, move the infected files to quarantine or even repair them. Antiviruses identify that a file is suspicious of being malicious mainly through signature-based detection, heuristics, sandboxing (i.e., testing it in a virtual and isolated environment) or machine learning. As malware evolves, some antivirus software also started to protect against other threats like malicious URLs, email spam and phishing.

The vital point to extract from this discussion is that all the above-mentioned security enforcing mechanisms have two things in common. Every single one of them is reactive to a cyber-attack and rely mainly on a database of known attack patterns, malicious files, or Internet Protocol (IP) addresses. Basically, having the same posture and behavior but searching for different threats in different attack surfaces. With the computer security landscape and some of its primary defensive mechanisms in mind, how and why should deception be inserted into computer security processes and infrastructure?

Information technology infrastructure has become the new battlefield in the twenty-first century and in this modern cyber warfare, the prevalence of cyber asymmetry is undeniably a serious problem to take into consideration. This means that the defender must not fail in any protection mechanism while the attacker needs only one vulnerability to exploit in order to disrupt an entire system. Existing reactive cyber defense techniques, such as perimeter-based and rule-based security measures like firewalls and intrusion detection systems, discussed previously, do not provide a proper level of security nowadays, as they depend heavily on known attack descriptions, patterns, and signatures. These types of strategies can often let a skilled adversary to be stealthy enough to learn about the system, discover further vulnerabilities and to perform lateral movement. They may also avoid static signature-based through reconnaissance, fingerprinting or simply by social engineering. Therefore, a proactive posture must be taken into account by the defenders, to break this asymmetry.

Cyber deception is an intentional (as it involves humans) misrepresentation of real IT systems, or parts of a system, to manipulate adversary's course of actions under the premises of the defenders' rules to inflict self-harm in the adversary, following the proposition of the adopted deception definition presented in Section 2.1, providing a proactive cyber defense that can balance or reverse the asymmetry. Deception-based solutions offer a promising complement (not alternative) to reactive and typical cybersecurity measures by providing an additional layer of protection through its proactiveness and active defense posture. Any

component in a computer system can be used for deception - a computer, a network, a service, a credential, a data item, etc.. It is important to notice that deception operations are not part of the normal operation and are only relevant upon attacker interaction or are only revealed upon attack detection.

For the purpose of demystifying deception technology, it is important to take a step into computer security history. One of the earliest recorded detailed descriptions dates back to 1989 when deception was used to detect and trace a cyber-attack. Cliff Stoll, a systems manager, wrote his first-person account “The Cuckoo’s Egg: Tracking a Spy Through the Maze of Computer Espionage” where a complex and detailed cyber-attack and defense was depicted. Stoll was intrigued by a 75-cent disparity in the billing for system usage [56]. The account causing this discrepancy was marked as dormant by the Unix system accounting and lead him to conclude that a hacker was in the system which must have acquired system privileges to modify the Unix accounting file. With the objective of tracking the hacker’s footsteps, Stoll physically wired teletypes and portable terminals to each of the modem lines which allowed the recording of every keystroke that came through and enabled the examination of the hacker without detection. When the hacker tried to extract files, Stoll slowed him down. In other machines, Stoll programmed trap doors to be notified when known hacked accounts were touched. To lure the attacker, the author created a complete believable persona of a secretary and a fake project based on the hacker’s behavior profile believing it would entice him to stay connected [20]. Stoll ended up tracking and identifying the perpetrator, simultaneously learning his procedures. Other representation on the use of deception is present on Bill Cheswick’s 1992 “An Evening with Berferd in Which a Cracker is Lured, Endured, and Studied” where similar procedures can be found [57].

As discussed in Section 2.2, deception does not only rely on misleading opponents, and so, the objective of the use of cyber deception is beyond misdirection, as other techniques can be used. Cyber deception can deflect its opponent away from its original target to a false or even non-existing target (avoiding confrontation), distort adversary’s perception of the IT infrastructure by adding entropy, ambiguity, and decoys to the network, deplete the enemy by consuming its computational power, delaying the attack propagation and finally, cyber deception can be used to engage with the adversaries to stir them down under the defender’s premises [58]. MILDEC use cases can also enrich cyber deception techniques e.g., causing confusion of adversary’s perception (as previously stated), causing the misallocation of forces by the adversary, or motivating the revelation of attacker’s strengths.



## Proactive Cybersecurity tailoring through deception techniques

It is of paramount importance to refer that deceptive maneuvers and techniques can be utilized in both offensive [59] and defensive perspectives, whilst only be discussed here to defender's advantage. Deceptive technology can be applied in a wide spectrum, from hardware and legacy applications to cloud-based and Internet of Things (IoT) environments. Adopting the use of deception into the cybersecurity practices enables an active defense posture as an attack can be prevented before its breach detection on the IT infrastructure, mitigating its intrinsic consequences. Alongside this advantage, others can also be listed:

- Contributing to lower the rate of false positives typical of reactive security appliances (discussed previously), reducing the fatigue of addressing security alerts.
- The raw and real-time gain of information about the attacker and its TTP. *Postmortem* forensics can also be performed.
- The discovery of technical zero-day vulnerabilities and operational vulnerabilities.
- No necessity of disrupting the business network as the “attack” is occurring in a separate isolated environment.
- Reducing the attacker dwell time (time since intrusion to its detection by defenders) on the system and response to threats.

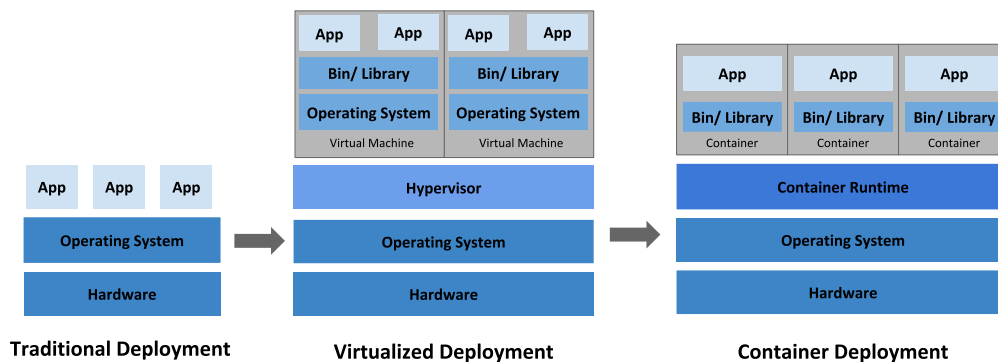
Recalling *Figure 7*, where a series of layers were identified as being part of cybersecurity affairs, it is viable to assert that deception can and should be used as complementary tool in all recognized layers. The disposition of using it as an instrument can bring multiple benefits that otherwise would not be achievable. Hence, how can the use of deception be instrumentalized in IT and security infrastructures?

With the aim of utilizing deception as a technology in computer security, some concepts need to be ascertained. Two pillars of technology enabling deception were identified, as it is hypothesized that “*given the current evolutions in computing (virtualization) and networking (software-defined networking), we now have a more mutable and forgiving environment to do computer deception across the entire network and operating system stack*” [60]. Both concepts will be outlined in the following sub-sections. The extensive study of these concepts is out of scope for this work, thus only an introduction of each is presented for acquaintance purposes.

## 2.4.1. Virtualization and Containerization

Virtualization constitutes the processes that allow a more efficient utilization of a computer's hardware, which is the main reason that justifies the existence of cloud computing. By using a software, called a hypervisor, to create an abstraction layer between the physical resources and the virtual environment, virtualization divides the use of hardware elements (processors, memory, storage, etc.) into multiple Virtual Machines (VMs). The hypervisor can be placed on top of an OS (e.g., when using virtualization on a personal computer) or directly onto the hardware (e.g., in a server). Each VM can then run its own OS behaving independently as a computer, consuming the resources that are allocated to it by the hypervisor. Some underlying benefits of using virtualization are resource efficiency, facilitated management by replacing the need of more physical computers with software defined computers (i.e., VMs), fault tolerance (redundancy of VMs) and faster provisioning [61].

VMs and Containers have both the capability of creating an isolated environment, being however, different on a technical point of view. The evolution of virtualization and deployment techniques are illustrated in *Figure 8*.



**Figure 8** - Deployment types, extracted from [62].

A Container is defined as an isolated group of processes that contain its own root filesystem and process namespace with kernel and services of the OS being only shared within the group [63]. Basically, a container image is a standard and portable unit of software that “*packages up code and all its dependencies so the application runs quickly and reliably*” [64] and becomes a Container in runtime. The main difference between a VM and a Container consists of the set-up of an application execution, where, with a VM, it is required the

installation of an entire OS, contrary to Containers, where application execution only requires a host operating system, having a much smaller size than a VM, resulting on a faster booting and execution process.

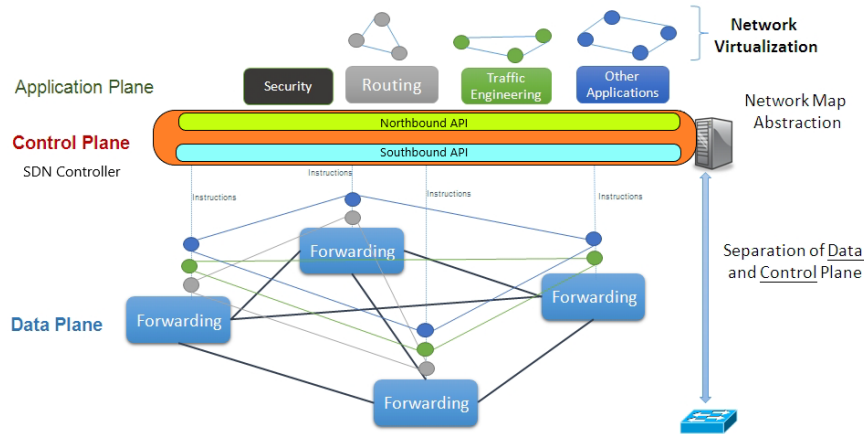
Virtualization is utmost necessary to possess and dynamically create virtual assets that are able to simulate real agents and to conduct other deceptive techniques involving the use of computerized elements, as the utilization of physical machines would not make deception flexibility achievable. Furthermore, containerization technique will be preferred over virtual machines as the advantages of containerization better suit the dynamic use of deception. The primary advantages of Containers are their lightweight nature, isolation, elasticity, and scalability.

## 2.4.2. Software Defined Networking

Software Defined Networking (SDN) is an architecture designed to make a network more flexible and easier to manage by centralizing management that abstracts the control plane from the data forwarding function. SDN enables the decoupling of the network control logic from the physical devices, such as routers or switches, which control the travelling of information in underlying network offering policy-based management [65] [66]. The idea of programmable networks *“has been proposed as a means to remedy this situation by promoting innovation in network management and the deployment of network services through programmability of the underlying network entities using some sort of an open network”* Application Programming Interface (API) [67]. A SDN architecture is composed of several elements, listed below:

- A controller is the core element of the architecture, responsible for the centralized management, control, and automation across physical and virtual network environments.
- A set of applications that interact with the controller to propagate or retrieve data concerning resource allocation.
- Southbound APIs are responsible for exchanging information between the controller and each physical network device.
- Northbound APIs are accountable for relaying data between the controller and applications concerning its policy engines.

An overview of the SDN architecture is illustrated in *Figure 9*.



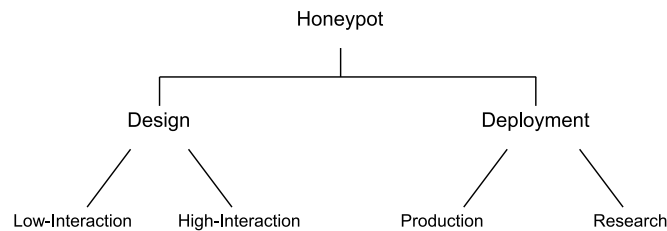
**Figure 9** - SDN architecture, adapted from [67].

It is important to notice that SDN has seen a wide adoption in data centers, Wide-Area Networks (WANs) and access networks, corresponding to an adoption rate of 64%, 58% and 40% respectively [68]. Moreover, a significant amount of capital is being injected on SDN development as its worldwide market size in 2022 corresponds to 26.02 billion U.S. dollars expecting to reach 51.08 billion U.S. dollars in 2027 [69]. Two examples of SDN controller technologies are Open Network Operating System (ONOS) [70] and OpenDaylight [71]. The use of SDN can provide a programmable environment over network configuration management that enables comprehensive diagnosis of the network elements enabling a quick deception assembling and dynamic response through its rapid resource allocation and deallocation.

## 2.5. Honeypots and Honeynets

An evolution in deception technology is recognized as, formerly, honeypots were synonymous of this technology in the cybersecurity landscape. Nowadays, honeypots are just a type of deceptive appliance and are computational or cyber assets, that are not a part of normal business processes, set up typically as a mimic to entice (create a perception) and lure (target action) a possible attacker to interact with it while away from legitimate targets. In case of interaction, this unit will detect the intrusion, alert about it, and perform some action or even an inaction depending on its configuration towards the attacker. A honeypot can be anything

from simple data to services (e.g., credentials, a database, a web server, etc.) and its value lies in simplicity since it is built with intention of being compromised. Common terminology refers to the terms “honeypot”, “honeycréd”, “honeypórt” and “spam trap” to types of honeypots that are only pieces of data, such as credentials, ports or fake email addresses, respectively, while service (e.g., Telnet, SSH, etc.) decoys are commonly addressed as honeypots [72]. Recalling deception taxonomy, specifically taxonomy by method, discussed in Section 2.2.1, honeypots (englobing all the above mentioned terminologies) fall into the “display” category within “active deception”, as seen on *Figure 3*. Honeypots and its variations can be seen as the most granular artifact used in deception technology, as numerous of implementations can be found for different services with different versions and for immense types of data. Honeypots can be classified by at least two factors, as seen on *Figure 10*.

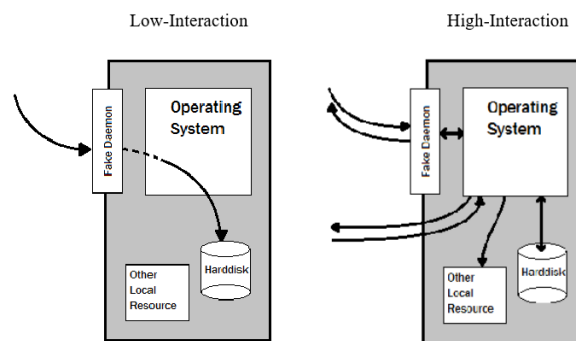


**Figure 10** - Honeypot classification.

Firstly, two types of honeypots can be distinguished based on their design, particularly by the level of interactivity that they have with the attacker. “*Low-interaction honeypots do not implement real functional services*” as they emulate simple services (and a single version of it) or network stack parts of an operating system and do not offer attackers the possibility to realize operations [73]. Its advantages are its easy set-up and not being resource intensive, as a single computer can execute various of these honeypots. For example, a honeypot can be a version of a given software running that has an identified vulnerability in Common Vulnerabilities and Exposures (CVE) database, thus tempting a possible attacker to contact with the honeypot. Due to its lightweight nature, low-interaction honeypots can be massively deployed to the Internet with the objective of providing quantitative global statistics on malicious activity. Their simpleness and easy administration are also its biggest disadvantage because not much can be learnt about the attacker, as it is limited to the detection and capture of known activity in a certain honeypot. Examples of low-interaction honeypots are Honeyd [74], Nepenthes [75] and mysql-honeypotd [76]. On the other hand, high-interaction honeypots do not simply emulate a service, instead they can be complex systems of physical computers or VMs that offer real

## Proactive Cybersecurity tailoring through deception techniques

services (e.g., *login accesses*) and a real operating system. These are suitable for monitoring the activity of attackers, as high-interaction honeypots offer a wider spectrum of services to exploit. A high-interaction decoy target, when compromised, enables better attacker progression by having bigger set of interactions that the same can perform in the honeypot. “A high-interaction honeypot is designed to get attackers to invest as much time as possible inside the honeypot” [77]. High-interaction honeypots present a clear evolution regarding deception sophistication, and in a way, they can be categorized as “dynamic” deception within sophistication taxonomy (as depicted in 2.2.3) while low-interaction honeypots can be considered as “static”. An example of possible interactions with both types of honeypots is visible on *Figure 11*.



**Figure 11** - Low-Interaction vs High-Interaction Honeypots, extracted from [78].

Examples of high interaction honeypots are Honeybow [79] and V.Nicomette’s *et al* experiment [73]. Some sources claim that other types of honeypots exist regarding its design. These refer to pure and mid-interaction honeypots. Summarily, pure honeypots exactly mimic the actual production environment of an organization and mid-interaction honeypots “*imitate elements of the application layer, but they do not have an operating system*” [77], being an intermediary type between low and high-interaction honeypots. For simplicity of conceptualization, mid-interaction honeypots are not taken into account, as there is not really a measurement to separate it from low or high-interaction. Their main purpose is “*to confuse an attacker or stall them so the organization has more time to ascertain how to react to the kind of attack in question*” [77]. Pure honeypots can be seen as high-interaction honeypots taken to its extreme, as they try to mimic a full-scale system.

Secondly, honeypots can be distinguished by the type of deployment, particularly on its placement in the network. The production honeypot is the most common type, and its purpose

is to collect information and intelligence (e.g., IPs, date and time of attempted intrusions, TTP, etc.) about cyber-attacks and are placed within the production network, i.e., behind a firewall and/or other defensive mechanism. They act as a warning system to suspicious activity that have surpassed the perimeter defenses and they can be low-interaction (only to detect intrusions) or high-interaction (to analyze the attacker’s exploitation). By contrast, research honeypots are mainly used for educational purposes and are placed directly facing the Internet with the main objective of gathering intelligence on the threats that an organization might face. Typically, these honeypots share the same public IP address subnet range as the production (real) network [20]. An extensive overview on various types of honeypots can be found on [80].

The initial enthusiasm with honeypots gradually faded when it became obvious that maintaining a credible deception environment capable of harvesting data from real threats was no easy solution. Deception technology has *“moved beyond the mere honeypot, and now has features that allow it to appear more credible, exfiltrate key data, and perform telemetry, and no longer in the tedious, manual manner of earlier honeypots”* [81]. Once the adversary accesses and gains unauthorized control of the honeypot, the same becomes useless and is eventually rendered inactive by the security team, not leaving much room to collect valuable data. The authors of *“Enhancing Honeypot Deception Capability”* built a threat model on honeypots to demonstrate the main drawbacks of honeypot implementation [82], illustrated on *Figure 12*. Threat modelling consists of proactively and systematically identifying potential threats of a given system, resulting on a conceptual representation or enumeration of these threats and needs to be kept up to date with system’s changes to be useful and valuable.

Attacks	Attacks Description	Root Causes	Attacks Surface	Possible Mitigation
Poisoning	Flooding the Honeypots	Honeypot fingerprinting	Multiple connections with same sources with no limit	Using LaBrea Tarpit
Compromising	Staging Internal Attacks	Honeypot fingerprinting	Honeypots that allow any activity entering or leaving the honeypot	Using Honeywall
Learning	Gaining private information	Honeypot fingerprinting	The honeypots are allowed to access the root system	Limit privileges

Figure 12 - Honeypots Threat Modelling, extracted from [82].

## Proactive Cybersecurity tailoring through deception techniques

As discussed in the same paper, the authors identify that one of the main counter deception strategies is to use fingerprinting techniques to positively identify a honeypot (i.e., root cause). The threats illustrated in *Figure 12* are the result of a successful fingerprinting by the attacker. Poisoning consists of invalidating the honeypot by flooding it with garbage packets, compromising the honeypot refers to turning the honeypot it into a bot controlled by the attacker and ultimately using against the organization or for spying and gaining more intel. IT security teams must be aware to when the attacker discovers that a honeypot is indeed a honeypot. To be able to do that, the defenders must acquire as most data as possible and in real-time, preferably. These tasks become extremely difficult when simply addressing honeypots. Considering “Demystifying Deception Technology : A Survey”, a set of more precise counter measures against honeypots were described [83], as seen on *Figure 13*.

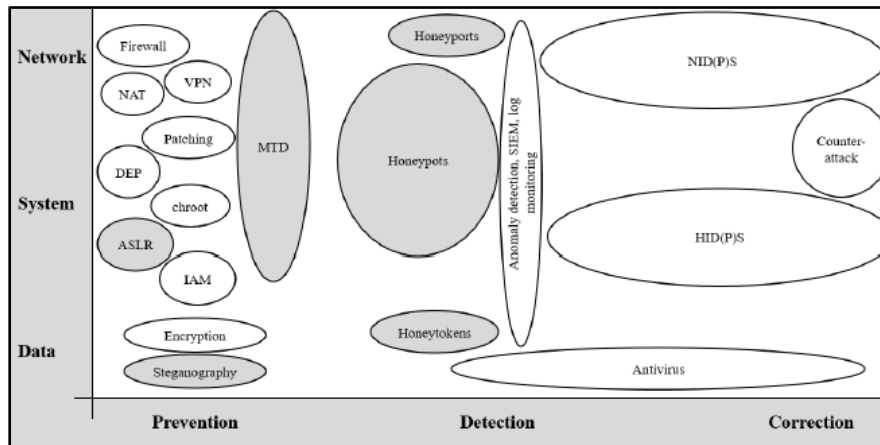
Method	Detail	Target	Mitigation	Ref.
Temporal behavior	Measure RTT to expose correlations between IP addresses	honeyd, virtual honeypots	Simulating timing behavior	[114, 115, 118, 121, 122]
Stack fingerprinting	Send corrupted packets and analyze responses	Simulated communication stacks	Implementation of full TCP/IP-stack	[27,123]
Functional probing	Use provided functions and verify status	SMTP and DNS	Implementation of full functionality	[113,121]
System call behaviour	Anomalies in temporal behavior or memory locations	Linux systems	Simulating timing behaviour, KASLR	[27, 121, 122,124]
Network traffic	Analyze RX and TX network traffic e.g. number of bytes	Network based data exfiltration e.g. Sebek	Hinder network monitoring, VMI, Proxy	[121]
UML detection	dmesg output, network device, /proc/, memory layout	UML based host isolation	Manipulating tools to show related information	[122]
VMware detection	Hardware e.g. MAC address, I/O backdoor	VMware based host isolation	Customize hardware, patch I/O backdoor [27]	[27, 122]
Debugger detection	Use ptrace() function, IsDebuggerPresent() function or memory search for 0xCC	e.g. Cuckoo	-	[122,125]
Semantic gap	Manipulate kernel data structure	VMI	-	[116]
Customiz.	Search for default strings	-	Customize systems	[27, 117, 120–122]

**Figure 13** - Overview of counter measures against honeypots, extracted from [83].

A comparison between typical security measures and honeypots concerning their purpose was also depicted by the authors of [83] can be observed on *Figure 14*. In their analysis, three different layers of resources to protect (network, system, and data) and three defensive objectives are taken into consideration (prevention, detection, and correction). The security techniques and primitives colored in grey are deception-based technologies. These include (Kernel) Address Space Layout Randomization ((K)ASLR), Moving Target Defense (MTD), Steganography, Honeytokens, Honeyports, and Honeypots.



## Proactive Cybersecurity tailoring through deception techniques



**Figure 14** - Deception based technologies vs typical security mechanisms, extracted from [83].

As illustrated on *Figure 14*, the focus of honeypots is to detect an attack as early as possible and prevent it. Other tools depicted in this document (IDS, IPS, etc.) center on correcting an already launched attack.

As analyzed previously, where many counter measures and threats were depicted in conjunction with the list of counter measures on *Figure 13*, it is feasible to state that a “standalone honeypot may not provide enough of an incentive for today's sophisticated cyber attacker. It may also not provide enough data to help IT security become stronger (...) On the other hand, it may be adequate to prompt an attacker to quickly leave” [84]. This quote infers that a honeypot can be used to deflect an adversary upon identification, this concept is presented also on [83] as a “fake honeypot”. This technique can be utilized when confrontation is not sought by the defender and may be proven useful, as discussed in Section 2.2, where deception techniques and practices were presented.

To address and overcome the above-mentioned difficulties, honeynets were conceptualized. An important aspect to note is that there is an entropy surrounding high-interaction honeypots and honeynet definitions. Some sources claim that these concepts are similar, notwithstanding, this thesis follows the definition that honeynets “are Information Systems that deploy Honeyports to track footprints of attackers” [85] and a “typical honeynet solution is an hybrid system consisting of a combination of high interaction and low interaction honeypots, which provides a good balance among scalability, performance, fidelity and containment” [86]. By creating a network of interconnected honeypots deployed following a specific network topology, a honeynet can be more difficult for the attacker to fingerprint it and conclude that he is being conned. Honeynets can be separated in generations according to architectural techniques evolution. Generation I honeynets were first developed in 1999 and

had simple “methodology, limited capability, and it is highly effective in detecting the automated attacks” [87]. It runs on the third OSI layer and for this reason was implemented in an isolated network. The 2<sup>nd</sup> generation of honeynets was built in 2002 by the HoneyNet Project [88] and “it runs on the second ISO/OSI layer and it inspects the outbound data and blocks or modifies data” [87]. The main objective of 2<sup>nd</sup> generation honeynets conception was to ensure that compromised honeypots within the honeynet are not used to attack other computers outside the honeynet. This is done by a containment wall called “Honeywall”. Generation III of honeynets, released in 2004, share the same architecture as the predecessor differing only in deployment and management. “The 3rd generation of honeynet offers a way of analysis of data from different sources without the person being obliged to go through different data sources manually and to try to determine the relationships” [89]. Every honeynet should contain three main capabilities: data control, data capture, and data collection in order to be useful for the defender. Data control consist of a set of measures implemented to mitigate the risk of the attacker using the compromised honeynet to attack other non-honeynet system. Data capture refers to logging all the attacker’s activity for investigation (e.g., system calls, modified files, network packets, etc.). Data collection include mechanisms to, securely and undetectably, forward all the captured data to a centralized secure data collection point. A typical architecture of Generation III is illustrated on *Figure 15*.

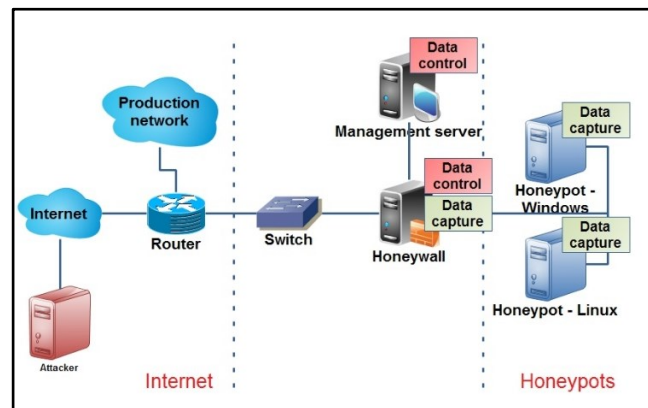
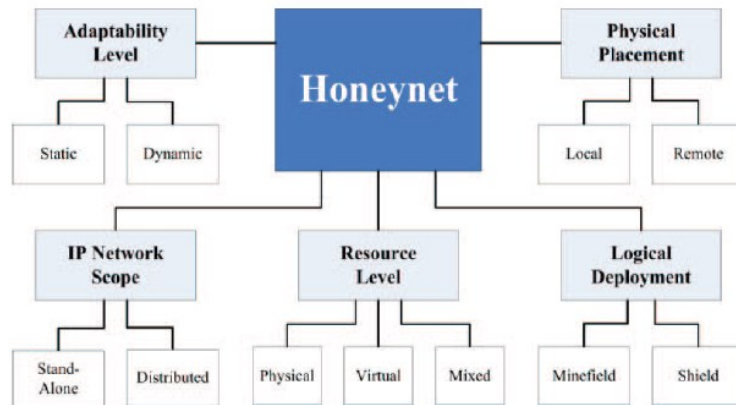


Figure 15 - Generation III Honeynet architecture, extracted from [87].

A honeynet’s “Honeywall” needs to keep the attacker contained in the honey environment by not making the perpetrator make outbound connections. This decision may be counter intuitive and counterproductive because, by not allowing these connections, knowledge about attack patterns and other type of information (e.g., a possible Command and Control server) is being discarded. In order to obtain as most valuable information as possible, a balance

in this mechanism must be found. Increasing the risk of compromising other systems will lead to a probable acquisition of a greater number of important attacker data.

Honeynet solution taxonomy and classification was presented in [86]. The authors classify honeynets through five main criteria: resource level, adaptability level, IP network scope, physical placement, and logical deployment as illustrated on *Figure 16*.



**Figure 16** - Honeynet classification scheme, extracted from [86].

A summary of each category and its sub-categories will now be depicted, as honeynet’s capabilities are extremely important to acknowledge. Regarding honeynet’s resource level, a honeynet can have its honeypots running on physical, virtual, or mixed environments. A honeynet’s adaptability level refers to the capability of dynamically change its configuration or topology. Static honeynets don’t have the capability of being reconfigured while dynamic honeynets are able to alter its topology (adding or removing nodes) and its honeypots’ configuration. This permits the existence of honeynets that can react to given events and reconfigure accordingly. IP network scope indicates in what way are IP addresses assigned to the honeypots within the honeynet. A stand-alone honeynet uses IP addresses from a common IP network prefix while a distributed honeynet will cover multiple networks concurrently. A distributed honeynet has the advantages of having a greater ability to capture network related suspicious activities as it is partitioned and some or even all components of the honeynet can be placed in geographically distinct locations. Concerning physical placement, a honeynet can be either local, if it is limited to one location, or remote, if it does not impose any physical limitations. Finally, logical deployment separates implementations regarding honeynet’s relationship with the production network. A “*minefield*” honeynet honeypots are often logically deployed among production systems as they function as a mine, capturing data upon interaction.

Using shield deployment strategy, the honeynet acts as mirror to the production network. The same authors of [86] also describe types of honeynet solutions and categorized them according to their proposed taxonomy that include Gen III honeynets and add Shadow, Potemkin, and anti-phishing types of solutions.

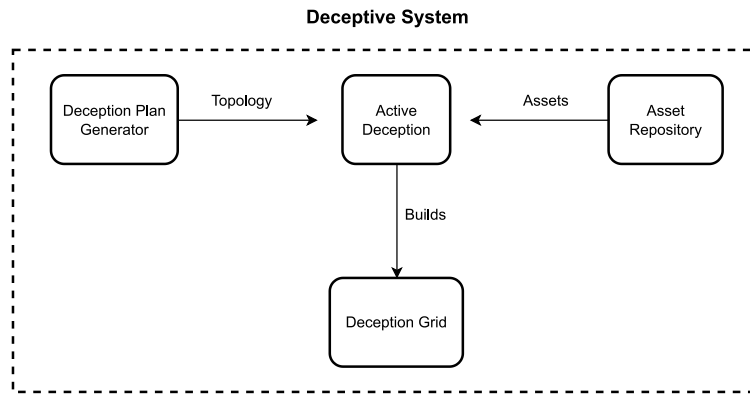
Literature concerning honeynet implementations is not difficult to find, as many solutions were developed addressing numerous purposes. For example, HoneyNettrap [90] was developed to detect and mitigate Distributed Denial-of-Service (DDoS) attacks. Other example is an IoT honeynet that was developed to detect attempts of compromising of IoT devices with specific vulnerabilities [91]. Other approaches to honeynet implementation do not focus on specific type of attack, and instead, try to automatically identify the type of attack and automatically reconfigure the honeynet. One example of this is an implementation that resorted to SDN to build an “intelligent honeynet” [92].

Honeynets enable a more sophisticated use of deception due to the possibility of dynamic reconfiguration, complex networking of honeypots and other depicted functionalities. The only aspect missing is a planned use of deception to premeditatedly build deception schemes through honeynet(s) implantation and to adapt and deal with ongoing attacks with the objective of capturing and collecting as much data as possible through attacker behavior tailoring and manipulation.

## 2.6. Deception Grids

Deception grids aim to utilize honeynets in conjunction with a planned use of deception to enable a more sophisticated active defense posture. Combining the honeynet’s honeypot clustering and topology reconfiguration abilities with a methodic deception planning conception, deceptive capabilities can be peaked to successfully manipulate the attacker’s course of actions and behavior within the deceptive grid(s) and obtain critical information about his TTP, while defending against the attack because it is occurring in an isolated environment from the business network. A typical deception system has the standard components observed in *Figure 17*.

