2025 IEEE Conference on Network Function Virtualization and Software-Defined Networking (NFV-SDN) | 978-1-6654-6577-9/25/$31.00 ©2025 IEEE | DOI: 10.1109/NFV-SDN66355.2025.11349579

# Harnessing Software-Defined Network for Cyber Deception Modeling and Analysis

Sukwha Kyung
*Arizona State University*
Tempe, USA
skyung1@asu.edu

Souradip Nath
*Arizona State University*
Tempe, USA
snath8@asu.edu

Jaejong Baek
*Arizona State University*
Tempe, USA
jbaek7@asu.edu

Gail-Joon Ahn
*Arizona State University*
Tempe, USA
gahn@asu.edu

***Abstract*—Recently, cyber deception has been continuously advancing, but implementing and testing advanced deception strategies have become considerably complex, requiring varying degrees of configurations. In addition, evaluation of the efficacy of the deception strategies still heavily relies on manual, ad-hoc analysis. In this paper, we present the design of a generic platform for cyber deception experiments and evaluations by leveraging software-defined networking (SDN). With centralized and fine-grained flow control provided by SDN, we address the challenge of managing various network-level deception mechanisms requiring different configurations. To demonstrate its feedback mechanism and data collection functionalities, we perform case studies using three different deception strategies in a research computing facility with $46$ nodes. The results showed that our platform is agnostic to the types of strategies, monitors network status and changes network policies to react to malicious activities in real time, and is capable of collecting meaningful data required to analyze the efficacy of the deception strategy.**

***Index Terms*—Cyber Deception, Software-Defined Networking, Network Security**

## I. Introduction

Cyber deception involves the deliberate creation of false or misleading information to deceive attackers, divert their attention, and impede their progress within a network, thereby inducing cognitive biases [1]–[3]. To achieve this, defenders employ various techniques such as honeypots [4], fabricated network topologies [5], and deliberately misleading data [6] to trigger alerts. However, despite the advancement of each deception mechanism, configuring, deploying, managing, and measuring the performance of multiple deception mechanisms deployed in a network still remains challenging, due to the difficulties in managing different types of deception mechanisms in a central manner. In addition, lack of flexible and reproducible experimental setups, inadequate feedback mechanisms, and thorough data collection and performance analysis are considered the major roadblocks in utilizing various deception mechanisms [7]–[9]. Currently, most cyber deception research efforts primarily focus on either analyzing the cognitive decision-making process of the players [10]–[12] or evaluating deception models using simple simulations that are designed only for domain-specific scenarios [13], [14] which may not adequately represent complex or diverse environments. As a result, researchers and developers attempting to assess the effectiveness of a cyber deception strategy often face significant difficulties in doing so.

Although their utilization of SDN still remains confined to specific scenarios, SDN has potential to be an entity that governs deception of the entire network with control, observability, and measurability regardless of the topology and functions. In this way, SDN can provide a way to transform ad-hoc deployments into a platform for systematic experimentation and quantitative evaluation. In this paper, we present an SDN-centric configurable cyber deception platform, which is designed to provide a flexible testbed that addresses the lack of central management of deception mechanisms, fine-grained flow control, and real-time data collection capabilities.

The goal of our platform is to enable researchers and developers to assess the efficacy and effectiveness of various cyber deception strategies by designing the real-time data collection capability with flow statistics generation of SDN. Based on the required capabilities, we implement a prototype of the platform leveraging SDN and assess its effectiveness through case studies involving three distinct deception strategies. Overall, the contribution of our work is as follows:

(i) We propose the design of our SDN-based cyber deception analysis platform that addresses key limitations in the current cyber deception research by providing flexible configuration, fine-grained control, and automated data collection. Designed as a strategy-agnostic system, our platform facilitates the deployment of test environments that incorporate a wide range of deception strategies, thereby overcoming the absence of standardized, flexible analysis environments in the field.
(ii) We implement a prototype of the platform by developing an SDN application, which implements deception strategies into practical, reproducible testbed configuration. We then validate the efficacy of our platform through three network-centric case studies simulating a real-world, large-scale corporate network environment. Our studies demonstrate that the platform enables comprehensive evaluation of deception strategies and provides key advantages in data collection and test environment configuration capabilities. We also measured the performance of the platform by deploying it in a research computing facility. Quantitative results further highlight the effectiveness of the platform as a platform for advancing cyber deception research.

