TangleNet: An advanced cyber deception model based on reinforcement learning

Tahani GAZDAR*, Lara MURAD, Safana AL-JAHDALI, Fai ZWAWI, Reman MANDILI

Cybersecurity Department, College of Computer Science and Engineering University of Jeddah, Jeddah, Saudi Arabia

***Corresponding Author:** Tahani GAZDAR

taalgazdar@uj.edu.sa

Abstract: Detecting network attacks is becoming a challenging task given that they are getting more complex. Almost all modern security systems can identify attacks in the early and final, but not in the middle stages. This limitation is caused by the fact that the detection of the nature of an attacker requires a deep investigation of the attack pattern. It means that the detection of the attack after its occurrence is useless since the harm has already been inflicted. Early signs can give quite a lot of false positive results. Consequently, cyber deception strategies are applied to fill this gap and improve the blue team's knowledge of the attackers' fundamental strategy. This research introduces an integrated cyber deception system called TangleNet that incorporates reinforcement learning to build a research honeynet that simulates real servers and attacker activity. The outcome of the experimentation conducted using Microsoft's CyberBattle platform and the Mininet library shows the level of deception of the cyber deception model is correlated to the number of commands executed by the attacker. This enhanced effectiveness extends the interactiveness time and allows for the tracking of potentially hostile entities with little dependency on human intervention.

Keywords: Cyber Deception, Reinforcement Learning, Agent, Action, Q-Learning.

1. Introduction

The security community continues to face threats from the growing frequency of sophisticated cyberattacks and subsequent disruption to the systems. Current security solutions frequently focus their detection on recognizing threats in discrete incidents, which is insufficient for identifying complex multi-stage attacks. The challenge of recognizing cyber-attacks has taken a new shift because of several factors. Specifically, cyber attackers are using new trends, which are multi-stage and composite attacks that cannot be captured by conventional detection solutions like firewalls, intrusion detection systems, or even traditional honeypots. Moreover, most of the attackers use stealth techniques such as encryption, disguising as other processes, or polymorphism with which to evade detection by the security systems. The computing environment also becomes heterogeneous due to which they are more complex in nature. Therefore, it is challenging to monitor their presence and prevent unauthorized access (Biplob, Marma & Akther, 2024). Nearly all security solutions available in the market can detect attacks in their initiation phase as well as final stages, but none address the middle stages. This limitation is well justified because identifying an attacker's primary methodologies demands further research. Finding that an attack has occurred once it has happened is of little use since the damage has already been done. Moreover, the issue of early detection can be very deceptive because the system often produces a high false positive and may change based on the network and environment. The cyber deception solutions are used by blue teams to bridge this gap and expand their perspective on attack patterns and techniques. Cyber-deception aims to provide fake and misleading information and create a kind of allusion to the attacker about the target system (Javadpour et al., 2024). The blue team deploys decoy systems to attract attackers and interact with them to gather knowledge about their tactics, techniques, and procedures (TTPs). Cyber deception techniques are efficient in detecting zero-day attacks with a nearly zero false positive rate because only attackers interact with a decoy system (Javadpour et al., 2024). Although conventional honeypots provide a certain level of defense, the attackers use sophisticated methods and techniques and this makes these simple decoy systems lose their effectiveness. A major drawback of the conventional decoy systems is the fact that the attackers can easily notice them due to their fingerprinting which depends on the specific and predictable

https://doi.org/10.33436/v35i1y202502

behavior of the decoy systems. Furthermore, most conventional honeypots are developed to interact with known attack scenarios, which limits their applicability. For example, many decoy systems only address the SSH behavioral scenarios since they can easily mimic the attack and since it is among popular targets. However, this somehow dismisses other vulnerabilities in the network protocols such as FTP and HTTP which are also commonly used. Additionally, most existing solutions rely on datasets and this decreases their interactivity and deceptiveness. Therefore, a new conceptual model for the design of such models is imperative. This paper presents TangleNet, a new research cyberdeception system that utilizes Reinforcement Learning (RL) to overcome the aforementioned limitations. TangleNet aims to provide an active, heterogeneous, highly interactive cyber deception system by merging the concept of deception with the adaptability of Reinforcement Learning techniques. Reinforcement learning (RL) allows an agent to learn through trial-and-error feedback in an interactive environment (Maddireddy Bharat Reddy & Maddireddy, Bhargava Reddy, 2024). Applying RL in cyber deception makes the decoy system more efficient in distracting the attackers' attention away from the real target. Thus, it extends the duration of interaction between the attacker and the target environment to collect more information about the attacker's behavior and tactics. Additionally, it minimizes the human intervention involved in recognizing potential intruders (Veluchamy & Kathavarayan, 2021).

TangleNet learns from past interactions with the attackers, adjusts its reaction according to the attack behavior, and optimizes its response over time. TangleNet can also collect information regarding the attacks' behavior like the used commands, and the time to compromise a server or to exploit a vulnerability. This information is invaluable for the defenders to enhance the security measures implemented in the operational environment (Javadpour et al., 2024). Unlike the existing cyberdeception solutions designed for only a specific server such as SSH, HTTP, etc, TangleNet is a heterogeneous cyberdeception system that consists of many decoy servers: SSH, FTP, SMTP, and a database server. More importantly, TangleNet goes a step forward compared to conventional honeypots because it is dynamic, particularly it adjusts its reaction to the attacker's action, unlike conventional honeypots which react in the same way independently of the attacker's behavior or action.

