Proactive Cybersecurity tailoring through deception techniques

Luís Guerra[a][0009−0008−4825−6909] and Luís Gonçalves[b][0000−0002−2411−979X]

Instituto Superior de Engenharia de Lisboa, Lisboa, Portugal
[a]a43755@alunos.isel.pt [b]luis.goncalves@isel.pt

Abstract. A proactive approach to cybersecurity can supplement a reactive approach 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 two use cases that demonstrate its utility.

Keywords: Cyber Deception · Deception Grid · Proactive Cybersecurity · Software-Defined Networking · Cyber Warfare · Network Virtualization.

1 Introduction

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 adopted by organizations. Including both panoramas in an enterprise's cyber defensive spectrum diminishes even more the perpetrator's attack success and hardens the technological infrastructure. Examples of measures that align with proactiveness in cybersecurity efforts are penetration testing, threat modeling processes and the adoption of deception technology in cyber warfare. 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 most often lack a comprehensive and complete architecture that responds to all required needs. In order to address and possibly revert the asymmetry present in cybersecurity efforts, the need to possess this kind of systems is imperative.

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 effort, which requires the identification of only 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.

Resorting to the state-of-the-art, 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 the expectation of reaching 7 billion U.S. dollars in 2030 [2]. Amongst cybersecurity landscape, deception technology is often narrowed to honeypots when these 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 versatility, 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. Works whose objectives align with the present are namely Active Deception Framework (ADF) [3], Adaptive Cyber Deception System (ACyDS) [4] and [5].

This paper's motivations are to present and discuss an architectural approach to deception technology with 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. To leverage the inherent benefits of adopting deception technology into an organization's cyber defensive pro-

cesses, the authors propose a deceptive system architecture that addresses cyber asymmetry, 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. In comparison to the referred related work, the proposed system presents a modular and comprehensive architecture that seizes a specific technological stack enabling the implementation of a complete and innovative system.

Therefore, the proposed approach manages to deploy deception in a versatile manner through the inclusion and execution of deception plans that further materialize in a set of specific resources and network topologies that compose a deception grid. Furthermore, in order to obtain crucial information about attacker's TTP, the system is prepared to extract such data due to the implementation of mechanisms that aggregate deception grid's outbound network and host logs. In the case where a plan modification needs to occur, caused by the unsatisfactory effectiveness of the selected deception plan observed by the security team through logging, means of dynamically configuring the deception grid are inplace, enabling flexibility within the deception process. Finally, the system and its associated mechanisms successfully defend the technological infrastructure as the attacker is prompted to engage with the deception grid and off business infrastructure, being then behaviorally tailored to defender's advantage.

The remainder of this paper is organized as follows. Section 2 discusses the proposed deception system's architecture and its particularities. Section 3 showcases how the deceptive system was materialized and what technologies were utilized to support its reference implementation. Section 4 evaluates the proposed system, proving the solution's value and applicability. Finally, Section 5 presents the main work conclusions and outlines future work.

2 System Architecture

The proposed system's architecture aims to leverage the established deception technology's benefits and tackles weaknesses found on honeypots, honeynets and similar solutions, capacitating the system with mechanisms that help peak cyber deceptive capabilities. Through a modular approach that segregates and delegates clear responsibilities to system's components, premeditated and tailored levels of deception are achieved, maximizing cyber defensive capacities for an organization. These aspects culminate on an architecture that follows honeypot's and honeynet's type of logical deployments, placement and purpose depicted on [6] and [7]. 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. 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 in-

formation to the system, shaping the deception grid's topology. It is crucial to emphasize that the utilization of the two aforementioned approaches is not mutually exclusive, as it is highly beneficial to employ them simultaneously in a complementary manner.