## Proactive Cybersecurity tailoring through deception techniques



**Figure 17** - Example of a typical deception system.

The deceptive system contains three principal components that capacitate the fabrication of a deception grid. Firstly, a “deception plan generator”, that for a given context develops a deception strategy that involve a certain set of assets. Secondly, at the opposite side, a repository of assets needs to be kept available to fetch the required deceptive artifacts. These can be virtual or physical assets, being majorly virtual due to recognized advantages discussed in Section 2.4.1. Lastly, given the inputs from the other modules, the core module builds and deploys the deception grid. The main difference between a deception grid and a honeynet is *de facto* the calculated and dynamic use of deception that is reflected in the topology and type of assets deployed. This use can be automated, e.g., with machine learning techniques, or by manually conceiving and applying the deceptive plans. A particular deception plan will deploy a certain topology with certain assets and a plan should be reconfigurable when deployed. A honeynet can be viewed as a build of a deception grid resultant of a particular deception plan. Deception grids, by using honeynets, are considered “*hybrid techniques*” of cyber deception as they involve host-based techniques (falsifying entities within a targeted device e.g., VM, filesystem, etc.) in conjunction with “*network-based techniques*” [60]. The conception of a deceptive system enables adaptive and premeditative levels of sophistication to the process of deception (Section 2.2.3).

Resorting to the state of the art of these types of solutions and systems to understand each implementation’s specificities is of paramount importance. However, it is duly noted that documentation surrounding this subject is not abundant. ACyDS or Adaptive Cyber Deception System [93] provides a unique virtual network view to each host in an enterprise network. This work is considered important as related work because of its dynamic and adaptive nature which aligns with the characteristics of the present work. To accomplish that, the authors rely on SDN technology and capabilities (introduced on Section 2.4.2) and use SDN Controllers, SDN

switches, and other components to create individual deceptive network views for hosts. To test their hypothesized system, a proof-of-concept prototype was implemented to test the system's capabilities in correctly handling Address Resolution Protocol (ARP), User Datagram Protocol (UDP), Transmission Control Protocol (TCP), ping and traceroute packets. ACyDS, however, does not present a solution that applies a planned and diversified use of deception to its system (defending only against reconnaissance attacks). For securing cyber-physical systems, a generic deceptive Defense-based Data Acquisition Framework (DDAF) [94] was developed for any hierarchical Supervisory Control And Data Acquisition (SCADA) network. This framework secures the data acquisition steps by shuffling the sensor IP addresses/IDs at each node in the hierarchy. With the use of SDN technology, DDAF can deceive an attacker by replacing the original sensors identifiers with deceptive IDs allowing the detection, mitigation and localize cyber-attacks. This framework focuses on the detection of cyber-attacks on SCADA networks with the use of deception and not on observing and learning perpetrator's patterns. Leaning on the idea of dynamic deception, a design of Advanced Persistent Threat (APT) attack defense system based on dynamic deception framework was developed [95]. This defense system can dynamically change and effectively defend against APT attacks through hybrid encryption communication mechanism based on sockets, a dynamic IP address generation method, a dynamic timing selection method and a dynamic policy allocation mechanism. From this paper, an idea of a defensive deception process is withdrawn and analyzed which provides a good insight for the present work. To facilitate the swift, safe, and effective cyber deception deployment enabling an open environment for developing sophisticated cyber deception applications an Active Deception Framework (ADF) was proposed [58]. ADF provides a deception API that can be used to build multi-strategy deception policies through SDN. This framework is possibly the most similar tool in terms of base functionalities to the present work's objective that was found. Its architecture, that involves communication with an adaptive deception module API and a SDN controller API is taken into consideration. The examination of the proposed adversary adaptive deception deployment strategy, the framework's architecture and API methods proves to be of vital importance to the present work's envisioning. In spite of being a solid foundation to enable active defense through dynamic and adaptive deception, the referred paper does not possess integration with a complete set of deceptive enabling components, being limited to its API operations.

Furthermore, other papers can be considered to address additional concerns of the present work. To plan the cyber deception, the examination of an autonomous planning and orchestration for deception called CHIMERA [96] and similar solutions are worth mentioning.

CHIMERA is a theoretical framework and implementation for an autonomous goal-oriented cyber deception planner that optimizes deception decision-making. Additionally, a dynamic scheduling and adaptive resource allocation process can be depicted through a software defined framework [97]. This enables intelligent decisions through different security functions. The multi-layered approach and responsibility depiction of each component in the presented layers of the referred paper enlightened the necessity of some core functionalities that needed to be discussed and implemented in the present contribution.

## 2.7. Cyber Intelligence and Deception Planning

Even though some level of uncertainty is present in the deceptive process that can inevitably alter the outcome (as discussed on Section 2.2), successful deceptions are not based on luck and do not happen by accident. Effective deception processes rely predominantly on solid planning and preparation phases. Deception planning englobes all the required steps and phases to build one or more deception plans. To efficiently understand the deception process, two concepts need to first be clarified: indicators and channels. *“Indicators are individual snippets of information about the capabilities, intentions, and actions of an actor created through the actor’s interaction with the environment”* [98]. These informational breadcrumbs can be real (truthful) or deceptive and provide the target with required data for him to draw the conclusion that he is facing reality. Channels are *“the specific ways in which information about a given subject reaches an audience”* [99]. The communication cycle between the deceiver and its target is present in *Figure 18*.

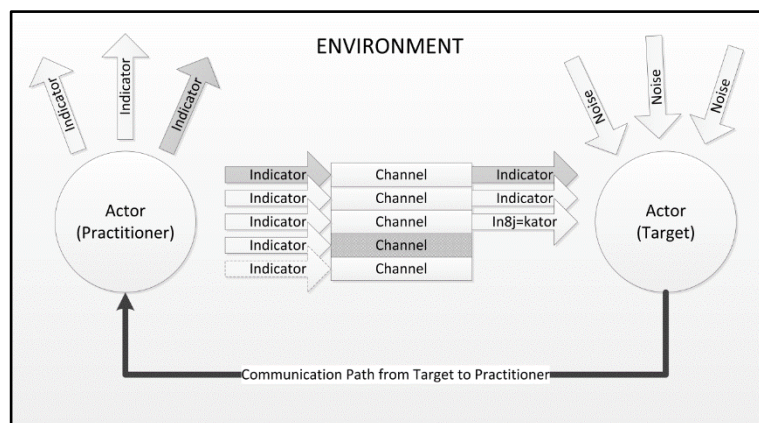
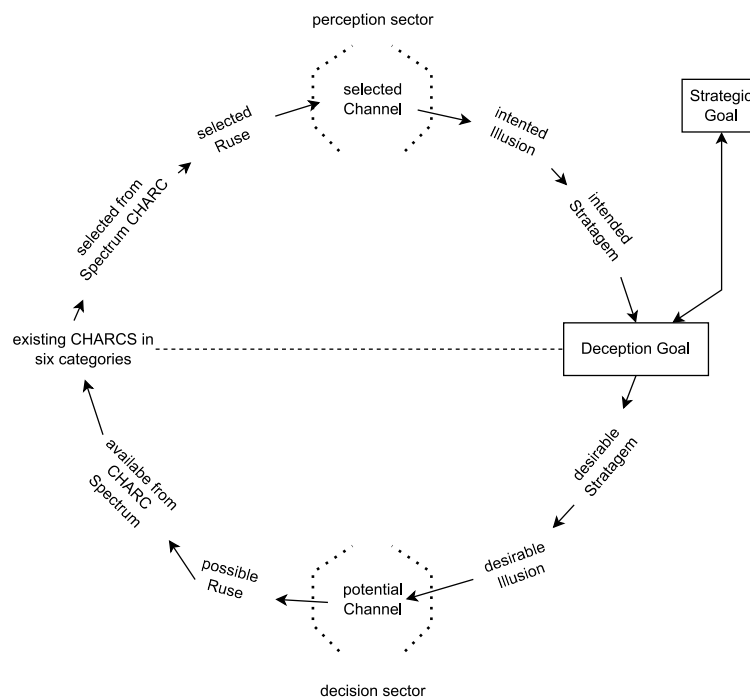


Figure 18 - Communication cycle, extracted from [100].

## Proactive Cybersecurity tailoring through deception techniques

The practitioner emits indicators through channels to convey them to the target. Some indicators can represent practitioner's true capabilities and intentions (represented in white) or be deceptive (in grey). The "Indicator" indicator represents an indicator that was corrupted by transmission errors, illustrating that errors may occur when delivering indicators to the target. Channels can be closed by denial strategies to prevent the target to acknowledge it (channel in grey). Finally, noise represent indicators that origin from other actors or the environment, that interfere with target's perceptions of the indicators. Channels may exist prior to the deceptive process or be created specifically for that purpose. Recalling to taxonomy by method (Section 2.2.1), a strategy belonging to "cover" category will minimize or limit the number of channels available to the adversary while "active deception" will increase it. Bell and Whaley propose a deception planning method based on a loop presented in *Figure 19*.



**Figure 19** - The deception planning loop, adapted from [12].

The bottom half of the loop, present in *Figure 19*, represents the "decision sector" where desirable stratagems and illusions, possible channels and ruses are planned. The upper half of the loop, the "perception sector", selects a ruse and a channel to generate the intended illusion and achieve the deceptive goal. Monroe considers that this process, while being thorough, falls short because the process "ends with what the attacker thinks, rather than what the attacker does" [101]. The same author proposes a revised process that merges Bell and Whaley's



planning loop (Figure 19) with MILDEC’s “See-Think-Do” process [102]. The resulting process of the described merge is illustrated on Figure 20.

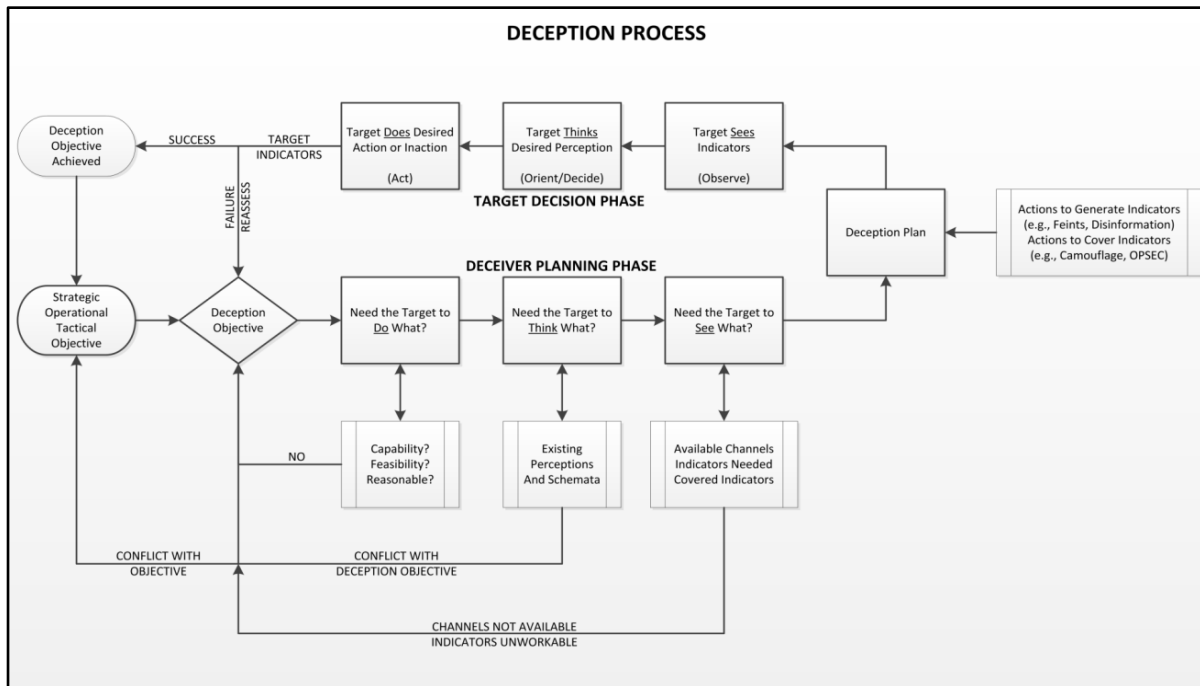
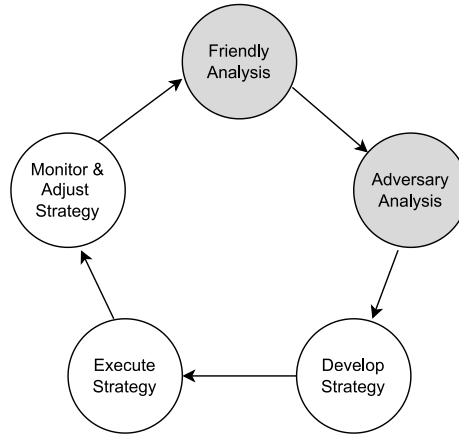


Figure 20 - Revised deception process, extracted from [101].

This revised process can provide a better understanding on the context of decision-making in the deception process. The process starts with the identification of a strategic, operational, or tactical objective, wherefrom a deception objective is derived. Then, an evaluation of what a target must see (indicators), think (perception), and do (action) needs to take place. After the adversary’s perception is decided, then the set of indicators necessary to create the perception must be generated. This set will build the scenario that the adversary will encounter, where it is vital to understand what the target already perceives “as the existing perception not only drives how new indicators are interpreted through filters like the expectancy and confirmation biases, but also what the target sees the indicators as” [101]. The set of channels need also to be managed to determine what channels are used, created, or denied to the target. Finally, once the deception plan is underway, collecting enemy indicators is of vital importance to help the deceiver to determine if the deception was successful or not. This information may enable the practitioner to adapt or simply abandon that deception and deal with the aftermath. Summarily, the deceptive process involves an analysis of the situation prior to the presentation of the deceptive scenario, the knowledge about the adversary, the

## Proactive Cybersecurity tailoring through deception techniques

development process of the strategy, its execution and in the end its monitoring. This cycle englobes Monroe's depicted process, represented in white, and reinforces the idea of friendly/adversary analysis, represented in grey, as seen on *Figure 21*.



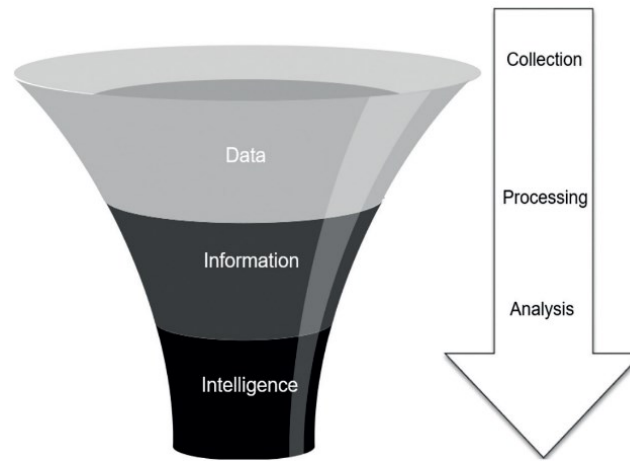
**Figure 21** - Deception strategy development and execution, adapted from [103].

As illustrated on *Figure 21*, there are two analysis that must occur before the development of a deception strategy. “Friendly Analysis” examines what information and/or assets most important within the infrastructure you are trying to defend (this same step is present in threat modelling process) and what is the objective of the deception that you want to execute (e.g., deflect, entice, etc.). On the other hand, “adversary analysis” means to “know your enemy” i.e., to recognize potential threat actors which may want to break into your system. This typically involves not only technical knowledge about the IT infrastructure but also (and specially) your business. Adversary analysis is important to a successful strategy development as deceptions plans are dependent on the target's characteristics. Questions like “Who are our main adversaries?”, “How they operate?” and “What are they trying to accomplish?” must be answered in order to prepare and execute successful deception plans. This is the point at which the branch of Intelligence in computer security becomes extremely relevant.

Intelligence can be defined as “*secret, state activity to understand or influence foreign entities*” [104]. While numerous definitions exist, finding an exact one is beyond the scope of this work. Nonetheless, it is important to comprehend the concept. Cyber Intelligence or Cyber Threat Intelligence (CTI) applies intelligence to the cyberspace. CTI is the process of data collection, processing and information analysis that enable the understanding of a specific threat actor concerning its motives, targets, and TTP. In order to generate evidence-based knowledge from any data, a process must occur that transforms data into intelligence. This

## Proactive Cybersecurity tailoring through deception techniques

process needs to turn raw data into valuable information to the defender and enables the adjustment or creation of new defensive measures (proactive or reactive, deceptive or “typical”). Three basic steps, presented in *Figure 22*, must be taken to accomplish that.



**Figure 22** - From raw data to Intelligence, extracted from [105].

Collection involves the gathering of raw data from a wide spectrum of sources which by itself is of limited use. Regarding computer security, this might mean a list of IP addresses or logs in bulk. Processing transforms data into information, as it is aggregated and compared with other sources to give meaning to the previously collected data, while at the same time asserting its relevance and reliability. Finally, the filtered information gets analyzed, verified and converted into intelligence. At this point, security analysts test and recognize trends, patterns, and other insights they can use to answer the previously asked questions. The intelligence generating process must be systematic and continuous (to keep up with emerging cyber threats), accessible, done in a timely fashion and in the worst-case scenario be responsive to an occurring cyber-attack. Intelligence helps secure the organization’s infrastructure against identified threat actors. In the context of cybersecurity, these artifacts can be materialized as:

- Indicators of Compromise (IoC) and Yet Another Recursive Acronym (YARA) rules.
- Client-derived data (e.g., information extracted from a client’s Security Information and Event Management (SIEM)).
- Information extracted from Deep/Dark Web (e.g., member-only hacking forums frequented by cybercriminals and/or hackers or malware marketplaces).

### Proactive Cybersecurity tailoring through deception techniques

- Open-source Intelligence (OSINT) (e.g., data retrieved from social media).
- Malware analysis and network forensics.
- Paste sites.
- Code repositories.
- Geopolitical strategic information.
- Public or private information from sharing platforms [106] [107] and/or reports (e.g., between companies' Computer Security Incident Response Teams (CSIRT)).

Intelligence can be an undoubtedly strong component in deception planning because, with a thorough deconstruction of the adversaries, a successful deceptive plan is more probable to be constructed. This leads to a premeditative deception regarding its level of sophistication (Section 2.2.3). These types of deceptions are based on experience and (mainly) in observations about the adversary's sensors and strategies and are the most prone to a successful outcome.

Succinctly, the perceptions used by the target to decide its course of actions rely on the indicators created and delivered by the practitioner, which implies knowledge of adversary tactics and intent. *"This is because deception is not a passive exercise. Deception is adversary engagement. As such, a successful deception program must employ people, processes, and technologies to engage the adversary in a manner that fits the program's objectives"* [60]. It is important to notice that deception plans do not necessarily always aim to actively interact with the adversary, in some circumstances the plan is to deviate from direct confrontation, as mentioned previously.

When considering cyber warfare and deception planning certain deceptive techniques may be implemented to achieve a desired outcome. Recalling deception techniques (listed on Section 2.2), Rowe [14] assesses the impact of the application of them to cyber warfare offense and defense in a scale from zero to ten, with zero being totally inappropriate while ten being most appropriate. This assessment is present on *Table 3*.

## Proactive Cybersecurity tailoring through deception techniques

**Table 3** - Summary of Rowe's assessment of deceptive types in information-system attack, extracted from [14].

Deception type	Useful for accomplishing a cyberwar attack?	Useful in defending against a cyberwar attack?
concealment of resources	2	2
concealment of intentions	7	10
camouflage	5	3
disinformation	2	4
lies	1	10
displays	1	6
ruses	10	1
demonstrations	3	1
feints	6	3
insights	8	10

Subsequently to the detailed selection of the deceptive technique to apply, a technological approach needs to happen to materialize deception in the cyberspace. This may be reflected in a plan that includes the deployment of certain assets and services that contain specific vulnerabilities that are commonly exploited by a certain threat actor, to entice him to interact with the deceptive grid(s). Other examples are storing, in a deception grid's deceptive artifact, specific types of data that some threat actor is actively pursuing (e.g., hacktivists), be consciously undefended against a threat actor's TTP or be permissible to certain network traffic from a set of already blacklisted IP addresses that belong to the deceived party, targeted to be manipulated. In the case of avoiding confrontation, deploying numerous defensive artifacts and countermeasures may discourage the attacker or at least stalemate him. As stated before, deception is not subsumed only to a defensive point of view as it can be weaponized in a possible cyber-attack.

As depicted throughout this section, Intelligence has, or should have, a great part of involvement when planning deception. This branch of computer security capacitates the deception architects with valuable information concerning the adversaries. With the proper comprehension of enemy capabilities, the chances of a successful deception plan increase greatly.

# 3. Proposed Solution

The purpose of this chapter is to comprehensively describe the proposed solution and has the following organization: Section 3.1 presents the proposed approach that tackles the problems discussed in the previous chapters, its added value regarding presented related work, and its materialization through a solution architecture and inherent conceptual proposals and mechanisms. Finally, Section 3.2 showcases the correspondence between the proposed system and the deception techniques and taxonomies previously presented in Section 2.2.

## 3.1. System Architecture

An architecture concerning a deception system was pondered and elaborated, tackling the problems and the lack of sophisticated deceptive capabilities presented and discussed in the introductory chapter (specifically on Section 1.1), and throughout the sections concerning the background and related work (presented on Chapter 2). It is important to notice that although deception can and should be applied to all cybersecurity layers (illustrated in *Figure 7*), only some are covered by this solution, namely data, application, and network. The particularities of the system's architecture aim to cover most of the unresolved problems found with state-of-the-art implementations encountered and in deception technology in general.

Firstly, both honeypot's and honeynet's weaknesses and threats were identified, as discussed in Section 2.5. Nevertheless, this depiction concluded that an evolution regarding deception technology sophistication is recognized to both aforementioned concepts (honeypots and honeynets). Deception grids and systems were then addressed (on Section 2.6), culminating in the acknowledgment that deception technology is thriving to implement such systems in order to help peak cyber deceptive capabilities for organizations. However, some negative considerations and barriers were identified to encountered similar solutions and related work of deceptive systems.

The proposed system and its architecture seize the most advantageous aspects of the referred technologies while solving the subsequent identified main issues:

- Honeypot's scarce TTP collection capabilities.
- Honeypot's associated threats (portrayed on *Figure 12*).
- Honeypot's low deceptive success.
- Honeynet's lack of deception flexibility (honeynets can be seen as one build of a deception grid).
- Deception system's non modular approach.
- Deception system's incompleteness, non-materialization, and lack of comprehensiveness.

Through a modular approach that segregates and delegates clear responsibilities to system's components, premeditated and tailored levels of deception can be achieved, maximizing cyber defensive capacities for an organization. As discussed in Section 2.5,

honeypots and honeynets can be classified according to its deployment procedure. The proposed solution follows the same logic and it is divided into two types of deployment that aggregates honeypot's and honeynet's type of logical deployments, placement and purpose depicted on [20] and [86]. On this basis, the proposed system's architecture varies on how the deception grid is deployed.

Firstly, the base architecture for this system embraces a research perspective that enables the manual and premeditated grid deployment whose particularities follow a certain attack information inputted by the security team to which it intends to defend against, addressed to as "Research Deployment". Alternatively, an extension to the base architecture is presented that supports a production environment where the deception grid is deployed on-demand when an attack is detected. In this case, detection and attack classification mechanisms send the attack information to the system, shaping the deception grid's topology, referred to as "Production Deployment".

It is crucial to emphasize that the two aforementioned approaches are not mutually exclusive, as it is highly beneficial and recommended for an organization to utilize them simultaneously in a complementary manner in their cyber defensive efforts.

### **3.1.1. Base Architecture (Research Deployment)**

Regarding the base architecture of the proposed system, the deployment of the deception grid is carried out manually by the security team, not depending on detection mechanisms. By eliminating cyber-attack detection requirements, this type of deployment avoids the dependency of typical signature-based detection appliances, attempting to not only act proactively upon a cyber security incident but also detect. A thorough study of the organization's infrastructure needs to take place beforehand, to then, strategically determine where should the deception grid(s) be deployed, consequently optimizing the system's deceptive efficiency. To achieve these objectives, it is proposed that, in this type of deployment, the deception grid can be allocated in two different locations regarding its placement on the IT infrastructure.

The deception grid can be deployed outside the business network, i.e., directly facing the Internet, sharing, preferably, the same public IP address subnet range (e.g., utilize unused IP addresses of the organization). In this case, the deception grid aims to gather valuable investigative and research information on possible adversaries, focusing on the collection of



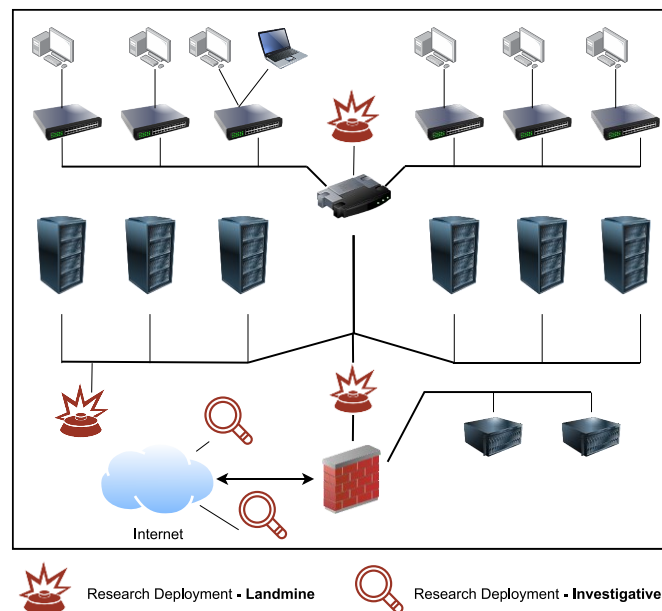
## Proactive Cybersecurity tailoring through deception techniques

statistical information that can be useful to the organization's threat intelligence efforts, referred to as "Investigative".

Research deployments can still happen within the business network boundaries placing the grid amongst and connected with infrastructural and network devices of the organization, such as servers and/or routers. In this context, the deployed deception grid acts as an alert, for the security team, and a trap, for the adversary, designated by "Landmine".

The base architecture's applicability of the proposed system is represented on *Figure 23* and its main purposes are:

- Defend the IT infrastructure with no use of reactive detection mechanisms.
- Zero-day vulnerability discovery-driven (technical and operational).
- Adversary identification (for Threat Intelligence purposes).
- Statistical and investigative information collection.



**Figure 23** - System's base architecture applicability.

Despite having two different conceptual purposes, both "Investigative" and "Landmine" deception grids are technically the same (being only distinguished by their placement) and all interactions with the grid are found suspicious and should be investigated by the security team. In the case where the attack suspicion is found to be pertinent, the security team should observe the attacker within the grid. If the suspicion of malicious intent is cleared

by the security team, the actor should be redirected to the real infrastructure. The two types of deception grid placement regarding research deployment placement will now be depicted.

### **Facing the Internet (“Investigative”)**

Across an organization’s cyber deceptive strategy, possessing deception grids that are deployed facing directly Internet can pose as a waste of resources at first glance. However, this work proposes that an organization and its security or intelligence teams should deploy one or more of its deception grids outside of their IT infrastructure and in direct contact with the Internet, which can bring immense value to the organization’s cyber defensive efforts. Statistical information can be processed from both data gathered from the observed attacker’s TTPs and IoCs (such as detected IP addresses that engage with the grid) enabling the discovering of valuable insights into the latest cyber-attack trends within the cyber space and identify possible adversaries that seek to attack the organization in the future. This information should then be translated to intelligence, as seen on Section 2.7. By actively collecting data on emerging threats and vulnerabilities, the organization can proactively enhance their cybersecurity posture, fortifying their network and systems against future attacks.

This type of deceptive deployment placement is mainly categorized as “research” and has been proven its worth in many works regarding collection of statistical information of honeypots. One example of this is present on [73], where a high-interaction honeypot is set-up and deployed to then gather vital quantitative information. The present work seizes the same idea but applying it to a deception grid context and not to the small granularity of a honeypot, increasing the level of the deceptive sophistication, its success rate and subsequently the collection of better-quality attacker information.

As stated before, public IP assignment and network management of the deception grid is of extreme importance since it is what enables the attacker to discover and interact with the deception grid. As discussed previously, sharing the same public IP address subnet range by utilizing unused IP addresses of the organization creates the perception that the deception grid(s) are actually part of the IT infrastructure of the organization. In a case where the attacker’s intentions are to strike a specific organization, this technique, visible on *Figure 24*, stirs the attacker to interact with the deceptive grid(s). In practice, the referred set of IP addresses should lead the attacker to the infrastructural devices (servers) where the deception grid(s) are deployed, running and ready for attacker interaction and handling.

## Proactive Cybersecurity tailoring through deception techniques

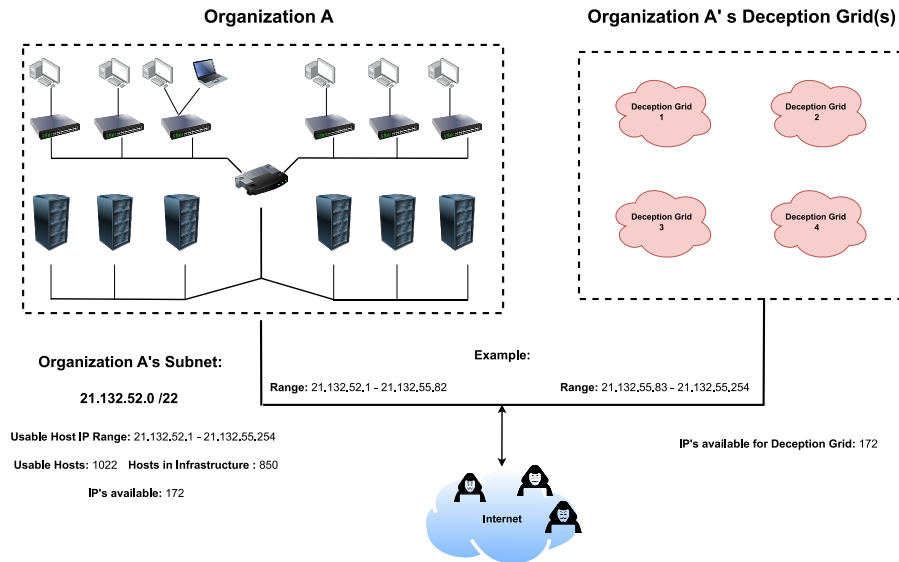


Figure 24 - Example of deception Grid(s) public IP address management.

“Investigative” placement of the deception grid(s) can ultimately be seen as the first line of defense for an organization, prompting the attacker with a grid of misleading artifacts for interaction before even reaching infrastructural boundaries, proactively defending against attackers.

### Within infrastructure boundaries (“Landmine”)

To complement external frontline defense of the “Investigative” placement, this work proposes that the cyber defensive efforts of an organization adopt, supplementarily, the deployment of deception grids as “landmines” inside infrastructural boundaries, strategically scattering them throughout the organization’s IT infrastructure. Placing deception grid(s) across a business network should not occur randomly or in an unconscious fashion. It is essential to conduct a comprehensive examination of the organization's infrastructure in order to identify the optimal locations for deception grid(s) implantation, which involves the consideration of the following key aspects:

- Network architecture that supports the organization.
- Systems and applications running on the IT infrastructure such as operating systems, web servers, etc..
- Data flows, referring to how data is transmitted, processed, and shared within business network.