## II. Background

**Challenges for deception platform:** Implementing a robust deception platform confronts a number of challenges mainly due to the limitations of traditional networking paradigms. First, the static, device-centric nature of conventional network configuration imposes severe operational overhead. Every new deceptive service or configuration must be manually provisioned on individual nodes, increasing the likelihood of misconfiguration and limiting the speed at which deception policies can be rolled out or revised. Moreover, the reliance on coarse-grained controls such as VLANs or access control lists tied to subnets or ports cannot distinguish threat behaviors at fine granularity.

Second challenge arises from the lack of real-time, unified telemetry in legacy deployments. Without centralized visibility into per-flow statistics, defenders lack timely feedback on how adversaries interact with deceptive hosts. Current approaches (e.g., SPAN ports and packet captures) are often siloed, introduce latency, and demand significant post-processing. Such approaches impede the ability to close the experimental loop required for adaptive deception, where hypotheses about attacker behavior must be tested, measured, and refined under controlled conditions.

Third, the operational agility and scalability of deception systems are severely constrained in heterogeneous, large-scale environments. As deceptive meshes expand to encompass multiple honeypots, virtualized decoys, and mimic services across physical, virtual, and cloud domains, coordinating configuration changes across different management interfaces becomes untenable and cumbersome. Moreover, static deception deployments quickly become stale and can be fingerprinted as attackers continuously evolve their tactics. Without a mechanism to rapidly reconfigure or rotate deceptive elements, defenders cannot measure and assess how a given strategy is effective. Finally, the coordination of deception logic with existing network policies introduces additional complexity. Routing, quality-of-service, and security policies must all coexist with deceptive redirections and mirroring rules, yet traditional systems lack provision of centralized policy verification or conflict resolution. In summary, the limitations of manual management, coarse controls, fragmented visibility, and poor agility and policy coordination present barriers to deploying rigorous and measurable cyber deception strategies.

**Deception strategies:** In our work, we implement three different strategies for our case studies [15]–[17]. Although many strategies have been developed, these strategies define tangible, and thus implementable, deception mechanisms clearly (e.g., high or low interaction honeypots, port reconfiguration, and flow redirection), while other existing works utilizes highly abstract or unimplementable deception, such as signals without specifying the context, partially disclosing defender's strategy to the adversary, or even allowing the attacker to take control of certain hosts within the network. Therefore, demonstrating the capability of our work by implementing these strategies provides a solid foundation, along with the potential impact that our work can bring to the field of cyber deception.

The first strategy [15] introduces a honeypot selection strategy which models a situation where an attacker decides which server in a network to attack. The defender deploys a set of honeypots to maximize the probability that the attacker chooses a honeypot instead of a real server to attack. The authors represented the configuration of the network as a vector of values where each value reflects the importance of a real server and its associated asset. In this strategy, the defender knows the value of each server in the network. The second strategy [16] utilizes deception in a sequential setting. More specifically, it modeled interactions between the attacker and the defender as a partially observable stochastic game utilizing a Markov-decision making model. The defender can know the true state of the game while the attacker can only have partial information. This is implemented by leveraging flow statistics collection capability of SDN and by utilizing an intrusion detection system (IDS) to capture the actions of adversary in the network. Finally, the third strategy [17] seeks to minimize damage incurred by attacker. The strategy proposes and evaluates a cyber deception system to defend virtualized wireless networks such as those used in IoT and CPS environments. The core idea is to dynamically divert attackers from legitimate network services to a deception mobile virtual network operators created on demand by an SDN controller. To do so, the strategy utilized isolation of traffic to a subnet of deceptive hosts. In our work, we leverage the traffic engineering capability of SDN to isolate malicious traffic and intentionally redirect it to a subset of deceptive hosts with intentional false information and configurations.