The paper is organized as follows: Section 2 presents a background on cyber deception, honeypots, Reinforcement Learning, and the OODA loop framework. Section 3 provides the related work that has been done to advance the field of cyber deception. Section 4 illustrates the design of the proposed Reinforcement Learning model. Section 5 demonstrates the experimentation environment setup and presents the obtained results. Section 6 concludes the paper and presents the future work.

2. Background

2.1. Cyber deception

Deception techniques have been employed for various purposes including military, espionage, and political propaganda. Deceptive tactics make attackers doubt their skills and judgment, unlike conventional defenses, which are frequently predictable and simple to overcome. In modern warfare, cyber deception is used for defensive purposes to detect and respond to cyber-attacks or for offensive purposes to gather intelligence (Soule et al., 2016). Cyber deception systems run on top of the OODA Loop framework (Priambodo et al., 2022). This loop consists of four phases: observe, orient, decide, and act as presented. It refers to an iterative decision-making process through which entities observe evolving information, contextualize it, decide on the next steps, implement an action plan, and adopt a strategy based on the observations and results obtained.

However, the effectiveness of cyber deception solutions depends on their ability to create realistic scenarios and maintain trickery over a long period (Brasoveanu, Moodie & Agrawal, 2020). Deceptive systems are designed to imitate real targets, such as servers, to allow defenders to capture, record, and observe the behavior of the attackers without their knowledge (Javadpour et al., 2024).

Cyber deception solutions entail deploying a honeypot which is a fake system or network aimed at drawing the attention and lure off attackers right away from the operational computing environment (Reynolds & Green, 2023). The types of data collected by honeypots include instructive and statistical information such as the top shell commands executed, tools downloaded, malware signatures, the operating system commonly used by attackers, the commonly used credentials, the classification of attacks by country, a list of attacker source IP addresses and associated ASNs (Autonomous Systems numbers), and so on. This information is analyzed later to detect the attack patterns and enhance the defense measures in the operational environment.

2.2. Reinforcement learning

Reinforcement learning (RL) is a machine learning technique that enables an agent to learn in an interactive environment by trial and error using feedback from its actions and experiences (Kadam & Ahmed, 2024). Further, RL finds its application in gameplay and development to create smarter NPCs that can learn new things from the field. Applying the concept in cyber deception makes attacks hard, complex, and lengthy tasks for attackers (Javadpour et al., 2024). Researchers have applied Reinforcement Learning in creating smart agents for use in cybersecurity aiming to detect and combat cyber security threats. For instance, the attackers will use one or many strategies to bypass the discovery and detection controls, so modern RL algorithms can adapt their reaction according to the dynamics of the attacker's actions and by considering the known patterns of attacks (Kadam & Ahmed, 2024).

Figure 1 shows the components of an RL model and their interactions. The design of an RL model involves defining the agent's actions, rewards, and states, in addition to selecting the appropriate algorithm to train the agent.

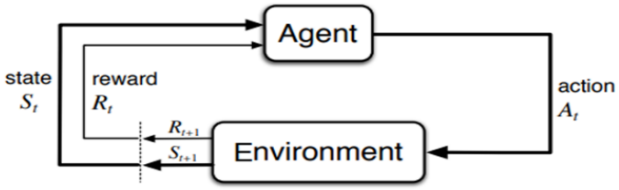


Figure 1. Design Components

The environment: it describes where the agent is expected to run. It consists of the state space, the action space, and the reward function. The tasks that the agent can carry out and how they will be paid must be specified in a way that motivates the agent to behave desirably and correctly and this is according to the objectives of the RL model:

- **State:** the observation that the agent makes on the environment; it could be a command entered by the attacker.
- **Action:** the agent acts on the environment according to the observation, for example allow, block, or substitute the command entered by the attacker.
- **Reward:** the feedback that the agent gets when it acts on the environment. If the action is positive, it gets a reward, otherwise, it gets a penalty.
- **Episode:** a single run or trial of the agent interacting with the environment to learn and improve its decision-making policy.
- **The agent:** The agent is the learner. It must find out autonomously how to act/react to the environment. The aim is to choose the most accurate actions that lead to a high cumulative reward.
- **Algorithms:** Agents are trained to learn how to behave depending on the rewards and punishments from their environment by using Reinforcement Learning (RL) algorithms. There are various RL algorithms such as Actor-Critical, deep Q-networks (DQN), Proximal Policy Optimization (PPO), Q-Learning, and SARSA. Q-learning algorithm is the most suitable for TangleNet. It is an off-policy Reinforcement Learning algorithm that immediately selects the best action, given a current state, out of all the possible actions.

The purpose is to identify the best series of activities based on the current state. To achieve this, the model develops a set of rules to reach the highest reward. The algorithm runs based on a matrix (state, action, reward) called Q-table. The Q-table describes for each pair of a state and an action (s, a) the reward associated with it. After that the agent takes either the action that has the maximum value of reward or a random action based on the ε-greedy method (ε, ε), and then it appends the new pair (action, state) to the Q-Table (Chadi & Mousannif, 2023).

