Deception by Design: A Configurable Platform for Flexible Cyber Deception Strategy Testing and Evaluation

Sukwha Kyung[0000−0002−9357−6249], Souradip Nath[0000−0001−7228−2316], Jaejong Baek[0000−0001−8588−3524], and Gail-Joon Ahn[0000−0002−4271−1666]

Arizona State University
Tempe AZ, USA
{skyung1,snath8,jaejong,gahn}@asu.edu

Abstract. Cyber deception has emerged as a promising paradigm for enhancing cybersecurity by introducing uncertainty and manipulating adversarial decision-making processes. However, research in this domain is hindered by the absence of flexible, reproducible, and measurable testbeds capable of supporting diverse deception strategies. Addressing this gap is essential to transforming cyber deception from a conceptual tool into an operational capability that can be reliably tested and deployed in various environments. This paper presents Cyber Deception Gym (*CDGym*), a configurable platform designed to address the disconnection between abstract deception strategies and their practical implementation. In order to enable scalable, modular, and automated deployment of deception strategies, we also introduce the Deception Configuration Schema (DCS), a structured approach to articulating configuration options and associated resource constraints of deception strategies. In addition, our platform supports comprehensive data collection and real-time feedback to facilitate effective evaluation. We prototype *CDGym* and evaluate it through case studies of three representative deception strategies, complemented by a large-scale penetration testing experiment with 80 professional testers and a usability study involving 15 deception researchers. The results demonstrate that *CDGym* facilitates reproducibility, scalability, and quantitative evaluation, thereby establishing itself as a robust platform for advancing cyber deception research.

Keywords: Cyber Deception · Network Security

1 Introduction

In today's cybersecurity landscape, organizations face persistent and evolving threats from increasingly sophisticated adversaries. Cyber deception has emerged as a promising approach to address these challenges. Rather than relying solely on detection and prevention, deception introduces uncertainty, doubt, and manipulation into the attacker's decision-making process [53]. It operates by deliberately providing misleading information, false targets, or interactive traps within a system, thereby increasing the cost and complexity of an attack while reducing its likelihood of success [6, 47, 48]. Moreover, deception techniques can

serve as a powerful detection mechanism, offering defenders visibility into attacker behavior and intentions that might otherwise go unnoticed [32, 52, 54].

Challenges in Cyber Deception Research: However, despite leveraging innovative frameworks such as game theory, the research endeavor to devise innovative cyber deception strategies still remains challenging. Key obstacles that impede progress in cyber deception research include (1) lack of demonstrating the realism of deception techniques, (2) limited adaptability to diverse environments, (3) difficulties in measuring performance, (4) lack of flexible and reproducible experimental setups, and (5) inadequate feedback mechanisms for thorough data collection and analysis [11,16,23,29,39]. In addition, to compare and analyze the effectiveness of different deception strategies, researchers must set up multiple experimental environments with fine-grained configurations at both the host and network levels. Moreover, most game-theoretic cyber deception research works have primarily focused on either analyzing the cognitive decision-making process of the players [14,15,21], or evaluating deception models using simple simulations that are designed only for specific scenarios [9, 10], which may not adequately represent complex or diverse environments. As a result, researchers attempting to assess the effectiveness of a cyber deception strategy or compare multiple strategies have been facing significant challenges. The following key factors contributed to the aforementioned challenges:
(1) The absence of an adaptive testbed that can support the analysis and implementation of a wide range of cyber deception strategies. Such a testbed should encompass comprehensive network and deception management features as well as robust data collection capabilities. The construction of a testbed for deception strategy analysis is nontrivial, as it is necessary to accommodate various types of strategies, each requiring a tailored environment and corresponding deception mechanisms.
(2) The models derived from game theory are frequently criticized for their high degree of abstraction, which complicates practical implementation. For instance, in signaling games [46], the *signal* is vaguely described as a piece of true or false information that is deliberately disclosed to the adversary to create confusion. Several aspects remain unclear in this context, including the methods for implementing and deploying the signal, the approaches for measuring the associated utility, and the criteria for assessing its performance. Consequently, the realism and effectiveness of existing game-theoretic cyber deception strategies in real-world scenarios remain inadequately understood.

Approach: In this paper, we present an overview of *CDGym*, a generic cyber deception platform designed to provide a flexible and environment-agnostic testbed for cyber deception research. Designed to aid researchers in exploring the complex interactions between attackers and deception mechanisms, *CDGym* addresses the current challenges by supporting scalability, reproducibility, and measurability.

However, even a single strategy can be realized through various combinations of deception mechanisms. Therefore, to enable scalable implementation of such strategies, it is essential to have a functionality that allows an abstract

strategy to be concretely defined in terms of its diverse practical realizations. To support such a functionality, we also design and implement a configuration schema for creating diverse deceptive environments. This design, named *Deception Configuration Schema* (DCS), allows users to define deception strategy and its mechanisms to be implemented within a simulated experiment network. Unlike existing configuration languages such as XML and JSON, DCS is designed to be less verbose, better-suited for host and network configurations and support deception provision. Additionally, DCS supports modularity through plug-and-play feature, enabling users (i.e., researchers experimenting with various deceptions) to group and manage related resources when implementing customized deception strategies. In this way, this work aims to reduce the gap between game-theoretic deception strategies and their practical application, providing a cohesive environment for testing and analysis of diverse deception strategies.

Based on the proposed design, we implement a prototype of *CDGym* and assess its effectiveness by conducting case studies with three distinct game-theoretic deception strategies. Among various technologies that can be leveraged to implement *CDGym*, we primarily employ virtualization and software-defined networking (SDN) technologies to realize the features of *CDGym*, automating environment configuration, control over flow and network status, and data collection. These technologies establish a foundational infrastructure that supports the dynamic simulation and evaluation of cyber deception strategies, thus enabling researchers and practitioners to adapt and optimize defensive strategies effectively.

Contribution: Overall, this work makes the following three contributions:

(i) We propose a generic cyber deception analysis platform *CDGym*, that is agnostic to the types of cyber deception models. *CDGym* facilitates the deployment of test environments, each incorporating different deception models along with appropriate deception mechanisms. This platform addresses the significant problem of the current cyber deception research, which is the absence of a unified, automated environment that can ensure reproducibility, measurability, and scalability. In developing *CDGym*, we also identify and address key design challenges and requirements.

(ii) We implement prototype of *CDGym* to embody the proposed design principles. By implementing a configuration schema that enables the translation of abstract strategy into practical implementations, *CDGym* establishes a comprehensive platform for evaluating diverse deception models that are based on game theory. Our implementation addresses the current challenges in cyber deception research, such as the absence of flexible test environments, limited support for diverse deception techniques, the complexity of automated testbed setup, and the lack of capabilities for real-time data collection. Our design makes *CDGym* a pivotal tool in advancing the field of cyber deception by providing a robust and flexible testing infrastructure.

(iii) We validate the *effectiveness* and *usefulness* of our platform by conducting case studies and qualitative user studies. The case studies involve a simulation of a real-world corporate network environment and 80 professional net-

work penetration testers. We also conduct a user study with 15 researchers in the field of cyber deception for the usability of *CDGym*. By analyzing and comparing the outcomes from these experiments, we demonstrate that *CDGym* enables a comprehensive analysis of performance for specific deception strategies.

The key distinction between *CDGym* and existing works is its *versatility*. While existing works on cyber deception testbed platforms suffer from inflexible and ad-hoc environments specific to a strategy under examination without data collection capability, *CDGym* can provide testbeds as defined by users with high scalability and measurability. These capabilities of *CDGym* to support a comprehensive range of deception measures, systems, and services allow users to simulate diverse and complex scenarios effectively.

2 Background

The foundational premise of cyber deception is based on controlling the adversary's perception. By shaping what attackers see, how they interpret system responses, and what paths they choose to follow, defenders can influence the attacker's trajectory, either to divert them from critical assets or to steer them into environments where their actions can be safely monitored and analyzed. This shift from reactive to proactive security aligns well with cyber resilience principles and supports adaptive threat management in complex environments.

A range of deception strategies has been proposed and implemented in both research and practice. These strategies differ in terms of technical mechanisms, deployment complexity, and security objectives. Table 1 presents a taxonomy of common cyber deception strategies, categorized by their core mechanisms and representative examples. This classification draws from foundational studies in deception theory and game-theoretic modeling [41, 42]. The table also highlights the specific security objectives that each strategy is typically designed to achieve, such as obfuscation, adversary engagement, misinformation, or tracking resistance.

Recently, game-theoretic models have been applied to cybersecurity problems due to the adversarial interactions among two or more parties [24, 31, 37]. Cyber deception game is one where the defender (e.g. network administrator) can use deceptive techniques to reduce the risks and improve the protection effectiveness of the systems. For example, in the works, the defender can deploy honeypots as decoys in her networks and distract the attackers from the real hosts. This scenario sets up a decision-making dilemma for the attacker, who must choose between continuing the attack or abandoning it, based on the perceived risks and rewards. The outcome of this interaction is influenced by the strategies and decisions of both the attacker and the defender. Game theory provides a framework to analyze and optimize these interactions, allowing for strategic modeling of potential outcomes based on various actions taken by each participant. Therefore, game theory can provide insight into when and how strategies should be

Table 1: Deception strategies with mechanisms and examples [41, 42].