## III. Design Overview

**Design Criteria:** Based on the deficiencies in the current cyber deception research, we define four essential criteria that guide the design principles of our platform as follows:
(**C1**) *Correctness* involves the capability to deploy and manage the deceptive network environment as intended. Correctness should also guarantee that any changes in network status made by deception strategy are implemented correctly. For example, suppose a strategy includes dynamic modifications, such as adding or removing subnets via gateway changes. It requires the testbed to detect any adversarial actions that trigger corresponding changes and subsequently adjust the network resources to redirect network flows to a target subnet.
(**C2**) *Adaptiveness* indicates the testbed should not be limited to a specific scenario, environment or deception mechanism. It should be versatile enough to accommodate a wide range of strategies, regardless of the particular deception mechanisms employed. In other words, the testbed should be environment-agnostic, capable of integrating various strategies.
(**C3**) *Measurability* pertains to the ability to collect relevant data effectively. Data collection processes often rely on manual and fragmented tasks that require significant human involvement. An automated, centralized data collection mechanism is

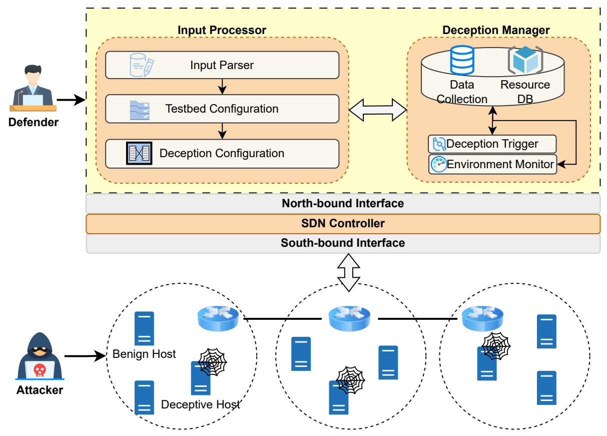

Fig. 1: Architecture of Proposed SDN-Based Deception Platform.

mandatory to correctly and comprehensively capture all the elements that can quantify the results of experiments.

**Architecture:** The platform is implemented as a collection of virtual functions interfacing with an SDN application to manage the deception mechanisms and strategies. The main components of the platform consist of *Input Processor* and *Deception Manager* functions (Figure 1). When a user inputs the environment object parameters along with specific deception mechanisms and triggering events, *Input Processor* parses the input and interfaces with the *Resource Database* to prepare and reserve resources for deception mechanisms to be deployed. The resource consists of virtual machine (VM) images of available deceptive systems implementing various deception mechanisms, virtual switches, and pools of IP addresses. The deception mechanisms include low and high interaction honeypots for various network services (e.g., ssh, email, web servers, and databases), along with any custom-made VMs added by the defender. Then, *Testbed Configuration* and *Deception Configuration* modules are called sequentially to deploy the test network environment and deception mechanisms as configured.

The input also allows the defender to assign priority scores to each asset in the network, where a set of assets consists of any valuable resources that should be protected (systems, links, services, etc). The importance of each asset is determined by its priority score, which reflects a defender's perspective. These scores can be utilized by different strategies. For example, the placement of honeypots can be determined by prioritizing the assets that require high (or low) protection. Another example is setting a specific goal of a strategy (e.g., minimize or maximize the compromise of the assets, etc) that can also be incorporated to utilize the priority scores.