3. Literature review

In this section, several studies have been reviewed. All the studies suggest adaptable cyber deception systems that utilize different algorithms with different focuses and objectives. Then a comparison of the proposed approaches has been conducted based on the design approach, type of system (specific or generic), level of interaction, level of deceptiveness, and whether it depends on datasets.

Many of the proposed cyber deception techniques are based on generic Machine Learning models, while few are based on RL. A cyber deception tool can be either heterogeneous, and works with various protocols and systems, or homogenous and customized to a single type of protocol, such as SSH or Web (HTTP). Regarding the level of interaction, cyberdeception systems are divided into three categories: high, medium, and low interaction. Conventional cyber deception techniques usually react to the attacker's behavior following a predefined and limited set of actions depending on the attack scenario and the attacker usually detects the decoy system due to its fingerprint as discussed above. For RL systems, the reaction of the RL agent is less likely to be expected by the attacker, because it depends on some observation done by the RL agent that leads to a dynamic state of the environment which results in a high interaction cyber deception system due to the action space of the RL agent. Furthermore, the level of deception exponentially rises when techniques like RL are used. Ghourabi, Abbes & Bouhoula (2011) proposed using honeypots to detect and study attacks against Web services. In the proposed solution, they deploy a honeypot as a web service application. This decoy system captures all request messages and analyzes them using ML techniques to detect the attacks.

Pauna & Patriciu (2014) developed an artificially intelligent model based on case-based reasoning (CBR). Their adaptive model was inspired by cognitive science and aimed to mimic human behavior. Case-based reasoning activities are sometimes divided into two categories: interpretation and problem-solving. Hence, it classifies or defines the current situation using precedents from earlier cases.

Dowling, Schukat & Barrett (2019) proposed an adaptive honeypot that learns the best answers to attack commands and overcomes the honeypot detection techniques incorporated into malware deployments. The suggested honeypot employs a state action space formalism to reward the learner for extending attack interactions, and it learns from repeated and automated attack or compromise attempts. They have shown that when compared to the normal high interaction honeypot, the adaptive honeypot captured a larger dataset with four times more attack command transitions after the initial learning phase. Torino (2021) sought to address the limitation of the Cowrie SSH honeypot (Cabral et al., 2019) by developing the Cannypot, a configurable honeypot that employs RL algorithms. Q-learning algorithm is embodied in the structure of Cannypot. The model is based on a database that stores the results of commands as a dictionary. A back-end service supplies a list of possible outputs to be appended to the dictionary and a front-end maintains the list of unidentified commands. The greatest disadvantage of this model is the amount of storage space required for the dictionary. Navarro (2021) has introduced a honeypot ‘Python RASSH’ which clones Kippo, a non-adaptive honeypot defeated by Cowrie (Medium-interaction honeypot). It is based on the SARSA algorithm, which is an e-greedy strategy and there is a new action called ‘delay’ in the algorithm which postpones the execution of the orders entered by the attacker. Also, unlike some other frameworks, it changes the action of a “substitute command” with the concept of a “fake command” which gives the attacker a wrong output.

Touch & Colin (2022) proposed an adaptive self-guarded honeypot: Asgard. It employs RL to monitor the tools and actions of the attackers and offers protection against deep compromise. The authors then contrast Asgard to the standard SSH honeypot Cowrie. Unlike Cowrie, Asgard could collect better quality information about the attacker's behavior at the time it could bypass detection techniques. One more advantage of Asgard compared to Cowrie is the fact that it could defend the system for as long as it could either through blocking or shading malicious programs or commands. Kunz et al., (2023) proposed MARLon: Multi-Agent Training ON CyberBattleSim. CyberBattale is a Python OpenAI Gym environment used to model network security issues (Walter, Ferguson-Walter & Ridle, 2021). The environment created by CyberBattleSim is a network of servers connected by several network connection protocols. The multi-agents model consists of two agents, a defender, and an attacker. They are trained to operate in the network environment using the proximal policy optimization PPO algorithm at the same time. The authors describe the changes and report on the results obtained when training the defending agents, either isolated or jointly with the attacker agents in the same environment.

In this research, the TangleNet framework is proposed. It uses Reinforcement Learning and advanced deception techniques to simulate a realistic multi-server network environment, dynamically engaging attackers to extend interactions, improve monitoring, and reduce human intervention while offering deep insights into the attacker's behavior. In TangleNet, the RL agent leverages the Q-learning algorithm to make realistic decisions: Allow, Block, or Substitute, based on the network's current state, causing transitions to new states for dynamic cyber security. Table 1 summarizes and compares the characteristics of the existing cyber deception models with TangleNet.