2.1 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. A thorough study of the organization's infrastructure needs to take place beforehand, to then, strategically determine where should the deception grid be deployed, consequently optimizing the system's efficiency. The security team may choose to place the grid within infrastructure boundaries or facing the Internet, visible on Fig. 1. When in direct contact with the Internet, the deception grid can extract valuable research information on possible adversaries of the organization, referred to as "investigative". If placed within the infrastructure boundaries, the deception grid can additionally act as an alert, for the security team, and a trap, for the adversary, referred to as "landmine".

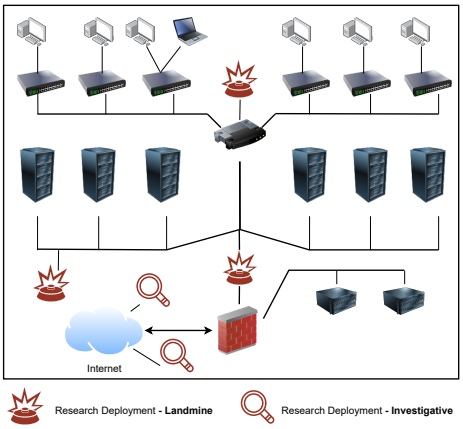

Fig. 1. System's base architecture applicability

The system architecture that supports research deployment is presented in Fig. 2, where several components and its interactions are displayed in a comprehensive way. Components marked green are implemented by the authors while red-marked components are external and need integration to be used by the system. The `Intelligence` component is mainly responsible for enabling communication to and from the deceptive grid and system, acting as gateway. This component provides an Application Programming Interface (API) that exposes a set of operations, that include the dynamic configuration of the deception grid, the output of the available grid's logs and other possible information of interest

(e.g., deception grid's current topology). The manual deployment of the deception grid in the context of a given attack information is also made available for the security team through `Intelligence`'s API. Upon research deployment instruction, this component forwards the inputted attack information to `Deception Generator`. Based on this input, it selects the most appropriated deception plan from a given playbook. Playbook determination relies on the quantity of attack-related information provided. The deception plan is materialized on a set of specific resources and network topology for the deception grid, that after being chosen accordingly, is deployed as a consequence of the interaction between the `Virtual Asset Repository` and `Network of Virtual Assets` and subsequently controlled via the `SDN` component. The `Virtual Asset Repository` contains all the virtual resources needed to implement any given deception strategy stored on `Deception Generator`. `Network of Virtual Assets` component connects the resources and builds a virtual network and the `SDN` controller allows the deception grid's management and configuration (e.g., dynamic configuration and traffic policy management).

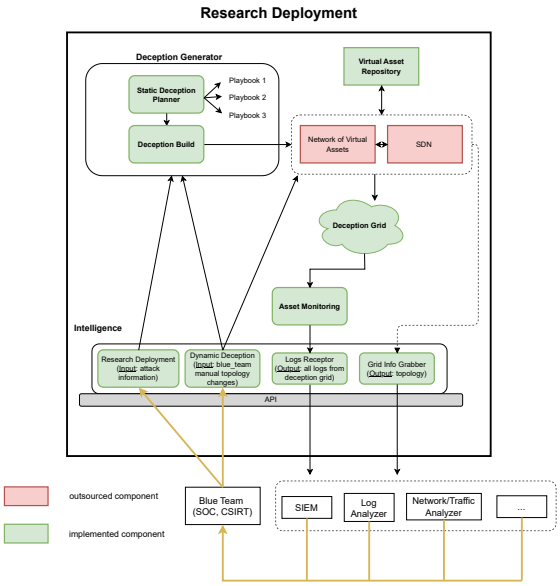

Fig. 2. System's base architecture

Following the deception grid's deployment, mechanisms that enable real-time visualization of grid-related information need to be implemented in order to provide logs. In that sense, `Asset Monitoring` aggregates all incoming host and network-based logs and forwards them to `Intelligence`. The decision of possessing a specific component for this functionality, instead of directly forwarding logs to `Intelligence`, is based on both modular responsibility model and security