### Proactive Cybersecurity tailoring through deception techniques

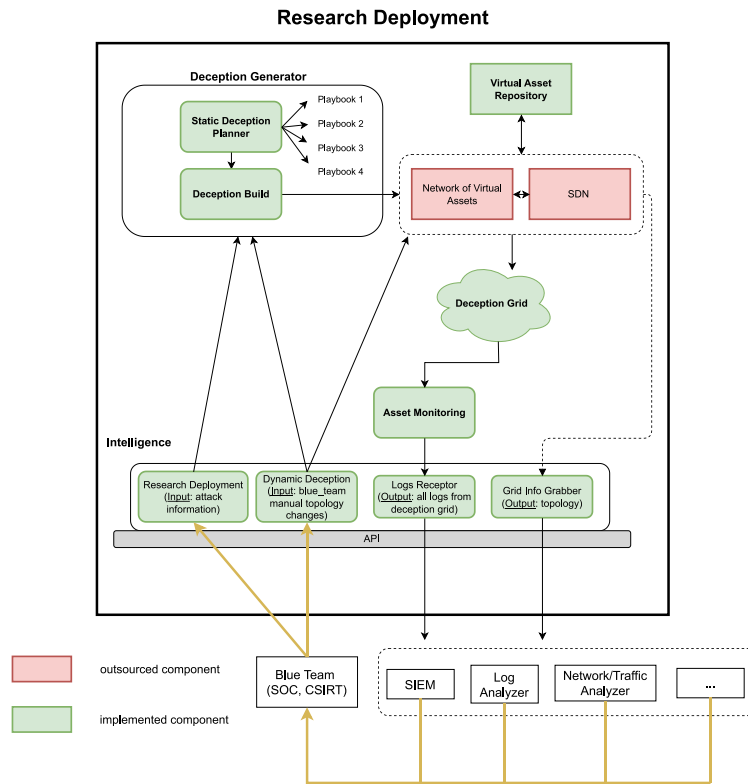
- Storage devices and appliances where critical organizational data is allocated.
- Security controls and its possible weaknesses, where a deception grid can help mitigate such gaps.
- Business processes, workflows, and operations.

Acknowledging that the selection of the location for the set of deployed deception grids directly impacts their fruitfulness and efficiency is of paramount importance. This should be taken into consideration when developing the cyber deception strategy, where several options must be pondered. Deploying a deception grid closer to the DMZ will allow a premature entrapment of a possible perpetrator and consequent alert to the security team. Thus, defending against a possible attack sooner in the cyber kill chain, taking less risks of compromising the infrastructure. On the other hand, in case that a deception grid is positioned closer to real assets, and given a positive identification of attacker interaction with the grid, it conveys a more precise idea on what the attacker was aiming to strike. Ultimately, the deceptive strategy of using “landmines” and its intrinsic decisions rely on a balance between risk of and reward. Some works that address honeypot selection and placement for increasing network defense effectiveness resort to game theory, such as [108] and [109], and should be conferred for deception grid strategic placement purposes.

It is worth noting that human awareness management may be advised, which can play an important role in the organization’s cyber defensive and deceptive efforts. This entails that not many people inside the organization should know about the existence and placement of the deception grids, being restricted to the security team or even a part of it. Firstly, these systems are not part of “normal” and business operations, discarding the necessity of its acknowledgement, e.g., by the software development team or others. Secondly, in a scenario where an employee is hacked and information is stolen from his workstation, the threat actors can collect sensitive information about the location of deception grids and in a later cyber attack can successfully avoid the deceptive appliances in-place and accomplish their malicious intents with more impact. By reducing the number of employees/collaborators who are aware of these cyber deceptive defensive mechanisms, that scenario is mitigated. Ultimately, the cluelessness of having these defensive mechanisms in place, enables in the worst-case scenario, the possibility of catching inside actors that want to compromise the organization’s infrastructure.

## Proactive Cybersecurity tailoring through deception techniques

With these considerations in mind, a conceptual software architecture was designed that allows to achieve excellent deceptive capabilities by segregating clear responsibilities amongst the presented components. The system's base architecture that supports research deployment is presented in *Figure 25*, where several components and its interactions are displayed in a comprehensive way.



**Figure 25** - Active Deception System base architecture.

This architecture is composed of seven main logical components: Deception Generator; SDN; Virtual Asset Repository; Network of Virtual Assets; Intelligence; Asset Monitoring and the Deception Grid. Components marked green are implemented by the authors while red-marked components are external and need integration to be used by the system.

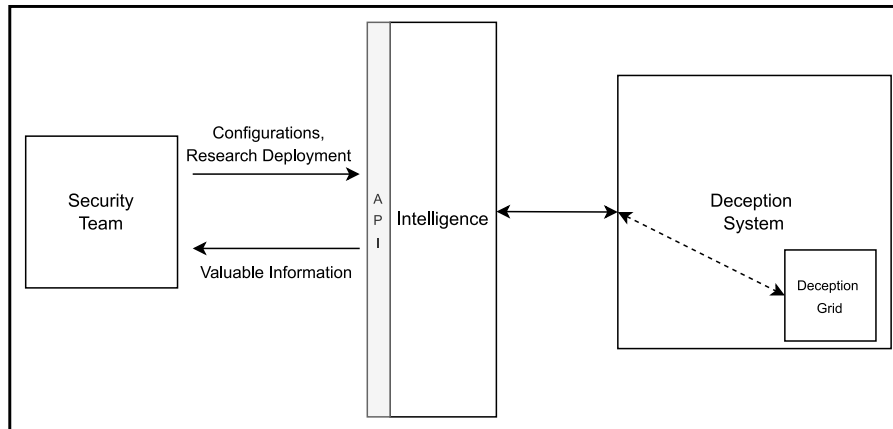
Components and respective modules will now be depicted, determining each one's responsibility and conceptual functionality within the proposed deceptive system.

### a. Intelligence

The Intelligence component is responsible for enabling communication to and from the deception grid(s) and system, acting as a gateway. The security team is able to communicate

## Proactive Cybersecurity tailoring through deception techniques

with the deception system through this component, enabling the research deployment (manual), making configuration adjustments to the grid and retrieving possible valuable information, as presented on *Figure 26*.



**Figure 26** - Intelligence component as a gateway.

As visible on *Figure 26*, Intelligence provides an API that establishes a set of rules, protocols and operations enabling external software applications (e.g., security team's frontend client) to communicate with it and with the deceptive system. Intelligence's API needs to provide the deceptive system's administrators with a considerable collection of operations for the security team to extract the most out of the system.

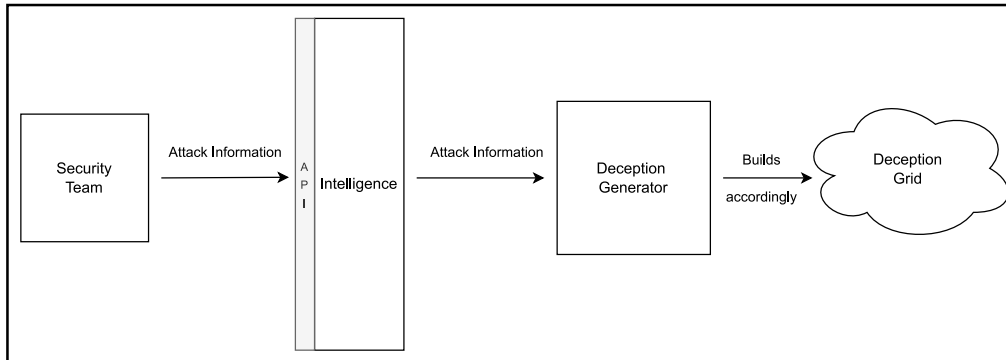
Thus, the Intelligence component should support the following base operations:

- `deployDeceptionGrid(attackInformation)`
- `getCurrentGridsTopologies()`
- `getAssetsLogs()`
- `addAsset(asset)`
- `removeAsset(asset)`
- `getRegisteredDeployments()` and `getRegisteredAttacks()`

Manual deployment of the deception grid (research deployment) occurs via Intelligence's API, specifically through its Research Deployment module represented on *Figure 25*. The security team is able to manually deploy a deception grid tailored by the attack information passed. The attack information refers to a set of cyber-attack parameters to what

## Proactive Cybersecurity tailoring through deception techniques

the security teams intends to defend against. These parameters are then forwarded to the Deception Generator component that, by specific mechanisms explained below, build the deception grid according to the received attack information. This process is portrayed in a simple manner on *Figure 27*.



**Figure 27** - Overview of research deployment via Intelligence's API.

This process allows the security team to premeditatively deploy tailored deception grids to defend against certain cyber-attacks and threat actors inside or outside infrastructural boundaries i.e., “Investigative” or “Landmine” types of grid placement.

Continuous deception grid logging processes are in place via the Asset Monitoring component. For the security team to receive those logs, it is proposed that the Intelligence component should provide the incoming deception grid logs via its API and/or directly dispatch them to other appliance administrated by the security team, such as a SIEM. This is represented in *Figure 25* through Intelligence’s Logs Receptor module.

Intelligence component needs to be equipped with operations that perform real-time adjustments to an already deployed deception grid, enabling a more refined use of deception through dynamic modification (Dynamic Deception module), in this case, mainly by adding or removing assets of the grid. The security team’s need of dynamic alteration of topology can be urged by the analysis of the grid’s logs received through Logs Receptor module, meaning that the previous automated deployed deception plan needs tweaks to increase its effectiveness enabling the last step on the deceptive process, depicted on *Figure 20* and *Figure 21*, specifically its adjustment possibility.

Finally, some pertinent information regarding the deception grid must be made available to the security team. The Intelligence component preserves information about research deployment requests, registered detected attacks that triggered the deployment the

deception grid (production deployment, depicted in Section 3.1.2) and can retrieve the topologies of the deployed deception grid(s). These informations can be utmost useful for examining and determining the current state of the system and its deception grid(s).

### b. Deception Generator

The Deception Generator component applies a planned deception strategy in a given security incident context. This component is “activated” under two circumstances: in case of research deployment (i.e., manual deployment of a deception grid) and when an attack is detected and the grid must be deployed on-demand (i.e., production deployment, discussed in Section 3.1.2). In both cases, attack information is supplied to this component, which immediately selects what is the appropriate deception plan for that cyber-attack context.

A deception plan consists of pre-planned set of virtual assets connected in a certain topology, desired traffic policies and asset configurations that the security team founds to be pertinent for the defense against a given cyber-attack, typified by certain parameters. The Deception Generator component is subdivided in two modules: Static Deception Planner module chooses a plan from a set of predefined deceptive plans for several possible cyber-attacks and Deception Build materializes that plan, building the network of deceptive artifacts (deception grid). An overview of this process is visible on *Figure 28*.

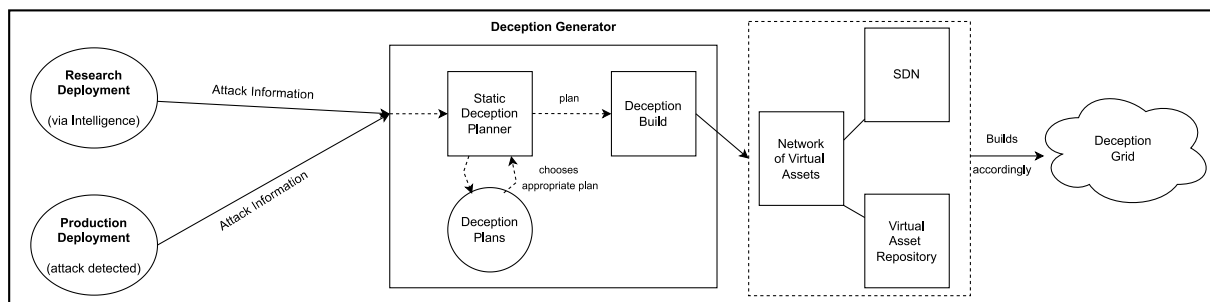


Figure 28 - Deception Generator's responsibility within the deceptive system.

Deception Build interacts with the triad of Network of Virtual Assets, SDN, and Virtual Asset Repository components in order to materialize a deceptive plan into a deployed deception grid (specifically through the Network of Virtual Assets component). These components will be outlined later.

Ideally, the set deceptive of plans should englobe all the techniques discussed in Section 2.2 and should follow the deception planning criteria and procedures presented in Section 2.7,



taking into account not only technical cyber-attack considerations but also the possible responsible threat actor. Furthermore, these plans should be crafted following some framework or guidelines of adversary engagement that optimizes the plan’s efficiency. However, the only encountered was MITRE’s engagement and deception process framework [110] that presented an undesired degree of specificity, limited to superficially mapping MITRE attack techniques with a restricted set of actions, only optimal for higher level cyber decisions and thus not being adequate for the present work’s context. Thus, it is suggested that deceptive planning needs to be manually conducted by the security and intelligence teams.

Additionally, the specific mechanism that selects the most appropriate deceptive plan regarding the received attack information can benefit of the use of machine learning or artificial intelligence techniques to automatize and perfect deception strategy selection for any attack context. Autonomous planning and orchestration of deception is what this component ideally should implement. An example of a system with the referred objective is theorized in CHIMERA [96].

### c. Virtual Asset Repository

Virtual Asset Repository consists of every virtual artifact that can be used to build the deceptive grid, i.e., that are present in at least one of the pre-planned deception strategies. These virtual resources can be materialized as virtual machines and/or containers and are mainly virtual hosts (e.g., used as honeypots or other type of deceptive auxiliary artifacts) or network devices. (e.g., OpenFlow switches). In a scenario where deception plan “A” is chosen, the Virtual Asset Repository should provide the Network of Virtual Assets with the necessary virtual assets to satisfy that deceptive plan, as portrayed in *Figure 29*.

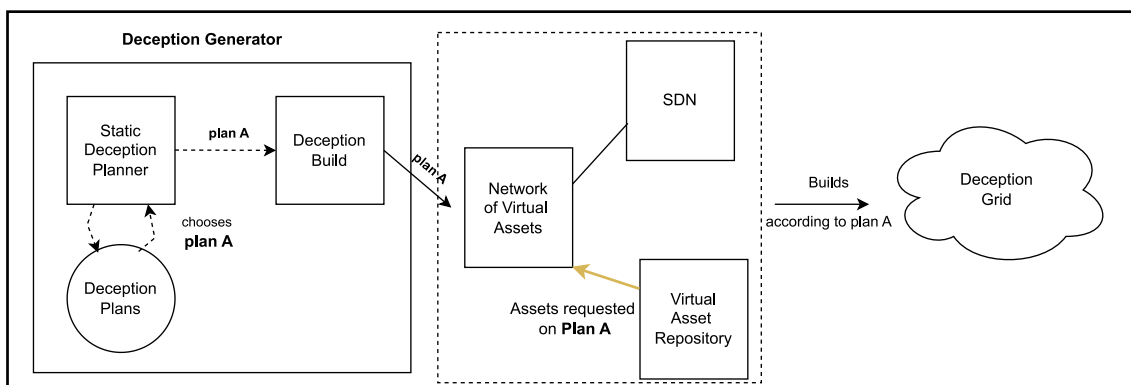


Figure 29 - Asset provisioning through Virtual Asset Repository.

## Proactive Cybersecurity tailoring through deception techniques

As introduced in Section 2.4.1, virtualization/containerization allows for increased flexibility and agility. In mutable deceptive environments where changes can occur frequently, virtualized assets can be easily created, modified, or deleted as needed. This enables organizations to quickly respond to evolving needs and adapt their infrastructure accordingly, as the one demanded in this scenario where the deception grids need be deployed in a timely fashion (specially on production type of deployment) and where dynamic allocation and deallocation (dynamic deception) of resources is enabled. Resource optimization and cost efficiency are also promoted by adopting virtualization or containerization. By running multiple virtual machines or containers on a single physical server, organizations can make more efficient use of their hardware resources when developing their cyber defensive and deception strategy. This reduces the need for additional physical assets, resulting in lower operational costs. One more advantage relates to the easier *postmortem* forensics of an asset in a scenario of full compromising.

### d. Network of Virtual Assets

Succeeding virtual asset provisioning by the Virtual Asset Repository, a network of these artifacts must be assembled. Network of Virtual Assets was inserted into the proposed system's architecture to fulfill that need. Upon receiving the selected deception plan to execute (from Deception Generator) and asserting that the virtual assets consisting of the plan are available for use, this component connects assigned deceptive assets in a specific topology, consistent with the deceptive plan and defines traffic policies of the network. These procedures should be done in an automated and programmatic manner to rapidly and preemptively construct the deceptive network of virtual assets.

### e. SDN

SDN techniques play a vital role in modern network management and infrastructure, as discussed on Section 2.4.2. In this scenario, the proposed deceptive system seizes the decoupling the control plane from the data plane for two main reasons: to centralize network management and control of the network(s) of the deceptive artifacts (deception grids) that ultimately allows to dynamically allocate and deallocate resources, and its provisioning of network programmability environment, facilitating automation of processes that might be vital to successfully deceive an attacker. SDN component is thus inserted into the conceptual system

architecture as a unit of grid orchestration and control of the final result of the interaction between the triad of components: Network of Virtual Assets, SDN, and Virtual Asset Repository which is the Deception Grid.

#### **f. Deception Grid**

The Deception Grid component represents the culmination of the mechanisms of the proposed deceptive system that, upon receiving attack information either from cyber-attack detection mechanisms or directly from the security team and choosing the appropriate deceptive plan, intentionally deploys misleading artifacts to construct an infrastructure that allows real-time investigation of an attacker's patterns and approaches allowing to proactively defend the real IT infrastructure. The Deception Grid consists of all selected virtual assets, topology, traffic policies and asset configurations present in the chosen deceptive plan.

Additionally, and as discussed on Section 2.5, honeynets' implementations introduced the concept of "honeywall" which mitigates the risk of the attacker using the deceptive network to attack other systems. The proposed system must attribute an element that has the same responsibilities to the deployed deception grid. The referred "honeywall" and its configuration needs to be specific on each deception plan, i.e., be properly configured in each deceptive plan to increase the effectiveness of a plan and allowing a more flexible and tailored deception.

Alongside adjusting traffic policies within the grid, generating "fake" traffic between the grid's assets and also traffic to outside the grid (e.g., to a cloud provider) creates an authenticity perception to the attacker regarding the infrastructure that he is attacking, leading to a higher probability of deceptive success. Traffic particularities should be assigned by each deception plan and adapted for each organization, for instance, an insurance company's traffic patterns differ from those of a streaming service organization.

Ideally, a deception grid should consist of the following assigned elements by the selected deception plan:

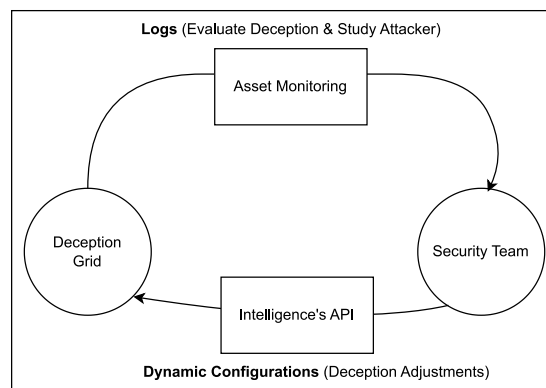
- Set of virtual assets.
- A network topology.
- Specific asset configurations and log shipment mechanisms.
- Traffic policies and management.
- Honeywall with specific rules.
- Fake traffic generation.

**g. Asset Monitoring**

Once the deception grid is in place (resultant of a specific deception plan), mechanisms that enable real-time visualization of grid-related information need to be implemented in order to provide logs. The process of logging the deployed deception grid allows two major objectives: to analyze the attacker’s TTPs and to evaluate the effectiveness of the deployed deceptive plan. For this purpose, resources within the grid must be configured to report continuous logs to the Asset Monitoring component. This component aggregates all asset and network logs (from the deception grid), parses them if necessary, and sends them to the Intelligence component that subsequently exposes them to the security team.

The fact that Asset Monitoring consists of a separate component is an architectural decision that is based on both a modular responsibility model and security concerns. In the event that the deception grid is fingerprinted, compromised and subsequently uncovered by an attacker, the perpetrator possesses the capability to expose the receptor responsible for logging information and may attempt to take it down. By adding Asset Monitoring as redundancy and by proxying log dispatchment, the referred threat can be mitigated by never directly revealing the location of the Intelligence component.

This process enables the eventual tracking of the attacker’s footprints within the deception grid allowing a defender to self-evaluate the effectiveness of its deception plan, as made pertinent on Section 2.1 and on *Figure 20* and *Figure 21*, specifically its deception monitoring possibility. As a consequence of log examination, a change or adjustment of the previously executed deception plan may be adequate, enabled through Intelligence’s API allowing adjustability to the deceptive process, creating an operational loop. An overview of the importance of this component is portrayed on *Figure 30*.

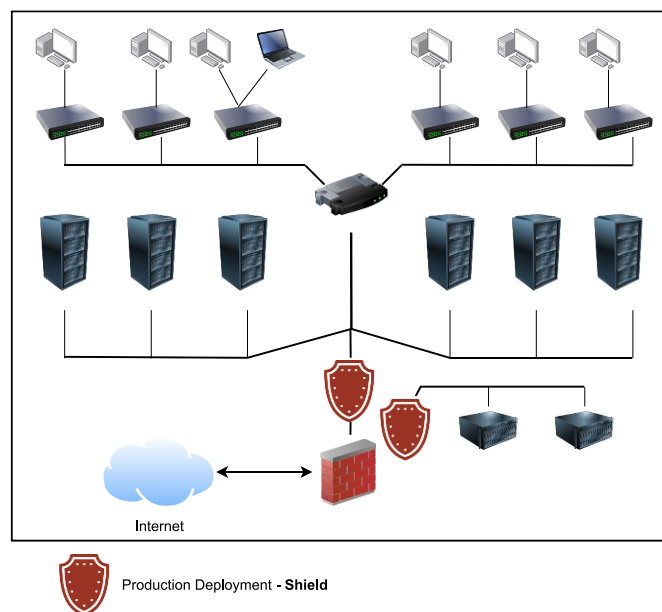


**Figure 30** - The role of Asset Monitoring component within the deceptive process.

### 3.1.2. Extended Architecture (Production Deployment)

Production deployment and the underlying extended architecture of the proposed system, supports on-demand deception deployment within business network, i.e., the grid's deployment occurs when an attack is detected, contrastingly to the research type of manual implantation (Section 3.1.1). Upon the identification of a cyber-attack launched against the organization's technological infrastructure, the deceptive system is alerted to deploy the deception grid whose resources and topology are modeled in conformity with the ongoing attack information received from detection and classification mechanisms. Therefore, enabling defenders to act and defend proactively against a security incident by providing a deception grid molded by the ongoing attack. The deception grid in this scenario acts as a "Shield" and must be swiftly prompted to the attacker to interact with it. Thus, deviating him away from the organization's real infrastructure and make him susceptible to behavior tailoring by the security team (deceiver). The extended architecture's applicability of the proposed system is represented on *Figure 31* and its main purposes are:

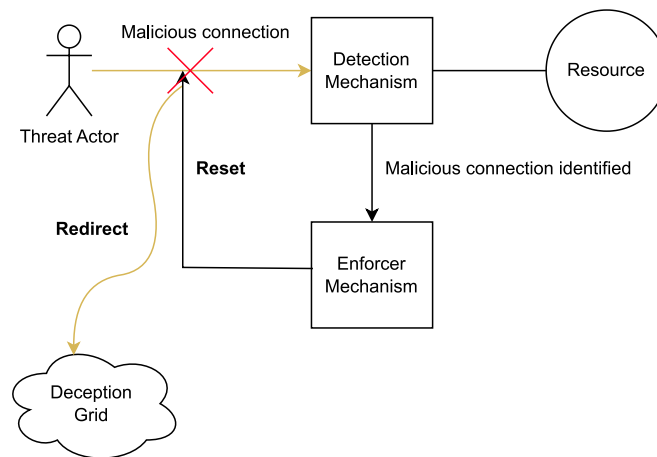
- Proactively defend the IT infrastructure against ongoing cyber-attacks with the assistance of attack detection and classification mechanisms.
- Study the attacker's TTP and intentions.



**Figure 31** - System's extended architecture applicability.

## Proactive Cybersecurity tailoring through deception techniques

Defending the business network in this type of deployment means that the deception grid must be quickly assembled and needs to be positioned directly facing the attacker, enabling his interaction with the deception grid instead of the organization's infrastructure. For that purpose, enforcing mechanisms need to additionally exist that redirect the identified malicious connection of the attacker to the deception grid, allowing the perpetrator to access it and be manipulated and tailored by the security team. Detection and enforcing mechanisms are typically found on IPS implementations. This process is illustrated in *Figure 32*.

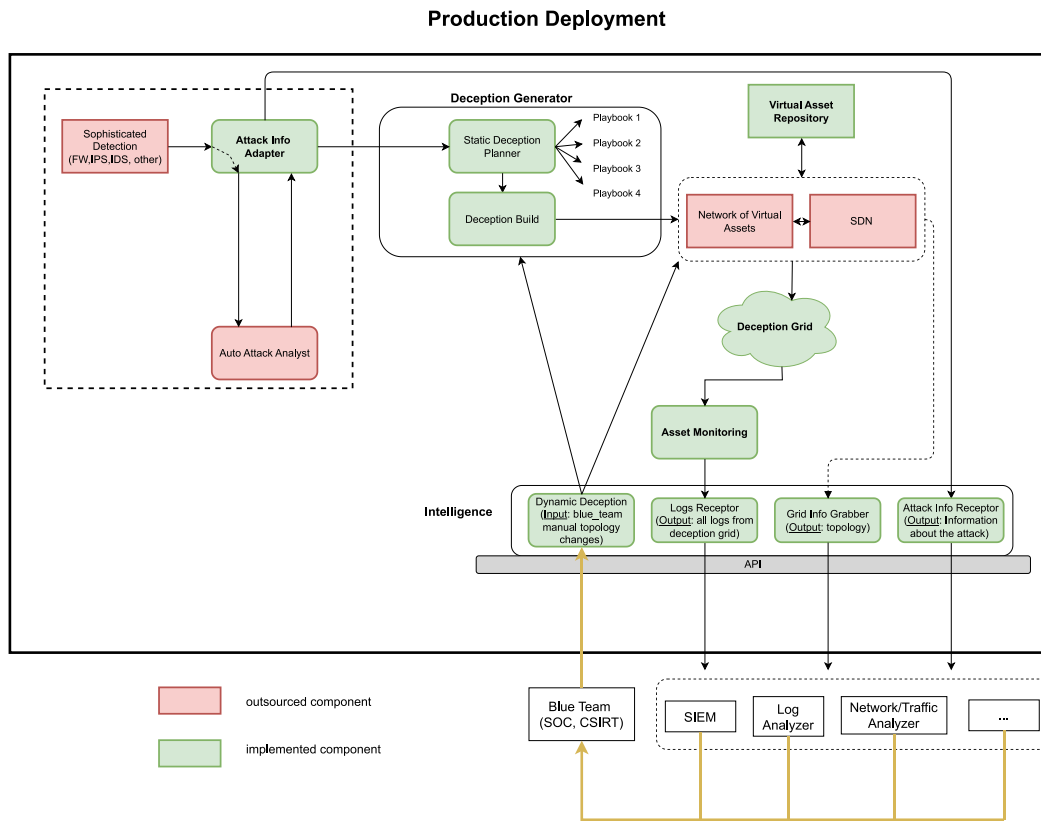


**Figure 32** - Detection and enforcing mechanisms prompting the attacker to the deception grid.

Following presumed threat actor's redirection to the deception grid and in the event of the security team recognizing that the referred deflection of the adversary occurred due to a "false positive" identification by the detection mechanism, the actor (without malicious intent) must be redirected to the real infrastructure.

The architecture that supports the production deployment extends the base architecture and is presented on *Figure 33*. Three components were added in comparison to the base architecture: Sophisticated Detection, Auto Attack Analyst and Attack Info Adapter. The remainder of components were already depicted (Section 3.1.1) and maintain the same responsibilities and functionalities.

## Proactive Cybersecurity tailoring through deception techniques



**Figure 33 - Active Deception System extended architecture.**

### h. Sophisticated Detection and Auto Attack Analyst

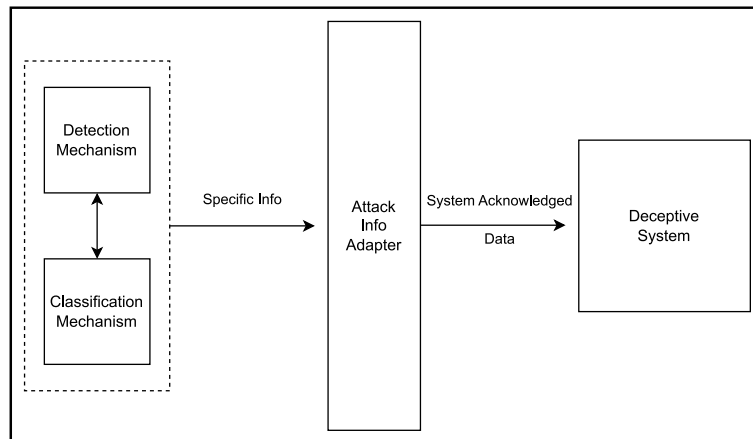
Sophisticated Detection refers to a tool that is running within the organization infrastructure that swiftly detects that a cyber-attack is probable to be occurring. This appliance must effectively detect APT attacks and be preferably distributed, to cover all the organization's infrastructure. In order to build the deception grid with a proper deceptive plan, the system must first know what type of attack is occurring. For that purpose, the integration of a system that automatically classifies a cyber-attack is utmost useful, referred to as Auto Attack Analyst. Operational exclusivity of these components can either only target cyber deception processes or be shared across the organization's infrastructure for standard security procedures. These two tools are represented as separate components, but they can symbolize the same appliance.

DISTDET [111] is a cost-effective detection system that detects and classifies APT attacks through distributed computing and provenance graphs. The integration of this system, or other similar systems, would enable the detection and classification of possible attacks and would bring immense value to the proposed deceptive system when in production deployment.

Alternatively, signature-based detection appliances such as typical IDS or IPS are also a viable option. It is duly noted that an IPS appliance, as discussed previously, can be a benefic choice since it typically possesses enforcing mechanisms, discarding the need of integrating other external component for enforcing purposes.

**i. Attack Info Adapter**

Attack Info Adapter acts as a broker between the attack detection and classification external components and the deceptive system, by adapting or parsing the possible proprietary and specific information (if needed). When an attack is positively detected, and classification processes are complete, information about the attack must be sent to Attack Info Adapter. If necessary, this component processes received information from the external components into system-acknowledged data, as represented on *Figure 34*, freeing the proposed system from the dependencies of each integrated tool’s particularities. Thus, abstracting and decoupling the rest of the system from external component’s attack typification specifications.



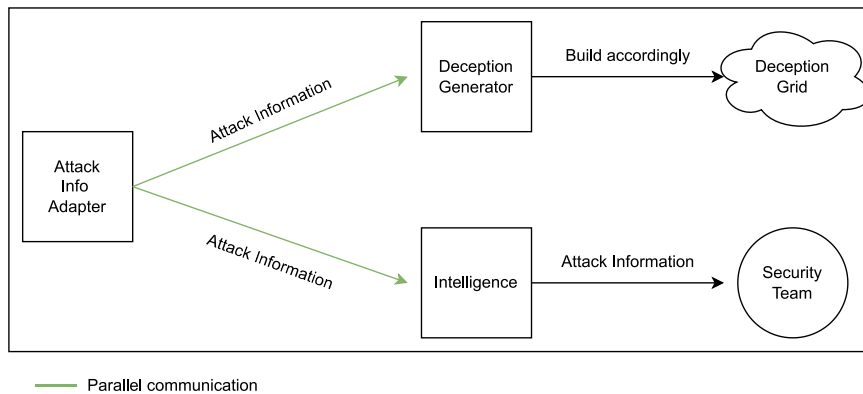
**Figure 34** - Attack Info Adapter responsibility as a broker.

Parsed attack information is then dispatched simultaneously (in parallel) to both Intelligence and Deception Generator components, allowing the security team (through Intelligence’s API) to visualize attack information whilst the deception grid is being deployed, as portrayed in *Figure 35*. Synchronous supply of this information is found pertinent due the security team’s urgency to be acquainted on going attack parameters, not only for notification purposes but also to discern if the in-progress deployment of the deception grid



## Proactive Cybersecurity tailoring through deception techniques

will need immediate tweaks when implanted, due to the fact that the security team knows anticipatorily what is the resultant deception grid for those attack parameters.



**Figure 35** - Parallel dispatchment of attack information through Attack Info Adapter.

In this type of deployment, the deception grid is assembled because Deception Generator component receives the attack information required from Attack Info Adapter, contrary to research deployment where such information is received directly from the security team's input.

On both types of deployment, the security team must consider when to eliminate certain deception grid(s) to, for example, optimize resources. Mechanisms such as timeouts or manual deallocations of the grid should be in place. Each deployed deception grid (via research or production deployment) should ultimately be treated differently, however, a balance between automated and manual deallocation mechanisms should be implemented to effectively dismantle deployed grids.