After the configuration process, the test environment is deployed, upon which *Deception Manager* starts monitoring, collecting, and storing system events and network status from the deployed environment. The collected data is used for analysis of the effectiveness of the given deception strategy and provided to *Deception Trigger* module that changes network status or flow policies based on the event defined by the defender to implement strategy. *Deception Trigger* module continuously monitors the host events and network flows to detect any triggering event defined by the defender and change network status to implement dynamic deception strategies. The triggering events consist of a set of pairs, each of which contains either IP addresses to detect the flow between two nodes representing unauthorized access to a target node, or ports for abnormal or unauthorized access to a service port. The defender can provide these triggering events as an input along with the corresponding modification that follows in the case of an event detection. For example, the platform can deploy new flow rules via SDN controller that simply block the traffic, changes the configuration of the node (e.g., change in IP address or port number), or redirects the flow to a deceptive host or subnet. Such real-time monitoring and updating capabilities can help fine-tune and improve the performance of a target deception strategy under experiment, considering the dynamics and evolution of engagement with adversaries. In addition, this feedback feature is particularly valuable in accommodating dynamic deception strategies, where it is beneficial for the user (from the defender's perspective) to monitor each action taken by the adversaries and observe the corresponding changes in the test environment.

**Implementation:** With centralized visibility and flow control provided by SDN, we implemented the prototype of our platform with multiple deception mechanisms and data collection capability. This enables users to select and utilize the most suitable deception mechanisms according to their specific needs and strategies they want to test. The platform also includes support for automated data collection and monitoring in a centralized manner. To that end, we leveraged ELK stack [18] and ONOS SDN controller [19]. In case any changes occur in the test environment, they are monitored and detected by *Environment Monitor* leveraging SNMP or NetFlow protocol. Depending on the strategy, it may trigger a report of these state changes through RESTful API to the *Deception Trigger* module. The module, in response, deploys its corresponding strategic action through dynamic scaling of resources and configuration changes to impede, delay, or confuse the attacker. After the experiment is completed, the network is decomposed, freeing the resources and listing them as available resources in *Resource Database*.

**Example of a strategy implementation:** To illustrate an example of how a strategy deployed can react to the environment change, we adopted a case of *Flow Manipulation* strategy [17]. This particular strategy is implemented by leveraging *Packet-In* event captured by ONOS SDN controller to detect triggering events while flow statistics is continuously polled by the controller from data plane switches to monitor and collect flow-level metrics. This is implemented by retrieving flow information from the data plane via RESTful API in real-time. If it detects malicious traffic, ONOS generates traffic redirection rules. The generated policies are pushed to the data

**Algorithm 1** Adaptive Flow Redirection

```
1: procedure FlowRedirection
2:     Initialize honeypots $H = \{h_1, h_2, ..., h_k\}$
3:     Initialize triggering event $T = \{t_1, t_2, ..., t_n\}$
4:     Initialize redirection rules $R = \{r_1, r_2, ..., r_m\}$
5:     while monitoring the network traffic do
6:         for each incoming packet $p$ do
7:             if $p \subseteq T$ then
8:                 Spawn a subset of honeypots $H'$
9:                 Apply $R'$ to direct traffic to $H'$
10:            else
11:                Continue monitoring the next packet
12:            end if
13:        end for
14:    end while
15: end procedure
```

plane using *Flow-Mod* messages [20]. Another approach to forward packets to a honeypot can be using an SDN proxy designed to select appropriate honeypots depending on the attacker's action [21]. In this way, traffic is redirected to a designated honeypot or subnetwork of honeypots.

In this strategy, any attempt to scan, gain unauthorized access (e.g., by sending a large number of packets for a given time window) is detected by *Environment Monitor*. Then, the *Detection Trigger* function generates corresponding flow rules ($R'$) that are pre-defined for each matching triggering event. It also selects the appropriate subset of honeypots ($H'$) among the entire registered honeypots ($H$). Algorithm 1 illustrates the redirection strategy. Once a packet ($p$) or a set of packets matches the triggering event defined in $T$, then a corresponding redirection rule $R'$ is applied, directing the flow to a subnet of honeypot $H'$. The registry of honeypots (including other types of deception mechanisms) is managed by *Deception Trigger* module and it can select the honeypots among the deployed ones or spawn them in real-time before pushing the new flow rules.