Table 1. Gap Analysis

Ref.	Approach	Specific or generic	Level of interaction	Level of Deceptiveness	Dataset	Algorithm
Ghourabi, Abbes & Bouhoula (2011)	ML	Web	M	M	✓	SVlvI, Apriori
Pauna & Patriciu (2014)	AI	SSH	M	M	✓	-
Dowling, Schukat & Barrett (2019)	RL	SSH	H	H	-	SARSA
Torino (2021)	RL	SSH	H	H	✓	Q-Learning
Navarro (2021)	RL	SSH	H	H	-	SARSA
Touch & Colin, 2022	RL	SSH	H	H	-	Q-Learning
Kunz et al., (2023)	RL	SSH, FTP, Mail, Database server	H	H	-	PPO
TangleNet	RL	SSH, FTP, Mail, Database server	H	H	-	Q-Learning

4. Proposed solution: TangleNet

4.1. Overview

TangleNet integrates the essence of deception in the cybersecurity field with the power of Reinforcement Learning to create a highly effective and efficient deception strategy that provides the blue team with visibility into the attacker's core techniques. The objective of TangleNet is to prolong and extend the attacker's interaction with the cyberdeception system by responding

dynamically, thereby elevating the level of deception, ensuring comprehensive monitoring, and minimizing human intervention.

In TangleNet, the considered network topology closely simulates real-life network environments. As illustrated in Figure 2, the topology comprises four servers: SSH server, FTP server, Mail server, and Database server. This will make TangleNet heterogenous, unlike conventional cyberdeception where only one server behavior is implemented. These servers are typical of the services that can be found in most networks. These types of servers are intended to develop an ideal condition that exhibits a variety of network behaviors and weaknesses. When an attacker tries to communicate with such servers using commands, the cyber deception starts by intercepting and analyzing. It then decides its action, whether to allow the attacker to continue attacking the current server, search for another server, or move to attack another server remotely. The learning process of the RL agent starts with the perception of the environment in this case the attacker's commands. It is out of this observation that the agent can learn and come up with relevant decisions about the course of action to undertake about a certain command.

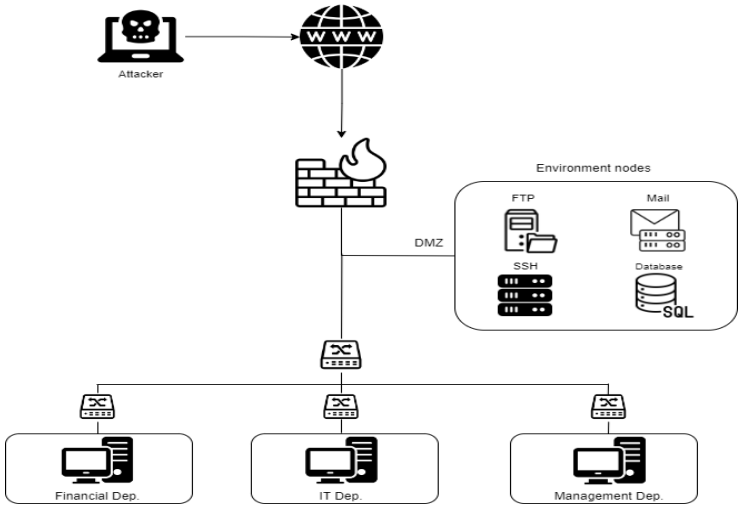


Figure 2. Network topology

4.2. TangleNet design and implementation

The design phase is crucial in the RL process. It comprises setting up the environment, defining the agent's actions and rewards, and selecting the most effective algorithm to train the agent. In this section, different RL model components are presented followed by the implementation and evaluation of TangleNet.

4.2.1. Environment

The CyberBattle platform (Walter, Ferguson-Walter & Ridle, 2021) and Mininet library Mininet/mininet 2.3.0 (2021) are used to create a realistic network environment. CyberBattle is a platform designed by Microsoft to simulate cyber-attack scenarios. Mininet is a network simulator that allows the creation of virtual network topologies. Mininet is useful in designing the network structure, configuring servers, and simulating the network interactions. This gives control over the network environment and facilitates the evaluation of the performance of TangleNet by mimicking real-life network scenarios. Combining the capabilities of the CyberBattle platform and the Mininet library helps obtain a robust and realistic network simulation environment.

In the simulated environment, four servers are considered: SSH, FTP, Mail, and Database servers, as illustrated in Figure 2. Each server was assigned specific functionalities and vulnerabilities to replicate real-world network scenarios. Each server is designed to have a number of vulnerabilities named *VulCounterX, X denotes the servers: SSH,FTP,Mail,DB*. A flag is assigned to each vulnerability, once it is exploited, this flag is checked by the RL agent, and *VulCounterX is decremented*.

The attacker's behavior consists of executing commands (shell commands: ls, cat, wget, pwd, ssh, nmap, dd, chmod or some shell scripts) on the different servers present in the network scenario. The attacker executes the command to achieve one of the following goals: (1) exploiting a vulnerability in the current server where the attacker resides, (2) exploiting a vulnerability in a remote server from the current server (from the location of the attacker), (3) discovering a new server, or (4) preparing for the exploitation phase in the target server.

Consequently, in TangleNet, the attacker enters commands that can be mapped to one of the following categories:

- Exploitation commands (denoted E): any command used to execute a malicious code in the target server (local or remote);
- Discovery commands (denoted D): any command used to discover the network, particularly the servers in the network;
- Other commands (denoted O): any other commands that cannot be mapped to E or D. It comprises all other commands used to prepare the exploitation of the vulnerabilities.