The proposed architecture and its variants aim to cover all the limitations of honeypots, honeynets and deception grids discussed throughout Chapter 2 and re-addressed in this section. The proposed architecture for a deceptive system follows a modular approach that separates and delegates clear responsibilities within the system, peaking defensive deceptive capabilities for organizations within computer security technology landscape. Conceptual mechanisms and important aspects were also presented with the objective of being considered in implementation phases, culminating in a complete and comprehensive deceptive system. Premeditative and tailored levels of deception sophistication can eventually be achieved in cyber defensive efforts.

## 3.2. Proposed System and Deception Techniques

The proposed system leverages various mechanisms that can ultimately help peak deceptive capabilities for an organization's cyber defensive efforts. In order to apply deception in a versatile manner, the deceptive system contains mechanisms that can implicitly and/or explicitly use deception techniques within each taxonomy introduced on Section 2.2.

Firstly, as discussed on the previous section (Section 3.1), the proposed system's Deception Generator component selects the most appropriate deception plan for a given context (in this scope, depending on received attack information). The set of available deception plans is of the security team's responsibility (deceptive system administrators), however, they should ideally comprise all deception techniques, to successfully defend against any attack. Even for the same attack, different deceptive defensive actions can theoretically be performed and should be taken into account on every deception plan. Thus, the set of all deception plans available to be selected by the Deception Generator component can and should embrace all deception techniques discussed on Section 2.2.

Apart from each plan's deceptive techniques, some system mechanisms and desired functionalities can be mapped into certain deception techniques. Regarding "Deception by Method" taxonomy, and by analyzing the definitions of the deception techniques on Section 2.2.1, depicted strategies can be mapped to some examples found within the system's intrinsic mechanisms and/or possible procedures, represented on *Table 4*.

"Deception by Commission-Omission" (addressed on Section 2.2.2) typifies deception by the level of involvement of the deceiver. Commission dictates that the deceiver takes active part of the deceived party's beliefs, decision-making processes, and propositions. Omission implies that no active contribution from the deceiver is performed, allowing the adversary to continue with his unaffected beliefs on the proposition. This taxonomy suggests that deceiver's actions or inactions, while the deception is occurring, are important to successfully mislead the attacker and can be transposed to the system's capability to allow the security team to dynamically modify the deception grid, thus, actively contributing for the target to acquire or continue to believe in the deceptive proposition or to continue or stop believing in the negation of the proposition. The security team must determine the appropriate moments to take action and decide whether and how to intervene in the existing deception plan (commission) or allow their adversary to continue making decisions within the established deception grid (omission).

## Proactive Cybersecurity tailoring through deception techniques

**Table 4 - Deceptive system and examples of deception.**

Plane	Technique	Sub-Technique	Examples
Active Deception	Displays	Decoys	Virtual Assets (honeypots).
Active Deception	Displays	Portrayals	Network topology and underlying software and hardware.
Active Deception	Feints	N/A	Fake traffic generation, dynamic configuration of deception grid.
Active Deception	Demonstrations	N/A	Deception plan to stall or deflect adversary.
Active Deception	Disinformation	N/A	Honeytokens (credentials, database data).
Cover	Camouflage	Hiding	Concealment of deceptive system (except deception grid).
Cover	Camouflage	Blending	Blending deception grid logs as “normal” traffic, avoiding fingerprinting.
Cover	Camouflage	Disguising	Research/Production deployment, preemptively disguised as real infrastructure.
Cover	Camouflage	Securing	Reducing indicators that compromise deception grid as a misleading artifact.
Cover	Denial	Counterreconnaissance	The use of tunneling mechanisms in system interactions to prevent sniffing.
Cover	Denial	Jamming	Adversary isolation in the deception grid through Honeywall policies.
Cover	Denial	Counterintelligence	N/A
Cover	Denial	Physical/Virtual destruction	Dynamic removal of deception grid assets.

Sophistication-based deception taxonomy (presented on Section 2.2.3) classifies deception according to the degree to which it incorporates the variables of a situation, being ultimately categorized as static, dynamic, adaptive, or premeditative. In Section 2.7 deception procedures and its correlation with Cyber Threat Intelligence was discussed, leading to the acknowledgement that by knowing possible adversaries and its characteristics, the highest degree of deception sophistication can be reached. The proposed system and particularly the resulting deception grid(s) can be placed within the adaptive sophistication when not informed of what threat actor is the security team defending against. However, in the case of adversary identification (e.g., “APT42”), premeditative levels of deception sophistication are reached as *“Premeditative deceptions are designed and implemented based on experience, (..) and, moreover, observations about the target’s sensors and search strategies”* [18]. Thus, allowing the security team to craft more precise and advanced deceptive scenarios for specific adversaries.

# 4. System Implementation

The present chapter showcases the proposed solution's implementation and has the following organization: Section 4.1 details every component of the deceptive system, exposing each's functionalities and disclosing corresponding proof-of-concept implementation, alongside with associated supporting mechanisms. Section 4.2 depicts how deceptive plans were elaborated and implemented. Section 4.3 presents a threat model of the proposed system and discusses possible mitigating countermeasures and improvements that although being out-of-scope for this work's implementation are duly noted and outlined.

## 4.1. System Components

The implementation of this system and its supporting components takes advantage of a number of existing technologies and resources, that combined with the specific development of each depicted component of the modular system architecture, presented on Section 3.1, compose the implementation of the proposed system. Conceptual mechanisms, depicted on the referred section, regarding each component functionalities and system-wide supporting mechanisms were taken into consideration when addressing this implementation, however, not all of them were implemented.

The development of the deceptive system's components was supported by Python programming language [112], thus, each component or module is materialized in a Python script. The selection of this language relied on its native Linux integration alongside its package management, its integration capabilities with systems developed in other languages and/or technologies (some pertinent for this work) and its rapid prototyping. More precisely, Python versions 3.9.12 and 3.6.9 were utilized to develop and execute the referred scripts that represent each component/module of the proposed deceptive system's architecture.

With the objective of enabling communication between components (in this scope, Python scripts), required for establishing inter-script connection, Python's `sockets` library was used. This low-level networking interface is a "*straightforward transliteration of the Unix system call and library interface to Python's object-oriented style*" [113], which capacitates the components with some network protocols such as TCP or UDP, in order to facilitate seamless communication and data exchange between different components/modules. Empowered by the utilization of socket communication, the system can become of distributed nature, eliminating the need for the components to reside in the same physical machine.

The development environment chosen for this proof-of-concept implementation was an Ubuntu 16.04 virtual machine where all scripts were built and executed, alongside supporting technologies that materialized the proposed conceptual components. This Linux-based development platform was chosen due to its integration with required tools for the system's implementation, namely SDN and virtual network technologies, alongside their smooth execution and straightforward configuration.

An overview of the proposed system's implementation is illustrated on *Figure 36*, displaying how each component/module is materialized, their core functionalities and communicational responsibilities.



## Proactive Cybersecurity tailoring through deception techniques

- **AttackType** – “Top-level” cyber-attack identifier.
- **AttackerIP** – IP address that can have two meanings but the same purpose. If network mask not provided, the default value is “/24”:
  - a) When in production deployment, this parameter refers to the detected malicious IP from the identified connection that can be used conveniently in a deception plan to dynamically assign IP addresses to the virtual assets.
  - b) When in research deployment (where there is not an ongoing attack detected), this parameter is the base IP address from where virtual assets dynamically get their IP address.
- **AttackDetails** [Optional] – Specific cyber-attack characteristics, ideally a sub-category of “AttackType”.
- **ThreatActor** [Optional] – Identifier of a malicious actor (individual or organization) from public sources, namely from [114], or private.
- **CVE** [Optional] – Public reference identifier of a weakness in the computational logic found in software and hardware components [115].

This implementation does not allow arbitrary combinations of these parameters due to the fact that it is assumed that, except “AttackType” and “AttackerIP”, the rest of the parameters follow an ascending order of specificity meaning that if, for example, “CVE” is provided, then all the other optional parameters are also present. An important aspect to consider is that the number of supplied attack parameters is directly related to the cyber deceptive success. As more parameters are supplied, the degree of specificity on attack identification is higher, leading to a more probable success of deception. The possible combinations are listed below:

<i>Attack_Info</i> = ( <i>AttackType</i> , <i>AttackerIp</i> )	-
<i>Attack_Info</i> = ( <i>AttackType</i> , <i>AttackerIp</i> , <i>AttackDetails</i> )	Deceptive
<i>Attack_Info</i> = ( <i>AttackType</i> , <i>AttackerIp</i> , <i>AttackDetails</i> , <i>ThreatActor</i> )	accuracy
<i>Attack_Info</i> = ( <i>AttackType</i> , <i>AttackerIp</i> , <i>AttackDetails</i> , <i>ThreatActor</i> , <i>CVE</i> )	+
	↓

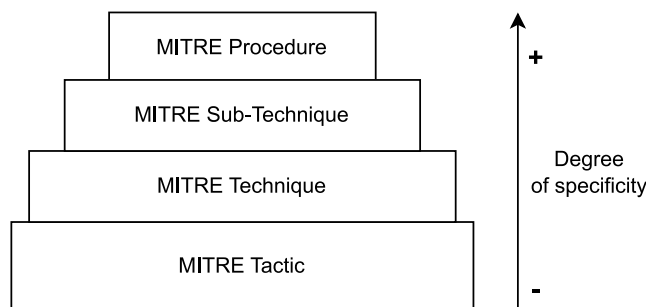
Regarding cyber-attack typification, namely for “AttackType” and “AttackDetails” parameters, following classification frameworks is of paramount importance. By standardizing attack classification processes, an organization can:

### Proactive Cybersecurity tailoring through deception techniques

- Integrate and smoothly adapt the use of external tools (in this work, detection and classification appliances).
- Apply reference terms publicly recognized, facilitating the development of deception plans.
- Share and collect cyber threat intelligence indicators with and from other sources.

Two widely adopted frameworks were encountered that meet the objectives of having a reasonably large spectrum of identified types of cyber-attacks alongside with being publicly recognized. Firstly, MITRE’s Adversarial Tactics, Techniques & Common Knowledge (ATT&CK) is “a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations” [116] that maps the reason (“tactical goal”) of an adversary to a technique or sub-technique. Secondly, Common Attack Pattern Enumeration and Classification (CAPEC) provides “a comprehensive dictionary of known patterns of attack employed by adversaries to exploit known weaknesses in cyber-enabled capabilities” [117]. Both frameworks have the similar goal of classifying adversary behavior. Consequently, a comparison between both frameworks is presented on [118], clarifying that the two frameworks have different scopes and summarily, CAPEC focuses on application security while MITRE ATT&CK centers on network and APT defense. Since a typical organization’s infrastructure consists of both networks, systems, and applications, and that a deception plan should respond to any attack surface, this implementation suggests the adoption both types of classification.

The MITRE ATT&CK framework has the levels of enterprise cyber-attack classification specificity illustrated on *Figure 37*, from tactic to procedure.

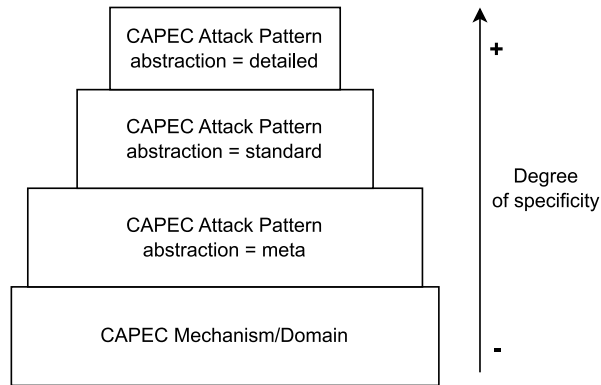


**Figure 37** - MITRE ATT&CK classification.



## Proactive Cybersecurity tailoring through deception techniques

CAPEC organizes its attack classification hierarchy in a rather similar manner, as portrayed in *Figure 38*, from mechanism/domain to attack pattern with the level of detailed regarding abstraction.



**Figure 38** - CAPEC attack classification.

Given the fact that this solution prompts the attacker with specific software artifacts and rely on more precise attack information, the highest degrees of attack classification specificity are advised for “AttackType” and “AttackDetails” data parameters. Lower levels of specificity are more prone for top-level security organizational guidelines. Thus, this implementation acknowledges the following data, for “AttackType” and “AttackDetails” parameters:

- **AttackType**
  - a. “MITRE\_” + {Attack\_Sub-Technique}
  - b. “CAPEC\_” + {Attack\_Pattern\_with\_abstraction=standard}
- **AttackDetails**
  - a. “MITRE\_” + {Attack\_Procedure}
  - b. “CAPEC\_” + {Attack\_Pattern\_with\_abstraction=detailed}

Instances of "Attack\_Info" serve as the data recognized by the system, where the terms "AttackType" and "AttackDetails" correspond to classifications present on the two frameworks that have been previously discussed where examples of such instances are:

```
Attack_Info=("MITRE_LSASS_Driver","21.62.71.123/24","MITRE_Pasam","","")
```

```
Attack_Info=("CAPEC_Overflow_Buffers","121.62.101.123/24","CAPEC_Buffer_Overflow_in_an_API_Call","APT32","CVE-2019-11510")
```

The subsequent sections present a comprehensive overview of each component comprising the deceptive system. Within each section, unique responsibilities of each component are showcased, reiterating their significance in the overall system. Furthermore, a detailed exposition of the implementation process for each component is provided, offering a glimpse into the efforts invested in constructing and integrating them cohesively.

#### **4.1.1. Sophisticated Detection and Auto Attack Analyst**

In order to maximize the value of production type of deployment, tools that detect and automatically classify an ongoing cyber-attack must be used by the deceptive system. As depicted on Section 3.1.2 and particularly in *Figure 33*, regarding the proposed system's extended architecture, this work acknowledges that external tools of attack detection and classification must be integrated instead of implemented. In the same section, specifically on h), DISTDET [111] is presented to be a serious option to take into consideration due to its distributed nature and APT classification capabilities. However, due to its newly development and closed-access, efforts were made to reach out to the authors with the objective of, at least, providing an interface to capacitate this proposed system with a later integration with DISDET. Unfortunately, these efforts were deemed unsuccessful.

Other options were also taken into account, e.g., the use of typical IDS/IPS systems, but due to time constraints and with the intention of avoiding conventional detection mechanisms, no appliances of this sort were adopted in this implementation. Instead, the interaction between the external tools and the implementation of the proposed system is simulated, used for demonstration and validation purposes.

Firstly, *Sophisticated Detection* creates a raw socket (through `socket` Python library) and binds it to all network interfaces expecting Internet Control Message Protocol (ICMP) pings. Upon receiving an incoming ICMP echo request message (type 8 and code 0) is received, this script grabs the IP address of the machine responsible for the ping and notifies *Auto Attack Analyst* with the identified IP via socket connection. The action of pinging the machine where *Sophisticated Detection* runs, in this context, simulates a detected attack.

Secondly, *Auto Attack Analyst* possesses a set of pre-configured cyber-attack parameters, that when provided with IP address information, are sent to *Attack Info Adapter*. The registered IP address is dynamically inserted into those parameters by *Auto Attack Analyst*. The referred parameters are not in conformity with the system-acknowledged

data (“Attack\_Info”) with the aim of enabling Attack Info Adapter to parse that information, depicted on Section 4.1.2. An overview of this process is presented on *Figure 39*.

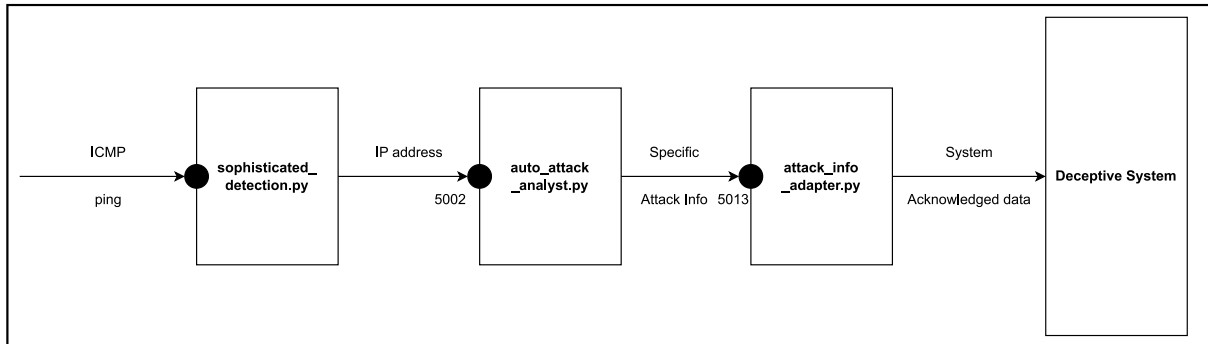


Figure 39 - Simulation of Sophisticated Detection and Auto Attack Analyst.

### 4.1.2. Attack Info Adapter

As conceptually discussed on Section 3.1.2 i), the Attack Info Adapter Python script functions as an intermediary by integrating the external components responsible for attack detection and classification with the deceptive system when in production deployment. Its primary role is to adapt or parse the information, if necessary, and parallelly dispatch the translated information to Deception Generator and Intelligence Python scripts. The detailed functionality of this component is present on *Figure 40*.

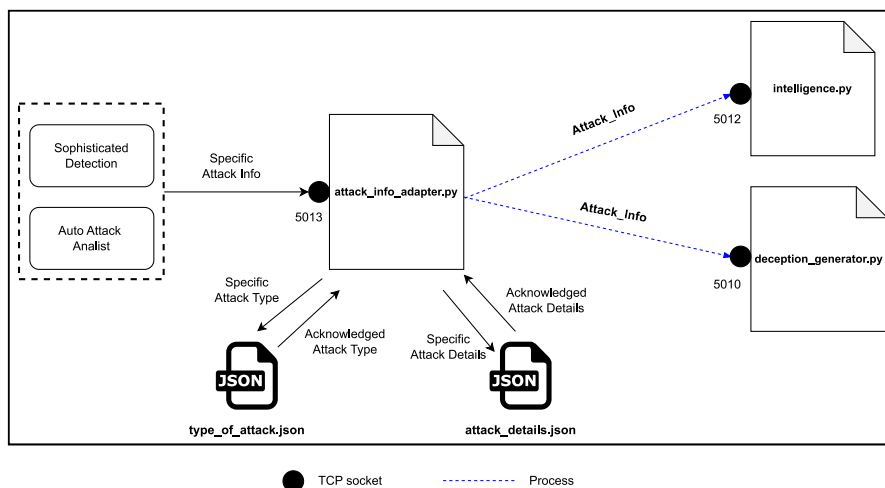


Figure 40 - Attack Info Adapter implementation and associated mechanisms.

## Proactive Cybersecurity tailoring through deception techniques

Supported by socket Python library, Attack Info Adapter possesses a TCP socket server listening on port 5013 for specific attack information from the integrated external tools. Upon its reception, Attack Info Adapter has the responsibility of translating the possible proprietary and specific classification information into system-acknowledge data (referred to as “Attack\_Info”). Particularly, the parsing of the specific information to the previously stated “Attack\_Type” and “Attack\_Details”, regarding MITRE and CAPEC. For this purpose, Attack Info Adapter resorts to two JavaScript Object Notation (JSON) separate files to exemplify the translation of “Attack\_Type” and “Attack\_Details” from proprietary classification to system-acknowledged, acting as dictionaries as illustrated on *Listing 1* and *Listing 2*.

**Listing 1** - Sample of type\_of\_attack.json parsing attack type external tool specific information (*key*) to system-acknowledged data (*value*).

```
{
  "sqli": "CAPEC_SQL_Injection",
  "DoS": "CAPEC_Sustained_Client_Engagement",
  "repoExfiltration": "MITRE_Exfiltration_to_Code_Repository"
}
```

**Listing 2** - Sample of attack\_details.json parsing attack details external tool specific information (*key*) to system-acknowledged data (*value*).

```
{
  "cleSQL": "CAPEC_Command_Line_Execution_through_SQL_Injection",
  "HTTPDoS": "CAPEC_HTTP_DoS",
  "empireTool": "MITRE_Empire"
}
```

Attack Info Adapter queries these two JSON files, using json Python library, and searches for the received specific information (from the integrated external tools), which is the key, and returns the associated value (system-acknowledged values). All supported attacks must be mapped in these files for the deceptive system to be able to employ the tailored deception grid(s) by ongoing cyber-attack parameters, regarding the production deployment.

In a scenario where the security team decides to integrate other external tool(s), only the keys (external tools’ attack nomenclature) of the referred JSON files need to be adapted for the deceptive system’s production deployment to be back operational. This aspect is the main reason why Attack Info Adapter is implemented, thus decoupling the proposed deceptive system from external tool particularities on attack classification.

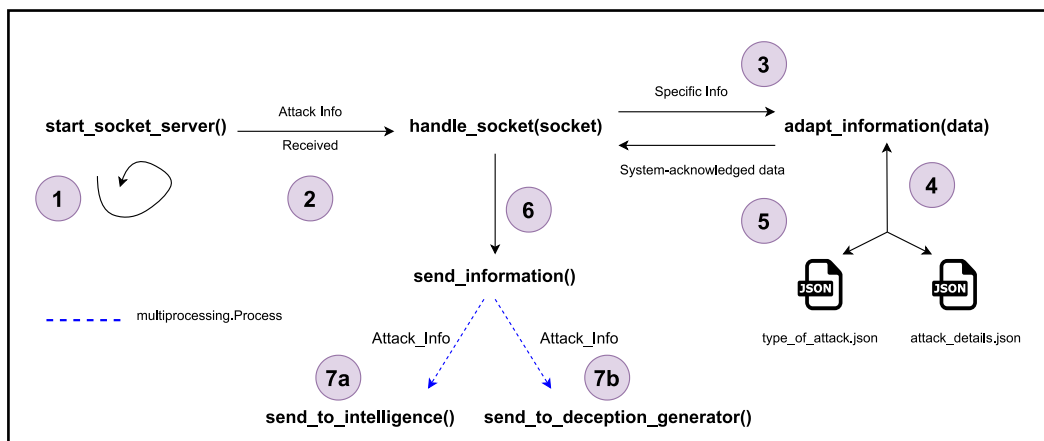
## Proactive Cybersecurity tailoring through deception techniques

Following attack information translation, Attack Info Adapter spawns two separate processes, supported by Python's multiprocessing library which offers process-based parallelism. Each process enables the dispatch of attack information for the respective receptor: Intelligence and Deception Generator. The creation of the separate processes and each's corresponding target is depicted on *Listing 3*, enabling the respective socket connection for the referred scripts to provide the security team with the ongoing attack parameters (in the case of Intelligence) and to deploy the attack parameter tailored deception grid (Deception Generator's responsibility).

**Listing 3** - Parallel process creation for respective socket connection.

```
def send_information():  
    # create separate processes for each socket connection  
    p1 = multiprocessing.Process(target=send_to_intelligence)  
    p2 = multiprocessing.Process(target=send_to_deception_generator)  
  
    # start the processes in parallel  
    p1.start()  
    p2.start()  
  
    # wait for the processes to finish  
    p1.join()  
    p2.join()
```

A numbered overview of the method calling sequence and flow of the Attack Info Adapter Python script is present on *Figure 41*, which materializes the conceptual concerns and responsibilities of this component.



**Figure 41** - Attack Info Adapter's method flow.

Since the thread that starts the socket server remains uninterrupted (represented by label 1 in *Figure 41*), this script continues to accept external connections indefinitely i.e., this script is only interrupted by force on the command line, which enables the system to, if required, deploy several consecutive deception grids resulting of a production type of deployment. This aspect is addressed in Section 4.1.5, providing the system with the possibility of limiting beforehand the number of possible grid research and production deployments.

### 4.1.3. Virtual Asset Repository

The proposed deceptive system requires that a set of virtual artifacts need to be available and accessible in order to be booted and later connected in a virtual network, which in this work's scope is the deception grid. In Section 2.4.1, two types of artifacts were discussed: virtual machines and containers. Technically, both types of these artifacts can be present on Virtual Asset Repository, however, as presented in the same section, advantages of containerization over virtualization are acknowledged. Thus, in this implementation containers are prioritized. For this purpose, the Docker [64] containerization engine is used, providing a fast and flexible platform for deploying deceptive artifacts.

The deceptive plan elected for the ongoing cyber-attack context selects the appropriate artifacts for that plan, specifically, it chooses what Docker images are used to materialize that deception plan. This means that every Docker image that is used in at least one deception plan must be present on the machine/server that later runs them and creates the deception grid (network of containers). Other approach considered was to dynamically generate the Dockerfile e.g., via an API call to a Large Language Model (LLM), to then produce the respective Docker image (build), however that was not implemented. Thus, the system requires that the images need to prematurely exist.

Docker containers are designed to be ephemeral and so, the ability to perform forensics is narrowed. *Postmortem* forensics of a compromised container within the deception grid can be a fruitful procedure to analyze the perpetrator's TTP. Docker does not support native process exportation, and so, this work allows the security team, if intended, to use the Docker `export` command when dynamically removing an asset (container) from the deception grid. This enables to export the content of the container's filesystem, allowing the security team to analyze it afterwards.

## Proactive Cybersecurity tailoring through deception techniques

Real-time forensic procedures are also in place, where every container used needs to be adapted to send logs to Asset Monitoring, which enables the security team to analyze the attacker's course of actions within each deployed host. Logging each container that can be a part of a deceptive strategy is not a simple process as each container runs specific services and so, needs its own method to extract valuable logs. For this reason, it is proposed that every container host possesses a Python script that aggregates the logs on the scope of a container and sends them to Asset Monitoring. To enable this functionality, the process of generation of each Docker image (`docker build`) should provide the container with the Python script. This occurs via the modification of the Dockerfile as exemplified on *Listing 4*. The remainder of Dockerfile instructions are explained on Section 4.1.4.

**Listing 4** - Example of Dockerfile modification to enable deceptive host logging.

```
FROM mysql:5.7
RUN apt-get update; exit 0
RUN apt-get install -y net-tools iproute2 iputils-ping
COPY ./send_logs_to_asset_monitoring.py /
```

Due to the specificity of each service running on each deception grid host (container), the logging scripts require to be unique for each deceptive virtual asset, being the responsibility of the security team to develop them. An example of a Python script inserted into a virtual asset used in a deceptive strategy is demonstrated on *Listing 5*.

**Listing 5** - Example of a Python script that sends logs from a container to Asset Monitoring.

```
@app.route('/send_log', methods=['POST'])
def get_data():
    data=request.get_json()
    message = f"[mn.h4 - WebPage Log] WebPage loaded!! Attacker has seen the page with language:
    {data.get('language')} and message: {data.get('msg')}"

    send_message(message, receiver_host, receiver_port) # send to Asset Monitoring
    return "Roger that!"

def send_message(message, host, port): # send to Asset Monitoring

    with socket.socket(socket.AF_INET, socket.SOCK_DGRAM) as s:
        try:

            s.sendto(message.encode(), (host, port))

            print("Message sent successfully!")
        except ConnectionRefusedError:
            print("Connection refused by Asset Monitoring.")
```

This script supplies a single endpoint API for the service of the given container (a simple frontend web page) to communicate with. The code associated with the frontend is altered to

send an HTTP request to notify the logging script that an attacker loaded the web page and supplies the information that the attacker is presented with. A snippet of the modified frontend's code is exhibited on *Listing 6*.

**Listing 6** - Associated frontend code that provides simplified logging.

```
fetch('http://localhost:8888/')
.then(function (response) {
  return response.json();
})
.then(function (json) {
  document.querySelector("#msg").innerHTML = json.msg;
  document.querySelector("#lang").innerHTML = '<i>-' + json.language + '</i>';

  fetch('http://localhost:8336/send_log', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify(json)
  })
  .then(response => response.json())
})
```

In addition to the introduction of a Python script on each container's filesystem, its execution must occur only upon container booting and deployment on the context of a deception grid. To accomplish this, in every deceptive plan a command to execute this script must be declared as demonstrated on *Listing 14*, for host h1.

#### 4.1.4. Virtual Asset Network and SDN Controller

Network virtualization is of significant importance in modern networking environments as it enables the creation of virtualized network components, such as virtual machines, switches, routers, and network links. These can be used to mimic and simulate complex network topologies without the need for physical hardware. Multiple virtual networks can coexist on a shared physical infrastructure, isolating and securing each network from others, which is an important feature for the present work. Additionally, network virtualization facilitates rapid deployment of network configurations, making it an essential tool for deceiving an attacker in real time.

The implementation of the Virtual Asset Network component needs to rely on the adoption of network virtualization tools such as Mininet [119], an open-source network emulator. The referred appliance “creates a realistic virtual network, running real kernel, switch and application code, on a single machine (VM, cloud or native), in seconds” [119]. Mininet hosts run “standard Linux network software”, and its network switches “support



*OpenFlow for highly flexible custom routing and Software-Defined Networking*” [119]. Summarily, Mininet accomplishes this task by creating host namespaces (h1, h2, etc.) and connecting them through virtual interfaces. Moreover, this tool provides a Python interface through libraries to facilitate a programmatic approach to network deployment. In the context of this work, where a deception grid must be tailored by cyber-attack parameters, this aspect clearly aligns with the system requirements and is conveniently seized.

Since the developed system’s Virtual Asset Repository seizes Docker containers as the deceptive assets suitable to be deployed in the deception grid, containers must be eligible as virtual hosts. For that purpose, Containernet [120] is adopted in this implementation, which forks Mininet and empowers “*Docker containers as hosts in emulated network topologies*” [120]. Containernet’s main functionalities include, extracted from [120]:

- Add, remove Docker containers to Mininet topologies.
- Connect Docker containers to topology (to switches, other containers, or legacy Mininet hosts).
- Execute commands inside containers by using the Mininet CLI.
- Dynamic topology changes.

Containers that are ran and managed by Containernet to materialize any deceptive plan should fulfill the following requirements, extracted from [120]:

- bash installed .
- ip installed (e.g., Ubuntu iproute package).
- ping installed (e.g., Ubuntu iputils-ping package).
- The main entry point script must not block the shell.

In the case of a container not supporting the referred requirements natively, they need to suffer modifications to be able to be included in a Mininet (in this case Containernet) network which in this context is a deployed deception grid, as depicted in [121]. An example of modifications on a MySQL container image to fit with Mininet’s requirements of a Docker host is exposed in *Listing 7*.

## Proactive Cybersecurity tailoring through deception techniques

**Listing 7** - Example of Dockerfile modification to fit Containernet's requirements.

```
FROM mysql:5.7
RUN apt-get update; exit 0
RUN apt-get install -y net-tools iproute2 iputils-ping
```

In this implementation, Containernet is used through its Docker container version (specifically its “:latest” version) running on the Ubuntu virtual machine, as depicted in *Figure 36*. Deception Generator and Static Deception Planner Python scripts are stored and executed inside this container to be able to import Containernet’s Python libraries. In order to efficiently run Containernet container, some ports are required to be open to enable communication with Deception Generator from the outside (from other components of the deceptive system), as the command illustrated in *Listing 8* demonstrates.

**Listing 8** - Docker command to run Containernet in the context of the proposed deceptive system.

```
docker run --name containernet -it --rm --privileged --pid='host' -p 5010:5010
-p 5014:5014 -p 5045:5045 -v /var/run/docker.sock:/var/run/docker.sock
containernet/containernet /bin/bash
```

Containernet is responsible for building the tailored network of virtual assets by connecting the Docker hosts and switches in a specific topology, given the chosen deceptive plan of a cyber-attack context.

Subsequently to the assembly of a network of container hosts (in this scope, the deception grid), this network and its inherent properties must be administered. Network management is a crucial part of administrating an IT infrastructure, where monitoring and configuration processes are in place. SDN plays an important role in modern networks, as presented in Section 2.4.2, which provides a streamlined and automated approach to network management presenting a great value to the present work since the deception grid, after its assembly, must be automatically configured and managed.