## IV. Evaluation

We now demonstrate that our work addresses the identified challenges by evaluating how it meets the design criteria (C1–C3) outlined in Section III. Specifically, we verify C1 (*Correctness*) and C3 (*Measurability*) by examining the correctness of deployed environments and the statistics generated by our platform, along with its utility in effectiveness analysis for deception strategies. C2 (*Adaptiveness*) is to verify that our platform is not confined to a certain scenario or environment. We demonstrate each strategy, which requires distinct configurations, is deployed by our platform, thereby showing its flexibility and adaptability. We selected the three strategies because they are well-established exemplars of the deception models and clearly specified concrete implementation methods [22]–[24]. We labeled the datasets from each strategy $D1$ [17], $D2$ [16], and $D3$ [15], respectively.

**Experiment environment:** Our experiment network implemented by the platform consisted of $46$ virtual hosts running Microsoft Windows and Linux operating systems. $24$ hosts were deployed with Windows XP, 7, or 10, while the remaining $22$ hosts had various Linux systems including Ubuntu (12.04, 14.04, 18.04, and 20.04), Fedora (22, 28), and CentOS (ver.7.7-1908). The network services deployed in the environment included web servers, database, ftp, shared file systems, and email servers. Snort [25] was also deployed as an IDS and a part of deception strategy, especially for the implementation of the strategy for $D2$ and $D3$. Overall, our platform simulates an existing corporate network as closely as possible while collecting data. In this way, we addressed the limitation of oversimplification of experimental environments (a network with only a few nodes) [14], [26].

### A. Correctness & Adaptiveness

Even though our platform can automate the deployment of deceptive networks, correctness of their configuration and management should be proven. In our evaluation, the correctness is demonstrated through the consistency analysis [27]. Specifically, we checked the consistency between conceptual deceptive network topology and the actual network deployment in terms of the number of nodes, edges, switches, along with the intended placement and connection of these elements by checking each neighboring nodes. Also, change of network status and modification of flow rules based on the deception triggering events should be correct to guarantee correct, automated, and dynamic deception strategy implementation. For adaptiveness for different types of deception, we deployed the three strategies discussed in Section II and attempted to demonstrate the consistency and correctness of deployment by our platform so that it can correctly manage different types of strategies. To that end, we conducted both empirical and consistency analysis to prove that our platform can interpret, implement, and deploy the deceptive network environments as intended throughout multiple trials. In this way, we attempt to demonstrate that the platform supports not only correctness but also reproducibility.

We first identified the components of the intended outcome as defined in our initial design of strategy. We then checked if the actual deployment by the platform aligned with our design. The components examined include the number of *nodes*, *edges*, *switches*, *neighboring nodes* of changed hosts (if any), and how identical the *entries of flow rule table* are. In case where a node was newly spawned or relocated to a different position, we counted its neighboring nodes and verified their assigned IDs. These neighbors served as components evaluated by the strategies that dynamically alter the network topology. The number of neighboring nodes is counted for any node that is newly spawned or relocated to different location in the network topology.

We tested five different topologies with the three deception strategies, generating in total 15 different test cases, and checked the correctness of the deployment. We performed the same deployment 30 times to ensure consistent deployment. The results showed that our platform deploys the network as intended, except that it generated 16 more flow rules. This was

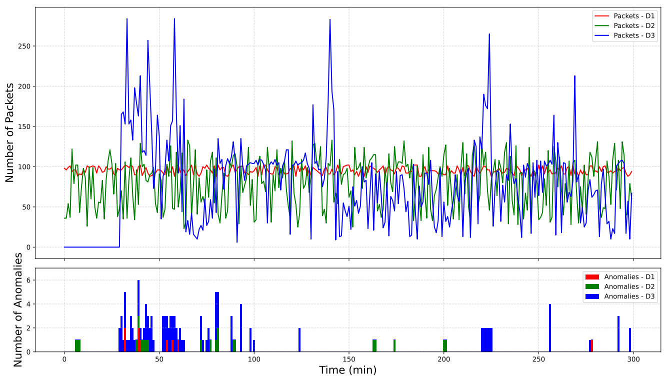

Fig. 2: Data Collected By the Deception Platform (*Packets*, *Timestamp*, and *Anomalies*)