Strategy	Mechanism	Examples	Objective
Permutation	Inject randomness/noise to data/responses	Fake sensor values in SCADA, dummy API errors	Misinformation, Obfuscation
Moving Target Defense	Dynamic IP/Port/OS	IP hopping, dynamic ports, OS reimaging	Confusion, Delay
Obfuscation	Encrypt, pad, or disguise real content	Traffic padding, obfuscated file names	Hiding patterns
Mixing	Combine flows to obscure identity/location	Route messages through multiple nodes, mix zones	Anonymity, Anti-tracking
Honey-X	Deploy decoys (systems, data, tokens)	Honeypot, honeytokens, decoy credentials	Detection, Profiling, Delay
Attack Engagement	Controlled interaction with attackers	Fake admin panels, sandbox malware	Engagement, Intelligence

adopted by a defender, or in our case an adaptive cyber defense system using automated deception techniques [18].

3 Design Overview

We first provide an explanation to highlight the need for a comprehensive platform for analyzing cyber deception strategies, along with our design of *CDGym*. We then discuss the high-level workflow and insights underlying the design of *CDGym*, along with the design of our configuration schema.

3.1 Challenges and Design Criteria

Current Challenges: In order to develop a robust deception strategy, researchers often perform comparative analysis to systematically evaluate and compare different methods, techniques, or frameworks, identifying their strengths, weaknesses, and applicability under various conditions. For example, comparative analysis helps researchers and developers determine which techniques are most effective in specific scenarios, study behavioral differences of adversaries, determine the cost-benefit balance, and assess how effectively different techniques mitigate the risk of successful cyber attacks [19, 23, 25, 30]. Therefore, analyzing the performance and efficacy of different cyber deception strategies is critical.

However, it is, at the same time, a complex endeavor due to several challenges. We first clarify the challenges that impede the effective analysis of deception strategies and explain why these challenges motivate the need for our work. The current challenges in the analysis of cyber deception strategies are as follows:

(1) *Difficulties in implementation of strategy:* In order to analyze deception strategies, it is imperative to implement deception strategies in a realistic environment and collect data. However, most strategies are based on highly theoretical or abstracted models without consideration of actual implementation, resulting in difficulties in realization of them in real-world testbeds [12]. Moreover, each strategy requires a distinct test environment equipped with different types of deception mechanisms. Preparing such an environment is not only cumbersome and time-consuming but also heavily reliant on manual intervention by human experts. Such complexities without automation make it difficult to assess and compare the performance of different strategies, increasing the need for a testbed that can automatically implement multiple types of deception strategies.

(2) *Lack of Configurability:* Deception strategies are designed to be reactive to adversarial actions. These reactive strategies require dynamic feedback of system or network status, with subsequent reconfiguration of the environment based on the feedback. Consequently, the capability to reconfigure the status of the system and network is essential for the implementation of deception strategies. This reconfiguration also needs to be performed in an automated and reproducible fashion. The absence of a testbed with such functionalities causes difficulty in implementing, reproducing, and analyzing these strategies.

(3) *Lack of data collection capability:* Effective quantitative analysis of the performance of a deception strategy is contingent upon robust data collection capabilities. For instance, if the cost to the attacker is defined as the time required to exploit a vulnerability in the target system, the efficacy of a given deception strategy can be quantitatively assessed in terms of time. To enable such an analysis, collecting metrics such as attack paths and keylog data are crucial. However, the existing testbeds notably lack these essential data collection functionalities, thereby hindering comprehensive analysis and evaluation of deception strategies.

Design Criteria: Based on these deficiencies in the current game-theoretic cyber deception research, we propose three essential criteria that should guide the design of a deception platform. These criteria are *Feedback & Reconfiguration*, *Scalability* and *Measurability*:

(1) *Feedback & Reconfiguration* involves the capability to both monitor and respond to changes within the environment based on the deployed strategy. For example, a strategy that includes dynamic modifications, such as the adding or removing of subnets via gateway changes, requires the testbed to detect any adversarial actions that trigger these changes (*Feedback*) and subsequently adjust the network configuration to redirect network flows to a target subnet (*Reconfiguration*). Our design implements this by detecting a triggering event from collected flow statistics and system logs that are constantly collected and monitored. The triggering event invokes a corresponding deceptive reaction defined by the strategy deployed.

(2) *Scalability* indicates that 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 deception mechanisms employed. In other words, the testbed should be environment-agnostic, capable of integrating various game-theoretic strategies.

(3) *Measurability* pertains to the ability to collect quantitative data effectively. Current processes often rely on manual and fragmented tasks that require significant human involvement. Effective measurability necessitates an automated, centralized data collection mechanism that operates independently of manual input.

These criteria ensure that the testbed is capable of reproducing research conditions, extensible to various deception strategies, providing real-time feedback and reconfiguration capabilities. In addition, the criteria also make sure that the platform can facilitate comprehensive data collection.

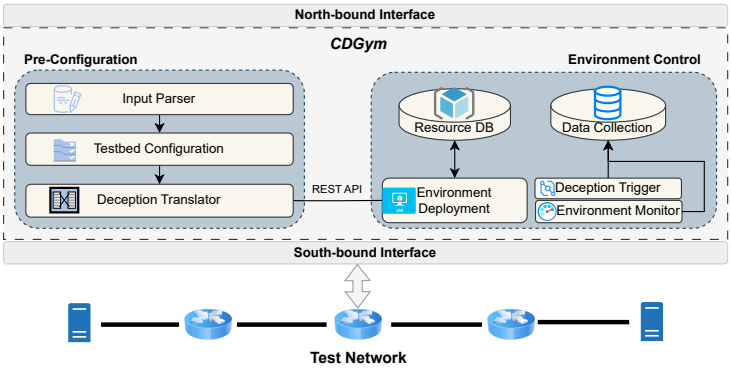


Fig. 1: Architecture of *CDGym*.

3.2 Design & Implementation of *CDGym*

For *Feedback & Reconfiguration*, *CDGym* actively monitors the system or network status and provides real-time reports on any changes. If a strategy requires dynamic adaptation based on this feedback, *CDGym* is equipped to modify the environmental configuration accordingly, ensuring that the strategy remains effective under varying conditions. In terms of *Measurability*, *CDGym* is designed to actively collect real-time data during experiments conducted within the deceptive network environment. This capability allows users to comprehensively measure and analyze the efficacy of their deception strategies, providing valuable insights into their performance. Lastly, to achieve *Scalability*, we design a configuration schema, DCS, which utilizes an attribute-value pair system similar to JSON. This design choice allows for flexible, precise, and user-friendly configuration of diverse deception strategies, making *CDGym* adaptable and extensible to a wide range of scenarios and requirements.

Architecture of *CDGym*: Figure 1 illustrates the architecture design. *CDGym* is composed of two main modules: *Pre-Configuration* and *Environment Control* modules. The initial input to *CDGym* via the user-facing north-bound interface defines an environment with information required to deploy the desired network configuration, specifying node, link, and topology configuration. The input also requires the defender to define the deception strategy to be deployed, specific deceptive mechanisms, and events triggering deceptive reactions in case of dynamic reactive strategy. Our comprehensive literature review, guided by the taxonomy [41,42], has led to the integration of six distinct strategies within *CDGym*, each corresponding to different types of game-theoretic models as shown in Table 1. These strategies are embedded into the platform, enabling versatile and dynamic deception tactic deployment.

Specifically, when a user inputs the environment and deception attributes, the *Input Parser* function of *Pre-Configuration* module first retrieves attributes of each element that constitutes the network environment to be deployed. Then, the strategy is also parsed and applied to the network configuration as the *Testbed*

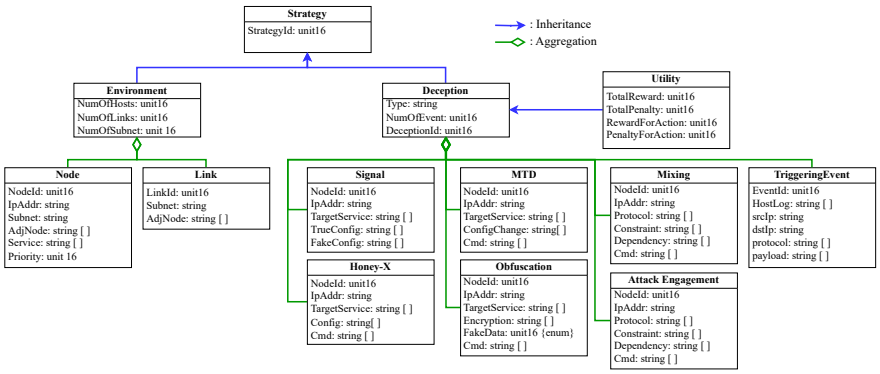


Fig. 2: Configuration Schema for Deception Strategy Definition.