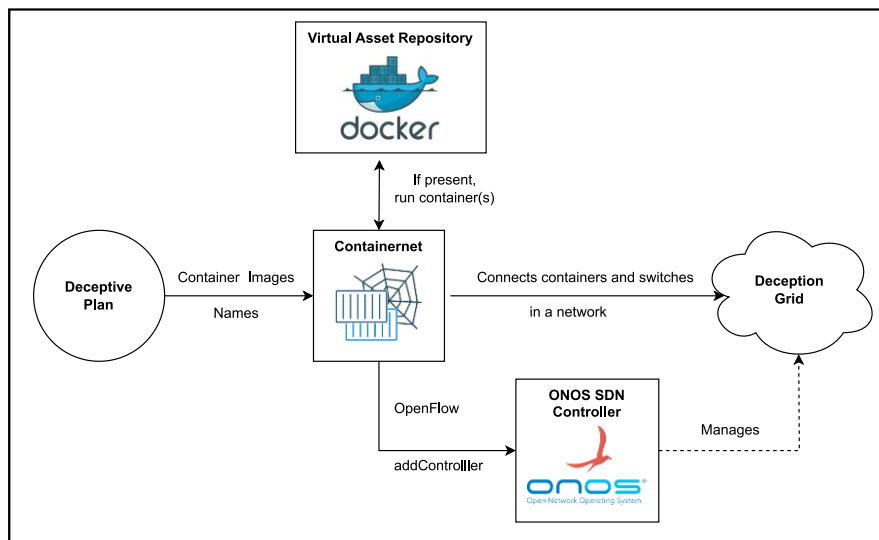
There are a few open-source SDN controllers available. The most well documented and accessible encountered was ONOS [70]. This controller enables deception grid configuring and its dynamic topology modification. This framework is Java-based and provides a modular architecture that allows the development of virtualized network features and functions by centralizing management, abstracting the control plane from the data forwarding function. SDN controllers possess northbound and southbound APIs. ONOS supports a wide variety of

## Proactive Cybersecurity tailoring through deception techniques

protocols, including REST-based northbound APIs and southbound protocols such as OpenFlow and NETCONF.

This implementation is performed on a pre-configured “ONOS SDN-enabled” Ubuntu 16.04 virtual machine, referenced in the ONOS documentation [122], being the base host for all components. This appliance enables the “plug-and-play” use of this technology, providing a cluster of three ONOS controller instances (version 1.15.0) materialized as Docker containers, along with the required configurations. ONOS is responsible for controlling the behavior of the network devices, such as switches, in the software-defined deception grid. It manages the flow of traffic, implements network policies, and provides a centralized control plane for network management.

The SDN controller and Mininet/Containernet operate on two different planes within the network architecture. While ONOS acts on the control plane of the deception grid, Containernet simulates the data/physical plane. An overview of the mechanisms of the depicted components is illustrated on *Figure 42*.



**Figure 42** - Virtual Asset Network and SDN implementation through Containernet and ONOS, respectively.

The two referred network appliances (SDN and virtual network emulator) must interconnect their planes to build a full stack network topology. ONOS SDN has the ability of managing OpenFlow switches. This protocol defines communication between a SDN controller and a OpenFlow-enabled network device, being the standard southbound protocol used between the SDN controller and a switch. Containernet creates a virtual network with switches and Docker hosts, using OpenFlow-enabled software switches. By assigning a controller to

those switches, via OpenFlow protocol, ONOS controller is connected to the switches in the Containernet network. This allows the controller to manage the switches and control the flow of traffic within the virtual network (deception grid). Containernet provides, through its Python library, the `addController(controllerIp)` method that allows to essentially connect a SDN controller to a network, providing the control plane to the deceptive virtual network.

From an infrastructure perspective, what is “seen” by an attacker is the SDN controller’s view over the network, since SDN possesses the infrastructural responsibility of controlling it. Thus, when, for example, a host is added or removed from the SDN controller (e.g., via its RESTCONF API), the host is added/deleted from the controller's view. However, the host still exists within the Containernet network because it manages the virtual network topology (a different plane from the SDN). In a production environment where a network is controlled by an SDN controller, if a host is added/removed from the network, the host would become reachable/unreachable from an infrastructure perspective, thus, ONOS decides on host accessibility for the attacker to be able or not to interact with. In *Figure 43* a list of network interfaces of a Containernet’s generated network host is depicted. As demonstrated, a host has at least three identified interfaces: loopback address (“127.0.0.1”); Docker network’s address (“172.17.0.10”) and the dynamically generated IP address for the deception grid (“192.168.200.140”) which is the one recognized by the SDN controller.

```
containernet> h2 ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid lft forever preferred lft forever
63: eth0@if64: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    group default
    link/ether 02:42:ac:11:00:0a brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.17.0.10/16 brd 172.17.255.255 scope global eth0
        valid lft forever preferred lft forever
118: h2-eth1@if117: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    group default qlen 1000
    link/ether 6c:c0:03:8f:ec:e1 brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet 192.168.200.140/8 scope global h2-eth1
        valid lft forever preferred lft forever
```

Figure 43 - Host in the Containernet network and associated network interfaces.

The present contribution seizes the effortless integration between these two technologies based on [123], which applies the same technological stack, providing an SDN-Docker based architecture for IoT. Additionally, the referred work uses Mininet-WiFi [124], which also forks “*Mininet SDN network emulator and extended the functionality of Mininet by adding virtualized WiFi Stations and Access Points based on the standard Linux wireless drivers and the 80211\_hwsim wireless simulation driver*” [124]. The adoption of this technology helps

widen the spectrum of possible deception plans by providing Wi-Fi based deception, through the implantation of deception grids containing Wi-Fi artifacts in a WLAN environment. However, this was not implemented in this work thus far due to time restrictions.

### 4.1.5. Deception Generator

Deception Generator is the starting point script for selection and application of deception plans in the aftermath of cyber-attack context provisioning from Attack Info Adapter (production deployment) or from the security team (via Intelligence). This script runs inside Containernet container (with Python 3.6.9) and is mainly responsible for receiving and handling connections on the following TCP sockets, each supported by its own process:

- 5010 – Deception grid deployment through Attack Info Adapter.
- 5014 – Deception grid deployment through Intelligence.
- 5045 – *Postmortem* command to export a container's (host on deception grid) filesystem after its removal from SDN network.

Given the fact that this script is running inside a container, port forwarding is needed in order to map the Docker network inner ports of the container to the outer host ports, thus, enabling outside communication, accomplished by executing the instruction on *Listing 8*.

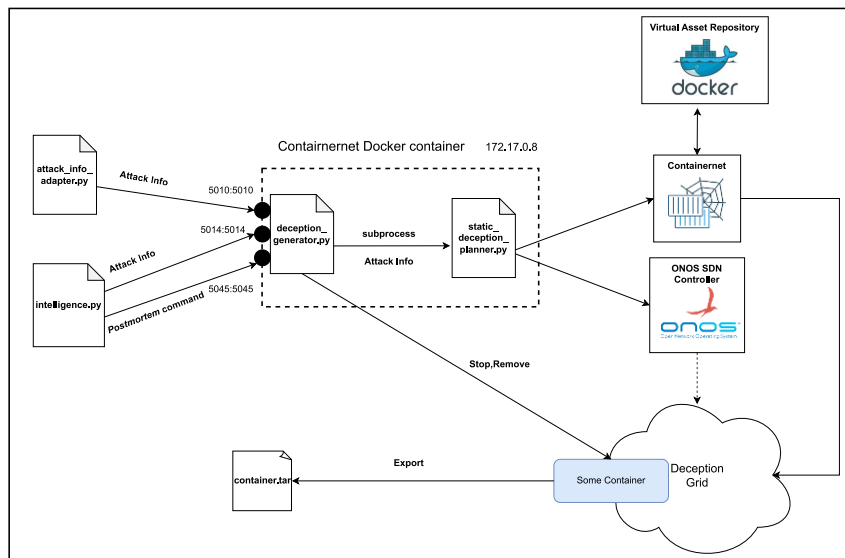
As discussed in Section 4.1.3, Docker offers a way to export a container's filesystem in a given moment which presents an opportunity for conducting *postmortem* forensics. Dynamic host removal from the deception grid can mean for example that the deceptive strategy is lacking effectiveness or that the host was fully compromised. In a scenario where the security team decides to shut down a host from the deception grid, the system provides a mechanism that enables them to, if needed, store the ceased host's filesystem through Intelligence's API. The operation responsible connects to Deception Generator's port 5045, supplying information about the container's ID or name intended to shut down and the name/path of the output file where its filesystem will be stored within Containernet's container. Thereafter, Deception Generator executes Docker commands `export`, `stop` and `rm` which stores the content of the grid's container (in a .tar file), stops it, and removes it, respectively.

As an entry point for deception execution, Deception Generator can and should make administrative decisions concerning deception grid deployment management. An example of

## Proactive Cybersecurity tailoring through deception techniques

that in this implementation is the ability to limit the number of deception grid deployments, production or research. Deception Generator has a `MAX_DECEPTION_GRIDS` and a `deception_grids_counter` global variables which keep track of the number of deployed grids and in the case of surpassing the stipulated value, blocks any new deployment.

Concerning deception plan implantation, Deception Generator relies on its Static Deception Planner module (materialized in another Python script) to actively build the deception grid. Whenever Deception Generator receives a deployment instruction either from port 5010 (production deployment) or 5014 (research deployment) with the associated cyber-attack information, it spawns a new process, supported by Python's subprocess library, that runs Static Deception Planner script supplying the cyber-attack information related to the requested deployment. Static Deception Planner then actively interacts with the Mininet/Containernet's packages to build the deceptive network, which is specified in Section 4.1.6. A summary of the Deception Generator's responsibilities and its association with Static Deception Planner is visible on *Figure 44*.



**Figure 44** - Deception Generator's responsibilities and interaction with Static Deception Planner.

Similarly to Attack Info Adapter, this script remains uninterrupted and so, it is capable of receiving and respond to consecutive deployment requests and delegate Static Deception Planner to fulfill them and deploy the corresponding deception grid(s).

### 4.1.6. Static Deception Planner

Summoned by Deception Generator, Static Deception Planner is one of the two cores of deception execution. This script possesses a direct link with Containernet through its Python library (by running inside Containernet container), to deploy the network of virtual deceptive artifacts according to cyber-attack parameters.

Firstly, as stated on Section 4.1, the “Attack\_Info” system-acknowledged data has four different possible combinations regarding cyber-attack parameters. Static Deception Planner upon receiving that information from Deception Generator, concerning the current deception grid deployment request, checks what parameters were made available to the deception generation component. The number of received parameters determines from what playbook a deception strategy is chosen from. A playbook refers to a set JSON files (in a directory) that contains deception strategies for each “Attack\_Info” parameter combination. Deceptive accuracy increases proportionally to the number of provided attack parameters. Since four different parameter combinations are defined in this implementation, four different playbooks can be selected, as represented on *Figure 45*. Playbook number four is acknowledged to have the highest degree of deceptive accuracy. Static Deception Planner contains a `playbook_mapping` variable which defines playbook selection given the attack parameter combination and a `playbook` variable that given that mapping and the received parameters registers from what playbook a deception strategy will be chosen.

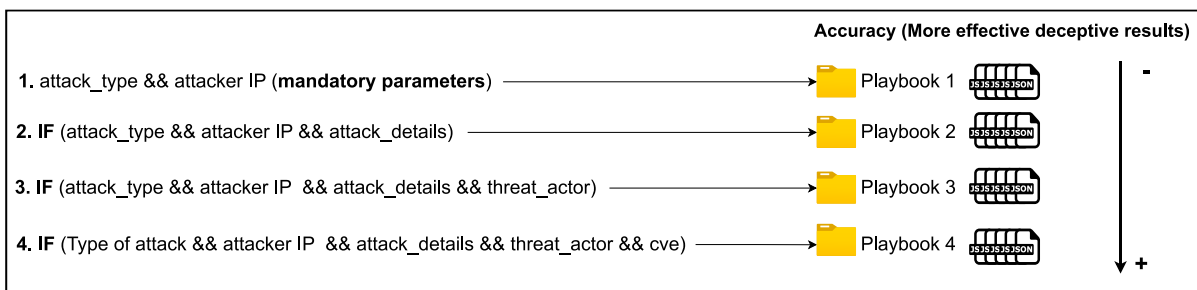


Figure 45 - Deception playbook selection process.

Deception plan selection within each playbook relies on the specific values for each attack parameter. Deception planning and strategies for this implementation are discussed and specified in Section 4.2. Summarily, the number of parameters determine from which set of

JSON files a deception strategy is selected and the specific strategy election is due to each cyber-attack parameter value.

Attacker IP address related information enables the system to not only record the identified attacker IP but additionally to use it in the deceptive process. For implementational purposes, the IP of the attacker is seized to assign conveniently IP addresses for deception grid hosts. For every host returned from a deception strategy in a playbook, `Static Deception Planner` assigns it a random IP address within the network of the attacker IP. This allows that the deception grid's hosts can be infrastructurally spawned "in front" of the attacker by the SDN, prompting the network of deceptive artifacts directly around the attacker facilitating its interaction. MAC addresses are also randomized for each host present in a deceptive plan.

Deceptive network assembly depends on the use of `Containernet` methods. The set of `Containernet/Mininet` functions used are: `addController()`, `addSwitch()`, `addDocker()` and `addLink()`. For this purpose, `Static Deception Planner` utilizes these methods to implement the specific SDN controllers, switches, Docker hosts and links (topology), respectively, from the selected pre-planned deception strategy. Commands to execute in each host are also supplied. Additionally, this script communicates with ONOS SDN Controller REST API (selected controller(s) for the given deception strategy) to implement traffic policy, flow and management rules. ONOS SDN offers policy-based directives via its Intent Framework, which allows "*to specify their network control desires in form of policy rather than mechanism*" [125] making possible to state data-paths at for a given host or switch port. Flows [126] are rules that define partial paths on a switch, assign packets to a port and what is the port's output. Configurations for Intents and Flows are automated due to the use of ONOS SDN REST API, specifically its `/intents` and `/flows` endpoints (POST method). In this case, each necessary HTTP request's body to configure the deception grid via Intents and Flows is provided by the JSON files which contain the specific configurations for the given "Attack\_Info" parameters.

Given the fact that all these assets (hosts and switches), links and controllers are specified in the JSON files, they need to be firstly mapped into `Containernet`-enabled objects after their fetching, which is one of the main `Static Deception Planner`'s responsibilities. A detailed representation of what `Static Deception Planner` entails by using the referred methods and JSON files to construct the deception grid is present on *Figure 46*. The variable number represented in the figure is the selected playbook (playbook variable, as stated before).



## Proactive Cybersecurity tailoring through deception techniques

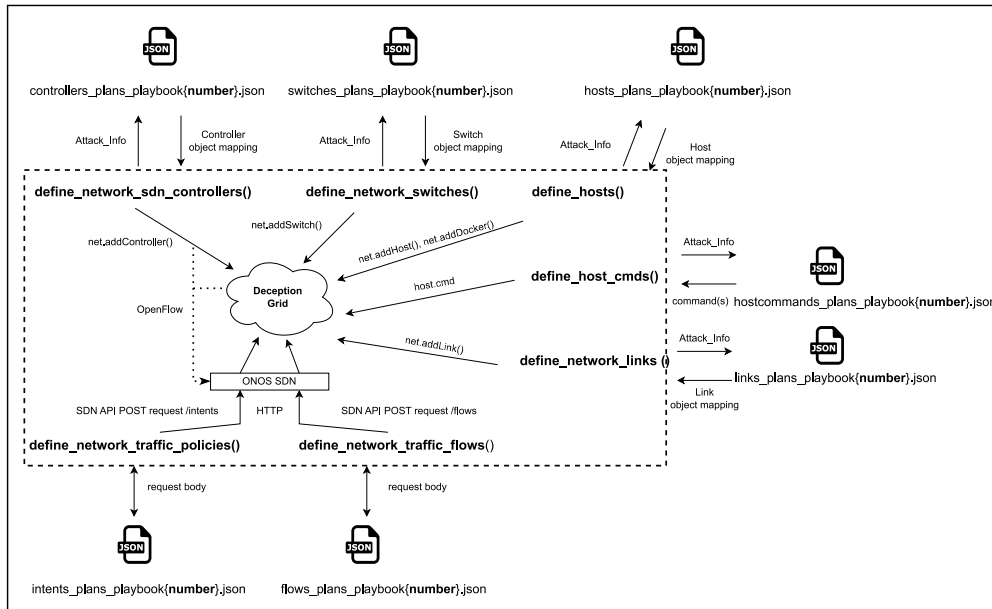


Figure 46 - Static Deception Planner and associated mechanisms.

*Listing 9* showcases the method that defines hosts for a given cyber-attack context. Hosts are retrieved from a specific JSON file and are subsequently mapped. Thereafter, random IP addresses within the attacker's network and MAC addresses are assigned and finally returned in an array (`specific_attacks_hosts`). Afterwards, each host in this array is added to the network as seen on *Listing 10*.

In order to not repeat already allocated IP and MAC addresses, the script possesses two set Python data structures to keep track of address assignments. `Hosts_ips_macs` is an array that maps each host with its corresponding IP and MAC addresses for host and traffic configuration purposes, as exemplified in *Listing 11*.

Listing 9 - Host definition for a given deception strategy.

```
def define_hosts(attack_type,attacker_ip,attack_details,threat_actor,cve,playbook):
    parse_json = load_json(f'./plans_config/playbook{playbook}/hosts_plans_playbook{playbook}.json')
    specific_attacks_hosts = parse_json.get(f'"{attack_type}|{attack_details}|{threat_actor}|{cve}"')

    if specific_attacks_hosts:
        for host in specific_attacks_hosts:
            random_ip = get_random_ip_within_network(attacker_ip)
            random_mac = get_random_mac()
            host['ip'] = random_ip
            host['mac'] = random_mac
            host = {"name":host['name'], "ip": random_ip, "mac": random_mac}
            hosts_ips_macs.append(host)
        return specific_attacks_hosts
    else:
        return "No attack corresponding given parameters was found."
```

**Listing 10** - Adding hosts to the network.

```
info('*** Adding Deception Grid Hosts ***\n')
hosts = define_hosts(type_of_attack,attacker_ip,attack_details,threat_actor,cve,playbook)
net_hosts = []

for host in hosts:
    if(host['isDocker'] == 1):
        if all(key in host for key in ('ports', 'port_bindings', 'environment')):
            net_hosts.append({ host['name'] :
net.addDocker(host['name'],ip=host['ip'],mac=host['mac'],dimage=host['dimage'],
ports=host['ports'],port_bindings=host['port_bindings'], environment=host['environment'])})
        else:
            (. . .)
    else:
        net_hosts.append({ host['name'] :net.addHost(host['name'],ip=host['ip'])})
```

**Listing 11** - Configuring network policies via ONOS SDN REST API (intents).

```
def define_network_traffic_policies(controllers,attack_type,attacker_ip,attack_details,threat_actor,cve,playbook):

    parse_json = load_json(f'./plans_config/playbook{playbook}/intents_plans_playbook{playbook}.json')
    specific_attacks_intents = parse_json.get(f"{attack_type}|{attack_details}|{threat_actor}|{cve}")

    if len(specific_attacks_intents) > 0:
        for controller in controllers:
            controller_ip = controller["ip"]
            url = f"http://{controller_ip}:8181/onos/v1/intents"
            headers = {'Accept' : 'application/json'}

            for intent in specific_attacks_intents:
                for criterion in intent["selector"]["criteria"]:
                    if criterion.get("type") == "IPV4_SRC" or criterion.get("type") == "IPV4_DST":
                        random_ip_host = ""
                        for obj in hosts_ips_macs:
                            if obj.get("name") == criterion.get("ip") :
                                random_ip_host = obj.get("ip")
                                criterion["ip"] = random_ip_host+"/32"
                        response = requests.post(url, headers=headers, data=json.dumps(intent), auth=('onos','rocks'))
                        print(f"Response from SDN Controller ({controller_ip}) for intent: ", response)
            else:
                print("No traffic policies were assigned.")
```

In a scenario where a deception strategy is required to be added, removed, or updated by the security team, this can be accomplished by only editing the JSON files, while Static Deception Planner and the rest of the system remain unaltered, which represents a crucial operational advantage due to its loose coupled nature. This is also true for the rest of the system, where to support a new cyber-attack, only the JSON files that translate specific information about attack classification into system-acknowledged data need to be altered in addition to the aforementioned deceptive strategies. Ultimately, these translating and deceptive strategizing JSON files can be made available to the system in an “online” manner.

### 4.1.7. Asset Monitoring

Adversary's tactics, techniques and procedures analysis are of extreme importance to profile the attacker and to track its steps throughout the deception grid. Ultimately, this can enable the discovery of zero-day vulnerabilities. Furthermore, this investigational procedure can suggest that the employed deceptive strategy is currently not having the intended impact and effectiveness desired by the security team.

Examining the adversary within the deception grid encompasses, in this context, two logging approaches: network and host logs. To provide the security team with the needed logs, a script that aggregates both sets of logs is implemented, referred to as `Asset Monitoring`.

`Asset Monitoring` aims to collect virtual asset logs and network logs which are provided to it from each host Docker container and from the SDN controller, respectively. Concerning network-centric logs, ONOS SDN provides logging functionality to external syslog servers or services, beside its command-interface, for production environments [127]. Initially, this seemed to be a straightforward approach, however, when attempting to receive SDN logs on `Asset Monitoring` Python syslog handler nothing was ever logged. There are two main possible reasons acknowledged for this. Either the containerized version of ONOS needs different or extra configurations to be able to send logs or the developed syslog server could not correctly manage the connection, thus, other approach was adopted. Docker command logs can fetch the logs of a container and in the case of logging the ONOS SDN controller, the information retrieved is the same as the initial logging method. So, `Asset Monitoring` seizes that functionality to extract and forward streaming network (SDN) logs to `Intelligence`.

As discussed in Section 4.1.3, each virtual asset possesses in its filesystem a Python script that aggregates all service valuable information and sends it to `Asset Monitoring`. For these type of logs, `Asset Monitoring` possesses a UDP socket listening on port 514 for every container's logs connections and forwards them to `Intelligence`.

*Listing 12* showcases `Asset Monitoring`'s methods that receive, extract and handle both logging functionalities. Each method runs in a different process to guarantee operation independence, and both call the `send_information()` method to dispatch the logs to `Intelligence` which are consecutively made available to the security team.

**Listing 12** - Asset Monitoring capability of extracting and dispatching network (SDN) and host logs.

```
def receive_sdn_log():
    while True:
        #command = ["docker", "container", "logs", "--details", "--follow", CONTAINER_NAME] #
        production mode
        command = ["docker", "container", "logs", "--details", "--tail", "10", CONTAINER_NAME] #
        demonstration mode delete to production mode)
        process = subprocess.Popen(command, stdout=subprocess.PIPE, stderr=subprocess.PIPE,
        text=True)
        try:
            for line in process.stdout:
                line = line.strip()
                log_message = "[ONOS SDN LOG] : " + line
                print("Received log line:", log_message)
                send_information(log_message.encode()) # send to intelligence
            except KeyboardInterrupt:
                print("Keyboard interrupt received. Exiting...")
            process.communicate()
            time.sleep(10) # for demonstration purposes (delete to production mode)

def receive_host_log():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_socket.bind(("172.17.0.1", 514))
    print("Syslog server started. Waiting for logs...")
    try:
        while True:
            data, address = server_socket.recvfrom(4096)
            send_information(data) # send to intelligence
            print(f"Received log from {address[0]}:\n{data.decode()}")
        except KeyboardInterrupt:
            print("Keyboard interrupt received. Exiting...")
    finally:
        server_socket.close()
```

#### 4.1.8. Intelligence

The role of the Intelligence component (materialized in a Python script) involves enabling bidirectional communication with the deception system and its grids, serving as a system gateway for the security team. To materialize its API, a Web Server Gateway Interface (WSGI) appliance named Flask [128] was utilized. This empowers the creation of a RESTful API enabling inbound and outbound communication of the deceptive system (e.g., from a client security team application). Flask library requires that version 3.8 or newer of Python is used. The Python version used to develop and execute the Intelligence script was 3.9.12.

For this implementation, six endpoints were established to enable the security team with important operations on the deceptive system, listed on *Table 5*. These follow the conceptual proposal addressed on Section 3.1.1 a), which determines a list of valuable operations that the security team must be able to execute on the deceptive system.

## Proactive Cybersecurity tailoring through deception techniques

**Table 5** - Intelligence's API operations.

HTTP Method	Endpoint	Body	Description
GET	/api/attacks	N/A	Return recorded detected attacks (production deployments).
GET	/api/deployments	N/A	Return recorded deployment requests by the security team (research deployments).
GET	/api/topology/{controllerIP}	N/A	Returns the current deployed grids' topologies.
POST	/api/hosts/{controllerIP}	JSON string with host parameters: "host_ip", "host_mac", switch_name", "vlan" and "port"	Dynamically adds a host to a deception grid.
DELETE	/api/hosts/{controllerIP} or /api/hosts/{controllerIP}?postmortem=true	JSON string with host parameters : "host_mac" and "vlan". If <i>postmortem</i> add: "container_id_or_name" and "output file".	Dynamically removes a host to a deception grid with the possibility of performing <i>postmortem</i> forensics.
POST	/api/topology	JSON string with "Attack_Info" parameters.	Deception grid deployment request (research deployment).

On every occasion where a deception grid is deployed as consequence of a detected attack (production) or by requirement of the security team (research), Intelligence has two global array variables whose objective is to keep track of such deployments by storing the triggering "Attack\_Info" parameters and its date/time. If the security team desires to inspect what deployments were performed, when they occurred and what attack parameters triggered each deployment, they can resort to /api/deployments (research) and /api/attacks (production) endpoints to obtain such data.

Topology information of the current deployed deception grid(s) is made available to the security team via /api/topology. Intelligence upon receiving an HTTP GET request for this endpoint, conveniently uses ONOS SDN RESTCONF northbound API, specifically its /devices and /hosts endpoints, aggregates both GET responses and presents them to the security team. ONOS SDN device information refers to brief switch-related data while host information concerns a summary of the virtual hosts.

Dynamic deception strategy modification is expressed via the addition and removal of deceptive artifacts of the deception grid. As discussed previously, the decision of dynamically modifying a deception grid can come from the lack of effectiveness recognized by the security team or the full compromising of a host. For these operations (host allocation and deallocation) Intelligence resorts to ONOS SDN RESTCONF API to add and remove hosts on the software-defined deception grid(s), through its /hosts endpoint by performing an HTTP POST or DELETE requests. To provide the required information to the SDN Controller for host allocation purposes, the Intelligence's /api/hosts endpoints requires the following

information from the security team, which is passed in the body of the POST request: host's IP address, host's MAC address, switch that the host will connect to, its assigned vlan and switch port. To deallocate a host, the security must perform a HTTP DELETE request to Intelligence's `/api/hosts` stating the host's MAC address and its vlan. Furthermore, in case that the security team opts to perform *postmortem* host analysis they must contain a request query parameter "postmortem" equaling it to "true", which instructs Intelligence to send a postmortem command via socket connection to Deception Generator (as referred on Section 4.1.5). Addition of a host only occurs on the ONOS SDN controller management plane while removing stops the container, thus, removing it from the data plane (Containernet network).

Research deployment of deception grids are premeditated, occurring by security team's command. In this implementation, the deceptive system allows research type of deployments via Intelligence and its REST API. The `/api/topology` endpoint is at the team's disposal to interact with via an HTTP POST request, passing "Attack\_Info" parameters in its body. Thereafter, via socket connection in a new thread, Intelligence dispatches the same information to Deception Generator which by its own mechanisms deploys the deception grid tailored by such parameters (as explained in Section 4.1.5 and 4.1.6). The responsible handling method for this operation is presented on *Listing 13*.

**Listing 13** - Research deployment enabled by Intelligence's REST API.

```
@app.route('/api/topology', methods=['POST'])
def deploy_deception_grid():
    data = request.get_json()
    attack_type = data.get('attack_type')
    ip = data.get('IP')
    attack_details = data.get('attack_details')
    threat_actor = data.get('threat_actor')
    cve = data.get('cve')

    date_to_record = datetime.datetime.now() # save current timestamp/date
    data['date/time'] = date_to_record

    global recorded_deployments
    recorded_deployments.append(data)
    threading.Thread(target=deploy_grid, args=(attack_type,ip,attack_details,threat_actor,cve,)).start()
    print('Deployment command and information received : ', data)
    response_data = {'status': 'success', 'message': 'Deployment command sent successfully for
attack type = {}, ip = {}, attack details = {} and threat actor = {}'.format(
    attack_type,ip,attack_details,threat_actor)}
    return jsonify(response_data)

def deploy_grid(attack_type,ip,attack_details,threat_actor,cve):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s2:
        s2.connect((HOST_DECEPTION_GENERATOR, PORT_DECEPTION_GENERATOR))
        params = '{} {} {} {} {}'.format(attack_type,ip,attack_details,threat_actor,cve)
        s2.sendall(params.encode())
        data = s2.recv(1024)
        print(repr(data.decode()))
```

## Proactive Cybersecurity tailoring through deception techniques

In addition to its REST API-related functionalities, Intelligence is the receptor for deception grid logging information. This data is supplied by Asset Monitoring regarding network and host logs. Intelligence has a UDP socket (port 514) constantly listening for log shipping. Upon receiving logs, they are immediately dispatched to other external component for the security team to store and analyze logs, referred to as SIEM. The security team can adopt any SIEM technology, only needing to define what is its IP address in the Intelligence script in order to enable communication (since the majority of SIEMs support syslog). A summary of Intelligence's mechanisms is represented on *Figure 47*.

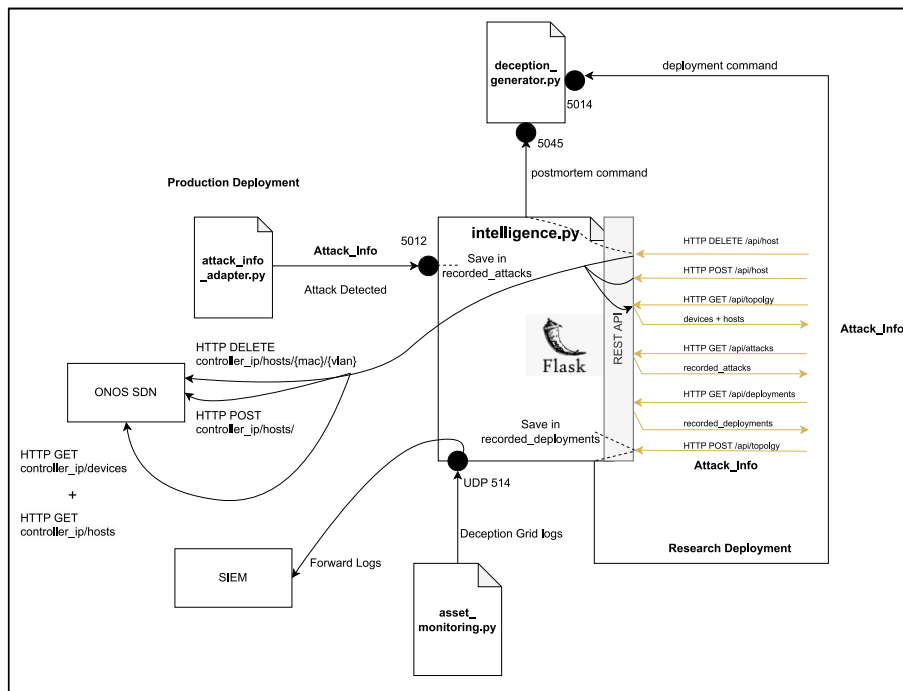


Figure 47 - Intelligence's implementational mechanisms.

## 4.2. Deception Plans

Deception Generator and particularly its Static Deception Planner module need to resort to pre-planned deceptive strategies to deploy a deception grid tailored by the received attack parameters (“Attack\_Info”). The set of deception plans made available to these components is considered to be the second core of deception execution of the system. These, which are materialized in several JSON files, provide Static Deception Planner with the required network and asset information in order to deploy the deception grid via Containernet

and ONOS SDN. There are four directories, each containing seven JSON files. These directories represent each available deception playbook of the system which is selected via a cyber-attack availability check (as portrayed in *Figure 45*). Following playbook selection, Static Deception Planner consults the JSON files which belong to the selected playbook and returns from each JSON file the assigned information for the context of “Attack\_Info” parameters’ values. The seven JSON files aim to seize Containernet and ONOS SDN mechanisms to deploy, manage and configure the deception grid are the following (variable *p* refers to the playbook selected):

- `hosts_plans_playbook{p}.json` – Set of deception grid hosts, Docker container or not.
- `switches_plans_playbook{p}.json` – Number and names of OpenFlow-enabled network switches.
- `controllers_plans_playbook{p}.json` – Set of ONOS SDN controllers.
- `links_plans_playbook{p}.json` – Set of network links between hosts and switches (hosts  $\Leftrightarrow$  switches and switches  $\Leftrightarrow$  switches) which form a specific deception grid topology.
- `hostcommands_plans_playbook{p}.json` – Set of virtual host commands to run when deploying the grid.
- `intents_plans_playbook{p}.json` – Set of ONOS SDN Intents to configure deception grid’s traffic policies.
- `flows_plans_playbook{p}.json` – Set of ONOS SDN Flows to configure deception grid’s traffic.