concerns. In the event that the deception grid is 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 the `Asset Monitoring` as redundancy, the `Intelligence` component remains undiscovered by the attacker, mitigating possible attacks. Collected logs are then sent to `Intelligence`, exposing them to the security team. This process enables the eventual tracking of the attacker's footprints within the deception grid and as a consequence of their examination, a change or adjustment of the previously executed deception plan may be adequate. Through `Intelligence`'s API, the security team is able to dynamically apply the needed deception grid's modifications. This operational loop aligns with the condensed deception process represented in Fig. 3. This diagram summarizes the extensive deception process presented on [8], adding friendly and adversary analysis (in gray color) as important stages that relate to cyber threat intelligence efforts. These efforts allow a more precise conception of the set of available deception plans, further enabling premeditated deception, which considerably increases the defensive success rate of a deception plan.

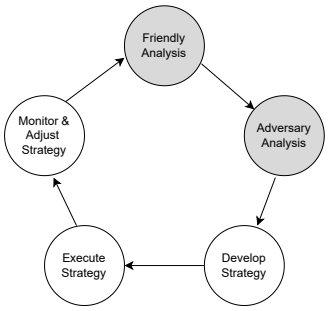

Fig. 3. Overview of deception process, adapted from [9]

2.2 Extended Architecture (Production Deployment)

Production deployment and the underlying extended architecture of the system, supports on-demand deception deployment, i.e., the grid's deployment occurs when an attack is detected, contrastingly to the research type of implantation. 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 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 modeled by the ongoing attack parameters. This type of deployment performs the function of a "shield" and needs to be placed behind the organization's attack detection mechanisms, as portrayed in Fig. 4.

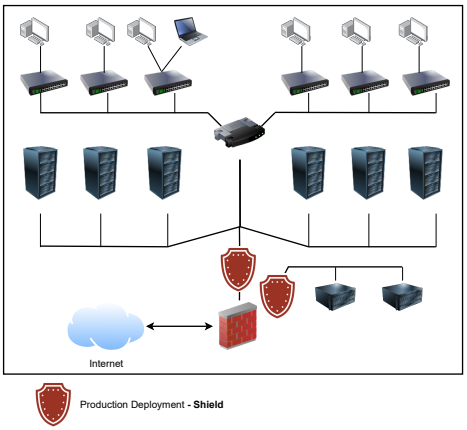

Fig. 4. System's extended architecture applicability

The architecture that supports the production deployment extends the base architecture (Fig. 2) by integrating three new components, as depicted on Fig. 5. External attack detection and automatic classification components are required to be utilized in combination with the system's mechanisms, referred in the diagram as `Sophisticated Detection` and `Auto Attack Analyst`, respectively. Additionally, `Attack Info Adapter` is presented to capacitate the system with functionalities that allow a production deployment. The referred component acts as broker between the attack detection and classification external components and the deceptive system, by adapting or parsing the information (if needed). Thus, abstracting and decoupling the rest of the system from each external component's specifications. When an attack is positively detected and classification processes are complete, information about the attack is sent to `Attack Info Adapter`. If necessary, this component processes received information from the external components into system-acknowledged data. Thenceforth, parsed attack information is dispatched simultaneously 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. The grid's deployment occurs since the `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. The remainder of the system's flow regarding deception plan selection, execution, monitoring and adjustment is performed in a similar manner as the research deployment, as explained on Section 2.1.

3 Implementation

The implementation of this system takes advantage of a number of existing technologies and resources, that combined with the development of the depicted components, compose the proposed deceptive system.