Configuration function constructs and finalizes the configuration. During the process of strategy parsing, the *Deception Translator* also retrieves the priority scores assigned to each asset, where an asset is a set of valuable resources (systems, links, services, etc) that should be protected. The importance of each asset is determined by its priority score, which is defined from a defender's perspective. These scores are used for the utility function of the input strategy provided by the user, where the utility function can be determined by the specific goal of the strategy (e.g., maximize the utility, minimize loss, or maximize the loss of the attacker, etc). Deployment of the environment is performed by the *Environment Deployment* function after it determines if the input can be implemented using the resources registered in the *Resource DB*.

After the simulated environment is deployed by the *Environment Deployment* function, *Environment Monitor* constantly monitors the environment, collects, and stores data. The stored data can also be fed into *Deception Trigger* module to trigger reaction from the specified deception strategy. In addition to supporting real-time and autonomous responses to deception triggers, *CDGym* enables step-by-step simulation of both attacker and defender actions by allowing user intervention at each triggered event. This feature is particularly valuable in the context of dynamic game models, where it is beneficial for the user (from defender's perspective) to monitor each action taken by the adversaries and observe the corresponding changes in the test environment. Such real-time monitoring and responsive updating capabilities can help fine-tune or improve the performance of deception strategy under assessment based on the evolving dynamics of the engagement.

Deception Configuration Schema: The definition of the desired network environment and deception strategy is represented by DCS. The primary purpose of DCS is to present the components, the relationship between them, and enable *Deception Translator* to formulate an optimized implementation of strategy. In addition, it is crucial to consider the resource constraints, including the inventory of system and network resources, and their utilization to determine the optimal strategy. Because the input should include these multiple factors, deploying an entire simulated network with a deception strategy can be a complex and tedious

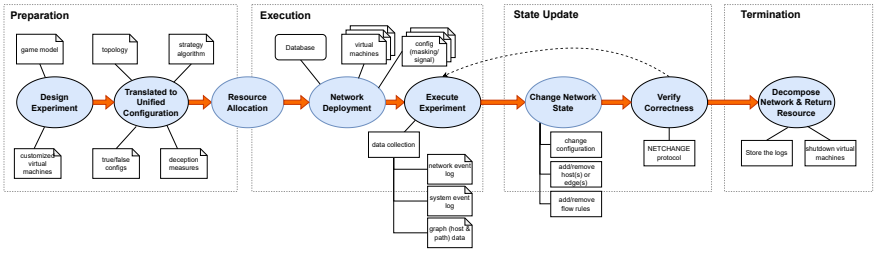


Fig. 3: Overall workflow of *CDGym*.

task due to the intricate details involved. Thus, we design DCS to systematically and comprehensively address these aspects and facilitate ease of use.

Representing a deception strategy with DCS involves two major objects—environment and strategy objects. Each object includes corresponding groups of attributes as its components. Figure 2 shows the schema for a deception strategy presented with environment and strategy objects. An *Environment* object consists of Nodes and Links to represent each node with priority assigned to it, and links between the nodes. Deception object, on the other hand, enables the user to specify one of the six strategies and a utility function to calculate the benefit (or gain, payoff, reward) or loss (or penalty) caused by an action. The attributes of each object contain the identity of the object, along with the target process or service to which a configuration is applied to, and the status to reach when the given deceptive action takes place.

For example, to implement the "Mixing" deception strategy schema in practice, one must architect a system capable of anonymizing communication patterns by obfuscating the origin, destination, and temporal correlation of transmitted data. This typically involves deploying a mix network architecture, where user messages are routed through a series of intermediate nodes to prevent end-to-end traffic correlation. Technologies such as Loopix [44] can provide a mechanism for message shuffling or layered encryption. Successful deployment of this strategy requires a relay infrastructure with distributed nodes to implement mix nodes. Additionally, coordination protocols must be established to maintain consistent mixing policies across distributed nodes while preserving unlinkability guarantees. The Mixing strategy offers high efficacy in resisting surveillance and metadata analysis, particularly in adversarial environments where privacy and anonymity are paramount.

In *CDGym*, this can be implemented by defining *Protocol* attribute for consistent communication between mix nodes while *Constraint* attribute defines the specific goal of the strategy (e.g., privacy through *encryption*) along with the associated *Cmd* (commands) to be run in the nodes for consistent deployment. Likewise, the "Attack Engagement" should implement a system that enables controlled, observable interaction with adversaries in a manner that isolates malicious behavior while capturing actionable intelligence. This involves deploying interactive deception environments, such as high-interaction honeypots, fake administrative portals, or sandboxed command-and-control (C2) emulation plat-

Table 2: Environment and Deception Attributes of DCS.

Environment		Deception	
NodeId	A unique identifier for each node.	TargetService	A set of services/assets the given deception mechanism covers.
LinkId	A unique identifier for each link.	Protocol	A set of target protocols the given deception mechanism covers.
AdjNode	A set of IDs of adjacent nodes to the given node.	TrueConfig	A set of actual configuration of the service/asset.
Service	A set of services/assets of the given node.	FakeConfig	A set of fake configuration used by deception to mask the true configuration.
Priority	Numeric priority score assigned to a node.	Constraints	A set of conditions to be met for a specific deception mechanism (e.g., isolation, record, delay, etc)
IpAddr	IP address of a node.	Dependency	A set of dependencies required to satisfy the given constraints.
Subnet	Subnet mask of a node.	Cmd	A set of subprocess/commands to be run.

forms, which are engineered to appear as legitimate targets to attackers. These systems must be indistinguishable from real assets, both in terms of their network behavior and service configurations, in order to entice engagement and sustain adversary interest. Therefore, the defender can set the *Constraint* attribute to isolation and *Dependency* attribute to a list of software packages such as Lyrebird [2].

Table 2 shows brief descriptions of environment and deception attributes. DCS is also designed to support flexibility and scalability by allowing the user to define and add customized objects. For example, if the defender devises a new strategy with a set of mechanisms that requires different configuration, processes, or commands, they can simply add those to the corresponding attribute fields so *CDGym* can interpret and implement them. This plug-and-play capability enabled by DCS provides flexibility and can be extended to implementation of any type of strategies.

Figure 3 illustrates the overall workflow of *CDGym*, structured into four main procedural stages: *Preparation*, *Execution*, *State Update*, and *Termination*. Each stage also consists of several subroutines. During the *Preparation* phase, initial planning and design activities for the experiment are undertaken. This phase may involve tasks such as preparing the network topology suitable for the intended deception strategies, specifying custom virtual machine images through the plug-in module if necessary, and defining the configuration parameters. These parameters include detailed host and network configurations that align with the actual test environment setup.

Once the design is finalized, resources such as VMs, virtual switches, and network links are allocated for the experiment, followed by network deployment in *Execution* phase. *Execution* phase is when *Environment Deployment* function deploys simulated environment including systems, services, and network, using the configuration parameters defined in the user input. This involves running specific protocols or software, and making adjustments such as adding or removing hosts or edges, modifying flow rules, and other network changes. After the deployment, *Environment Monitor* immediately starts running to collect and generate logs that record the states of the test environment.

The purpose of *Execution* phase is to deploy the intended strategy in the test environment correctly. If there occurs any change in the test environment, it can be detected by *Environment Monitor* through its monitoring mechanisms such as NETCHANGE protocol [3]. Depending on the strategy, it may trigger a report of these state changes. The updated state is then provided as an updated input to *Deception Trigger* function. The *Deception Trigger*, in response, gen-

Algorithm 1 Adaptive Flow Redirection

```
procedure FlowRedirection
    Initialize honeypots H = {h_1, h_2, ..., h_k}
    Initialize triggering event T = {t_1, t_2, ..., t_n}
    Initialize redirection rules R = {r_1, r_2, ..., r_m}
    while monitoring the network traffic do
        for each incoming packet p do
            if p ⊆ T then
                Spawn a subset of honeypots H'
                Apply R' to direct traffic to H'
            else
                Continue monitoring the next packet
            end if
        end for
    end while
end procedure
```

erates and dispatches updated strategy attributes for deployment based on the selected strategy. The strategy implements its corresponding deception mechanisms through dynamic scaling of resources and configuration changes to impede, delay, or confuse the attacker. After the experiment is completed, the network is decomposed, meaning that the temporary configurations are removed, and resources are freed up and listed as available in the *Resource Database*.

Implementation of *CDGym*: We implemented the prototype of *CDGym* with various deception mechanisms as virtual machines (VMs) and containers with simulated flows. This enables users to select and utilize the most suitable deception strategies and mechanisms according to their specific needs and strategies they want to test. *CDGym* should also include support for automated data collection and monitoring in a centralized manner. There exists multiple ways to fulfill this requirement. In our implementation, we leveraged ELK stack [1] and ONOS software-defined networking (SDN) controller [4]. Each control module we introduced in Section 3.2 was implemented in Python and Java. These tools and technologies enable streamlined and centralized control over the data collection and monitoring processes within the test environment.

Strategies under Testing: We use three representative deception strategies for our evaluation. We labeled the datasets from each strategy *D0 – D3*. An explanation for each environment is as follows:
D0 (No Deception Deployed): This environment serves as the baseline condition.