TangleNet is preloaded with a huge set of commands categorized into E, D, and O. This categorization is based on many threat intelligence sources and the analysis of different commands that may be used by the attacker on different servers. Each time the attacker executes a command, TangleNet first categorizes the command as E, D, or O. Particularly, a similarity score between the command and the set of preloaded commands is calculated to decide on its category.

In TangleNet the states are defined based on the category of commands entered by the attacker in addition to information about the target server and the attacker's location. Particularly, a state S is defined as follows:

S:[Category of the command, environmental information]

Where the Category of command can be E, D, or O and the environmental information comprises:

- The current server (location of the attacker)
- The target server (target of the attacker)
- The number of vulnerabilities not yet exploited V*ulCounterX* in the target server *X*.

An example of State is presented below:

$S_i = (E, SQL\ Server, SSH\ Server, 4)$

Where:

- E: the attacker entered an exploitation command
- SQL Server: The current location of the attacker
- SSH server is the target server
- 4 is the number of vulnerabilities not yet exploited on the SSH Server.

4.2.2. RL agent

In TangleNet, the RL agent will respond realistically by following a clear decision-making process. It uses the Q-learning algorithm, which begins when the agent receives a state from the environment. The states in TangleNet represent the current configuration and status of the network as explained in the previous section. Once the RL agent observes a state S_i, it must decide on one of the three actions: *Allow, Block, or substitute*, which makes the system transit to a new State S_j (Touch & Colin, 2022). The actions of the RL agent are as follows:

- *Allow* the command to be executed to exploit the vulnerability in the target server. An example of a transition from state S_i (equation 1) to a potential state S_j upon the execution of an *Allow*:

$$S_i \xrightarrow{Allow} S_j\ (E,\ SQLServer,\ SSH\ Server,\ 3)$$

- *Block* the command to give the attacker the allusion that it is a real environment and avoid the detection of the decoy servers. An example of a transition from state S_i (equation 1) to a potential state S_k upon the execution of a *Block*:

$$S_i \xrightarrow{Block} S_k = (O,\ SQL\ Server,\ SSH\ Server,\ 4)$$

- *Substitute:* to make the attacker change its attack strategy by entering new commands, Consequently, the interaction period with the attacker will be extended and the RL agent will be able to collect more information about it. An example of a transition from state S_i (equation 1) to a potential state S_l upon the execution of a *Substitute*:

$$S_i \xrightarrow{substitute} S_l = (D,\ SQLServer,\ FTP\ Server,\ 4)$$

4.2.3. Learning algorithm

Q-learning is a Reinforcement Learning technique that enables the agent to learn the optimal action selection based on the current state (Chadi & Mousannif, 2023). In the implementation of TangleNet the Q-learning algorithm is used to implement the behavior of the RL agent. Initially, the Q-table is empty, as the learning process begins, the RL agent will store a pair of (state, action) associated with the maximum reward obtained over time.

The number of states depends on the type of the commands (E, D, or O), the current server, the target server, and the number of vulnerabilities in each server as explained above. The RL agent can take one of three actions {Allow, Block, Substitute}. During each learning cycle, the agent stores in the Q-Table a pair consisting of an action and a state, along with the accumulated reward obtained from performing that specific action in that state.

A description of the procedural form of the algorithm is provided in Algorithm 1.

Algorithm 1 Q-Learning: Learn function

```
Initialize the Q-table with random values
Set the learning rate (alpha), discount factor (gamma), and exploration rate
(epsilon)

For each episode:
    Initialize the current state

    While the episode is not complete:
        Select an action based on the current state and the Q-table using
epsilon-greedy exploration

        Execute the selected action in the network

        Observe the new state, vulnerabilities, and attack duration

        Calculate the reward based on the vulnerabilities and attack
duration:
        reward = calculate_reward(vulnerabilities, attack_duration)

        Update the Q-table using the Q-learning update rule:
        Q(state, action) = (1 - alpha) * Q(state, action) + alpha * (reward +
gamma * max(Q(new_state, :)))

        Update the current state to the new state

    Reduce the exploration rate (epsilon) over time to balance exploration
and exploitation

Repeat the above steps for a sufficient number of episodes until convergence
```

4.2.4. Reward function

The reward system encourages the agent to trick the attacker into staying in the network and prolonging the attack. The reward increases whenever the agent successfully deceives the attacker.

The model's reward function is based on the states and the actions the RL agent can take in the environment. The agent will receive a reward when it performs the correct action in a specific state.

Initially, the agent may not receive a high reward as it only starts the learning process. Over time, the agent will learn to take the correct action in each state and will get rewarded consistently.