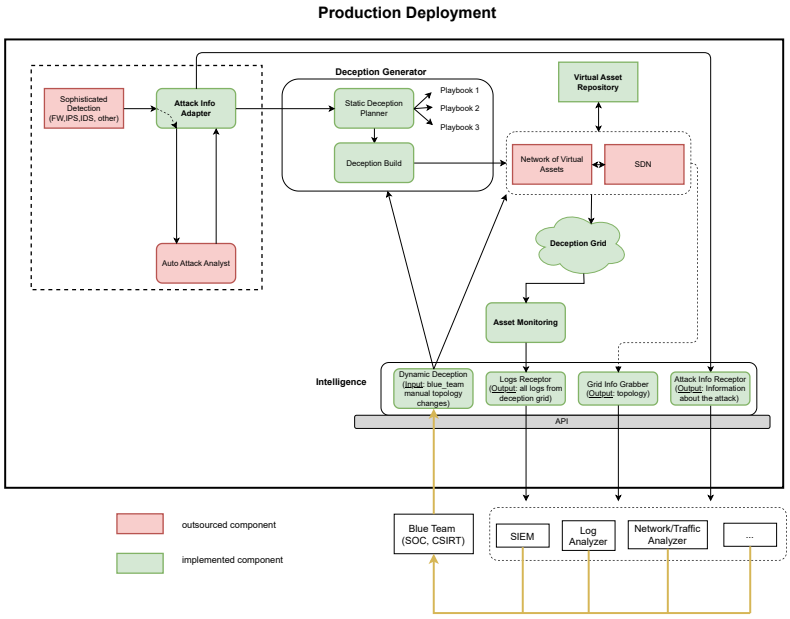

Fig. 5. System's extended architecture

Python programming language is used to support the development of the system's components, as each component is materialized in a Python script and their interaction is accomplished via Transmission Control Protocol (TCP), through `socket` Python library, ensuring the provisionament of an object-oriented low-level networking interface. `Intelligence` component uses Flask [10], a Web Server Gateway Interface (WSGI), that enables the creation of a RESTful API with a set of operations listed on Table 1, allowing outside communication via HTTP request handling (e.g., from and to a security team's client application). `Virtual Asset Repository` refers to a set of virtual artifacts that are available to be used on a deception plan. In this context, Docker containers are preferred over virtual machines due to their faster startup and deployment. Containernet [11] is used to connect these assets in a realistic virtual network. This technology forks Mininet [12], adding the possibility to use Docker containers as hosts in the automated and emulated network topologies for the deception plans.

For deception grid's network management and configuration purposes, SDN technology is used, specifically Open Network Operating System (ONOS) SDN controller [13]. SDN's decoupling of the control plane from the data plane is of advantageous use as it improves network agility and simplified management. This work seizes Mininet's effortless integration with the SDN controller via OpenFlow protocol allowing ONOS, as a centralized point, to rapidly prototype and manage the Mininet's network, that in this scope, is a deception grid. This technological stack was also used on [14] to build a SDN-Docker based infrastructure.

Table 1. Intelligence's REST API operations

HTTP Method	Path	Description
GET	`/attacks`	Returns list of registered attacks.
GET	`/topology`	Returns grid's topology.
POST	`/topology`	Deploys grid following certain parameters.
POST	`/hosts`	Adds virtual host from grid.
DELETE	`/hosts`	Removes virtual host from grid.
GET	`/deployments`	Returns a list of research deployments.

Deception playbooks, invoked by `Deception Generator`, are Python scripts that utilize Mininet's (and Containernet's) network programmatic capabilities, through Python libraries, to automate the assembling of the deception grid as a result of the execution of a deception plan. There are numerous final objectives of a given conceived deception plan including luring or deflecting the attacker. Two factors determine what topology and set of hosts are deployed. Firstly, the quantity of information provided to `Deception Generator` dictates from what playbook a deception plan is chosen. The implementation of this system is prepared to receive the following attack information, from either the security team or the `Attack Info Adapter` component:

$$AttackInformation = (attackType, attackerIp, attackDetails, threatActor) \tag{1}$$