D1 (Flow Redirection) [45]: This strategy is based on a zero-sum game model and seeks to minimize the cumulative loss of the defender. To do so, the authors of the strategy utilized *isolation* of traffic to a subnetwork of deceptive hosts. The authors also leveraged the traffic engineering capability of SDN to isolate malicious traffic and redirect it to a subset of honeypots. We implemented this strategy through getting traffic information from the data plane via RESTful API in real-time. If it detects malicious traffic, ONOS can generate traffic redi-

rection rules. The generated policies are pushed to the data plane using *FLOW MOD* messages [38]. The other approach to implement for forwarding the packets to a honeypot is to use an SDN proxy designed to select appropriate honeypots depending on the attacker's action [35]. In this way, the traffic is redirected to a designated honeypot or subnet of honeypots (H'), which is a subset of entire honeypots ($H = \{h1, h2, ..., h_k\}$). The entire adaptive flow redirection algorithm is described in Algorithm 1.

D2 (Active Manipulation of Attacker's Belief) [27]: This strategy actively manipulates the attacker's belief by following a probabilistic model based on Markov decision process [26]. As an example of a potential strategy, the authors proposed traffic control techniques, such as blocking the traffic, that follow the probabilistic model to implant a sense of being monitored in the attacker's mind and ultimately influence the attacker's belief. This work also proposed the generation of deceptive information to manipulate the cognitive decision-making process of the attacker, but it is suggested as another utilization of a probabilistic model and does not present a concrete strategy. Based on this observation, we implemented traffic control using a probabilistic model. We model the defender's deception controller as a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ that selects a traffic-control action at each decision epoch (e.g., per suspicious flow per time window Δt) to steer the attacker toward the belief of being monitored while still preserving operational goals. Specifically, the state $s \in \mathcal{S}$ summarizes (i) an estimated attacker belief level $b \in \{0, 1, 2\}$ (low/medium/high "I am monitored"), (ii) an intrusion stage $k \in \{0, 1, 2\}$ (recon/exploit/post-exploit), and (iii) an alert level $\ell \in \{0, 1\}$ (IDS low/high). The action set $\mathcal{A}$ is $\{\mathsf{ALLOW}, \mathsf{DELAY}, \mathsf{REDIRECT}, \mathsf{BLOCK}\}$ applied to the flow for the next Δt (e.g., $\Delta t = 5\,\mathrm{s}$), where DELAY injects latency/jitter (e.g., $+150 \pm 50$ ms), REDIRECT moves the flow to a monitored decoy subnet, and BLOCK drops packets. The transition kernel $P(s'|s, a)$ captures how actions probabilistically change b and k. For example, when $\ell = 1$ (high suspicion), DELAY increases belief with probability $P(b{+}1|b, \mathsf{DELAY}) = 0.40$, REDIRECT with 0.55, and BLOCK with 0.25 (often triggering "detected" but also aborting interaction), while ALLOW yields a smaller increase 0.10.

D3 (Honeypot Selection) [43]: The strategy is based on a static honeypot placement problem using priority scores assigned to each node (asset) in the network. The authors consider a scenario where a defender can deploy a limited number of honeypots to deceive and delay potential attackers. They also compare the quality of the game-theoretic solution to baseline strategies and test their strategies against simple heuristic attackers. Based on their analysis, the authors provide recommendations to network administrators applying honeypots in their networks. Overall, the authors highlight the importance of considering both technical and strategic issues in honeypot design and use.

Note that *D1* and *D2* take active deception measures (manipulating the flow and attacker's belief) upon detection of adversarial activity. While outright blocking of malicious traffic may effectively terminate an attack, deception serves a fundamentally different objective. Rather than solely preventing access,

deception aims to strategically delay, mislead, and impair adversarial actions while simultaneously enabling the defender to observe, analyze, and collect detailed intelligence on attacker behavior, tactics, and decision-making processes. This intelligence-gathering capability is a key motivation for employing deception instead of immediate blacklisting.

4 Evaluation of *CDGym*

In this section, we present a comprehensive evaluation of the efficacy of *CDGym* and demonstrate that it fulfills the design goals defined in Section 3. With the three strategies implemented, we also mimic a research experiment in which researchers conduct a comparative analysis of the three strategies. To that end, we performed network penetration experiments with human subjects. Our evaluation aims to prove the following items:

- **Performance:** Through the case studies, we demonstrate that *CDGym* seamlessly collects data in real time during the execution penetration tests. Using these data, we analyze and compare the performance of the three different deception strategies excerpted from the existing works. The goal is to prove that *CDGym* provides *measurable* environments that enable the users to analyze the performance of various deception strategies.
- **Scalability:** In addition to the network deployment and data collection capabilities, we also aim to prove that *CDGym* can interpret, generate, and deploy the intended deception strategy regardless of the type of strategy. In addition, we demonstrate the correctness of the deployed strategy through NETCHANGE protocol to verify network information whenever the network goes through changes.
- **Usability:** Lastly, we performed user survey regarding the usability of *CDGym*. We invited 20 researchers working in the field of cyber deception who used our prototype of *CDGym* and collected their feedback on our survey questionnaire.

4.1 Network Penetration Test

Experiment Environment: For our case studies, we designed and conducted a network penetration test in a simulated enterprise network with 46 hosts running Microsoft Windows and Linux operating systems. 24 hosts were installed 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. We employed ONOS SDN controller for our experiment to centrally manage the experimental environment. This choice was made to leverage the innate capability of SDN, which enables the monitoring and collection of traffic data, the selection of the optimal deception mechanism for each strategy, and the enforcement of flow rules in real-time. Snort [5] was also deployed as an IDS and a part of deception

techniques, especially for *D3*. Among the 46 hosts, 8 hosts were deployed with the target data. The target data consisted of made-up client credentials, which was one of the high-value assets that must be protected.

The participant was given access to a Kali Linux instance connected to the test environment to perform the penetration test. Overall, *CDGym* simulated an existing corporate network as closely as possible while conducting the penetration tests with human participants from cyber security industry. In this way, we addressed the limitation of oversimplification of experimental environments (a network with only a few nodes) and the expertise of human subjects (arbitrarily recruited participants with little or no security background) in previous works on deception strategies [8, 10].

Recruitment: We recruited 80 participants by contacting individuals working in the cybersecurity field through multiple channels, including known contacts, employment platforms (e.g., LinkedIn), cybersecurity events and conferences. The participants were recruited from multiple sub-domains of the cybersecurity field, including security consulting, professional penetration testing, malware and vulnerability analysis, and security research. Once the participant responded to our email with a written agreement, we sent consent forms along with a demographic questionnaire to collect demographic information, including age, education level, duration of career in the cybersecurity field, and job title. After recruiting the first small set of participants, we expanded our participant pool by combining ongoing recruitment strategies with snowball sampling, where existing participants referred additional subjects who would be interested in participating in our study. [22, 34, 40].

Experiment Procedure: Each penetration test took place with one participant playing as an attacker over the period of 24 hours. Each environment (i.e., *D0* - *D3*) had 20 participants randomly assigned to ensure an even distribution of the sample. Each participant performed the test for only one deception strategy to prevent learning bias, which can degrade the validity of the data. Specifically, if one human participant repeatedly performs the penetration tests in the same network environment with all three strategies, the participant is affected at both conscious and subconscious levels [15]. This occurs because a learning bias is created by the information that the participant acquires from the previous round of tests. Therefore, data obtained from one participant performing multiple rounds of tests for different deception strategies is not valid. In addition, it is not feasible to have one participant perform three separate 24-hour long experiments.

The participants were given two main objectives: 1) Find any vulnerable systems and exploit as many as possible, and 2) Find and exfiltrate the 8 target data, each of which is stored in 8 different hosts among the vulnerable system. A monitoring agent answered to any questions the participant asked during the experiment regarding the details of the procedure. Note that we did *not* reveal the true goal of our research but simply introduced our research as a study of decision-making process in network penetration tests. In this way, we could

prepare a realistic sample of attackers that had no information or bias regarding the target network before conducting reconnaissance.

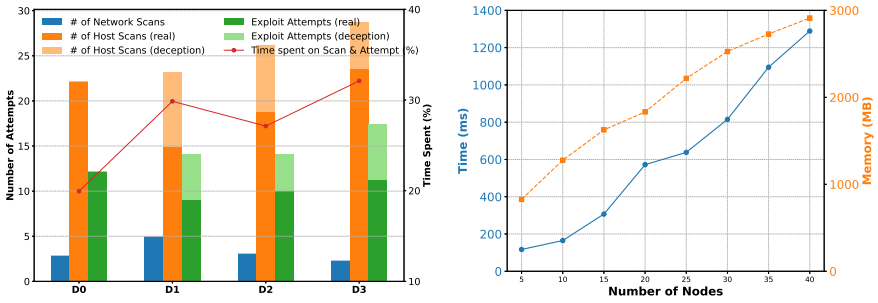
Research Ethics: The experiments and participant recruitment process received an exemption from our Institutional Review Board (IRB). Even so, we followed policies and procedures designed for human research studies that are specified by our institution. We did not use any personally identifiable information (PII) in our analysis and anonymized the data. In addition, the data is stored in a physical hard drive, which is not connected to or accessible through external networks. We received the exemption for both our penetration test and usability study.