It is worth recalling that the main objective of TangleNet is to prolong the existence of the attacker in the network and trick him into executing more commands on the target server. Consequently, TangleNet will be highly interactive and can learn more about the attack patterns executed to compromise the servers in the network. The reward function must consolidate TangleNet to achieve this goal. Hence the RL must select the adequate action each time the attacker executes a command on the target environment. The proper action must trick the attacker into frequently substituting the command mainly if it is in the exploitation and discovery phases (commands type E and D) before allowing or blocking it. Hence, the reward function R at a time t is given as follows:

$$Rt\left(s_t, a_t\right) = \begin{cases} 100 \;\; if \;\; s_t \in \{E, D\} \; and \; a_t = \{substitute\} \\ 10 \; if \;\; s_t \in \{E\} \; and \; a_t = \{Allow, Block\} \; or \; s_t \in \{D\} \; and \; a_t = \{Allow\} \\ 0 \; otherwise \end{cases} \quad (1)$$

5. Experimentation

5.1. Experimentation setup

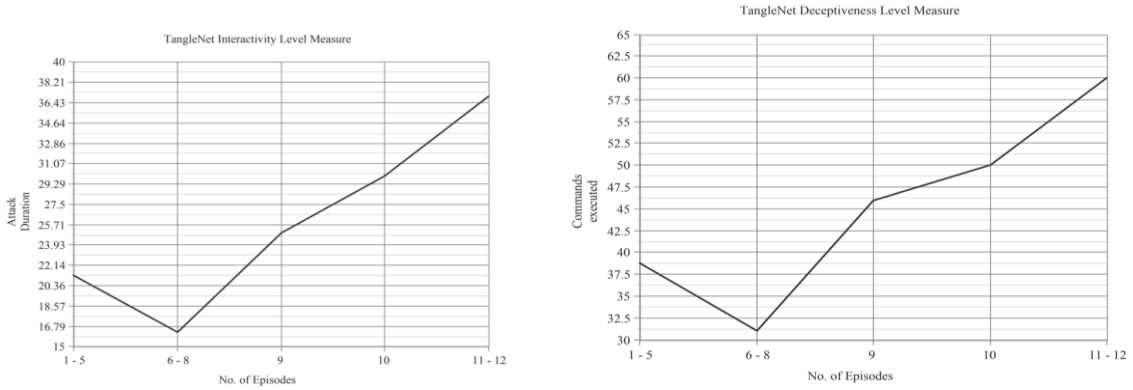
The network configuration consists of four servers: FTP, SSH, MySQL, and SMTP. They depict typical services that exist in the networks. A firewall is used to protect the network from unauthorized access as depicted in Figure 2. VLAN is enabled on the firewall with the VLAN mode specified as dot1q. In this topology, VLANs are configured on the firewall interfaces with VLAN tags of 10,20,30 & 40. Moreover, the IP forwarding is configured on the firewall to let forward packets to any of the interfaces. On each server, there is a setting of the default gateway, and traffic is directed through the VLAN interfaces of the firewall. Moreover, the system of NAT (Network Address Translation) rules is integrated into the firewall to perform the masquerading on the external interface. This makes it possible for the firewall to mask the internal IP addresses to be forwarded to the external network.

5.2. Results

TangleNet is designed in a way that would effectively extend attacker's experience in the network and capture crucial Tactics, Techniques, and Procedures (TTPs). The RL agent is designed to generate deceptive responses that mimic real servers' behavior, confounding attackers and prolonging their interaction. Two metrics are considered to assess the effectiveness of the RL agent: the deceptiveness level and the interaction time. The level of deceptiveness is the number of commands executed by the attacker. A higher number of executed commands indicates a higher level of deception in the environment. This metric serves as a quantifiable measure of the RL agent's ability to prolong the attack. The interaction time is the average duration of interaction between the attacker and the RL agent either by completing all the episodes or compromising the target server or the whole network. This metric reflects the agent's efficacy in making timely decisions and executing actions properly.

The number of commands entered by the attacker and the duration of the attack are illustrated as a function of the number of episodes in Figures 3a and 3b respectively. Figure 3 shows a positive correlation between the attack duration in minutes and the number of commands entered by the attacker. Figure 3a shows that the average duration required to attack a server is about 21 minutes between the first and the fifth episode, then decreases to around 16 minutes between episodes 6-8 before increasing exponentially starting from episode 9. Similarly, Figure 3b shows that the initial number of commands is around 39 between the first and the fifth episodes,

then decreases to around 31 commands between episodes 6-8 before increasing exponentially starting from episode 9. The decrease in the duration and the number of commands between episodes 6 and 8 can be explained by the fact that the RL is just getting started with the learning process, the Q-table is initially empty so it needs a certain time to initialize it with adequate pair of (state, action). Overall, the attack duration and the number of commands reflect the high deceptiveness level of TangleNet: the higher the duration of the attack, the higher the number of commands is, consequently the deceptiveness.



(a) The duration of the attack for each trial

(b) The number of commands entered for each episode

Figure 3. Correlation between the attack duration and the number of executed commands

Further experiments are conducted to investigate the performance of TangleNet. Particularly, the deceptiveness of TagleNet is measured using the average amount of time needed for the attacker to compromise a server or the entire network.

Table 2 presents the performance of the TangleNet on each server separately. The average duration of the attack and the average number of commands are calculated for each server.