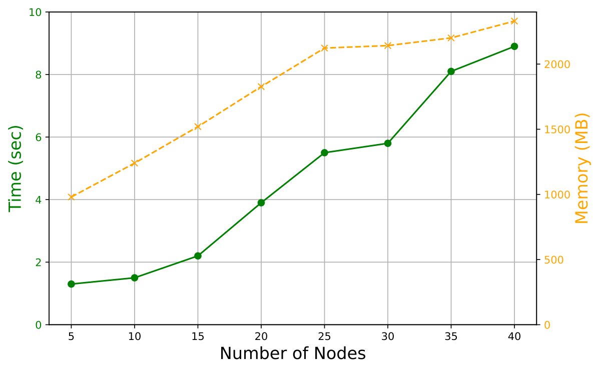

Fig. 3: Data Collected By the Deception Platform (*Packets*, *Timestamp*, and *Anomalies*)

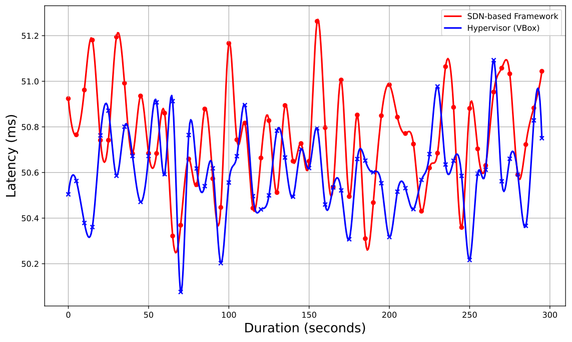

Fig. 4: Data Collected By the Deception Platform (*Packets*, *Timestamp*, and *Anomalies*)

because duplicate flow rules were pushed for every possible link whenever flows are redirected to a deceptive node or a subnet of deceptive nodes. The user can still specify links or switches to which the intended flow rules should be applied to eliminate duplicate flow rules. Therefore, the correctness of the outcome was validated along with its capability to manage different types of strategies.

## B. Measurability

The data collected serves as a critical lens through which the effectiveness of employed deception strategies is scrutinized. To provide valuable data for analyzing the effectiveness of a strategy, we implemented our platform to collect the packets exchanged in the network, timestamps associated, flow statistics through OpenFlow protocol, and system event logs in every node via Logstash [18].

Figure 2 illustrates the data captured throughout 5-hour experiment. We observed that the rate of anomalies (the number of triggering events and detection by IDS) collected by $D3$ was the highest ($< 40\%$). $D3$ also showed a higher number of packets exchanged with the deception deployed. Apart from the detailed comparisons between the effectiveness of strategies, the defenders and researchers can leverage the data collection capability provided by our platform to identify and analyze the patterns of quantitative data. Combining with the captured anomalies and timestamps associated with them, researchers can associate these quantitative data with qualitative analysis such as in-depth interview with user studies, the analysis may further provide insights into how adversaries made decisions, at which point such decisions were made, and which deceptive (or non-deceptive) actions of the strategy affected the decision-making process of the adversaries [24], [28]. Consequently, our platform can increase the efficiency in deception strategy analysis and provide a basis for the advancement of deception research by leveraging SDN.

## C. Performance

We also measured the performance of our implementation. We deployed networks with numbers of hosts ranging between 5 and 40 at an interval of 5 nodes. As the number of nodes increased for the testbed, our implementation showed linear growth in terms of deployment time and memory consumption as shown in Figure 3. For latency, the performance was similar to deploying the equivalent number of VMs with a hypervisor as illustrated in Figure 4. However, considering the strategy deployment, resource management, and data-collection capabilities, the performance overhead remains negligible relative to a stand-alone hypervisor. All the systems, services, and network topologies are identical, but one is a virtual network managed by VirtualBox (VBox) hypervisor, while the other is deployed and managed by our platform. The variation in packet latency in our platform was higher than the one with VBox, reaching up to $51.2$ ms. In contrast, the latency of the one with hypervisor shows moderate variation in the latency with $50.2 - 51.1$ ms. This was due to the flow statistics polling from control planes and flow rules installation based on deceptive strategy. Overall, the overhead incurred was minimal, primarily resulting from data collection and flow management functionalities that enable centralized and fine-grained control.