4.2 Case Study

The main goal of the penetration tests is demonstration of the performance and efficacy of *CDGym* to collect data in real-time for performance analysis, and ultimately, provide a quantitatively measurable platform.

Data Analysis: The efficacy of any cyber deception platform lies not only in its ability to deploy deceptive measures but also in its capacity to gather real-time and meaningful data during active engagement. The data collected serves as a critical lens through which the effectiveness of employed deception strategies is scrutinized. We first aim to demonstrate the *CDGym*'s data collection capability, examining its capacity to capture relevant information, facilitate post-experiment analysis, and contribute invaluable insights into the dynamics of cyber deception strategies.

Although there is no standardized way to measure the effectiveness of a deception strategy, there is a universal agreement that the effectiveness can be assessed by how successful the strategy delays attacks, disrupting the progress of an attacker. In our case studies, we plan to measure the effectiveness in terms of attack costs. The attack costs include time, effort, and resources invested to compromise the target. Higher costs incurred by deception mean the deception impedes the attacker's progress effectively. To that end, we measured the quantifiable data that indicated how long a successful attack was delayed. The evaluation metrics include: (1) *Network scans*, detected by surges in traffic from a host to multiple destinations; (2) *Host scans*, captured by abnormal traffic to a single host's ports; (3) *Exploit attempts* and (4) *Successful exploits*, both tracked through system events; (5) *Time to first exploit*, indicating deception effectiveness; (6) *Proportion of reconnaissance*, measuring the time spent on reconnaissance relative to the experiment duration; (7) *Login attempts*, the number of login attempts to any systems; (8) *Time to target* represents time taken to start interacting with the target system or service by monitoring the traffic exchanged; (9) *Number of data exfiltration*, the number of attempts to transfer or exfiltrate data; (10) *Duration for exfiltration*, the duration of time taken to exfiltrate the data; and (11) *Network traffic statistics* (packet counts, frame sizes, timestamps) collected in *pcap* format.



(a) Evaluation Metrics and Data Collected

(b) Performance of *CDGym*

Fig. 4: Data Collected and Performance of *CDGym*

Figure 4 (a) shows the mean values of the first six data metrics collected by *CDGym* during the experiments. The defender or analyst can compare the performance of different deception strategies comparing how much the adversaries reacted with deceptive measures. For example, one can clearly see that the attacker's process is delayed by the increased attempts for host scans and decreased exploit attempts against the real hosts, and time spent on reconnaissance. Note that *D1* shows significantly decreased number of host scans and number of exploit attempts against the real hosts, as well as the increased attempts of network scans. These results can be attributed to the impaired decision-making process of the attacker due to the deception strategies.

We also evaluated the performance of *CDGym* with test environments of different sizes. The size of a network refers to the total number of hosts or VMs deployed simultaneously. We deployed fully-configured networks with numbers of hosts ranging between 5 and 40 at an interval of 5 nodes. Note that the test environment for our case studies with *D0* – *D3* is different from the ones deployed for the performance test. The size of environment for the case studies remained constant, while the one for the performance test varied to measure the performance of our implementation in terms of deployment time and memory consumption. Figure 4 (b) illustrates that our implementation of *CDGym* showed linear growth in both deployment time and memory consumption as the number of nodes increased. The performance was similar to deploying the equivalent number of VMs with a hypervisor. However, with the strategy deployment, resource management, and data-collection capabilities of *CDGym* are considered, the performance overhead remains negligible relative to a stand-alone hypervisor.

In addition, *CDGym* can identify scanning attempts and successful exploit of the vulnerable nodes at the specific point in time utilizing the network traffic metrics collected. Figure 5 shows the example of such data. Utilizing this data, we also mapped the behavioral footprint in a temporal sequence by leveraging *Network traffic statistics*, including timestamps, packet counts, and scanning and exploitation attempts, to align each action with its corresponding point in time

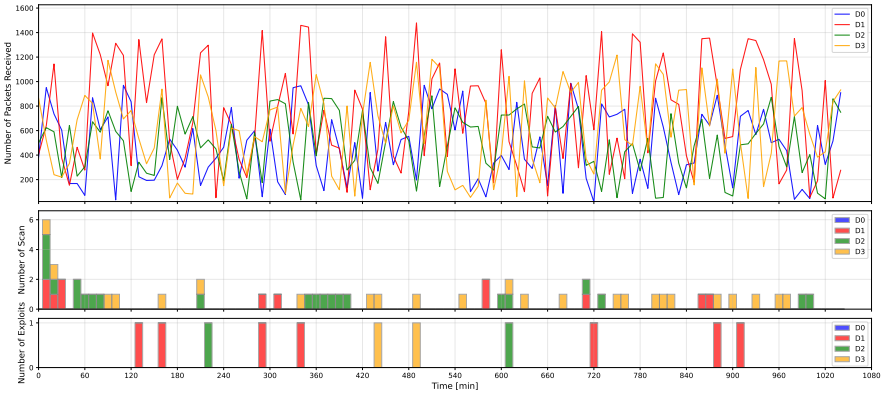
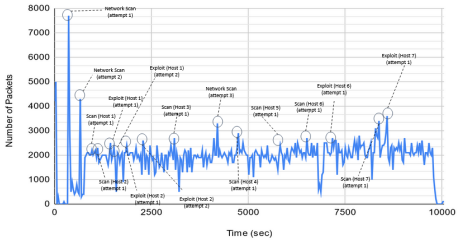


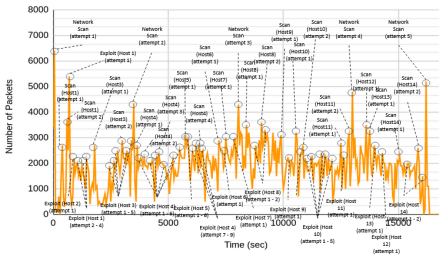
Fig. 5: Data Collected by *CDGym*. The data can be used to perform temporal analysis of the adversarial activities. Combining these data with qualitative analysis such as in-depth interview with user studies may further provide insights into how adversaries made their decisions, at which point such decisions were made, and which deceptive (or non-deceptive) actions of the strategy affected the decision-making process of the adversaries.

during the entire experiment. Figure 6 shows an examples of such footprints with *D1* - *D3* data. This feature offers significant advantages in deception strategy analysis. First, it facilitates data collection by automating the analytical process, thereby greatly enhancing convenience. Second, when used in conjunction with qualitative data, it enables a more balanced and robust analytical approach, unlike conventional methods that often rely heavily on either qualitative or quantitative data alone.

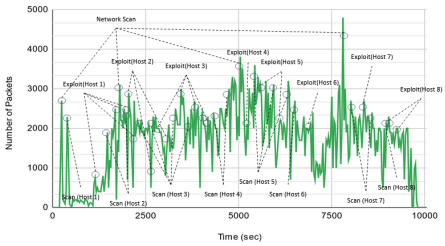
Another example of comparative analysis of the three strategies is shown in Table 3. In addition to the data collected by *CDGym*, p-values for each metric is calculated to compare the effect of deploying deception for each strategy. These p-values indicate the degree of statistical significance of the change in each metric, signifying the probability that the observed results occurred by random chance if there is truly no effect or difference. A p-value less than 0.05 means the observed results are strong evidence against the null hypothesis. In other words, the values indicate that there exist significant changes in the attack progress for *D1* and *D2*, while *D3* did not have significant impact on the attack progress. *CDGym* enables deception researchers and developers to perform a comparative analysis of different strategies. Therefore, the results indicate how the cost of adversarial actions was increased before and after deceptions were deployed. To perform the analysis, we collected additional data including number of login attempts to any systems (*Login attempts*), time taken to start interacting with the target system or service by monitoring the traffic exchanged (*Time to target*), Number of attempts to transfer or exfiltrate data (*Number of data exfiltration*), and total duration of time taken to exfiltrate the data (*Duration for exfiltration*). The result shows that, while *D1* and *D2* show significant increase in efforts to perform adversarial actions, *D3* showed less impact of deception on



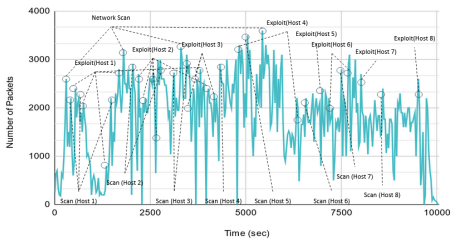
(a) Behavior Analysis with Non-Deceptive Network



(b) Behavior Analysis with D1



(c) Behavior Analysis with D2



(d) Behavior Analysis with D3

Fig. 6: Behavior Analysis using *CDGym* Data

the cost of adversarial actions. Overall, these results may imply that the second strategy (*D2*) is more effective than the first and third in the given environment. Through this exemplary analysis, we attempted to demonstrate that *CDGym* can help researchers and developers to perform a analysis of different strategies and obtain insights in their endeavor to devise innovative deception strategies.

4.3 Scalability and Correctness

Even though *CDGym* can automate the deployment of deceptive networks, its capability to interpret and implement the user input should be proven. The correctness of an implementation can be proven through an empirical analysis when the expected outcome is known, or through a consistency analysis [20].