A deception strategy or plan is essentially the combined values extracted from each of the seven JSON files for a key that is the cyber-attack context (“Attack\_Info” parameters’ values). Placing deception strategies in different JSON files within each playbook and the decision of dividing them into playbooks provides a more granular environment for the security to refine, add or remove specific aspects of a deception strategy.

As presented in Section 4.1.6, Static Deception Planner has a `host_ips_macs` array in memory to track each host’s IP and MAC addresses. This is utmost useful due to the operational decision to assign dynamically both addresses to each host. Thus, those fields can’t be premeditatively assigned in the deception strategies in the JSON files. Instead, these parameters are either omitted in the case of `hosts_plans_playbook{p}.json`, they contain



## Proactive Cybersecurity tailoring through deception techniques

the host identifier, on intent and flow definition, which are later dynamically replaced for the corresponding IP address generated by Static Deception Planner, as already demonstrated by *Listing 11*, or contain the keyword “ip” on the command list to replace with the corresponding host upon command execution in each host, as outlined in red on *Listing 14*.

*Listing 14* and *Listing 15* showcase an example of a deceptive strategy, for host command and SDN controller assignment respectively, for all recognized “Attack\_Info” parameters (Playbook 4). Within the cyber-attack key on host command assignment, the security team must provide an array of objects. Each object contains two properties: the command to execute on the host and the host identifier. This enables Static Deception Planner to map these instructions to the corresponding host and execute them via Containernet’s libraries.

**Listing 14** - Example of host command assignment for a given deceptive strategy.

```
{
  "CAPEC_SQL_Injection|CAPEC_Command_Line_Execution_through_SQL_Injection|APT42|CVE-2022-30927": [
    {
      "cmd" : "/entrypoint.sh mysqld &" ,
      "host" : "h1"
    },
    {
      "cmd" : "python3 send_logs_to_asset_monitoring.py &" ,
      "host" : "h1"
    },
    {
      "cmd" : "/usr/local/bin/docker-entrypoint.sh apache2-foreground &",
      "host" : "h2"
    },
    {
      "cmd" : "ifconfig h3-eth1 ip",
      "host" : "h3"
    },
    {
      "cmd" : "python3 /usr/src/app/app.py &",
      "host" : "h3"
    },
    {
      "cmd" : "service nginx start& ",
      "host" : "h4"
    }
  ]
}
```

**Listing 15** - Example of SDN controller assignment for a given deceptive strategy.

```
{
  "CAPEC_SQL_Injection|CAPEC_Command_Line_Execution_through_SQL_Injection|APT42|CVE-2022-30927": [
    {
      "name": "c0", "ip": "172.17.0.5", "port": 6633
    }
  ]
}
```

Listing 16 demonstrates a host assignment of a deceptive strategy that belongs to “Playbook 3”, thus no CVE was identified and so, that information must be deducted from the key. These deductions occur for the other deceptive playbooks regarding non-available parameters.

Listing 16 - Example of host assignment for a given deceptive strategy.

```
{
  "MITRE_Data_Encoding|MITRE_BADNEWS|Patchwork": [
    {
      "isDocker": 1,
      "name": "h10",
      "ip": "",
      "mac": "",
      "dimage": "ubuntu:trusty"
    },
    {
      "isDocker": 1,
      "name": "h11",
      "ip": "",
      "mac": "",
      "dimage": "ubuntu:trusty"
    }
  ]
}
```

### 4.3. Threat Modeling

Threat modeling is a strategic and systematic procedure which objectives are to “*identify, communicate, and understand threats*” [129], possible attack scenarios and mitigations. This process is an example of proactiveness in security practices, as stated previously, and should occur in the early stages of application/system development, approach known as “Shift-Left” [130]. By recognizing possible threats to a system, the responsible security team can define mitigating countermeasures to tackle identified threats. Since this system attacker-oriented, i.e., involves the direct interaction with the perpetrator, this threat modelling process is found to be of extreme importance.

In the following sections a threat model of the proposed system is presented, and security and hardening concerns are addressed.

### 4.3.1. Threat Model

Representing a threat model of any system is not a consensual process, although it is recommended that standards should be followed for effectiveness purposes. By standardizing threat modelling processes organizations can easily manage it (update and share) and make it more understandable for the responsible security teams. For this reason, the first approach to the proposed system’s threat modelling is to use OWASP’s recommendation [131] which materializes in a threat diagram with the STRIDE methodology, visible on *Figure 48*.

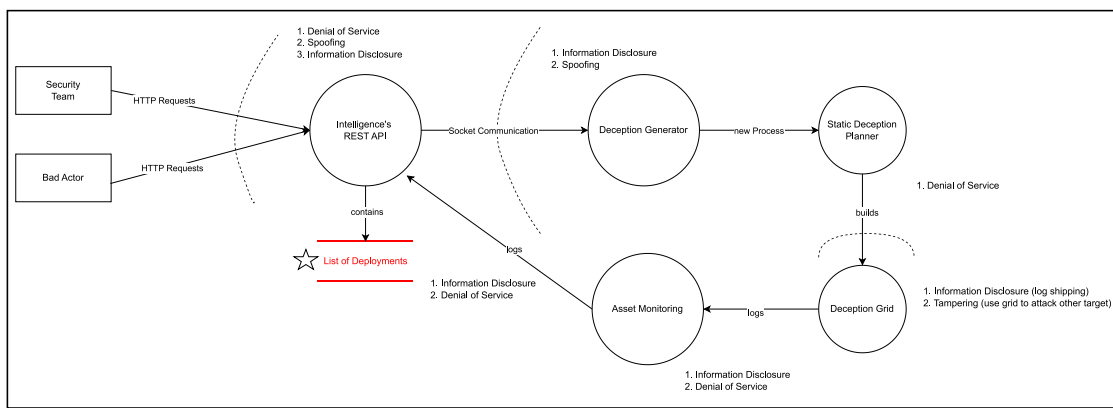


Figure 48 - Proposed system's threat model STRIDE diagram.

Intelligence poses the most considerable threat to the deceptive system due to its interactive nature, i.e., its main objective is to enable communication from the outside to the deceptive system through its REST API. By exposing an API to the outside, this component widens the attack surface for the system and can be an identified vulnerable entry point for a bad actor.

**Denial of Service** – Two types of DoS threats with two different consequences are identified. Firstly, API flooding may take down Intelligence’s interface and making it offline, jeopardizing the security team’s deceptive operations. Secondly, a bad actor might be able to arbitrarily deploy deception grids through research (API) or production (simulating bogus attacks). In this case, triggering numerous times deception grid deployments can be a possible attack. Given the fact that a deception grid consumes several hardware/software resources, uncontrollably deploying several grids might overload an organization’s infrastructure and make deceptive operations unavailable. This aspect is already mitigated by giving Deception Generator the ability of limiting deception grid deployments (Section

4.1.5.). This value however, if tampered with, can eventually pose a serious threat to the availability of the system.

**Information Disclosure** – Inter-component communication is accomplished by a simple unencrypted socket connection. Eavesdropping and sniffing are serious threats to consider, from deception execution to log shipment procedures.

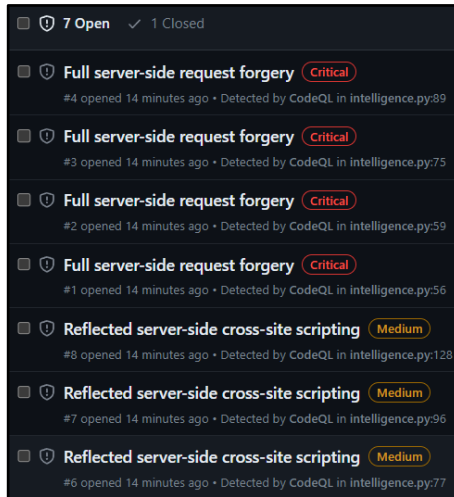
**Spoofing** – As TCP/IP basic communication cannot authenticate the source or destination of a packet, a bad actor can perform man-in-the-middle attacks to leverage this threat. API abuse (through replay attacks or session hijacking) and impersonation (through token forgery or user spoofing) malicious procedures can also compromise the system via spoofing techniques.

**Tampering** – The main tampering concerns comes from when the attacker is interacting with the deception grid. Depending on the deployed assets and in the case of full grid compromising, the attacker can use the deception grid’s resources to attack other targets. This aspect should be mitigated by deploying a tailored honeywall for each deception grid.

Additionally, code vulnerabilities can have far-reaching consequences for the security and integrity of a system. Thus, a code vulnerability assessment was performed via two external services: Snyk [132] and CodeQL [133] (via GitHub Action). These tools statically analyze all code uploaded to a GitHub repository, and list all encountered vulnerabilities which enable the correction of such weaknesses. The identified vulnerabilities of the deceptive system’s code to this date are visible on *Figure 49* (identified by Snyk) and on *Figure 50* (showcased by CodeQL).

File	Issue Type	Line	Score
intelligence.py	Server-Side Request Forgery (SSRF)	Line 56	Score 430
intelligence.py	Server-Side Request Forgery (SSRF)	Line 59	Score 430
intelligence.py	Server-Side Request Forgery (SSRF)	Line 75	Score 430
intelligence.py	Server-Side Request Forgery (SSRF)	Line 89	Score 430
static_deception_planner.py	Server-Side Request Forgery (SSRF)	Line 182	Score 430
static_deception_planner.py	Server-Side Request Forgery (SSRF)	Line 218	Score 430
static_deception_planner.py	Server-Side Request Forgery (SSRF)	Line 297	Score 430
static_deception_planner.py	Server-Side Request Forgery (SSRF)	Line 301	Score 430

Figure 49 - Proposed system's code vulnerability assessment by Snyk.



**Figure 50** - Proposed system's code vulnerability assessment by CodeQL.

Both tools flag the similar vulnerabilities regarding threats of server-side request forgery on Intelligence, although marked with different degrees of severity. However, CodeQL also flags four identified reflected server-side cross-site scripting vulnerabilities in Intelligence (unidentified by Snyk), while Snyk identifies a set of four server-side request forgery vulnerabilities in Static Deception Planner (unidentified by CodeQL). This proves the volatility of results in vulnerability assessment and the importance of utilizing different tools. Furthermore, a dynamic security analysis may also prove to be extremely useful.

Finally, as third-party software and tools are seized by this implementation such as ONOS SDN, Containernet and several other Python libraries, vulnerabilities present on such software are inherited by this system, increasing its attack surface exposing a supply chain attack threat. Organizations that adopt external software must be aware and prepared to mitigate possible threats to its system coming from third-party inherited vulnerabilities.

### 4.3.2. Hardening and Resilience Concerns

Implementing a hardened and resilient solution is out-of-scope for this work, however, to complete the threat modeling process, threat mitigating countermeasures alongside resilience suggestions to the proposed system are presented and duly noted. Some referred measures and mechanisms are later addressed on future work depiction (Section 6.2).

Security hardening considerations are listed below:

### Proactive Cybersecurity tailoring through deception techniques

- Tunneling/encryption mechanisms for inter-component communication.
- Infrastructural barriers to separate and isolate the deceptive system from the deception grid(s).
- Preventing the use of the deception grid to attack outside targets with an appropriate honeywall deployment in each deceptive plan.
- The use of API security measures to protect Intelligence, e.g., through tokens, encryption, using an API gateway and the utilization of quotas and throttling.
- Virtual asset security (containers) to prevent tampering.
- Sanitization of parameters to address vulnerabilities such as the flagged by Snyk (*Figure 49*) and CodeQL (*Figure 50*).

System resilience concerns are the following:

- Asynchronous communication when possible (e.g., message broking).
- Apply the most suitable communication protocol for all given system operations.
- Periodic/reactive backup of data in external databases (e.g., registered deployments and detected attacks).
- Error handling and acknowledgment mechanisms covering all system's code and communications.

# 5. System Validation

This chapter aims to present a validation of the proposed deceptive system and is hereby organized: Section 5.1 catalogs the specifications of the test environment used to conduct the system validation. Section 5.2 dissects several use-case scenarios that prove the value and validity of the proposed system's architecture and implementation.

## 5.1. Test Environment

Proposed system's implementation and testing were carried out on an Ubuntu virtual machine running on the type-2 hypervisor for x86 virtualization VirtualBox [134], particularly its 6.1.38 version. The virtual machine under consideration is, specifically, an Ubuntu 16.04.5 LTS previously configured and referenced by ONOS in their documentation [122]. Regarding virtual machine's configuration, 30 GB of virtual storage were allocated alongside with 3 CPU processors and 9 GB of Random Access Memory (RAM). The VirtualBox hypervisor was installed on an ASUS GL552VX laptop running Windows 10 Home (version 22H2, 10.0.19045) as the underlying operating system. The referred machine has the following hardware specifications:

- x64-bit Processor Architecture
- Intel Core i7-6700HQ CPU with 2.60GHZ, 2592 MHz, 4 Core(s), 8 Logical Processor(s)
- 16 GB of RAM
- 256 GB of external memory on Solid State Drives (SSD)

## 5.2. Use Case Evaluation

Use case evaluation is of extreme importance for any type of software solution because it proves and materializes the efficacy and validity of the any proposed system and its functionalities. Use cases not only outline the various interactions and scenarios that the system is designed to handle but also provide a roadmap for assessing its functionality in real operational situations. In the following sections several operational scenarios that the proposed system's implementation can answer are depicted. These mechanisms were already addressed and discussed on Chapter 3 and Chapter 4, thus, are not extensively re-introduced here. The testing environment involves two virtual machines hosted on VirtualBox: the already discussed ONOS provided Ubuntu and a Kali Linux 2023.2. ONOS Ubuntu is mainly responsible for executing all development scripts (deceptive system), instantiating the deception grid(s) and hosting supplementary services (ONOS, Containernet, Docker, etc.). Kali Linux plays the part of a simulated attacker and the receiving end of the logs (simulating a SIEM appliance). Both



virtual machines belong to the same NAT network and are identified by the following IP addresses: 10.0.2.15 (ONOS) and 10.0.2.11 (Kali).

A performance analysis was pondered, however, its possible outcomes are deemed irrelevant given the fact that all the components of the deceptive system are executed and booted on the same machine, thus, not adding any valuable performance results since the ultimate purpose of the system is to be implemented in a real IT infrastructure.

### 5.2.1. Production Deployment

Deception grid production type of deployment is triggered when an attack is detected and classified. External tools are responsible for such mechanisms, as depicted on Section 3.1.2 h), and in the corresponding implementation are simulated, as described in Section 4.1.1. Attack execution simulation occurs when Kali pings the ONOS virtual machine. Attack detection procedures occur when Sophisticated Detection script detects the launched ping and then extracts its associated IP address and sends it to Auto Attack Analyst (as previously illustrated on *Figure 39*).

In a first stance, the simulated attacker performs a NMAP scan to see available services and discovers none (ignoring the pre-configured SSH service on ONOS VM), as portrayed in *Figure 51*.

```
(kali@kali)-[~/Desktop]
└─$ nmap 10.0.2.15
Starting Nmap 7.93 ( https://nmap.org ) at 2023-08-21 16:08 EDT
Nmap scan report for 10.0.2.15
Host is up (0.0099s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT      STATE SERVICE
22/tcp    open  ssh
```

Figure 51 - Simulated attacker performing NMAP network discovery on ONOS IP address.

To begin the attack launching and detection simulation, the attacker pings the ONOS VM host, as seen on *Figure 52* which is then detected by Sophisticated Detection, and its associated IP address is extracted as visible on *Figure 53*.

```
(kali@kali)-[~/Desktop]
└─$ ping 10.0.2.15
PING 10.0.2.15 (10.0.2.15) 56(84) bytes of data.
64 bytes from 10.0.2.15: icmp_seq=1 ttl=64 time=1.44 ms
64 bytes from 10.0.2.15: icmp_seq=2 ttl=64 time=1.88 ms
64 bytes from 10.0.2.15: icmp_seq=3 ttl=64 time=1.07 ms
^C
```

Figure 52 - Cyber-attack launching simulation via ICMP ping.

## Proactive Cybersecurity tailoring through deception techniques

```
sdn@onos-tutorial:~/Desktop$ sudo python3.9 sophisticated_detection.py
Attack Detected from IP Address: 10.0.2.11
```

Figure 53 - IP address extraction from detected ping.

The associated IP address is then fed to Auto Attack Analyst which selects a certain pre-determined cyber-attack to simulate classification procedures. IP-related information is annexed to the cyber-attack parameters and are sent to Attack Info Adapter, as illustrated on *Figure 54*, and are not in conformity with system-acknowledged data.

```
sdn@onos-tutorial:~/Desktop$ sudo python3.9 auto_attack_analyst.py
Socket server listening on 127.0.0.1:5002
Socket connection established from: ('127.0.0.1', 47370)
Classifying cyber-attack ...
Attack classified with the following proprietary typification:
sqli with IP: 10.0.2.11, details: cleSQL by APT42 with CVE: CVE-2022-30927
Attack Info adapter received the information. '{"status": "success", "message":
"Attack information received successfully"}'
```

Figure 54 - Cyber-attack classification procedures after IP address identification.

Attack Info Adapter translates received “proprietary” cyber-attack information into system-acknowledged data via its associated supporting JSON files. Attack Info Adapter receives, in this demonstration, the following specific data from Auto Attack Analyst:

```
type_of_attack='sqli', attacker_ip='10.0.2.11', attack_details= 'cleSQL' ,
threat_actor= 'APT42', cve='CVE-2022-30927'
```

As portrayed in *Listing 1* and *Listing 2*, the identified “type\_of\_attack” and “attack\_details” parameters’ values are respectively adapted to: "CAPEC\_SQL\_Injection" and "CAPEC\_Command\_Line\_Execution\_through\_SQL\_Injection". After parsing, Attack Info Adapter parallelly dispatches resulting parsed (system-acknowledged) data to Intelligence and Deception Generator, as presented on *Figure 55*.

```
sdn@onos-tutorial:~/Desktop$ sudo python3.9 attack_info_adapter.py
Socket server listening on 127.0.0.1:5013
Socket connection established from: ('127.0.0.1', 53634)
Parsed cyber-attack information:
Attack Type: CAPEC_SQL_Injection
IP Address: 10.0.2.11
Attack Details: CAPEC_Command_Line_Execution_through_SQL_Injection
Threat Group: APT42
CVE: CVE-2022-30927
Sending parsed cyber-attack information to Deception Generator and Intelligence
simultaneously ...
Intelligence received the information: '{"status": "success", "message": "Attack
information received successfully"}'
'Deception Generator received the deployment command'
```

Figure 55 - Attack information parsing and dispatching to Intelligence and Deception Generator.

## Proactive Cybersecurity tailoring through deception techniques

Intelligence receives attack information (*Figure 56*) and makes it available for the security team, while Deception Generator upon its reception (*Figure 57*), builds the deception grid according to the attack parameters.

```
sdn@onos-tutorial:~/Desktop$ sudo python3.9 intelligence.py
Socket server listening on 127.0.0.1:5012
* Serving Flask app 'intelligence'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
Syslog server started. Waiting for logs...
Socket connection established from: ('127.0.0.1', 53636)
Attack triggered and information received : CAPEC SQL Injection 10.0.2.11 CAPEC
Command_Line_Execution_through_SQL_Injection APT42 CVE-2022-30927
```

**Figure 56** - Intelligence's reception of the detected cyber-attack information.

```
root@cb9e3c45e8d8:/containernet# python3 deception_generator.py
Listening on 172.17.0.8:5010
Listening on 172.17.0.8:5014
Listening on 172.17.0.8:5045
Connected by ('172.17.0.1', 43780)
Full data : CAPEC SQL Injection 10.0.2.11 CAPEC_Command_Line_Execution_through_
SQL_Injection APT42 CVE-2022-30927
Type of attack : CAPEC_SQL_Injection
Attacker IP Address : 10.0.2.11
Details of the attack : CAPEC_Command_Line_Execution_through_SQL_Injection
Threat actor : APT42
CVE : CVE-2022-30927
```

**Figure 57** - Deception Generator's reception of the detected cyber-attack information.

Given the identified cyber-attack parameters, the pre-planned deception strategy developed by the security/intelligence team, to defend against it, is selected from the “Playbook 4” set of JSON files (since all attack parameters are available). The specific plan selected in conformity with attack parameters’ values dictates that four hosts are to be deployed:

- A MySQL container.
- A Wordpress container.
- A web API running on an Alpine Linux container.
- A web page running on a Nginx container.

Furthermore, seven OpenFlow-enabled are assigned to the deception grid to connect the virtual hosts in a certain topology which is managed by the plan-designated SDN controller with the “172.17.0.5” IP address. The resulting deception grid is visible via the ONOS visual interface on *Figure 58*. Additionally, a set commands are executed upon container booting and deception grid traffic policies and flows are configured on *Figure 59*. These assets and their base configuration are imported and adapted from [121].

## Proactive Cybersecurity tailoring through deception techniques

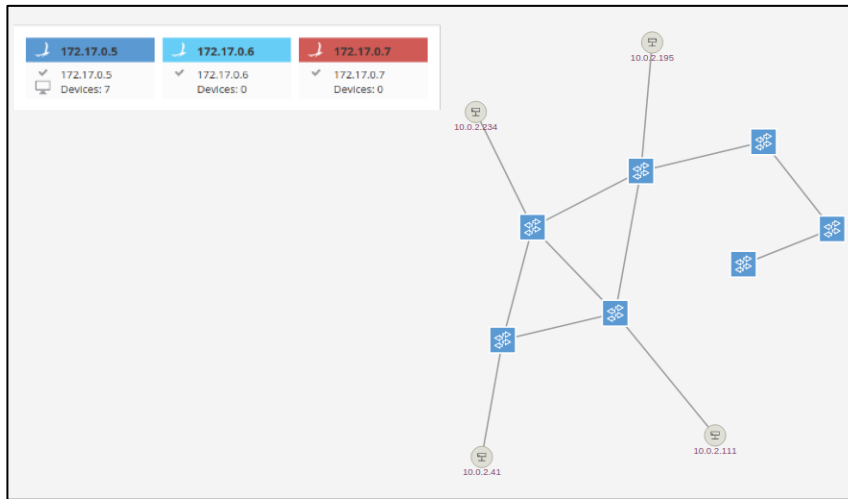


Figure 58 - Resulting deception grid in the context of the ongoing cyber-attack.

```
*** Injecting Commands in Hosts ***
{'cmd': '/entrypoint.sh mysqld &', 'host': 'h1'}
{'cmd': 'python3 send_logs_to_asset_monitoring.py &', 'host': 'h1'}
{'cmd': '/usr/local/bin/docker-entrypoint.sh apache2-foreground &', 'host': 'h2'}
}
{'cmd': 'ifconfig h3-eth1 10.0.2.178', 'host': 'h3'}
{'cmd': 'python3 /usr/src/app/app.py &', 'host': 'h3'}
{'cmd': 'service nginx start &', 'host': 'h4'}
*** Creating Deception Grid Traffic Policies ***
Response from SDN Controller (172.17.0.5) for intent: <Response [201]>
Response from SDN Controller (172.17.0.5) for intent: <Response [201]>
*** Creating Deception Grid Traffic Flows ***
Response from SDN Controller (172.17.0.5) for flow: <Response [200]>
```

Figure 59 - Command, traffic policy and flow injection on deception grid hosts.

To prove host and switch correct establishment on the network, the state of the links between every host and switch is tested via the Containernet’s command-line interface (CLI), as presented on *Figure 60*.

```
containernet> links
h1-eth1<->s20-eth1 (OK OK)
h2-eth1<->s21-eth1 (OK OK)
h3-eth1<->s22-eth1 (OK OK)
h4-eth1<->s23-eth1 (OK OK)
s20-eth2<->s21-eth2 (OK OK)
s20-eth3<->s22-eth2 (OK OK)
s21-eth3<->s22-eth3 (OK OK)
s21-eth4<->s23-eth2 (OK OK)
s22-eth4<->s23-eth3 (OK OK)
s23-eth4<->s24-eth3 (OK OK)
s24-eth4<->s25-eth3 (OK OK)
s25-eth4<->s26-eth3 (OK OK)
```

Figure 60 - Link connection testing of network devices and hosts of the deception grid.

In [121] it is suggested that to test internal service connection and networking, consequence of traffic policy and flow attribution, an HTTP request should be performed to the assigned internal IP address and port of the WEB API service (in this case “10.0.2.111:5000”) from “h4” host. The confirmation that the network (deception grid) is operating as intended is present on *Figure 61*.

## Proactive Cybersecurity tailoring through deception techniques

```
sdn@onos-tutorial:~/Desktop$ docker exec -it mn.h4 bash
root@h4:/# curl 10.0.2.111:5000/; echo;
{"msg": "Hello, World!", "language": "English"}
root@h4:/# curl 10.0.2.111:5000/; echo;
{"msg": "Прывітанне Сусвет!", "language": "Belarussian"}
root@h4:/# curl 10.0.2.111:5000/; echo;
{"msg": "Hallo Welt!", "language": "German"}
root@h4:/# curl 10.0.2.111:5000/; echo;
{"msg": "Ciao mondo!", "language": "Italian"}
```

Figure 61 - Service testing after traffic policy and flow implantation.

As visible on *Figure 58*, virtual asset IP address assignment is congruent with the network of the detected attacker (“10.0.2.11”), thus, the process of conveniently assigning SDN-controlled IP addresses to the deceptive virtual assets of the grid is successful. The resulting booted-up containers are listed on *Figure 62*.

```
sdn@onos-tutorial:~/Desktop$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
dc845075b287   lab-web:latest /bin/bash               8 minutes ago Up 8 minutes 0.0.0.0:8081->80/tcp, 0.0.0.0:32772->8081/tcp
b1a7527d7e7d   lab-api:latest /bin/bash               8 minutes ago Up 8 minutes 0.0.0.0:8888->5000/tcp, 0.0.0.0:32771->8888/tcp
f74e001ad3fc   mywordpress:latest /bin/bash              8 minutes ago Up 8 minutes 0.0.0.0:8080->80/tcp, 0.0.0.0:32776->8080/tcp
9acebd69883e   mysql:latest   /bin/bash               8 minutes ago Up 8 minutes 0.0.0.0:32769->3306/tcp, 0.0.0.0:32768->33060/tcp
```

Figure 62 - Resulting containers that form the hosts on the deception grid.

When the attacker re-runs the NMAP command, three new services are exposed: Wordpress (port 8080), Web API (port 8888) and the Web page (port 8081) as portrayed in *Figure 63*.

```
(kali@kali)-[~/Desktop]
└─$ nmap 10.0.2.15
Starting Nmap 7.93 ( https://nmap.org ) at 2023-08-21 16:15 EDT
Nmap scan report for 10.0.2.15
Host is up (0.0067s latency).
Not shown: 991 closed tcp ports (conn-refused)
PORT      STATE SERVICE
22/tcp    open  ssh
8080/tcp   filtered http-proxy
8081/tcp   filtered blackice-icecap
8888/tcp   filtered sun-answerbook
```

Figure 63 - Service discovery by the attacker after deception grid deployment.

The aforementioned services are running and available for attacker interaction. The web page and the Web API services are visible on *Figure 64* and *Figure 65*, respectively.

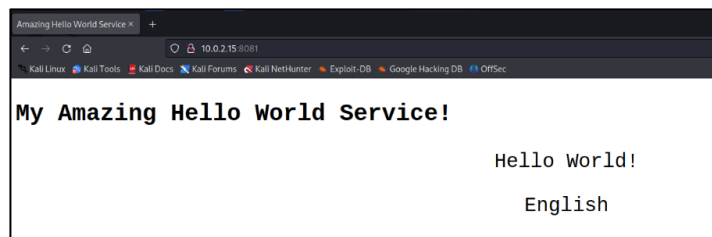


Figure 64 - Deception grid web page service.

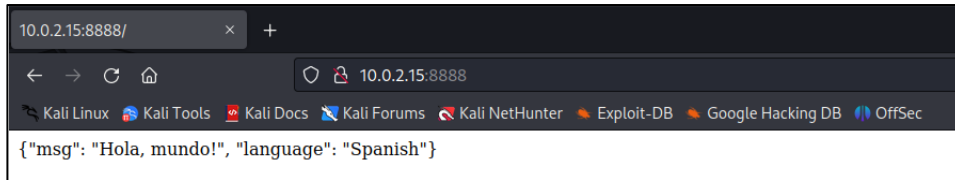


Figure 65 - Deception grid web API service.

This scenario enables that, for any attack supported by the security team i.e., present in the JSON files of deceptive planning and attack information adapters, a deception grid can be prompted with tailored virtual assets, topologies, traffic policies and flows congruent with the defensive strategy responsible for a detected ongoing cyber-attack and its parameters. Generally, the deception grid should mimic the IT infrastructure/system/service that the attacker was originally targeting or a non-existing infrastructure ideal for the identified attack.

### 5.2.2. Research Deployment

Deception grid research type of deployment is prompted by the security team via the Intelligence's REST API. When it is opportune for the security team to launch a deception grid tailored by cyber-attack parameters, they must perform an HTTP POST request to `/api/topology` Intelligence's API endpoint with attack information present on the request's body. In *Figure 66* an example of a request to deploy a deception grid is presented.

```
sdn@onos-tutorial:~/Desktop$ curl -X POST -u onos:rocks -H "Content-Type: application/json" -d '{"attack type": "MITRE_Data_Encoding", "IP": "192.168.200.150/24", "attack details": "MITRE_BADNEWS", "threat actor": "Patchwork", "cve": ""}' http://127.0.0.1:5000/api/topology
{"message": "Deployment command sent successfully for attack type = MITRE_Data_Encoding, ip = 192.168.200.150/24, attack details = MITRE_BADNEWS and threat actor = Patchwork", "status": "success"}
```

Figure 66 - Execution of research deployment procedures.

Upon extraction of the attack parameters transmitted on the request's body, Intelligence sends the same information (already compliant with the system-acknowledged data since it comes directly from the security team) to Deception Generator to begin with the grid deployment procedures (*Figure 67* and *Figure 68*). Given the fact that the supplied information does not include the "CVE" value, "Playbook 3" set of JSON files selects the deceptive plan that tailors the grid.

```
Deployment command and information received : {'attack type': 'MITRE_Data_Encoding', 'IP': '192.168.200.150/24', 'attack details': 'MITRE_BADNEWS', 'threat actor': 'Patchwork', 'cve': '', 'date/time': datetime.datetime(2023, 8, 22, 5, 49, 3, 29868)}
127.0.0.1 - - [22/Aug/2023 05:49:03] "POST /api/topology HTTP/1.1" 200 -
'Deception Generator received the deployment command'
```

Figure 67 - Intelligence recognizing the research deployment request.