## V. Discussion and Future Direction

Currently, our prototype implementation primarily supports network-based deception techniques and strategies. Consequently, it does not consider host-level deception mechanisms, even though our design is capable of managing them. One example of such a deception is a fake configuration of an operating system to make it appear as a different one (e.g., disguising Windows systems as Linux). We plan to incorporate this capability into our implementation as part of our future development efforts. In addition, our future direction includes the adoption of AI and machine learning for zero-touch automation of dynamic deception management.

It is also important to note that our objective in this work is to demonstrate the efficacy of generic deception management framework leveraging SDN as a means to enhance research efforts in the field of cyber deception, rather than to compare the effectiveness of any specific strategies employed during our evaluation. Consequently, the actual effectiveness of these strategies and the influence of associated variables on the observed results (e.g., how the skillset of adversary impacts the effectiveness of each strategy) fall beyond the scope of this work. Instead, our work provides the infrastructure needed for rigorous empirical measurement and subsequent scientific investigation.

## VI. Conclusion

We presented the design of a cyber deception platform, which leveraged the central visibility, fine-grained flow control, and programmability of SDN. We also utilized control plane decoupling of SDN to address the challenges of designing a platform that is agnostic to specific scenarios. Lastly, we demonstrated the efficacy of our prototype by conducting case studies implementing three different deception strategies, each of which requires different configurations with negligible overhead.

## VII. Acknowledgments

This work was partially supported by the grants from US National Science Foundation (NSF-CICI-2232911 and NSF-SFS-1663651).

## References

[1] C. K. Johnson, R. S. Gutzwiller, J. Gervais, and K. J. Ferguson-Walter, "Decision-making biases and cyber attackers," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*. IEEE, 2021, pp. 140–144.

[2] M. Zhu, A. H. Anwar, Z. Wan, J.-H. Cho, C. A. Kamhoua, and M. P. Singh, "A survey of defensive deception: Approaches using game theory and machine learning," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2460–2493, 2021.

[3] C. Wang and Z. Lu, "Cyber deception: Overview and the road ahead," *IEEE Security & Privacy*, vol. 16, no. 2, pp. 80–85, 2018.

[4] M. A. Sayed, A. H. Anwar, C. Kiekintveld, and C. Kamhoua, "Honeypot allocation for cyber deception in dynamic tactical networks: A game theoretic approach," in *International Conference on Decision and Game Theory for Security*. Springer, 2023, pp. 195–214.

[5] S. Achleitner, T. La Porta, P. McDaniel, S. Sugrim, S. V. Krishnamurthy, and R. Chadha, "Cyber deception: Virtual networks to defend insider reconnaissance," in *Proceedings of the 8th ACM CCS international workshop on managing insider security threats*, 2016, pp. 57–68.

[6] S. Srinivasa, J. M. Pedersen, and E. Vasilomanolakis, "Towards systematic honeytoken fingerprinting," in *13th International Conference on Security of Information and Networks*, 2020, pp. 1–5.

[7] P. V. Mohan, S. Dixit, A. Gyaneshwar, U. Chadha, K. Srinivasan, and J. T. Seo, "Leveraging computational intelligence techniques for defensive deception: a review, recent advances, open problems and future directions," *Sensors*, vol. 22, no. 6, p. 2194, 2022.

[8] A. Alshammari, D. B. Rawat, M. Garuba, C. A. Kamhoua, and L. L. Njilla, "Deception for cyber adversaries: status, challenges, and perspectives," *Modeling and Design of Secure Internet of Things*, pp. 141–160, 2020.

[9] X. Han, N. Kheir, and D. Balzarotti, "Deception techniques in computer security: A research perspective," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–36, 2018.