We conducted both empirical and consistency analysis to prove that our design of *CDGym* can interpret, implement, and deploy the deceptive network environments as intended. Comparing the initial input, we followed the method used by NETCHANGE protocol [50] to check the correctness. The input defined through DCS enables the users to configure each host including system and service, network addresses, open ports, adjacent hosts (links), and the location and configuration of deception mechanisms. Leveraging how NETCHANGE tracks the changes in a network topology allows us to verify the correctness as well as to verify that of any future changes in the network.

We first identify the components of the intended outcome, including those defined in NETCHANGE protocol. We then check if the actual deployment by

Table 3: Heatmap of the p-values for Each Quantitative Metric indicating the significance of differences between before and after deploying the deception.

	Network scans	Host scan	Login attempts	Time to target	Successful exploits	Number of data exfiltration	Duration for exfiltration	Proportion of reconnaissance
D1	Before = 1.57 After = 2.29 $p = 0.107$	Before = 17.85 After = 27.43 $p = 0.036$	Before = 69.71 After = 45.29 $p = 0.024$	Before = 7482.23 sec After = 8826.22 sec $p = 0.029$	Before = 10.35 After = 13.54 $p = 0.032$	Before = 1.86 After = 2.71 $p = 0.188$	Before = 8018.36 sec After = 10091.356 sec $p = 0.026$	Increased by 52.4% $p = 0.004$
D2	Before = 2.24 After = 2.5 $p = 0.309$	Before = 17.85 After = 42.14 $p = 0.003$	Before = 78.79 After = 35.57 $p = 0.004$	Before = 8291.37 sec After = 9237.35 sec $p = 0.038$	Before = 9.52 After = 11.88 $p = 0.033$	Before = 2.07 After = 2.36 $p = 0.390$	Before = 8856.21 sec After = 11372.82 $p = 0.034$	Increased by 42.48% $p = 0.004$
D3	Before = 2.2 After = 2.9 $p = 0.073$	Before = 27 After = 27.8 $p = 0.237$	Before = 98.2 After = 43.9 $p = 0.004$	Before = 9510.38 sec After = 10836.27 sec $p = 0.180$	Before = 6.82 After =5.91 $p = 0.313$	Before = 1.7 After = 0.6 $p = 0.040$	Before = 12521.14 sec After = 11735.56 sec $p = 0.270$	Increased by 14.4% $p = 0.360$

CDGym matches it. The components we checked include the number of nodes, edges, switches, neighboring nodes of changed hosts (if any), how identical the entries of distance table are, and how identical the entries of flow rule table are. The number of neighboring nodes is counted for any node that is moved to different location in the network topology. The node IDs of the neighboring nodes are also verified. The neighboring nodes are checked for the strategies that change the network states dynamically. Therefore, the neighboring nodes are counted for only the Topology Manipulation strategies.

We generated 3 different topologies for four different deception strategies defined in Section 3 (Mixing, Permutation, MTD, and Obfuscation) and compared the above-mentioned items. In addition, we performed the same deployment 30 times each to ensure consistent deployment. Table 4 summarizes the results. The results show that *CDGym* deploys the network as intended, except that it generated 16 more flow rules for the Topology Manipulation strategy. The result is because *CDGym* pushed duplicate flow rules for every possible path to the flow rule tables when new nodes are added or deleted. The intended outcome is still achieved, but the user can specify paths or switches on which the intended flow rules should be applied to. Therefore, the correctness of the outcome is still intact.

4.4 Usability of *CDGym*

Lastly, we conducted a user survey regarding the usability of *CDGym*. We recruited additional 15 participants who are researchers who are currently working in the field of cyber deception. Their research fields include resilient deception adaptive adversarial strategies, adaptive strategy in large-scale networks, and applying different deception mechanisms in specific type of networks, etc. The participants are given access to *CDGym* for a week and conducted a survey to answer the questions regarding its usability.

In the examination of our survey data, we employed the iterative open coding approach to ensure a comprehensive analysis [49]. Through comprehensive analysis, we can perform thorough examination of relevant qualitative factors regarding the usability of our implementation. The relevant factors in this analysis included *Ease of Usage*, *Correctness*, *Coverage*, and *Data Collection* capabilities (Appendix A). In our analysis, two independent coders were engaged to code all the answers to mitigate bias in our qualitative analysis. "Coding" refers to the

Table 4: Results for the correctness of strategy implementation. The table shows the percentage of configurations translated into actual deployment by *CDGym*.

Components	**Intended outcome/Platform's outcome**			
	Camouflage	Honey-X	Mixing	Signaling
Number of Nodes	64/64	64/64	64/64	64/64
Number of Edges	79/79	79/79	79/79	79/79
Number of Switches	15/15	15/15	15/15	15/15
Neighboring Nodes	N/A	N/A	18/18	N/A
Distance Table	100/100	100/100	125/125	100/100
Flow Rule Table	15/15	15/15	**26/42**	25/25

systematic process of assigning labels (known as *codes*) to segments of textual data such as interview responses or open-ended survey answers. These codes represent meaningful concepts, themes, or patterns that the researcher identifies in the data. This step is essential for enabling structured interpretation and supporting subsequent reliability assessments.

The primary researcher was responsible for conducting the survey and collecting the answers from the participants. Following the completion of surveys, the primary researcher developed a codebook by systematically coding the answers. Subsequently, the second coder joined and conducted the separate analysis and produced a conclusion. The second coder was guided by the primary coder, who imparted insights into qualitative research methodologies, coding techniques, and knowledge required to understand the workings of *CDGym*. Upon completion of the coding process, we computed the Cohen's Kappa coefficient as a metric for inter-coder reliability. Unlike simple percent agreement, Kappa incorporates the expected probability of random concordance based on each coder's marginal distributions, making it a more robust indicator of reliability in classification tasks. Computationally, the coefficient is defined as $\kappa = \frac{P_o - P_e}{1 - P_e}$, where P_o represents the observed agreement and P_e denotes the expected agreement under random assignment. A Kappa value of 1 indicates perfect agreement, while a value of 0 corresponds to chance-level agreement. The resulting Cohen's Kappa coefficient for this study is 0.79, which indicates a substantial agreement for categorical data [36].

To assess each factor (i.e., *Ease of Usage*, *Correctness*, *Coverage*, and *Data Collection*), the responses were recorded using a five-point Likert scale [33], where 1 denotes "Highly Unusable" and 5 denotes "Highly Useful". Under this scale, an average score of 4.12 indicates a strong positive assessment of the implementation's usability and perceived effectiveness (Appendix A). Overall, 80% of the participants answered our implementation is *useful* with the average score of 4.12. For the benefits of using *CDGym* (strengths), 80% of the participants answered its capability to automate the deployment of test environments, followed by its data collection capability (60%), and its correctness (40%). On the contrary, for its shortcomings, 90% of the participants answered that it still lacks the coverage for the wide range of deception measures. For example, one participant noted that the platform does not support "*OS- or host-level masking*", a point that was consistently raised across all participants. *CDGym* can be ex-

panded to support those layers, and our future work is to add those layers and accommodate the participants' feedback.

While automation was identified as the most valuable benefit in the survey results, the written responses indicated that participants found the data collection capability of *CDGym* to be its most significant feature. Additionally, two participants answered that ease of usage was the most innovative feature of *CDGym*, as it only requires "*running an input file*" based on DCS.

Further investigation through additional interviews with four participants who highlighted this response revealed that although other existing testbeds (discussed in Section 5) support automation capabilities, the unique data collection functionality is a distinctive feature exclusive to *CDGym*, which is highly helpful. This response offers insights into the necessity of our work, suggesting that existing testbeds lack the functionality required in the field of cyber deception research. It thus supports our problem statement and underscores the need for testbeds such as ours.

In summary, our evaluation demonstrates the efficacy of *CDGym* as a comprehensive platform for assessing the effectiveness of deception strategies based on various game-theoretic models. The data collection capability stands out as a key contribution, underscoring its value in enhancing the precision and applicability of deception strategies.

5 Related Work

There exist a few research efforts to develop a platform for cyber deception experiments and research. We discuss platforms that are similar to *CDGym* and compare it with them in this section.

Cyber Security Virtual Assured Network (CyberVAN) [13] testbed is designed to support general cyber security experimentation by enabling the setup of high-fidelity cyber security scenarios. Key functionalities include transparent IP-layer packet forwarding, scenario creation using simulation models, and time synchronization technology. The testbed aims to support a wide range of cyber experiments, including DDoS experiments, and plans to incorporate human factors research into its capabilities. CyberVAN is mainly leveraged for training and educational purposes [51], but it is also recognized and used by researchers in cyber deception field and leveraged to conduct a research test [10]. However, CyberVAN is designed to test scenario-based security tests and does not support any deception measures, strategy implementation, and data collection.