The system requires that at least **attackType** and **attackerIp** are present (mandatory), while the rest being optional. If only the mandatory parameters are received, a deception plan within "Playbook 1" is selected. In the case that, in addition to the mandatory parameters, the **attackDetails** is supplied, "Playbook 2" will determine which plan to execute. Finally, if all parameters are received, "Playbook 3" elects the deceptive plan. It is duly noted that, the rate of deceptive success and precision increases with the number of provided parameters. Specific plan selection depends entirely on each parameters and its combinations. Given a type of attack and optionally its details and possible responsible threat actor, the system picks the most suitable plan for the given tuple, triple or quartet attack information. Attacker IP-related information enables its use on deceptive plan , e.g., to preemptively assign convenient IP addressing to the grid's hosts.

Ultimately, `Asset Monitoring` listens for network and/or host-based logs, aggregates them and forwards parsed logs to `Intelligence`, which exposes them to the security team. Network-based logs are supplied by the SDN controller while host-based need to come from each individual host. In the scope of this work, each Docker Container that can be used in a deception plan needs to be modified to send logs to `Asset Monitoring`'s TCP port. An overview of the system's implementation is visible on Fig. 6.

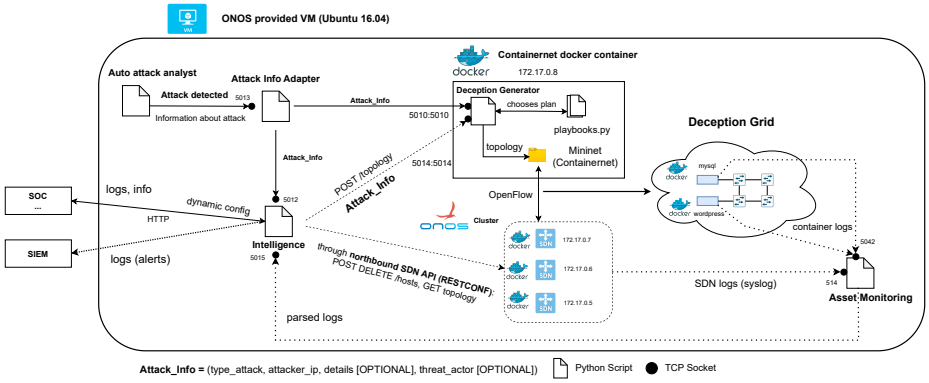

Fig. 6. System's implementation overview

4 System Evaluation

The proposed system is here evaluated through two use cases that prove its cyber defensive utility. Each use case represents a type of deployment discussed on Section 2.

Production deployment, as stated before, depends on detection and automatic attack classification. For demonstration purposes, the interaction between `Auto Attack Analist` and `Attack Info Adapter` components that provides the detected attack information is simulated. In this case, `Attack Info Adapter` receives the information shown on Listing 1.1 through its TCP socket.

Listing 1.1. Information received by `Attack Info Adapter`

```
type_of_attack = 'sql_injection'
attacker_ip = '192.168.200.150/24'
attack_details = 'login_attempt'
threat_actor = 'APT42'
```

`Attack Info Adapter` upon attack information reception, dispatches the information in parallel (supported by Python's `multiprocessing` library) to both `Deception Generator` and `Intelligence`. `Deception Generator`'s deception playbook and plan selection occurs in conformity with the received information. Given the fact that all attack parameters are provided, a deception plan covered in "Playbook 3" is chosen. Plan execution defines hosts and topology for the deception, presented on Fig. 7 and Fig. 8. In this case, `Deception Generator` opts for two hosts that are adequate to defend against the recorded attack: a MySQL and a WordPress Docker containers. Afterwards, a network topology is formed by connecting the hosts with SDN-enabled OpenFlow switches in a particular topology.