Table 2. TangleNet attack trial statistics per server

Server	**Average Attack Duration**	**Average # of Executed Commands**
SSH	28min	43
FTP	19min	36
SMTP	35min	26
MySQL	28min	51

As shown in Table 2, the attacker was able to compromise the SSH server after an average of 28 minutes by executing 43 commands. For the FTP server, the attack duration is 19 minutes and the attacker executed 36 commands. Table 2 points out the RL agent succeeds to extend the attack duration and this is due to the substitute action performed as reaction to the commands executed by the attacker. The extended attack duration allows the attacker to explore more attack patterns and execute more commands.

Table 3 provides the results of these experiments used to deeply evaluate the deceptiveness of the TangleNet agent as a function of the initial location of the attacker. It shows the following information:

- The initial location of the attacker before pivoting in the network;
- Average Attack Duration: This column shows the average attack duration in minutes for each trial;
- Number of Commands: it represents the number of commands executed by the attacker;
- Compromise the whole network: it indicates whether the attacker succeeds in compromising the entire network.

Table 3. TangleNet attack trial statistics for the whole network

Attacker Initial Location	Attacker Trial (# of attempts)	Average Attack Duration	#of Command Executed	Compromise the whole Network
SSH Server	1-5	25min	45	x
	6-8	18min	32	x
	9	**30min**	**50**	✓
FTP Server	1-5	22min	38	x
	6-8	15 min	28	x
	9	**20min**	**42**	**x**
SMTP Server	1-5	10 min	20	x
	6-8	12 min	24	x
	9	**15min**	**32**	**x**
MySQL Server	1-5	28min	52	x
	6-8	20min	40	x
	9	**35min**	**60**	✓

As shown in Table 3, the attacker was able to compromise the network from the SSH server and the MySQL server as initial locations before pivoting to other servers. It needed an average of 30 minutes to execute 50 commands which was the required number of commands needed to compromise the network. Similarly, it needed around 60 commands to compromise the whole network if it started from MySQL server.

On the other hand, Table 3 shows that the attacker failed to compromise the network when it started from server FTP and SMTP. The number of trial 9 was not sufficient for the RL agent to extend the attack until the whole network was compromised, the duration of 20 minutes was insufficient to compromise the whole network by referring to Table 2, the duration of 19 minutes was required to compromise only the FTP server. More time would be given to the RL agent to learn how to prolong the attack duration.

6. Conclusion and future work

TangleNet offers an innovative solution by integrating deception techniques and Reinforcement Learning to effectively extend the attacker's engagement, capture crucial TTPs, and enable proactive defense strategies. TangleNet reduces human interference, automates threat intelligence gathering, and improves overall security posture. It generates deceptive responses that mimic genuine server behavior, confounding attackers and prolonging their interaction. The experimentation results show a positive correlation between the level of deception and the number of commands executed by the attacker which effectively increases the confusion. The TangleNet interactiveness is measured by the average time it takes an attacker to compromise a server or an entire network. Longer session time indicates a higher level of interaction, resulting in longer attack time. This extended attack window allows for more comprehensive data collection of changing attack patterns.

Although TangleNet improves the current application of RL in cyber deception, it does not consider the previous or following situations, because a single command may not lead to a malicious situation, but a series of commands may do. Further research efforts should explore potential approaches to achieve state correlation in the decision-making process.

As future work, it is expected to use RL to implement the attacker's behavior. Furthermore, more experiments would be conducted to compare TangleNet to other cyberdeception tools.

Acknowledgment

This research was supported by the University of Jeddah, Saudi Arabia, grant number UJ-23-DR-21.

REFERENCES

Biplob, M. B., Marma, S. M. & Akther, M. (2024) Securing Tomorrow's Digital World: Key Trends in Cyber security for 2024. *Preprints*. https://doi.org/10.20944/preprints202409.0576.v1.

Brasoveanu, A., Moodie, M. & Agrawal, R. (2018) Science of Cyber Deception: Experimental Design and Implementation. CEUR Workshop Proceedings, https://www.osti.gov/biblio/1648653.

Cabral, W., Valli, C., Sikos, L. & Wakeling S. (2019) Review and Analysis of Cowrie Artefacts and Their Potential to be Used Deceptively. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2019*. pp. 166-171. doi: 10.1109/CSCI49370.2019.00035.

Chadi, M. A. & Mousannif, H. (2023) Understanding Reinforcement Learning Algorithms: The Progress from Basic Q-Learning to Proximal Policy Optimization. To be published in *Machine Learning.* [Preprint] https://doi.org/10.48550/arXiv.2304.00026 [Accessed 31st March 2023].

Dowling, S., Schukat, M. & Barrett, E. (2019) Using reinforcement learning to conceal honeypot functionality. In: Brefeld, U. et al., (eds.) Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2018). *Lecture Notes in Computer Science* (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 11053, 341–355. https://doi.org/10.1007/978-3-030-10997-4_21.

Ghourabi, A., Abbes, T. & Bouhoula, A. (2011) Design and implementation of Web service honeypot. *SoftCOM 2011. 19th International Conference on Software, Telecommunications and Computer Networks, Split, Croatia, 2011.* pp. 1–5.

Javadpour, A., Ja'fari, F., Taleb, T., Shojafar, M. & Benzaïd, C. (2024) A comprehensive survey on cyber deception techniques to improve honeypot performance. *Computers & Security*. 140(4), 103792. doi: 10.1016/j.cose.2024.103792.