CONCEAL is a multi-strategy cyber deception framework that argues that no single deception technique (e.g., IP mutation or honeypots alone) is sufficient against skilled reconnaissance adversaries [17]. Instead, effective deception requires an optimal composition of complementary techniques. In order to prove its argument, CONCEAL composes three strategies (IP address, fingerprint anonymization, and heterogeneous decoy services) to improve concealability, detectability, and deterrence. The work also provided extensive scalability and effectiveness evaluation across up to 250 services. Although CONCEAL re-

Table 5: Comparisons of existing testbeds and *CDGym*.
(✔: fully supported, ✔: partially supported, ✘: not supported)

	Reproducibility	Feedback & Reconfiguration	Extensibility	Measurability
CyberVAN [13]	✔	✘	✘	✔
HackIt [8]	✔	✘	✘	✘
CONCEAL [17]	✔	✘	✔	✘
ADF [28]	✔	✔	✔	✔
CDES [7]	✔	✔	✔	✘
CDGym	✔	✔	✔	✔

lies on static attacker models without demonstrating its applicability in various scenarios, it navigated the area of deception strategy optimization through the combination of multiple deception types.

Active Deception Framework (ADF) is proposed to address the lack of management for adaptive cyber deception by designing a set of high-level APIs [28]. The modular development of APIs for different types of deception mechanisms enhances extensibility of deception implementation. By enabling modular development of APIs for different classes of deception mechanisms, ADF enhances the extensibility of deception implementations. While ADF represents a meaningful advancement toward automated and extensible deception systems, it primarily focuses on the control and orchestration of deception mechanisms and does not incorporate strategic efficiency into its design.

Another platform for deception research is HackIt [8]. HackIt is a real-time simulation tool that is used to study real-world cyberattacks in a laboratory setting. It allows researchers to create dynamic cyberattack scenarios and investigate the behavior of attackers and defenders in complex cyber situations. HackIt simulates the two phases of hacking websites: probing for vulnerabilities and attacking computer workstations. In the probe phase, the tool scans web servers in the network for vulnerabilities. In the attack phase, it involves gaining access to different computers and stealing information or compromising computer systems. The tool provides the flexibility to create networks of different sizes, configure honeypots, and set up various fictitious ports, services, operating systems, and files on honeypots. Even though HackIt provides a realistic environment, it is still confined to a specific scenario, which is an attacking web server, and therefore, lacks comprehensiveness.

CDES [7] is another system dedicated to testing dynamic deception strategies. The specific aims of CDES include the development of a testbed that offers a realistic, scalable, and controlled environment suitable for dynamic deception. This also involves enhancing the effectiveness of honeypots against sophisticated adversaries by making them more unpredictable and targeted. The capabilities of CDES are showcased through three distinct scenarios: gateway mimicking, multi-gate, and intrusion detection. However, CDES exhibits limitations in adaptability, particularly noted in its inability to efficiently deploy various types of strategies. Additionally, it lacks a robust data collection feature, thereby limiting its measurability and the comprehensive evaluation of deception strategies.

Table 5 presents a summary of the comparisons between *CDGym* and existing testbeds, organized according to the identified categories. In our work, we have designed DCS supporting reproducibility, feedback & reconfiguration, extensibility, and measurability by enabling users to efficiently define and verify the test environment and deception mechanisms.

6 Discussion

Our implementation of *CDGym* showed that it can enhance the efficiency in diverse types of deception strategy analysis, along with high usability and data collection capability. Despite these features, however, our work has some limitations. We discuss these limitations in this section.

Our prototype primarily supports network-based deception techniques and strategies, lacking host-level deception mechanisms to implement all types of deceptions categorized by Pawlick et al. [42]. For example, a fake configuration of an operating system to make it appear as a different one (e.g., disguising Windows systems as Linux) is not supported. Although DCS is designed to be capable of defining such configurations, our implementation is still in its early stages. We are actively working to incorporate this capability into our implementation as part of our future development efforts.

It is also important to note that our objective is to demonstrate the efficacy of *CDGym* as a means to enhance research efforts in the field of cyber deception, rather than to showcase the effectiveness of the specific strategies employed during our evaluation. Consequently, the actual effectiveness of the strategies implemented in this study and influence of associated variables on the observed results (e.g., how the skillset of each participant impacts the effectiveness of each strategy) fall outside the scope of this work.

7 Conclusion

This paper presents the design of a cyber deception analysis platform named *CDGym*. *CDGym* is devised to reduce the gap between abstract design of deception strategies and their implementation in the real world. In order to make the platform agnostic to a specific strategy, we proposed a deception configuration schema, DCS that enables the users to define strategy and required deception measures. We also demonstrated the efficacy and usability of *CDGym* by conducting case studies with different deception strategies, each of which requires different configurations. Based on our evaluation, *CDGym* can deploy network environments as intended, seamlessly collect quantitative data in real-time, supporting various deception measures.

Acknowledgments. This work was partially supported by the grants from US National Science Foundation (NSF-CICI-2232911 and NSF-SFS-1663651) and the Institute of Information & Communications Technology Planning & Evaluation (IITP) (RS-2024-004398199 and RS-2024-00442085).

References

1. ELK stack. https://www.elastic.co/elastic-stack, [Online; accessed 2024-1-14]
2. Lyrebird Honeypot. https://hub.docker.com/r/lyrebird/honeypot-base/, [Online; accessed 2025-7-21]
3. NETCHANGE. https://efficientip.com/products/netchange/, [Online; accessed 2024-3-8]
4. Open Network Operating System (ONOS). https://opennetworking.org/onos/
5. Snort. https://www.snort.org/, [Online; accessed 2024-4-28]
6. Achleitner, S., La Porta, T., McDaniel, P., Sugrim, S., Krishnamurthy, S.V., Chadha, R.: Cyber deception: Virtual networks to defend insider reconnaissance. In: Proceedings of the 8th ACM CCS international workshop on managing insider security threats. pp. 57–68 (2016)
7. Acosta, J.C., Basak, A., Kiekintveld, C., Leslie, N., Kamhoua, C.: Cybersecurity deception experimentation system. In: 2020 IEEE Secure Development (SecDev). pp. 34–40. IEEE (2020)
8. Aggarwal, P., Gonzalez, C., Dutt, V.: Hackit: a real-time simulation tool for studying real-world cyberattacks in the laboratory. In: Handbook of Computer Networks and Cyber Security, pp. 949–959. Springer (2020)
9. Aggarwal, P., Jabbari, S., Thakoor, O., Cranford, E.A., Vayanos, P., Lebiere, C., Tambe, M., Gonzalez, C.: Human-subject experiments on risk-based cyber camouflage games. In: Cyber Deception: Techniques, Strategies, and Human Aspects, pp. 25–40. Springer (2022)
10. Aggarwal, P., Thakoor, O., Mate, A., Tambe, M., Cranford, E.A., Lebiere, C., Gonzalez, C.: An exploratory study of a masking strategy of cyberdeception using cybervan. In: Proceedings of the human factors and ergonomics society annual meeting. vol. 64, pp. 446–450. SAGE Publications Sage CA: Los Angeles, CA (2020)
11. Alshammari, A., Rawat, D.B., Garuba, M., Kamhoua, C.A., Njilla, L.L.: Deception for cyber adversaries: status, challenges, and perspectives. Modeling and Design of Secure Internet of Things pp. 141–160 (2020)
12. Ashenden, D., Black, R., Reid, I., Henderson, S.: Design thinking for cyber deception (2021)
13. Chadha, R., Bowen, T., Chiang, C.Y.J., Gottlieb, Y.M., Poylisher, A., Sapello, A., Serban, C., Sugrim, S., Walther, G., Marvel, L.M., et al.: Cybervan: A cyber security virtual assured network testbed. In: MILCOM 2016-2016 IEEE Military Communications Conference. pp. 1125–1130. IEEE (2016)
14. Cranford, E.A., Gonzalez, C., Aggarwal, P., Cooney, S., Tambe, M., Lebiere, C.: Toward personalized deceptive signaling for cyber defense using cognitive models. Topics in Cognitive Science **12**(3), 992–1011 (2020)
15. Cranford, E.A., Gonzalez, C., Aggarwal, P., Tambe, M., Cooney, S., Lebiere, C.: Towards a cognitive theory of cyber deception. Cognitive Science **45**(7), e13013 (2021)
16. De Faveri, C., Moreira, A.: Designing adaptive deception strategies. In: 2016 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C). pp. 77–84. IEEE (2016)
17. Duan, Q., Al-Shaer, E., Islam, M., Jafarian, H.: Conceal: A strategy composition for resilient cyber deception-framework, metrics and deployment. In: 2018 IEEE Conference on Communications and Network Security (CNS). pp. 1–9. IEEE (2018)