Following host and network topology solving, Mininet's/Containernet's virtual network is created by using `addDocker`, `addHost` (if not a container), `addSwitch` and `addLink` functions, provided by Containernet's Python interfaces, on each

```
if attack_type == 'sqli' and attack_details == 'login_attempt' and threat_actor == 'APT42':
    h1_ip= get_random_ip_within_network(attacker_ip)
    h2_ip= get_random_ip_within_network(attacker_ip)
    h1 = {'isDocker':1,'name': 'h1','ip': h1_ip, 'mac':get_random_mac(), 'dimage':'mymysql:latest'}
    h2 = {'isDocker':1,'name': 'h2','ip': h2_ip, 'mac':get_random_mac(), 'dimage':'mywordpress:latest'}
    return [h1,h2]
```

Fig. 7. Deception plan host selection

```
if attack_type == 'sqli' and attack_details == 'login_attempt' and threat_actor == 'APT42':
    link1 = {'device1': net_hosts[0], 'device2': network_switches[0], 'port1': 1, 'port2':1}
    link2 = {'device1': net_hosts[1], 'device2': network_switches[1], 'port1': 1, 'port2':1}
    link3 = {'device1': network_switches[0], 'device2': network_switches[1], 'port1': 2, 'port2':2}
    link4 = {'device1': network_switches[0], 'device2': network_switches[2], 'port1': 3, 'port2':2}
    link5 = {'device1': network_switches[1], 'device2': network_switches[2], 'port1': 3, 'port2':3}
    link6 = {'device1': network_switches[1], 'device2': network_switches[3], 'port1': 4, 'port2':2}
    link7 = {'device1': network_switches[2], 'device2': network_switches[3], 'port1': 4, 'port2':3}
    return [link1,link2,link3,link4,link5,link6,link7]
```

Fig. 8. Deception plan topology selection

host/switch/link assigned by the deception plan, as well as `addController` to connect with ONOS SDN controller. The resulting deception grid deployed for the detected attack (Listing 1.1) is visible via ONOS visual interface, in Fig. 9.

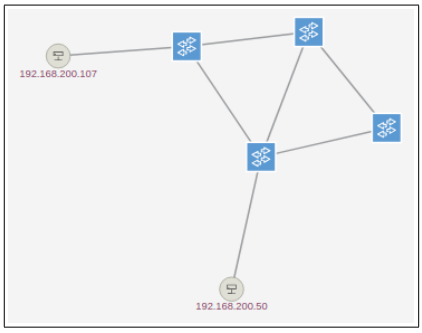

Fig. 9. Deception grid deployed, via production deployment

Research deployment allows the security team to deploy the deception grid according to their inputted attack parameters. For this to be accomplished, a HTTP POST request to **/topology** must be sent to `Intelligence`'s REST API. In this scenario, attack information is transmitted on POST request's body. In order to showcase the adaptability and simultaneous operability of the system, this deployment follows the previous production deployment to show that both deception grids, although deployed in different manners and with differing parameters, co-exist. Simulated request, using command line tool `CURL` is por-

trayed on Fig. 10, while resulting deception grid (outlined in red) composed by two Ubuntu hosts is visible on Fig. 11, alongside the previously deployed grid.

```
sdn@onos-tutorial:~/Desktop$     curl -X POST -u onos:rocks -H "Content-Type: ap
plication/json" -d '{"attack_type":"ubuntu_exploitation","IP":"192.168.200.150/2
4","attack_details":"kernel_exploit","threat_actor":""}' http://127.0.0.1:5000/a
pi/topology
```

Fig. 10. Deploying deception grid following HTTP POST request body attack parameters