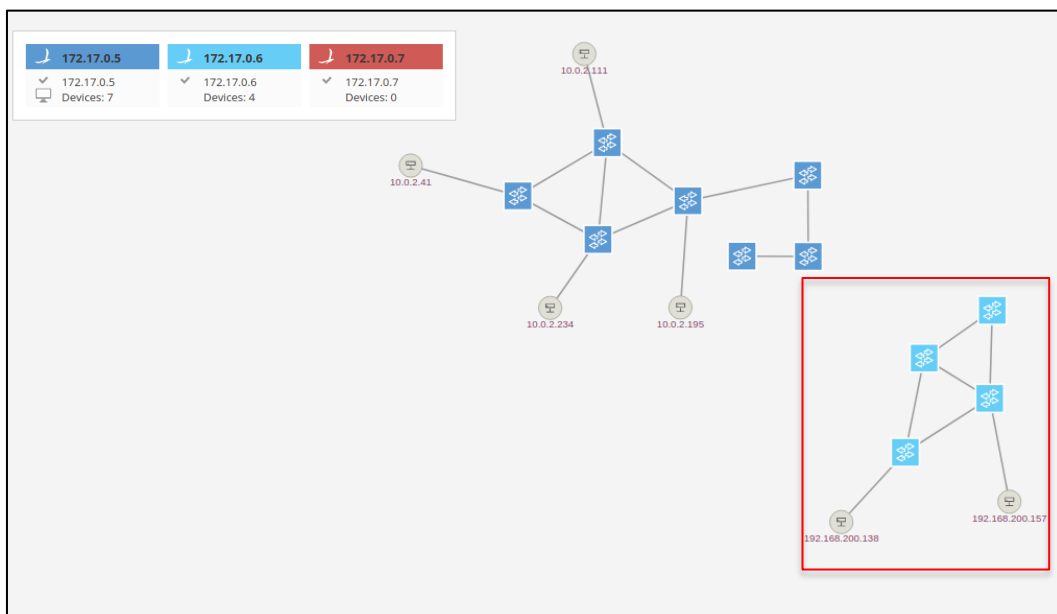
## Proactive Cybersecurity tailoring through deception techniques

```
containernet> Connected by ('172.17.0.1', 56434)
No CVE was identified.
Full data : MITRE Data Encoding 192.168.200.150/24 MITRE_BADNEWS Patchwork
Type of attack : MITRE Data Encoding
Attacker IP Address : 192.168.200.150/24
Details of the attack : MITRE_BADNEWS
Threat actor : Patchwork
CVE :
```

**Figure 68** - Deception Generator receiving attack parameters to begin deployment procedures.

The deceptive plan requires that, in this context, two Ubuntu containers (“ubuntu:trusty”) are deployed, connected to a set of four switches managed by a different SDN controller from the previous deception grid (in this case “172.17.0.6”). Additionally, convenient IP distribution to the virtual assets occurs because the security team also sends the base IP on the request, enabling dynamic host IP assignment in the same network.

This deployment consecutively follows the production one presented on Section 5.2.1, thus, proving that, although deployed in different manners and with differing parameters, two or more grids can co-exist in the management plane of a SDN’s view over the network. Both deployed deception grids are visible on *Figure 69*, where the resulting deception grid of the present (research) deployment is outlined in red. It is duly noted that IP assignment of both hosts in this deployment is different from the production deployment (with base IP address of “192.168.200.150”). The deceptive system enables consecutive fulfillment of deployment requests, production or research, and its management and visual tracking via ONOS SDN controllers and its visual interface.



**Figure 69** - Deception grids resultant of both deployments.

## Proactive Cybersecurity tailoring through deception techniques

The booted-up containers from both deployments are listed on *Figure 70*, as these are running on the same virtual machine and all research deployment services are available for attacker interaction.

```
sdneonos-tutorial:~/Desktop$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
8233da1a6983	ubuntu:trusty	"/bin/bash"	15 minutes ago	Up 15 minutes		mn.h11
d96334123e9c	ubuntu:trusty	"/bin/bash"	15 minutes ago	Up 15 minutes		mn.h10
f9a9091869e2	lab-web:latest	"/bin/bash"	24 minutes ago	Up 24 minutes	0.0.0.0:8081->80/tcp, 0.0.0.0:32777->8081/tcp	mn.h4
55157a66cd67	lab-api:latest	"/bin/bash"	24 minutes ago	Up 24 minutes	0.0.0.0:8888->5000/tcp, 0.0.0.0:32776->8888/tcp	mn.h3
ec9661402f60	mywordpress:latest	"/bin/bash"	24 minutes ago	Up 24 minutes	0.0.0.0:8080->80/tcp, 0.0.0.0:32775->8080/tcp	mn.h2
b2d1eaa4578	mysql:latest	"/bin/bash"	24 minutes ago	Up 24 minutes	0.0.0.0:32774->3306/tcp, 0.0.0.0:32773->33060/tcp	mn.h1

**Figure 70** - Resulting containers that form the hosts on both deception grids.

As discussed on Section 4.1.5, it is plausible for Deception Generator to make operational decisions such as limiting the number of deception grid deployments. To evaluate such functionality, a research deployment request was attempted again. That deployment request was denied and unfulfilled by Deception Generator since the stipulated maximum value of permitted deployments is set to two. Deployment request denial notification is presented on *Figure 71* and its decision communicated to Intelligence, visible on *Figure 72*.

```
containernet> SConnected by ('172.17.0.1', 56452)
No CVE was identified.
Maximum number of deployed deception grids was hit.
```

**Figure 71** - Deployment request denied by Deception Generator.

```
127.0.0.1 - - [22/Aug/2023 05:51:09] "POST /api/topology HTTP/1.1" 200 -
'Maximum number of deployed deception grids was hit.'
```

**Figure 72** - Notification to Intelligence of the deployment blocking.

This type of deployment enables the security team to arbitrarily and strategically deploy tailored deception grid(s) with a set of supplied cyber-attack parameters and a base IP address. The objective of deception grids deployed in a research manner is extensively described in Section 3.1.1.

### 5.2.3. System Information Retrieval

Intelligence's API provides the security team with additional information regarding the deceptive system and the deployed deception grid(s). Following the previously described scenarios where two deception grids were deployed via each deployment procedure, three types of valuable information of the deceptive system can be extracted.



## Proactive Cybersecurity tailoring through deception techniques

Firstly, every deployment request is recorded by Intelligence (research or production). Intelligence’s API `/api/attacks` endpoint returns a list of production deployment requests, providing the associated cyber-attack parameters and a timestamp of the received deployment instruction, as seen on *Figure 73*. Contrastingly, Intelligence’s API `/api/deployments` endpoints returns a list of registered research deployments, as portrayed on *Figure 74*. Both lists contain the performed deployments on Section 5.2.1 and Section 5.2.2. All system components, including the REST API, are running on “127.0.0.1” and not in all interfaces (“0.0.0.0”, as the deception grids’ containers) to simulate that these must not be reachable by an attacker (in this case from the Kali machine on the “10.0.2.0” network), only by the security team.

```
127.0.0.1:5000/api/attacks
[{"attack_details": "CAPEC_Command_Line_Execution_through_SQL_Injection", "attacker_ip": "10.0.2.11", "cve": "CVE-2022-30927", "date/time": "Mon, 21 Aug 2023 20:08:22 GMT", "threat_actor": "APT42", "type_of_attack": "CAPEC_SQL_Injection"}]
```

Figure 73 - Fetching the list of recorded production deployments.

```
127.0.0.1:5000/api/deployments
[{"IP": "192.168.200.150/24", "attack_details": "MITRE_BADNEWS", "attack_type": "MITRE_Data_Encoding", "cve": "", "date/time": "Tue, 22 Aug 2023 05:36:42 GMT", "threat_actor": "Patchwork"}]
```

Figure 74 - Fetching the list of recorded research deployments.

Additionally, an overview of the current switches and hosts that form all currently deployed deception grids is available for visualization for the security team. *Figure 75* showcases the set of hosts and switches deployed as a consequence of the deployments described and executed on Section 5.2.1 and Section 5.2.2.

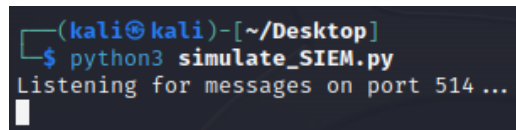
```
127.0.0.1:5000/api/topology/172.17.0.5
{"HOSTS": [{"hosts": [{"configured": false, "id": "76:09:88:88:25:9F/None", "innerVlan": "None", "ipAddresses": ["10.0.2.188"], "locations": [{"elementId": "of:00000000000000017", "port": "1"}, {"mac": "76:09:88:88:25:9F", "outerTpid": "unknown", "vlan": "None"}]}, {"configured": false, "id": "96:AF:55:FA:CF:AF/None", "innerVlan": "None", "ipAddresses": ["192.168.200.3"], "locations": [{"elementId": "of:00000000000000001", "port": "1"}, {"mac": "96:AF:55:FA:CF:AF", "outerTpid": "unknown", "vlan": "None"}]}, {"configured": false, "id": "1E:83:68:07:8C:ED/None", "innerVlan": "None", "ipAddresses": ["10.0.2.137"], "locations": [{"elementId": "of:00000000000000015", "port": "1"}, {"mac": "1E:83:68:07:8C:ED", "outerTpid": "unknown", "vlan": "None"}]}, {"configured": false, "id": "58:EB:34:28:55:3B/None", "innerVlan": "None", "ipAddresses": ["10.0.2.248"], "locations": [{"elementId": "of:00000000000000014", "port": "1"}, {"mac": "58:EB:34:28:55:3B", "outerTpid": "unknown", "vlan": "None"}]}, {"configured": false, "id": "B2:6E:93:BE:06:BC/None", "innerVlan": "None", "ipAddresses": ["10.0.2.230"], "locations": [{"elementId": "of:00000000000000016", "port": "1"}, {"mac": "B2:6E:93:BE:06:BC", "outerTpid": "unknown", "vlan": "None"}]}, {"configured": false, "id": "82:56:54:69:1E:59/None", "innerVlan": "None", "ipAddresses": ["192.168.200.180"], "locations": [{"elementId": "of:00000000000000002", "port": "1"}, {"mac": "82:56:54:69:1E:59", "outerTpid": "unknown", "vlan": "None"}]}], "NETWORK_DEVICES": [{"devices": [{"annotations": [{"channelId": "172.17.0.8:44368", "managementAddress": "172.17.0.8", "protocol": "OF_14", "available": true, "chassisId": "1a", "driver": "ovs", "humanReadableLastUpdate": "connected 14m1s ago", "hw": "Open vSwitch", "id": "of:0000000000000001a", "lastUpdate": "1692708036799", "mfr": "Nicira, Inc.", "role": "MASTER", "serial": "None", "sw": "2.9.8", "type": "SWITCH"}, {"channelId": "172.17.0.8:44366", "managementAddress": "172.17.0.8", "protocol": "OF_14", "available": true, "chassisId": "14", "driver": "ovs", "humanReadableLastUpdate": "connected 14m1s ago", "hw": "Open vSwitch", "id": "of:00000000000000014", "lastUpdate": "1692708036799", "mfr": "Nicira, Inc.", "role": "MASTER", "serial": "None", "sw": "2.9.8", "type": "SWITCH"}, {"channelId": "172.17.0.8:48916", "managementAddress": "172.17.0.8", "protocol": "OF_14", "available": true, "chassisId": "3", "driver": "ovs", "humanReadableLastUpdate": "No Record", "hw": "Open vSwitch", "id": "of:00000000000000003", "lastUpdate": "0", "mfr": "Nicira, Inc.", "role": "NONE", "serial": "None", "sw": "2.9.8", "type": "SWITCH"}, {"channelId": "172.17.0.8:44358", "managementAddress": "172.17.0.8", "protocol": "OF_14", "available": true, "chassisId": "15", "driver": "ovs", "humanReadableLastUpdate": "connected 14m1s ago", "hw": "Open vSwitch", "id": "of:00000000000000015", "lastUpdate": "1692708036799", "mfr": "Nicira, Inc.", "role": "MASTER", "serial": "None", "sw": "2.9.8", "type": "SWITCH"}, {"channelId": "172.17.0.8:48918", "managementAddress": "172.17.0.8", "protocol": "OF_14", "available": true, "chassisId": "4", "driver": "ovs", "humanReadableLastUpdate": "No Record", "hw": "Open vSwitch", "id": "of:00000000000000012", "lastUpdate": "0", "mfr": "Nicira, Inc.", "role": "NONE", "serial": "None", "sw": "2.9.8", "type": "SWITCH"}, {"channelId": "172.17.0.8:48912", "managementAddress": "172.17.0.8", "protocol": "OF_14", "available": true, "chassisId": "11", "driver": "ovs", "humanReadableLastUpdate": "No Record", "hw": "Open vSwitch", "id": "of:00000000000000011", "lastUpdate": "0", "mfr": "Nicira, Inc.", "role": "NONE", "serial": "None", "sw": "2.9.8", "type": "SWITCH"}, {"channelId": "172.17.0.8:48914", "managementAddress": "172.17.0.8", "protocol": "OF_14", "available": true, "chassisId": "2", "driver": "ovs", "humanReadableLastUpdate": "No Record", "hw": "Open vSwitch", "id": "of:00000000000000002", "lastUpdate": "0", "mfr": "Nicira, Inc.", "role": "NONE", "serial": "None", "sw": "2.9.8", "type": "SWITCH"}, {"channelId": "172.17.0.8:44364", "managementAddress": "172.17.0.8", "protocol": "OF_14", "available": true, "chassisId": "18", "driver": "ovs", "humanReadableLastUpdate": "connected 14m1s ago", "hw": "Open vSwitch", "id": "of:00000000000000018", "lastUpdate": "1692708036799", "mfr": "Nicira, Inc.", "role": "MASTER", "serial": "None", "sw": "2.9.8", "type": "SWITCH"}, {"channelId": "172.17.0.8:44360", "managementAddress": "172.17.0.8", "protocol": "OF_14", "available": true, "chassisId": "19", "driver": "ovs", "humanReadableLastUpdate": "connected 14m1s ago", "hw": "Open vSwitch", "id": "of:00000000000000019", "lastUpdate": "1692708036799", "mfr": "Nicira, Inc.", "role": "MASTER", "serial": "None", "sw": "2.9.8", "type": "SWITCH"}, {"channelId": "172.17.0.8:44362", "managementAddress": "172.17.0.8", "protocol": "OF_14", "available": true, "chassisId": "17", "driver": "ovs", "humanReadableLastUpdate": "connected 14m1s ago", "hw": "Open vSwitch", "id": "of:00000000000000017", "lastUpdate": "1692708036799", "mfr": "Nicira, Inc.", "role": "MASTER", "serial": "None", "sw": "2.9.8", "type": "SWITCH"}]}]}
```

Figure 75 - Deception grids topology retrieval.

The availability of this information enables the security team to evaluate the current state of the deceptive system by determining what deployment requests were executed and what is the current devices and hosts (deception grids) managed by the SDN controller.

#### 5.2.4. Deception Grid's Log Observability

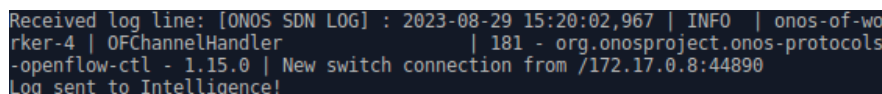
In order to try to profile an attacker by tracking its footsteps and analyzing TTP, discover possible zero-day vulnerabilities and evaluate the in-place deception strategy's effectiveness, deception grid's network and host logs must be supplied to the security team. Network logs are essentially materialized as SDN controller logs while host logs are aggregated in a Python script present in each container (host) which sends them to Asset Monitoring, and this one forwards them to Intelligence, which consecutively dispatches them to a SIEM. A realistic SIEM appliance, in this case QRadar, was attempted to be installed and used, however, its installation requirements greatly surpassed the test environment's capacities. To simulate a SIEM appliance, the Kali Linux virtual machine runs a Python script prepared to receive UDP log connections on port 514, as visible on *Figure 76*.



```
(kali@kali)-[~/Desktop]
└─$ python3 simulate_SIEM.py
Listening for messages on port 514 ...
```

Figure 76 - Kali virtual machine prepared to receive logs, simulating a SIEM.

Following the deception grid's production deployment request and execution described on Section 5.2.1, Asset Monitoring starts to acknowledge logs that are provided by the ONOS SDN controller, as visible on *Figure 77*. Consecutively, it forwards them to Intelligence.



```
Received log line: [ONOS SDN LOG] : 2023-08-29 15:20:02,967 | INFO | onos-of-worcker-4 | OFChannelHandler | 181 - org.onosproject.onos-protocols -openflow-ctl - 1.15.0 | New switch connection from /172.17.0.8:44890
Log sent to Intelligence!
```

Figure 77 - Asset Monitoring receiving ONOS SDN logs and forwarding them to Intelligence.

Intelligence's reception of the referred log is seen on *Figure 78*. This component then dispatches it to the simulated SIEM appliance.

## Proactive Cybersecurity tailoring through deception techniques

```
Received log from 127.0.0.1:  
[ONOS SDN LOG] : 2023-08-29 15:20:02,967 | INFO | onos-of-worker-4 | OFChannelH  
andler | 181 - org.onosproject.onos-protocols-openflow-ctl - 1.1  
5.0 | New switch connection from /172.17.0.8:44890  
Log sent to SIEM!
```

**Figure 78** - Intelligence receiving ONOS SDN logs and dispatching them to the SIEM.

In the last instance of the deceptive system's logging procedure, the SIEM appliance recognizes the incoming logs, supplying them to the security team. Kali receiving the referred log sent by Intelligence is demonstrated on *Figure 79*.

```
Listening for messages on port 514 ...  
Received message: [ONOS SDN LOG] : 2023-08-29 15:20:02,967 | INFO | onos-of-  
worker-4 | OFChannelHandler | 181 - org.onosproject.onos-prot  
ocols-openflow-ctl - 1.15.0 | New switch connection from /172.17.0.8:44890 fr  
om 10.0.2.12:54894
```

**Figure 79** - Simulated SIEM receiving and presenting ONOS SDN logs sent by Intelligence.

In addition to SDN logs, host (container) logs are also provided to and via Asset Monitoring. These are sent from each container once deception grid implantation procedures are complete. Following the referred production deployment's (Section 5.2.1) deception grid and host implantation, hosts start to ship logs to Asset Monitoring. Deception grid's host h4 (web page) is configured to send a reactive logs each time the web page is loaded by the attacker, and associated with that alert, the content (*Figure 64*) of what the attacker is presented with (this host's configurations were already addressed on Section 4.1.3). This aspect aims to entail that the security team should track what the attacker does and visualizes within the deception grid.

Deception grid's host h3 (MySQL) is configured to send "hello world" periodic logs for demonstration purposes. Host logging procedures are similar to SDN, as Asset Monitoring aggregates container shipped logs and sends them to Intelligence, as depicted on *Figure 80*.

```
Received log from 172.17.0.9:  
[mn.h1 - MySQL Log] hello world  
Log sent to Intelligence!  
Received log from 172.17.0.12:  
[mn.h4 - WebPage Log] WebPage loaded!! Attacker has seen the page with language:  
French and message: Bonjour le monde!
```

**Figure 80** - Asset Monitoring aggregating host logs and forwarding them to Intelligence.

Intelligence's reception of the referred logs is represented on *Figure 81*. This component then directly dispatches them to the Kali's Python script which simulates a SIEM appliance.

```
Received log from 127.0.0.1:
[mn.h1 - MySQL Log] hello world
Log sent to SIEM!
Received log from 127.0.0.1:
[mn.h4 - WebPage Log] WebPage loaded!! Attacker has seen the page with language:
French and message: Bonjour le monde!
```

**Figure 81** - Intelligence receiving and dispatching host logs to the SIEM.

Lastly, the simulated SIEM receives referred host logs, as exhibited on *Figure 82*.

```
Listening for messages on port 514 ...
Received message: [mn.h1 - MySQL Log] hello world from 10.0.2.12:51959
Listening for messages on port 514 ...
Received message: [mn.h4 - WebPage Log] WebPage loaded!! Attacker has seen th
e page with language: French and message: Bonjour le monde! from 10.0.2.12:51
917
```

**Figure 82** - Simulated SIEM receiving and presenting host logs sent by Intelligence.

Deceptive grid logging procedures allow deception plan monitoring by the security team, enabling the evaluation of its effectiveness. Additionally, it presents a way to analyze and profile an attacker as a consequence of its interaction with the deception grid. These mechanisms are an essential part of the deception process, especially its monitoring phase, that may lead to dynamic deception grid configuration to tailor the attacker's behavior.

### 5.2.5. Deception Strategy Adjustments

Deception plan monitoring via the implanted logging procedures might indicate that live adjustments should occur to increase deception process's effectiveness and provide better results for the security team. Dynamic deception materializes on adding or removing deception grid hosts on the SDN's controller management plane. Intelligence's API empowers these actions for the security team to apply planned live deceptive modifications via its `/api/hosts` endpoint. For use case evaluation purposes, a scenario where, succeeding the production deployment of Section 5.2.1, the security team decides to apply modifications to the in-place deceptive strategy of a deployed deception grid by adding or removing hosts is presented.

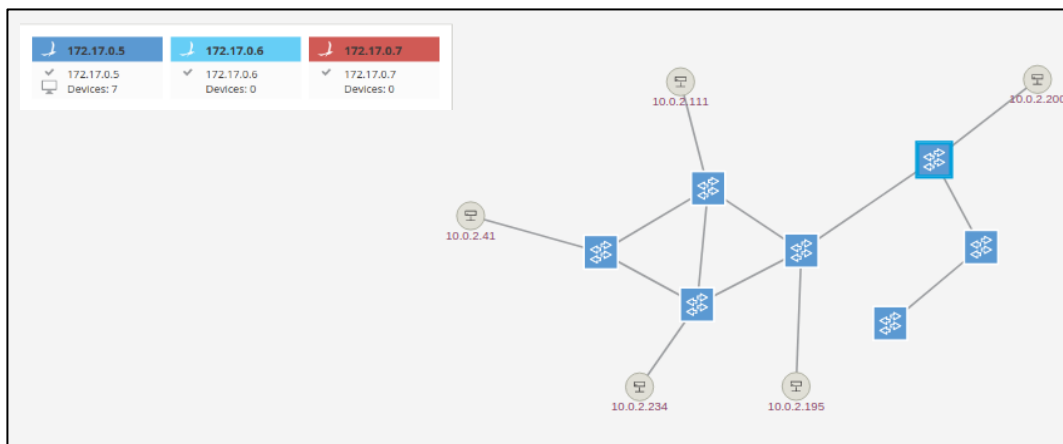
## Proactive Cybersecurity tailoring through deception techniques

To add a new host to a deployed deception grid, a HTTP POST request to `/api/hosts` must be performed by the security team. This request must contain a complete host and associated network information to dynamically add it to a designated live deception grid, as portrayed on *Figure 83*. The deceptive system determines to what deception grid is the host being added through switch identification.

```
sdn@onos-tutorial:~/Desktop$ curl -X POST -u onos:rocks -H "Content-Type: application/json" -d '{"host ip" : "10.0.2.200", "host_mac" : "97:BB:19:27:08:87", "switch name" : "of:00000000000000018", "vlan" : "None", "port" : "2"}' http://172.0.0.1:5000/api/hosts/172.17.0.5
```

**Figure 83** - HTTP request to dynamically add a host to a deception grid.

*Figure 58* showcases the initial state of the deployed deception grid and upon Intelligence’s API request (*Figure 83*) reception and handling, a new host is added to the assigned switch on the designated port and with the attributed IP and MAC addresses, as visible on *Figure 84*.



**Figure 84** - Resulting deception grid after dynamic allocation of new host.

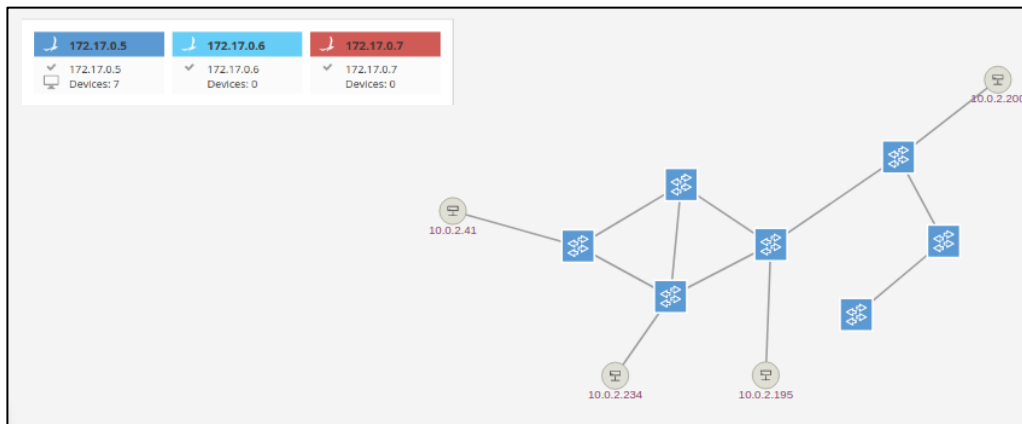
Dynamically deallocation procedures of a deception grid’s host are activated in the same manner as the described allocation. The security team must execute an HTTP DELETE request to `/api/hosts`. Host information must contain the basic host data of its MAC address and VLAN. Additionally, in order to stop and remove the running container (from the data plane), the container id or name must be provided. The security team also decides whether a host is subject to *postmortem* forensics. The example of a request that, in this case, is removing the host responsible for the web API service (h3 with “10.0.2.111” IP address) from the deception grid without *postmortem* is represented on *Figure 85*.

## Proactive Cybersecurity tailoring through deception techniques

```
sdn@onos-tutorial:~/Desktop$ curl -X DELETE -u onos:rocks -H "Content-Type: application/json" -d '{"host_mac": "46:5c:93:2a:af:21", "vlan": "None", "container_id_or_name": "mn.h3"}' "http://127.0.0.1:5000/api/hosts/172.17.0.5"
```

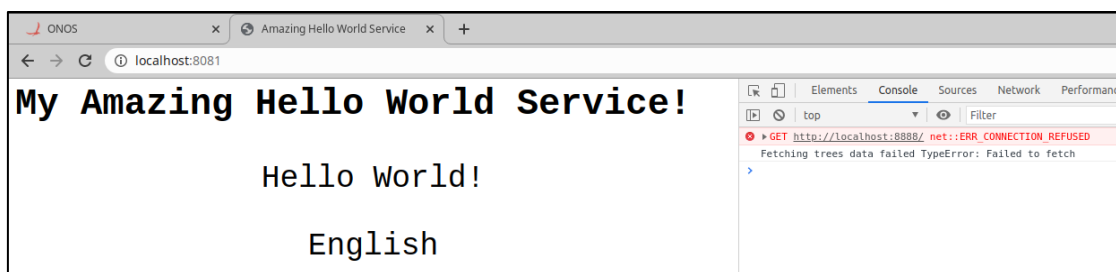
**Figure 85** - HTTP request to dynamically remove a host of a deception grid without *postmortem*.

*Figure 84* showcases the current state of the deception grid and upon Intelligence’s API request (*Figure 85*) reception and handling, the host with “10.0.2.111” IP address and “46:5c:93:2a:af:21” MAC address is deallocated, as shown on *Figure 86*.



**Figure 86** - Resulting deception grid after dynamic deallocation of the web API host.

As a consequence of the host deallocation, the deception grid’s web API service is no longer online and is unavailable for the web page and the attacker to interact with, as visible on *Figure 87*.



**Figure 87** - Web API unavailable after dynamic host removal.

Host deallocation with the possibility *postmortem* procedures involves that the security explicitly informs about that preference on the HTTP DELETE request via the query parameter. Additionally, the name of the file that composes the extracted container filesystem for analysis must be supplied. *Figure 88* showcases a request that instructs the removal of the web page service (h4 with “10.0.2.195” IP address) and extracts its filesystem content.

```
sdn@onos-tutorial:~/Desktop$ curl -X DELETE -u onos:rocks -H "Content-Type: application/json" -d '{"host_mac": "ca:1a:07:ea:69:a8", "vlan": "None", "container_id_or_name": "mn.h4", "output_file": "./web_page_postmortem.tar"}' "http://172.0.0.1:5000/api/hosts/172.17.0.5?postmortem=true"
```

Figure 88 - HTTP request to dynamically remove a host of a deception grid with postmortem.

Figure 86 demonstrates the current state of the deception grid and upon Intelligence’s API request (Figure 88) reception and handling, the host with “10.0.2.195” IP address and “ca:1a:07:ea:69:a8” MAC address is deallocated, as shown on Figure 89.

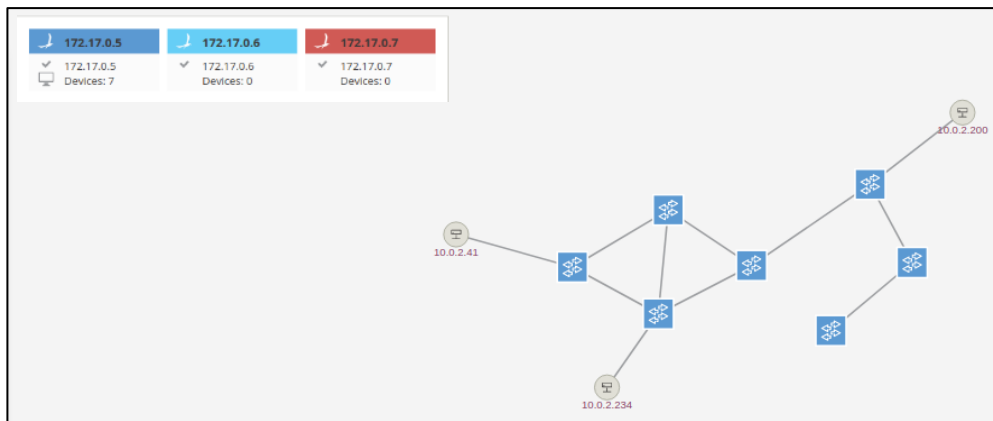


Figure 89 - Resulting deception grid after dynamic deallocation of the web page host.

Intelligence sends a *postmortem* command to Deception Generator which exports the container’s filesystem, stops its execution, and removes it. These actions are illustrated on Deception Generator’s command line in Figure 90.