18. Ferguson-Walter, K., Fugate, S., Mauger, J., Major, M.: Game theory for adaptive defensive cyber deception. In: Proceedings of the 6th Annual Symposium on Hot Topics in the Science of Security. pp. 1–8 (2019)
19. Gao, C., Wang, Y., Xiong, X.: Comparison of defense effectiveness between moving target defense and cyber deception defense. In: 2021 4th International Conference on Data Science and Information Technology. pp. 119–124 (2021)
20. Ghorbani, S., Godfrey, B.: Towards correct network virtualization. ACM SIGCOMM Computer Communication Review **44**(4) (2014)
21. Gonzalez, C., Aggarwal, P., Cranford, E.A., Lebiere, C.: Adaptive cyberdefense with deception: A human–ai cognitive approach. Cyber Deception: Techniques, Strategies, and Human Aspects pp. 41–57 (2022)
22. Goodman, L.A.: Snowball sampling. The annals of mathematical statistics pp. 148–170 (1961)
23. Han, X., Kheir, N., Balzarotti, D.: Deception techniques in computer security: A research perspective. ACM Computing Surveys (CSUR) **51**(4), 1–36 (2018)
24. Hemida, A., Asghar, A.B., Kamhoua, C., Kleinberg, J.: A game theoretic framework for multi domain cyber deception. In: 2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). pp. 443–447. IEEE (2024)
25. Hong, Q., Li, J., Guo, X., Xie, P., Zhai, L.: Assessing the effectiveness of deception-based cyber defense with cyberbattlesim. In: International Conference on Digital Forensics and Cyber Crime. pp. 224–243. Springer (2023)
26. Horák, K., Bošanskỳ, B., Pěchouček, M.: Heuristic search value iteration for one-sided partially observable stochastic games. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 31 (2017)
27. Horák, K., Zhu, Q., Bošanskỳ, B.: 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. pp. 273–294. Springer (2017)
28. Islam, M.M., Al-Shaer, E.: Active deception framework: An extensible development environment for adaptive cyber deception. In: 2020 IEEE Secure Development (SecDev). pp. 41–48. IEEE (2020)
29. Iyer, K.I.: Adaptive honeypots: Dynamic deception tactics in modern cyber defense (2021)
30. Javadpour, A., Ja'Fari, F., Taleb, T., Benzaïd, C.: A mathematical model for analyzing honeynets and their cyber deception techniques. In: 2023 27th International Conference on Engineering of Complex Computer Systems (ICECCS). pp. 81–88. IEEE (2023)
31. Jia, G., Wang, X., Liu, H., Wu, H.: A deception defense strategy based on game theory. In: 2024 IEEE International Conference on Security, Privacy, Anonymity in Computation and Communication and Storage (SpaCCS). pp. 27–33. IEEE (2024)
32. Johnson, C.K., Gutzwiller, R.S., Gervais, J., Ferguson-Walter, K.J.: Decision-making biases and cyber attackers. In: 2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW). pp. 140–144. IEEE (2021)
33. Joshi, A., Kale, S., Chandel, S., Pal, D.K.: Likert scale: Explored and explained. British journal of applied science & technology **7**(4), 396 (2015)
34. Kokulu, F.B., Soneji, A., Bao, T., Shoshitaishvili, Y., Zhao, Z., Doupé, A., Ahn, G.J.: Matched and mismatched socs: A qualitative study on security operations center issues. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 1955–1970 (2019)

35. Kyung, S., Han, W., Tiwari, N., Dixit, V.H., Srinivas, L., Zhao, Z., Doupé, A., Ahn, G.J.: Honeyproxy: Design and implementation of next-generation honeynet via sdn. In: 2017 IEEE Conference on Communications and Network Security (CNS). pp. 1–9. IEEE (2017)
36. Landis, J.R., Koch, G.G.: The measurement of observer agreement for categorical data. biometrics pp. 159–174 (1977)
37. Li, T., Zhu, Q.: Symbiotic game and foundation models for cyber deception operations in strategic cyber warfare. arXiv preprint arXiv:2403.10570 (2024)
38. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: Openflow: enabling innovation in campus networks. ACM SIGCOMM computer communication review **38**(2), 69–74 (2008)
39. Mohan, P.V., Dixit, S., Gyaneshwar, A., Chadha, U., Srinivasan, K., Seo, J.T.: Leveraging computational intelligence techniques for defensive deception: a review, recent advances, open problems and future directions. Sensors **22**(6), 2194 (2022)
40. Nath, S., Soneji, A., Baek, J., Bao, T., Doupé, A., Rubio-Medrano, C., Ahn, G.J.: "It's almost like Frankenstein": Investigating the Complexities of Scientific Collaboration and Privilege Management within Research Computing Infrastructures . In: 2025 IEEE Symposium on Security and Privacy (SP). pp. 2995–3013 (May 2025). https://doi.org/10.1109/SP61157.2025.00197, https://doi.ieeecomputersociety.org/10.1109/SP61157.2025.00197
41. Pawlick, J., Colbert, E., Zhu, Q.: A game-theoretic taxonomy and survey of defensive deception for cybersecurity and privacy. ACM Computing Surveys (CSUR) **52**(4), 1–28 (2019)
42. Pawlick, J., Zhu, Q.: A taxonomy of defensive deception. Game Theory for Cyber Deception: From Theory to Applications pp. 37–48 (2021)
43. Píbil, R., Lisỳ, V., Kiekintveld, C., Bošanskỳ, B., Pěchouček, M.: Game theoretic model of strategic honeypot selection in computer networks. In: International Conference on Decision and Game Theory for Security. pp. 201–220. Springer (2012)
44. Piotrowska, A.M., Hayes, J., Elahi, T., Meiser, S., Danezis, G.: The loopix anonymity system. In: 26th USENIX Security Symposium (USENIX Security 17). pp. 1199–1216. USENIX Association, Vancouver, BC (Aug 2017), https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/piotrowska
45. Rawat, D.B., Sapavath, N., Song, M.: 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. pp. 401–406 (2019)
46. Sasahara, H., Sandberg, H.: Epistemic signaling games for cyber deception with asymmetric recognition. IEEE Control Systems Letters **6**, 854–859 (2021)
47. Sayed, M.A., Anwar, A.H., Kiekintveld, C., Kamhoua, C.: Honeypot allocation for cyber deception in dynamic tactical networks: A game theoretic approach. In: International Conference on Decision and Game Theory for Security. pp. 195–214. Springer (2023)
48. Srinivasa, S., Pedersen, J.M., Vasilomanolakis, E.: Towards systematic honeytoken fingerprinting. In: 13th International Conference on Security of Information and Networks. pp. 1–5 (2020)
49. Strauss, A., Corbin, J.: Basics of qualitative research: Procedures and techniques for developing grounded theory (1998)
50. Tajibnapis, W.D.: A correctness proof of a topology information maintenance protocol for a distributed computer network. Communications of the ACM **20**(7), 477–485 (1977)

51. Venkatesan, S., Youzwak, J.A., Chiang, C.Y.J., Sugrim, S., Poylisher, A., Witkowski, M., Walther, G., Wolberg, M., Chadha, R., Newcomb, E.A., et al.: Vulnervan: Automating the generation of vulnerable network scenarios for cybersecurity events and training exercises. Tech. rep., Section: Technical Reports (2021)
52. Wang, C., Lu, Z.: Cyber deception: Overview and the road ahead. IEEE Security & Privacy **16**(2), 80–85 (2018)
53. Zhang, L., Thing, V.L.: Three decades of deception techniques in active cyber defense-retrospect and outlook. Computers & Security **106**, 102288 (2021)
54. Zhu, M., Anwar, A.H., Wan, Z., Cho, J.H., Kamhoua, C.A., Singh, M.P.: A survey of defensive deception: Approaches using game theory and machine learning. IEEE Communications Surveys & Tutorials **23**(4), 2460–2493 (2021)

Appendix

Appendix A. Usability Survey Questions for *CDGym*.

1. Are you currently working on/researching cyber deception?
   (a) If YES: Have you ever implemented a network-based cyber deception strategy?
   (b) If NO: What is your area of research? Is it related to cyber deception?

2. Which data metrics do you require (or expect to be collected) for your cyber deception research?

3. How do you implement your testbed?

4. Do you think CDGym is significantly different from your testbed?

   (a) If YES: What do you think are the differences?
   (b) If NO: Is there anything you wish CDGym provided?

5. **Ease of Usage:** On a scale of 1 to 5, 1 being Highly Unusable and 5 Highly Useful, how would you rate CDGym in terms of Automation? In other words, was it easy to deploy your network successfully using CDGym in an automated fashion?

6. **Correctness:** On a scale of 1 to 5, 1 being Highly Unusable and 5 Highly Useful, how would you rate CDGym in terms of Network Deployment/Management? In other words, was the network correctly deployed and changed its status using CDGym?

7. **Coverage:** On a scale of 1 to 5, 1 being Highly Unusable and 5 Highly Useful, how would you rate CDGym in terms of Support for Deception Mechanisms? In other words, did CDGym cover the necessary deception mechanisms required for cyber deception strategy research?

8. **Data Collection:** On a scale of 1 to 5, 1 being Highly Unusable and 5 Highly Useful, how would you rate CDGym in terms of Data Collection? In other words, Did CDGym collect sufficient data required for cyber deception strategy research?

9. On a scale of 1 - 5, 1 being Highly unusable and 5 Highly useful, how would you rate the Overall Performance CDGym?

10. Which of the following(s) do you think are the strengths of CDGym?
    (a) Ease of Usage
    (b) Correctness
    (c) Coverage

(d) Data Collection
(e) Other

11. Which of the following(s) do you think are the weaknesses of CDGym?
    (a) Ease of Usage
    (b) Correctness
    (c) Coverage
    (d) Data Collection
    (e) Other

12. Please provide any suggestion of feedback to improve *CDGym*.