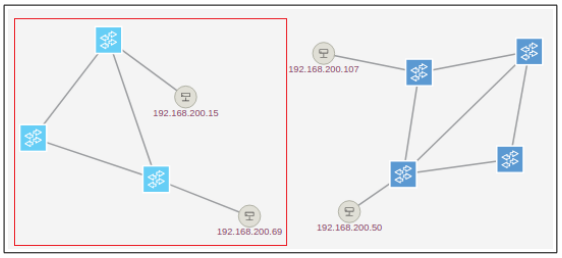

Fig. 11. Deception grid deployed, via research deployment

5 Conclusions and Future Work

In this paper, a deceptive system's architecture and implementation was presented and discussed that can peak deceptive capabilities for organizations. Through SDN, virtualization, and containerization techniques, rapid assembling of the deception plans is possible. Dynamic configuration of the deception grid offers flexibility within the deception process because, when the deployed deceptive strategy is lacking effectiveness, changes may be operated by the security team. The discussed deceptive system, by leveraging depicted technologies and complementing with the development of supporting components, enables deception execution versatility, substantial information collection (for attacker's profiling) and ultimately defends the technological infrastructure of an organization.

As future work, the authors recognize that integration with external tools that detect and automatically classify a cyber attack is of extreme importance. Additionally, to redirect the attacker to the grid, the use of mechanisms that reset the detected malicious connection and replaces it for a grid connection must be implemented. Finally, some considerations regarding the establishment of this system in a real technological infrastructure must be clarified.

References

1. Check Point Research: Cyber Attacks Increased 50% Year over Year, https://blog.checkpoint.com/security/check-point-research-cyber-attacks-increased-50-year-over-year/. Last accessed 14 May 2023
2. Deception Technology: Global Strategic Business Report, https://www.researchandmarkets.com/reports/4804186/deception-technology-global-market-trajectory. Last accessed 13 Jul 2023
3. Islam, M., Al-Shaer, E.: "Active Deception Framework: An Extensible Development Environment for Adaptive Cyber Deception". In: IEEE, 2020 IEEE Secure Development (SecDev), pp. 41-48, Atlanta, GA, USA, 2020. https://doi.org/10.1109/SecDev45635.2020.00023
4. -Y., C., Chiang, J., et al.: "ACyDS: An adaptive cyber deception system". In: IEEE, 2016 IEEE Military Communications Conference, pp. 800-805, Baltimore, MD, USA, 2016. https://doi.org/10.1109/MILCOM.2016.7795427
5. Meng, X., Zhao, Z., et al.: "An intelligent honeynet architecture based on software defined security". In: IEEE, 9th International Conference on Wireless Communications and Signal Processing (WCSP), pp. 1-6, Nanjing, China, 2017. https://doi.org/10.1109/WCSP.2017.8171066
6. Acalvio Technologies,: Definitive Guide to Cyber Deception 2.0., Santa Clara, CA, U.S. (2017)
7. Fan, W., Du, Z., Fernández, D.: "Taxonomy of honeynet solutions". In: IEEE, SAI Intelligent Systems Conference (IntelliSys) 2015, pp. 1002-1009. London, UK, 2015. https://doi.org/10.1109/IntelliSys.2015.7361266
8. Monroe, J.: Deception: Theory and Practice, pp.72-76, Calhoun, Monterey, U.S. (2012)
9. Brown, B.: Deception Strategy Development and Execution, TrapX Security, San Jose, U.S. (2021)
10. Flask, https://flask.palletsprojects.com/en/2.3.x/. Last accessed 21 May 2023
11. M. Peuster, H. Karl, and S. v. Rossem: MeDICINE: Rapid Prototyping of Production-Ready Network Services in Multi-PoP Environments. IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Palo Alto, CA, USA, pp. 148-153. doi: 10.1109/NFV-SDN.2016.7919490. (2016)
12. Mininet: An Instant Virtual Network on your Laptop (or other PC), https://mininet.org/. Last accessed 18 May 2023
13. ONOS(Open Network Operating System), https://opennetworking.org/onos/. Last accessed 18 May 2023
14. Bedhief, I., Kassar, M., Aguili, T. Empowering SDN-Docker Based Architecture for Internet of Things Heterogeneity. J Netw Syst Manage 31, 14 (2023). https://doi.org/10.1007/s10922-022-09702-3