```
containernet> Connected by ('172.17.0.1', 33070) to perform postmortem host saving.  
Preserving host state and storing of post-mortem forensics...  
Exporting container file to {...} ./web_page_postmortem.tar  
Stopping and removing container...
```

Figure 90 - Deception Generator exporting a container's filesystem for postmortem forensics.

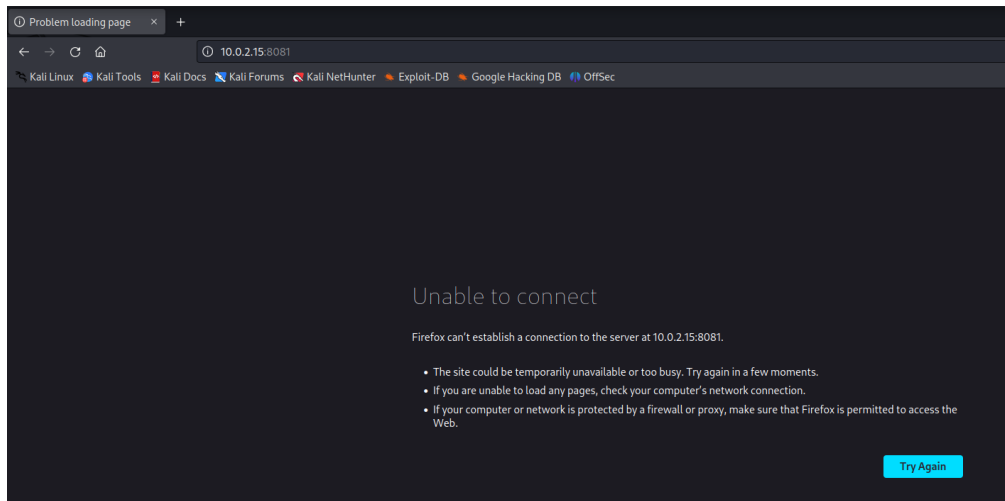
Since the socket connection of *postmortem* forensics is sent to Deception Generator and that this component is running on the Containernet Docker container, the exported “web\_page\_postmortem.tar” file containing the ex-deception grid host container’s filesystem is present on this container’s filesystem, as highlighted in red and portrayed on Figure 91.

## Proactive Cybersecurity tailoring through deception techniques

```
root@3dc2c77dcb6b:/containernet# ls
CONTRIBUTORS      ansible      mn.1
Dockerfile        bin          mnexec
Dockerfile.centos build        mnexec.1
INSTALL           deception_generator.py mnexec.c
LICENSE           dist        plans_config
Makefile          doc         setup.py
README.md         examples    static_deception_planner.py
StandaloneVagrantfile mininet     util
Vagrantfile       mininet.egg-info web_page_postmortem.tar
```

**Figure 91** - Exported filesystem of dynamically removed host with postmortem.

As a consequence of the host deallocation, the deception grid's web page service is no longer online and is unavailable for the attacker to interact with, as visible on *Figure 92*.



**Figure 92** - Deception grid's web page host service unavailability after its dynamic deallocation.

Live and dynamic deception adjustments to deployed deception grids is of extreme importance to increase the deceptive effectiveness. By allocating or deallocating deception assets, the security team can manipulate and tailor an attacker to their own advantage, actively defending and learning the perpetrator's patterns.



# 6. Conclusions and Future Work

In this chapter, main conclusions regarding the elaboration of this document and the development of the associated work are outlined on Section 6.1. Additionally, a description of the future work related to this thesis and corresponding contribution is listed in Section 6.2.

## 6.1. Conclusions

The elaboration of the present document and associated work provided valuable information about the particularities of this thesis' subject. The meticulous examination undertaken throughout the underlying research process has shed light on various nuances and aspects central to deception technology and its interconnection with computer security. This work culminated on a proposed architecture, presentation and discussion of conceptual mechanisms, the development of a proof-of-concept implementation and its validation.

In a first instance, a glance through the definition of what is deception is discussed. The depiction of several proposed definitions by some authors from distinct branches of study was found crucial to understand what deception entails. Acknowledging the spectrum of deceptive techniques in conjunction with the study of proposed taxonomies helped comprehend the variety of methods and ways that deception can be weaponized. Furthermore, the extensive enumeration of deception practical examples on historical and everyday references cemented the diversity of situations in which deception can and should be applied to gain any type of advantage. Following the conceptual overview of deception, the utilization of this instrument in computer security technology was addressed. Firstly, a summary of what computer security is, its purposes and layers, was conducted. Secondly, a comparison between typical defensive security mechanisms and deception technology was then performed. The result evidenced an obvious contrast between them, primarily on their proactive/reactive posture regarding cyber-attack handling where reactive mechanisms were deemed insufficient when acting individually. From this, advantages of adopting deception technology as a complementary procedure in computer security affairs were outlined. Two pillars that support the use of deception in technology were identified. Virtualization and SDN techniques were found to have a vital role in adopting deception into the IT infrastructure for cyber defense purposes.

Diving into the background on deception technology permitted the study of each implementation's technological particularities. Honeypots, honeynets and deception grids/systems were technically disclosed, which prompted the study on how deception is and was technologically implanted. Afterwards, a state-of-the-art analysis was performed, enabling the acquaintance with solutions that have similar objectives as the ones proposed in this work. This analysis revealed a need for a more meticulous approach to deception, highlighting gaps in comprehensive solutions and incomplete architectures, while also acknowledging that the incorporation of certain aspects discussed and implemented on existing solutions is fruitful. A

deception planning methodology considered important to prepare and administer cyber deception strategies was presented. This process included an adversary analysis' phase, which prompted the introduction of Intelligence as a fundamental branch of computer security. The correlation of Intelligence with deception planning is found to be crucial to achieve successful deception plans with premeditative levels of sophistication.

With all the background and related work in consideration, an architecture of a proposed deceptive system was designed to meet the core requirements of successful deceptive cyber defensive procedures in an IT infrastructure. Firstly, two complementary approaches to deception grid deployment were proposed. Both approaches should be adopted by an organization in its cyber defensive efforts and differ on what triggers a deception grid deployment and where should it be placed. Through a modular responsibility approach, system components were drafted and composed with different responsibilities within the deceptive system. Defining each component responsibilities allowed to enumerate all the functionalities that the deceptive system should provide to fulfill flexible and complete deceptive capabilities. Correlated mechanisms found pertinent to deceptive success on both types of deployment were also presented and discussed. A correlation between the proposed system and the different deceptive techniques is established to underline the importance of the theoretical study of deception previously delineated and its application on the deceptive system.

The implementation phase of this work allowed to execute the majority of the discussed conceptual components and mechanisms of the proposed solution and architecture. Each system-implemented component that enables both types of deception grid deployment was extensively dissected and implanted. To ensure the development and implementation of each fragment of the deceptive system, a technological stack was seized given the already discussed enabling pillars of deception technology culminating on the use of ONOS SDN, Contairnernet and Docker technologies, alongside with Python for the development of the implemented components. This technological stack enables the construction of tailored virtual networks of containers (deception grids), that combined with the development of each Python component, forms the deceptive system. Supplementarily to the implementation of each system component, the depiction on how deceptive plans are crafted is addressed. Deceptive planning and deceptive execution (by the system) are considered two different planes, thus, decoupling the deceptive system from the deceptive plans was found crucial. In this context, deceptive plans are materialized on available Contairnernet and SDN functionalities, thus, being naturally dependent on their capabilities. Additionally, threat modelling processes on the implementation

of the proposed deceptive system were performed, which highlighted several security concerns surrounding the system and was found pertinent to acknowledge.

The deceptive system's validation procedures, via use case evaluation, proved that the presented implementation meets and satisfies core deceptive functionalities and requirements of implanting tailored deception grids, i.e., a network of customized virtual assets, topologies and configurations aligned to a pre-planned deceptive strategy that proactively defends against ongoing cyber-attack contexts, manipulating an attacker's behavior to defender's advantage. Non-implemented mechanisms and unanswered concerns are also recognized and are duly depicted on future work.

## 6.2. Future Work

The presented solution has natural limitations of any complex system that was developed from ground up and with stringent time restrictions. As described on Chapter 4, a simulation of detection and classification appliances for implementational and validation purposes was performed. Integration with real external tools/software that detect, classify, and supply the deceptive system with ongoing cyber-attack parameters for deception grid's production deployment is of paramount significance. Additionally, to completely capacitate the system with viable production deployments, the use of reset and redirection enforcing mechanisms from security tools (e.g., Suricata) to force the attacker to the deception grid after attack detection is utmost useful. Regarding deceptive system's additional functionalities, the Contairnernet object tracking of deployed deception grids enables the dynamic allocation of hosts on the physical plane and the possibility of strategically and programmatically deleting already deployed deception grids instead of using Contairnernet's CLI. Regarding deceptive planning, the insertion of a "Honeywall" virtual asset or by assigning SDN ACLs on each deception strategy and corresponding deception grid mitigates security issues of weaponizing a deception grid. Fake traffic generation allows a more authentic attacker's perception on the deception grid and its traffic particularities can help elaborate finer deceptive strategies. In order to interlope and incorporate the proposed system in a real cyber defensive environment, the clarification of important infrastructural aspects is needed alongside with the dispatchment of deception grid logs to a real SIEM appliance is opportune. Finally, the deceptive system must undergo a thorough examination following the security and resilience concerns listed on Section 4.3 which need to be actively addressed in order to fully harden the system.

## 7. References

- [1] Check Point, "Check Point Research: Cyber Attacks Increased 50% Year over Year," Check Point, 10 January 2023. [Online]. Available: <https://blog.checkpoint.com/2022/01/10/check-point-research-cyber-attacks-increased-50-year-over-year/>. [Accessed 13 July 2023].
- [2] Research and Markets, "Deception Technology - Global Market Trajectory & Analytics," Research and Markets, January 2023. [Online]. Available: <https://www.researchandmarkets.com/reports/4804186/deception-technology-global-market-trajectory>. [Accessed 13 July 2023].
- [3] Cambridge English Dictionary, "Deception," [Online]. Available: <https://dictionary.cambridge.org/us/dictionary/english/deception>. [Accessed 1 October 2022].
- [4] S. Tzu, *The art of war*, Filiquarian Pub., 2006.
- [5] A. Evans and K. Lee, "Emergence of Lying in Very Young Children," *Developmental psychology*, vol. 49, January 2013.
- [6] J. D. Monroe, "Deception: Truth and Lies," in *Deception: Theory and Practice*, Monterey, Calhoun, 2012, p. 42.
- [7] J. D. Monroe, "Deception Defined," in *Deception: Theory and Practice*, Monterey, Calhoun, 2012, p. 40.
- [8] D. Daniel and K. Herbig, "Propositions on Military Deception," *Journal of Strategic Studies - J STRATEGIC STUD*, vol. 5, March 1982.
- [9] Psychology Today, "Deception," [Online]. Available: <https://www.psychologytoday.com/us/basics/deception>. [Accessed 12 October 2022].
- [10] J. H. Korn, *Illusions of reality: A history of deception in social psychology*, State University of New York Press, 1997.
- [11] J. D. Monroe, "Deception Defined," in *Deception: Theory and Practice*, Monterey, Calhoun, 2012, p. 41.
- [12] J. B. Bell and B. Whaley, "Cheating and Deception," in *Cheating and Deception*, Routledge, 1991, p. 60.

## Proactive Cybersecurity tailoring through deception techniques

- [13] J. F. Dunnigan and A. A. Nofi, *Victory and Deceit: Dirty Tricks at War*, 1st edition ed., William Morrow & Co, 1995, p. 350.
- [14] N. C. Rowe and H. S. Rothstein, "Two Taxonomies of Deception," July 2004. [Online]. Available: <https://faculty.nps.edu/ncrowe/mildec.htm>. [Accessed 24 November 2022].
- [15] J. D. Monroe, "Interrelationship of Truth, Deception, and Lies," in *Deception: Theory and Practice*, Monterey, Calhoun, 2012, p. 43.
- [16] J. D. Monroe, "TAXONOMY OF DECEPTION," in *Deception: Theory and Practice*, Monterey, Calhoun, 2012, pp. 43-52.
- [17] R. Chisholm and T. Feehan, *The Intent to Deceive*, *The Journal of Philosophy*, 1977.
- [18] S. Gerwehr and R. W. Glenn, *Unweaving the Web: Deception and Adaptation in Future Urban Operations*, RAND Corporation, 2003.
- [19] H. G. Lerner, *The Dance of Deception: Pretending and Truth-Telling in Women's Lives*, HarperCollins, 1993.
- [20] Acalvio Technologies, *Definitive Guide to Cyber Deception 2.0*, 2017.
- [21] Britannica, "Journalism," 13 Jul 2023. [Online]. Available: <https://www.britannica.com/topic/journalism>. [Accessed 14 July 2023].
- [22] Ethical Journalism Network, "Ethical Journalism Network," [Online]. Available: <https://ethicaljournalismnetwork.org/>. [Accessed 28 October 2022].
- [23] National Union of Journalists, "Code of conduct," [Online]. Available: <https://www.nuj.org.uk/about-us/rules-and-guidance/code-of-conduct.html>. [Accessed 28 October 2022].
- [24] BBC News, "The rise and rise of fake news," 6 November 2016. [Online]. Available: <https://www.bbc.com/news/blogs-trending-37846860>. [Accessed 28 October 2022].
- [25] Britannica, "Advertising," 13 June 2023. [Online]. Available: <https://www.britannica.com/topic/advertising>. [Accessed 14 July 2023].
- [26] J. G. Navarro, "Global advertising spending from 2014 to 2022," Statista, 6 January 2023. [Online]. Available: <https://www.statista.com/statistics/273288/advertising-spending-worldwide/>. [Accessed 18 January 2023].
- [27] M. Belleme, "He Spent 25 Years Infiltrating Nazis, the Klan, and Biker Gangs," *Rolling Stone*, 30 January 2022. [Online]. Available: <https://www.rollingstone.com/culture/culture-features/fbi-infiltrator-nazis-kkk-biker-gangs-1280830/>. [Accessed 2 December 2022].

- [28] C. J. Najdowski and C. L. Bonventre, "Deception in the interrogation room," American Psychological Association, May 2014. [Online]. Available: <https://www.apa.org/monitor/2014/05/jn>. [Accessed 2 December 2022].
- [29] D. Easton, *The political system : an inquiry into the state of political science*, New York : Knopf, 1953.
- [30] *Introduction to Sociology: Understanding and Changing the Social World*, University of Minnesota Libraries Publishing, 2016.
- [31] F. Cantú, *The Fingerprints of Fraud: Evidence from Mexico's 1988 Presidential Election*, Cambridge University Press, 2019.
- [32] University of the District of Columbia, "The Importance of Psychology in Today's World," 7 March 2018. [Online]. Available: [https://www.udc.edu/social-udc/2018/03/07/importance\\_psychology\\_todays\\_world/](https://www.udc.edu/social-udc/2018/03/07/importance_psychology_todays_world/). [Accessed 3 December 2022].
- [33] Department of Psychology - The Ohio State University, "What is Psychology?," [Online]. Available: <https://psychology.osu.edu/about/what-psychology>. [Accessed 3 December 2022].
- [34] Health Research Funding, "11 Pros and Cons of Deception in Psychological Research," [Online]. Available: <https://healthresearchfunding.org/11-pros-and-cons-of-deception-in-psychological-research/>. [Accessed 3 December 2022].
- [35] J. J. Mark, "Warfare - Definition," World History Encyclopedia, 2 September 2009. [Online]. Available: <https://www.worldhistory.org/warfare/>. [Accessed 5 December 2022].
- [36] T. E. Peet, "The Legend of the Capture of Joppa and the Story of the Foredoomed Prince. Being a Translation of the Verso of Papyrus Harris 500," *The Journal of Egyptian Archaeology*, vol. 11, pp. 225-229, 1925.
- [37] Homer, *The Odyssey*.
- [38] S. H. Verstappen, *The Thirty-Six strategies*, China Books, 1999.
- [39] D. C. Daniel, *Strategic Military Deception: Pergamon Policy Studies on Security Affairs*, Pergamon , 2013.
- [40] J. J. Myers, "GEORGE WASHINGTON: DEFEATED AT THE BATTLE OF LONG ISLAND," HistoryNet, 12 June 2006. [Online]. Available: <https://www.historynet.com/george-washington-defeated-at-the-battle-of-long-island/?f>. [Accessed 5 December 2022].
- [41] A. Zegart, "George Washington Was a Master of Deception," *The Atlantic*, 25 November 2018. [Online]. Available:

## Proactive Cybersecurity tailoring through deception techniques

<https://www.theatlantic.com/ideas/archive/2018/11/george-washington-was-master-deception/576565/>. [Accessed 5 December 2022].

- [42] S. C. Tucker, *Middle East Conflicts from Ancient Egypt to the 21st Century: An Encyclopedia and Document Collection*, ABC-CLIO, 2019.
- [43] J. Diamond, "Magic in the Desert," *Warfare History Network*, August 2013. [Online]. Available: <https://warfarehistorynetwork.com/article/magic-in-the-desert/>. [Accessed 5 December 2022].
- [44] F. Stech, K. Heckman, P. Hilliard and J. Ballo, "Scientometrics of Deception, Counter-deception, and Deception Detection in Cyber-space.," *PsychNology Journal*, no. 9, pp. 79-122, 2011.
- [45] Britannica, "Computer Security," 17 June 2023. [Online]. Available: <https://www.britannica.com/technology/computer-security>. [Accessed 14 July 2023].
- [46] D. Schatz, R. Bashroush and J. Wall, "Towards a More Representative Definition of Cyber Security," *The Journal of Digital Forensics, Security and Law*, vol. 12, 2017.
- [47] National Institute of Standards and Technology (NIST), "Data Integrity: Identifying and Protecting Assets Against Ransomware and Other Destructive Events," [Online]. Available: <https://www.nccoe.nist.gov/publication/1800-25/VolA/index.html>. [Accessed 23 January 2023].
- [48] R. Miller, *The OSI Model: An Overview*, SANS Institute, 2001.
- [49] P. Rawat, "Common Security Attacks in the OSI Layer Model," *InfosecTrain*, 25 January 2023. [Online]. Available: <https://www.infosectrain.com/blog/common-security-attacks-in-the-osi-layer-model/>. [Accessed 2 February 2023].
- [50] D. Gkoutzamanis, "Defense in Depth – The Layered Approach to Cybersecurity," *The CISO Times*, 17 April 2020. [Online]. Available: <https://cisotimes.com/defense-in-depth-the-layered-approach-to-cybersecurity/>. [Accessed 2 February 2023].
- [51] S. Climer and M. Khan, "What Are The 7 Layers Of Security? A Cybersecurity Report," *Mindsight*, 14 July 2020. [Online]. Available: <https://gomindsight.com/insights/blog/what-are-the-7-layers-of-security/>. [Accessed 2 February 2023].
- [52] D. Antonenko, "Disaster Recovery and Cybersecurity: Integrating Cyber Security and Business Continuity," *BusinessTechWeekly*, 17 February 2022. [Online]. Available: <https://www.businesstechweekly.com/operational-efficiency/business-continuity/disaster-recovery-cybersecurity/>. [Accessed 3 February 2023].



## Proactive Cybersecurity tailoring through deception techniques

- [53] Fortinet, "Threat Modeling," [Online]. Available: <https://www.fortinet.com/resources/cyberglossary/threat-modeling>. [Accessed 5 February 2023].
- [54] IBM, "What is incident response?," [Online]. Available: <https://www.ibm.com/topics/incident-response>. [Accessed 5 February 2023].
- [55] H. Andrea, "Comparison and Differences Between IPS vs IDS vs Firewall vs WAF," Networks Training, [Online]. Available: <https://www.networkstraining.com/firewall-vs-ips-vs-ids-vs-waf/>. [Accessed 6 February 2023].
- [56] C. Stoll, *The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage*, The Bodley Head, 1989.
- [57] B. Cheswick, *An Evening with Berferd In Which a Cracker is Lured, Endured, and Studied*, AT&T Bell Laboratories, 1992.
- [58] M. M. Islam and E. Al-Shaer, "Active Deception Framework: An Extensible Development Environment for Adaptive Cyber Deception," in *2020 IEEE Secure Development (SecDev)*, Atlanta, GA, USA, 2020.
- [59] A. Greenberg, "The Untold Story of the 2018 Olympics Cyberattack, the Most Deceptive Hack in History," *Wired*, 17 October 2019. [Online]. Available: <https://www.wired.com/story/untold-story-2018-olympics-destroyer-cyberattack/>. [Accessed 7 February 2023].
- [60] V. E. Urias, W. M. Stout, L. Liebrock, M. Merza, J. Luc-Watson and C. Grim, "Technologies to enable cyber deception," in *International Carnahan Conference on Security Technology (ICCST)*, Madrid, Spain, 2017.
- [61] IBM, "What is virtualization?," [Online]. Available: <https://www.ibm.com/topics/virtualization>. [Accessed 8 February 2023].
- [62] Kubernetes, "Kubernetes Overview," [Online]. Available: <https://kubernetes.io/docs/concepts/overview/>. [Accessed 9 February 2023].
- [63] E. Nemeth, G. Snyder, T. R. Hein, B. Whaley and D. Mackin, *UNIX and Linux System Administration Handbook*, Addison-Wesley Professional, 5th edition, 2017.
- [64] Docker, "Use containers to Build, Share and Run your applications," [Online]. Available: <https://www.docker.com/resources/what-container/>. [Accessed 9 February 2023].
- [65] Cisco, "Software-Defined Networking," [Online]. Available: <https://www.cisco.com/c/en/us/solutions/software-defined-networking/overview.html>. [Accessed 10 February 2023].

- [66] IBM Cloud Education, "What is Software-Defined Networking (SDN)?," [Online]. Available: <https://www.ibm.com/cloud/blog/software-defined-networking>. [Accessed 10 February 2023].
- [67] M. S. Olimjonovich, "Software Defined Networking: Management of network resources and data flow," in *International Conference on Information Science and Communications Technologies (ICISCT)*, Tashkent, Uzbekistan, 2016.
- [68] Cisco, "Global Networking Trends," 2020.
- [69] L. S. Vailshery, "Software-defined networking (SDN) market size worldwide from 2021 to 2027," Statista, 7 July 2023. [Online]. Available: <https://www.statista.com/statistics/468636/global-sdn-market-size/>. [Accessed 14 July 2023].
- [70] Open Networking Foundation, "Open Network Operating System (ONOS)," Open Networking Foundation, 2023. [Online]. Available: <https://opennetworking.org/onos/>. [Accessed February 2023].
- [71] OpenDaylight Project The Linux Foundation, "Platform Overview," OpenDaylight Project The Linux Foundation, 2023. [Online]. Available: <https://www.opendaylight.org/about/platform-overview>. [Accessed February 2023].
- [72] J. Vilanova, "Deception as a Form of Defense," Cloudtango, Cloud Security Alliance, 1 April 2022. [Online]. Available: <https://cloudsecurityalliance.org/blog/2022/01/04/deception-as-a-form-of-defense/>. [Accessed 15 February 2023].
- [73] V. Nicomette, M. Kaâniche, E. Alata and M. Herrb, "Set-up and deployment of a high-interaction honeypot: experiment and lessons learned," Springer, 2010, pp. 143-157.
- [74] N. Provos, "Developments of the Honeyd Virtual Honeypot," 2004. [Online]. Available: <https://www.honeyd.org/>. [Accessed 24 February 2023].
- [75] P. Baecher, M. Koetter, T. Hoz, M. Dornseif and F. Freiling, "The Nepenthes Platform: An Efficient Approach to Collect Malware," in *Proceedings of the 9th international conference on Recent Advances in Intrusion Detection*, 2006.
- [76] V. Kolesnykov and J. Senkerik, "mysql-honeypotd," [Online]. Available: <https://github.com/sjinks/mysql-honeypotd>. [Accessed 26 February 2023].
- [77] Fortinet, "What Are Honeypots (Computing)?," [Online]. Available: <https://www.fortinet.com/resources/cyberglossary/what-is-honeypot>. [Accessed 16 February 2023].

- [78] D. K. Rahmatullah, S. M. Nasution and F. Azmi, "Implementation of low interaction web server honeypot using cubieboard," in *International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*, 2016.
- [79] J. Zhuge, T. Holz, X. Han, C. Song and W. Zouw, "Collecting Autonomous Spreading Malware Using High-Interaction Honeypots," in *Proceedings of 9th International Conference on Information and Communications Security (ICICS'07)*, Zhengzhou, China, 2007.
- [80] M. Nawrocki, M. Wählisch, T. . C. Schmidt, . C. Keil and . J. Schönfelder, "A Survey on Honeypot Software and Data Analysis," 2016.
- [81] Counter Craft, "What's the Real Difference Between Cyber Deception and Honeypots?," 2023. [Online]. Available: <https://www.countercraftsec.com/blog/whats-real-difference-between-cyber-deception-and-honeypots/>. [Accessed 16 February 2023].
- [82] R. Dahbul, C. Lim and J. Purnama, "Enhancing Honeypot Deception Capability Through Network Service Fingerprinting," *Journal of Physics Conference Series*, 2017.
- [83] D. Fraunholz, S. D. D. Anton, C. Lipps, D. Reti, D. Krohmer, F. Pohl, M. Tammen and H. D. Schotten, *Demystifying Deception Technology: A Survey*, 2018.
- [84] Fortinet, "What is Deception Technology?," [Online]. Available: <https://www.fortinet.com/resources/cyberglossary/what-is-deception-technology>. [Accessed 17 February 2023].
- [85] M. Adeel, A. A. Chaudhry, E. Ahmed, K. Samad and N. M. Shaikh, "Honeynets: An Architectural Overview," in *2005 Student Conference on Engineering Sciences and Technology*, Karachi, Pakistan, 2005.
- [86] W. Fan, Z. Du and D. Fernández, "Taxonomy of honeynet solutions," in *2015 SAI Intelligent Systems Conference (IntelliSys)*, London, UK, 2015.
- [87] P. Sokol, "Legal issues of honeynet's generations," in *Proceedings of the 2014 6th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, 2014.
- [88] The Honeynet Project, "The Honeynet Project," [Online]. Available: <https://www.honeynet.org/>. [Accessed 18 February 2023].
- [89] L. Spitzner, "The Honeynet Project: trapping the hackers," in *IEEE Security & Privacy*, vol. 1, no. 2, 2003.
- [90] A. Gupta and B. B. Gupta, "Honeynettrap: Framework to detect and mitigate ddos attacks using heterogeneous honeynet," in *International Conference on Communication and Signal Processing (ICCSP)*, Chennai, India, 2017.

- [91] W. Zhang, B. Zhang, Y. Zhou, H. He and Z. Ding, "An IoT Honeynet Based on Multiport Honeypots for Capturing IoT Attacks," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 3991-3999, 2020.
- [92] R. Li, M. Zheng, D. Bai and Z. Chen, "SDN Based Intelligent Honeynet Network Model Design and Verification," in *International Conference on Machine Learning and Intelligent Systems Engineering (MLISE)*, Chongqing, China, 2021.
- [93] C.-Y. J. Chiang, Y. M. Gottlieb, S. J. Sugrim, R. Chadha, C. Serban, A. Poylisher, L. M. Marvel and J. Santos, "ACyDS: An adaptive cyber deception system," in *MILCOM 2016 - 2016 IEEE Military Communications Conference*, Baltimore, MD, USA, 2016.
- [94] M. H. Shahriar, M. A. Rahman, N. I. Haque and B. Chowdhury, "DDAF: Deceptive Data Acquisition Framework against Stealthy Attacks in Cyber-Physical Systems," in *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, Madrid, Spain, 2021.
- [95] X. Liu, L. Li, Z. Ma, X. Lin and J. Cao, "Design of APT Attack Defense System Based on Dynamic Deception," in *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*, Chengdu, China, 2019.
- [96] M. . M. Islam, A. Dutta, M. S. I. Sajid, E. Al-Shaer, J. Wei and S. Farhang, "CHIMERA: Autonomous Planning and Orchestration for Malware Deception," in *2021 IEEE Conference on Communications and Network Security (CNS)*, Tempe, AZ, USA, 2021.
- [97] X. Meng, Z. Zhifeng, R. Li and H. Zhang, "An intelligent honeynet architecture based on software defined security," in *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*, Nanjing, China, 2017.
- [98] J. D. Monroe, "Indicators," in *Deception: Theory and Practice*, Monterey, Calhoun, 2012, pp. 61-63.
- [99] R. Godson and J. J. Wirtz, "Strategic Denial and Deception," *International Journal of Intelligence and CounterIntelligence*, pp. 424-437, 2000.
- [100] J. D. Monroe, "COMMUNICATIONS PROCESS OF DECEPTION," in *Deception: Theory and Practice*, Monterey, Calhoun, 2012, p. 61.
- [101] J. D. Monroe, "DECEPTION PROCESS," in *Deception: Theory and Practice*, Monterey, Calhoun, 2012, pp. 72-76.
- [102] Joint Chiefs of Staff , Military Deception - Joint Publication 3-13.4, 2012.
- [103] B. Brown, "Deception Strategy Development and Execution," TrapX Security, San Jose, 2021.

- [104] M. Warner, "Wanted: A Definition of "Intelligence"," *Studies in Intelligence Vol. 46 No. 3*, 2002.
- [105] CTIPS (CREST Threat Intelligence Professionals), "What is Cyber Threat Intelligence and how is it used?," CREST, 2019.
- [106] C. Johnson, L. Badger , D. Waltermire, J. Snyder and C. Skorupka, "Guide to Cyber Threat Information Sharing - NIST Special Publication 800-150," National Institute of Standards and Technology (NIST), 2016.
- [107] MISP project, "Open Source Threat Intelligence and Sharing Platform," 2023. [Online]. Available: <https://www.misp-project.org/>. [Accessed 5 March 2023].
- [108] M. Bilinski, R. Gabrys and J. Mauger, "Optimal Placement of Honeypots for Network Defense," in *International Conference on Decision and Game Theory for Security GameSec 2018: Decision and Game Theory for Security*, 2018.
- [109] R. Píbil, V. Lisý, C. Kiekintveld, B. Bošanský and M. Pěchouček , "Game Theoretic Model of Strategic Honeypot Selection in Computer Networks," in *International Conference on Decision and Game Theory for Security GameSec 2012: Decision and Game Theory for Security*, Berlin, Heidelberg, 2012.
- [110] The MITRE Corporation, "MITRE Engage," The MITRE Corporation, 2022. [Online]. Available: <https://engage.mitre.org/>. [Accessed 7 March 2023].
- [111] F. Dong, L. Wang, X. Nie, F. Shao, H. Wang, D. Li, X. Luo and X. Xiao, "DISTDET: A Cost-Effective Distributed Cyber Threat Detection System," in *In Proceedings of the USENIX Security Symposium (USENIX Security 2023)*, 2023.
- [112] Python Software Foundation, "Welcome to Python.org," [Online]. Available: <https://www.python.org/>. [Accessed 18 July 2023].
- [113] Python Software Foundation, "socket — Low-level networking interface," [Online]. Available: <https://docs.python.org/3/library/socket.html>. [Accessed 22 July 2023].
- [114] The MITRE Corporation, "Groups," 2023. [Online]. Available: <https://attack.mitre.org/groups/>. [Accessed 26 July 2023].
- [115] The MITRE Corporation, "CVE® Program Mission," 2023. [Online]. Available: <https://www.cve.org/>. [Accessed 26 July 2023].
- [116] The MITRE Corporation, "ATT&CK," 2023. [Online]. Available: <https://attack.mitre.org/>. [Accessed 26 July 2023].
- [117] The MITRE Corporation , "Common Attack Pattern Enumerations and Classifications - A Community Resource for Identifying and Understanding Attacks," 2023. [Online]. Available: <https://capec.mitre.org/index.html>. [Accessed 26 July 2023].

- [118] The MITRE Corporation , "ATT&CK Comparison," 2023. [Online]. Available: [https://capec.mitre.org/about/attack\\_comparison.html](https://capec.mitre.org/about/attack_comparison.html). [Accessed 26 July 2023].
- [119] Mininet Project Contributors, "Mininet - An Instant Virtual Network on your Laptop (or other PC)," 2022. [Online]. Available: <https://mininet.org/>. [Accessed 1 August 2023].
- [120] M. Peuster, H. Karl and S. v. Rossem, "MeDICINE: Rapid prototyping of production-ready network services in multi-PoP environments," in *IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Palo Alto, CA, USA, 2016.
- [121] J. L. Coronado, "SDNLab," 2020. [Online]. Available: <https://github.com/jorgelopezcoronado/SDNLab>. [Accessed 2 August 2023].
- [122] T. Vachuska , "Basic ONOS Tutorial," Open Network Operating System, 2019. [Online]. Available: <https://wiki.onosproject.org/display/ONOS/Basic+ONOS+Tutorial>. [Accessed 2 August 2023].
- [123] I. Bedhief, M. Kassar and T. Aguilí , "Empowering SDN-Docker Based Architecture for Internet of Things Heterogeneity," *Journal of Network and Systems Management* , vol. 31, p. 14, 2022.
- [124] R. Fontes, "Mininet-WiFi - Emulation Platform for Software-Defined Wireless Networks," 2022. [Online]. Available: <https://mininet-wifi.github.io/>. [Accessed 2 August 2023].
- [125] J. Hart , "Intent Framework," Open Network Operating System (ONOS), 2016. [Online]. Available: <https://wiki.onosproject.org/display/ONOS/Intent+Framework>. [Accessed 7 August 2023].
- [126] J. Hart, "Flow Rule Subsystem," Open Network Operating System (ONOS), 2016. [Online]. Available: <https://wiki.onosproject.org/display/ONOS/Flow+Rule+Subsystem>. [Accessed 7 August 2023].
- [127] C. Chan, "How to log on external syslog servers," Open Network Operating System (ONOS), 2015. [Online]. Available: <https://wiki.onosproject.org/display/ONOS/How+to+log+on+external+syslog+servers>. [Accessed 16 August 2023].
- [128] Pallets, "Flask - Quickstart," 2010. [Online]. Available: <https://flask.palletsprojects.com/en/2.3.x/quickstart/>. [Accessed 8 August 2023].

## Proactive Cybersecurity tailoring through deception techniques

- [129] V. Drake, "Threat Modeling," Open Web Application Security Project (OWASP), [Online]. Available: [https://owasp.org/www-community/Threat\\_Modeling](https://owasp.org/www-community/Threat_Modeling). [Accessed 14 July 2023].
- [130] R. Seiersen, "A Modern Shift-Left Security Approach," Forbes, January 2021. [Online]. Available: <https://www.forbes.com/sites/forbestechcouncil/2021/01/04/a-modern-shift-left-security-approach/>. [Accessed 14 July 2023].
- [131] Open Web Application Security Project (OWASP), "Threat Modeling Cheat Sheet," 2021. [Online]. Available: [https://cheatsheetsseries.owasp.org/cheatsheets/Threat\\_Modeling\\_Cheat\\_Sheet.html](https://cheatsheetsseries.owasp.org/cheatsheets/Threat_Modeling_Cheat_Sheet.html). [Accessed 10 August 2023].
- [132] Snyk , "Developer-focused, real-time SAST," 2023. [Online]. Available: <https://snyk.io/product/snyk-code/>. [Accessed 10 August 2023].
- [133] Github, "CodeQL," 2023. [Online]. Available: <https://codeql.github.com/>. [Accessed 10 August 2023].
- [134] Oracle, "Welcome to VirtualBox.org!," 2023. [Online]. Available: <https://www.virtualbox.org/>. [Accessed 13 August 2023].