Kadam, S. J. & Ahmed, H. M. (2024) Ransomware Detection and Prevention Using Machine Learning and Honeypots: A Short Review. *Iraqi Journal of Computers, Communications, Control and Systems Engineering*. 24(2), 29-40. doi:10.33103/uot.ijccce.24.2.3.

Kunz, T., Fisher, C., Novara-Gsell, J., Nguyen, C. & Li, L. (2023) A Multiagent Cyberbattlesim for RL Cyber Operation Agents. To be published in *Cryptography and Security.* [Preprint] https://arxiv.org/ftp/arxiv/papers/2304/2304.11052.pdf [Accessed 3th April 2023].

Maddireddy Bharat Reddy & Maddireddy, Bhargava Reddy (2024) The Role of Reinforcement Learning in Dynamic Cyber Defense Strategies. *International Journal of Advanced Engineering Technologies and Innovations*. 1(2), 267-292. https://ijaeti.com/index.php/Journal/article/view/306.

Mininet/minet 2.3.0 (2021). Emulator for rapid prototyping of software defined networks. GitHub. https://github.com/mininet/mininet [Accessed 11th June 2024].

Navarro, F. O. (2021) *Analysis of reinforcement learning techniques applied to honeypot systems*. Master's Thesis. Universitat Oberta de Catalunya. http://hdl.handle.net/10609/126948.

Pauna, A. & Patriciu, V.V. (2014) Self-Adaptive SSH Honeypot Model Capable of Reasoning. *Academia.* https://www.academia.edu/5538993/ Self_adaptive_SSH_Honeypot_Model_ Capable_of_Reasoning [Accessed 09th July 2023].

Priambodo, D. F., Pramadi, Y. R., Briliyant, O. C., Hasbi, M. & Yahya, M. A. (2022) Observe-Orient-Decide-Act (OODA) for Cyber Security Education. *International Journal of Advanced Computer Science and Applications*. 13(12), 246-255. doi:10.14569/IJACSA.2022.0131031.

Reynolds, C. & Green, A. (2023) Analysis of Honeypots in detecting Tactics, Techniques, and Procedure (TTP) changes in Threat Actors based on Source IP Address. *The 27th Annual Symposium of Student Scholars - 2023 DigitalCommons@Kennesaw State University*.

Soule, N., Pal, P., Clark, S., Krisler, B. & Macera, A. (2016) Enabling defensive deception in distributed system environments. In *Proceedings of the 2016 Resilience Week (RWS), Chicago, IL, USA, 2016*. pp. 73-76. doi:10.1109/RWEEK.2016.7573310.

Torino, P. D. I. (2021) Cannypot: A reinforcement learning-based adaptive SSH honeypot. Politecnico di Torino, Corso di laurea magistrale. *Ingegneria Informatica (Computer Engineering)*, 2021.

Touch, S. & Colin, J. (2021) Asguard: Adaptive Self-guarded Honeypot. In *Proceedings of the 17th International Conference on Web Information Systems and Technologies (WEBIST 2021).* Volume 1: DMMLACS, pp. 565-574. doi:10.5220/0010719100003058.

Touch, S. & Colin, J. N. (2022) A Comparison of an Adaptive Self-Guarded Honeypot with Conventional Honeypots. *Applied Sciences.* 12(10), 5224. doi:10.3390/app12105224.

Veluchamy, S. & Kathavarayan, R. S. (2021) Deep reinforcement learning for building honeypots against runtime DoS attack. *International Journal of Intelligent Systems*. 37(7), 3981-4007. doi:10.1002/int.22708.

Walter, E., Ferguson-Walter, K. J. & Ridley, A. (2021) Incorporating Deception into CyberBattleSim for Autonomous Defense. To be published in *Cryptography and Security*. [Preprint] https://doi.org/10.48550/arXiv.2108.13980. [Accessed 31st August 2021].

Wang, C. & Lu, Z. (2018) Cyber Deception: Overview and the Road Ahead. IEEE Security & Privacy, https://doi.org/10.1109/MSP.2018.1870866.

* * *

Tahani GAZDAR received the Engineering, M.Sc., and Ph.D. degrees in computer science from the National School of Computer Sciences, University of Manouba, Tunisia, in 2009, 2010, and 2015, respectively. She is an Associate Professor at the College of Computer Science and Engineering, University of Jeddah, Saudi Arabia. Her research interests include Applying Machine Learning in Cybersecurity, Intrusion detection systems, Reinforcement Learning, and Penetration Testing.

* * *

Lara MURAD is a graduate student at the College of Computer Science and Engineering, Cybersecurity Department, University of Jeddah, Jeddah, Saudi Arabia.

* * *

Safana AL-JAHDALI is a graduate student at the College of Computer Science and Engineering, Cybersecurity Department, University of Jeddah, Jeddah, Saudi Arabia.

* * *

Fai ZWAWI is a graduate student at the College of Computer Science and Engineering, Cybersecurity Department, University of Jeddah, Jeddah, Saudi Arabia.

* * *

Reman MANDILI is a graduate student at the College of Computer Science and Engineering, Cybersecurity Department, University of Jeddah, Jeddah, Saudi Arabia.