[10] C. Gonzalez, P. Aggarwal, E. A. Cranford, and C. Lebiere, "Adaptive cyberdefense with deception: A human–ai cognitive approach," *Cyber Deception: Techniques, Strategies, and Human Aspects*, pp. 41–57, 2022.

[11] E. A. Cranford, C. Gonzalez, P. Aggarwal, M. Tambe, S. Cooney, and C. Lebiere, "Towards a cognitive theory of cyber deception," *Cognitive Science*, vol. 45, no. 7, p. e13013, 2021.

[12] E. A. Cranford, C. Gonzalez, P. Aggarwal, S. Cooney, M. Tambe, and C. Lebiere, "Toward personalized deceptive signaling for cyber defense using cognitive models," *Topics in Cognitive Science*, vol. 12, no. 3, pp. 992–1011, 2020.

[13] P. Aggarwal, S. Jabbari, O. Thakoor, E. A. Cranford, P. Vayanos, C. Lebiere, M. Tambe, and C. Gonzalez, "Human-subject experiments on risk-based cyber camouflage games," in *Cyber Deception: Techniques, Strategies, and Human Aspects*. Springer, 2022, pp. 25–40.

[14] P. Aggarwal, O. Thakoor, A. Mate, M. Tambe, E. A. Cranford, C. Lebiere, and C. Gonzalez, "An exploratory study of a masking strategy of cyberdeception using cybervan," in *Proceedings of the human factors and ergonomics society annual meeting*, vol. 64, no. 1. SAGE Publications Sage CA: Los Angeles, CA, 2020, pp. 446–450.

[15] R. Píbil, V. Lisỳ, C. Kiekintveld, B. Bošanskỳ, and M. Pěchouček, "Game theoretic model of strategic honeypot selection in computer networks," in *International Conference on Decision and Game Theory for Security*. Springer, 2012, pp. 201–220.

[16] K. Horák, Q. Zhu, and B. Bošanskỳ, "Manipulating adversary's belief: A dynamic game approach to deception by design for proactive network security," in *International Conference on Decision and Game Theory for Security*. Springer, 2017, pp. 273–294.

[17] D. B. Rawat, N. Sapavath, and M. Song, "Performance evaluation of deception system for deceiving cyber adversaries in adaptive virtualized wireless networks," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, 2019, pp. 401–406.

[18] "ELK stack," https://www.elastic.co/elastic-stack, [Online; accessed 2024-1-14].

[19] "Open Network Operating System (ONOS)," https://opennetworking.org/onos/.

[20] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM computer communication review*, vol. 38, no. 2, pp. 69–74, 2008.

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

[22] S. Vyas, V. Mavroudis, and P. Burnap, "Towards the deployment of realistic autonomous cyber network defence: A systematic review," *ACM Computing Surveys*, 2025.

[23] J. Pawlick, E. Colbert, and Q. Zhu, "A game-theoretic taxonomy and survey of defensive deception for cybersecurity and privacy," *ACM Computing Surveys (CSUR)*, vol. 52, no. 4, pp. 1–28, 2019.

[24] K. Ferguson-Walter, T. Shade, A. Rogers, M. C. S. Trumbo, K. S. Nauer, K. M. Divis, A. Jones, A. Combs, and R. G. Abbott, "The tularosa study: An experimental design and implementation to quantify the effectiveness of cyber deception." Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), Tech. Rep., 2018.

[25] "Snort," https://www.snort.org/, [Online; accessed 2024-4-28].

[26] P. Aggarwal, C. Gonzalez, and V. Dutt, "Hackit: a real-time simulation tool for studying real-world cyberattacks in the laboratory," in *Handbook of Computer Networks and Cyber Security*. Springer, 2020, pp. 949–959.

[27] S. Ghorbani and B. Godfrey, "Towards correct network virtualization," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, 2014.

[28] K. J. Ferguson-Walter, M. M. Major, C. K. Johnson, and D. H. Muhleman, "Examining the efficacy of decoy-based and psychological cyber deception," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021.