A New Method of Cyber Deception Defense

Chaochao Liu[1,2], Degang Sun[1,3], Zhixin Shi[1,2], Ning Zhang[1,2]
[1] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[2] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[3] Computer Network Information Center, Chinese Academy of Sciences, Beijing, China
{liuchaochao, shizhixin, zhangning}@iie.ac.cn, dgsun@cnic.cn

2024 IEEE Symposium on Computers and Communications (ISCC) | 979-8-3503-5423-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISCC61673.2024.10733695

***Abstract*—New network attacks such as Advanced Persistent Threats (APTs) have created an asymmetric dynamic between attackers and defenders. Traditional defense mechanisms typically focus on perimeter security, rendering them ineffective once attackers infiltrate the network. Cyber deception defense, on the other hand, proactively establishes a deceptive environment to manipulate attackers' perceptions and decisions, thereby delaying, detecting, and potentially thwarting attacks. This paper introduces a novel approach rooted in cyber deception defense, utilizing FPGA technology. By harnessing network packet rewriting techniques on FPGA, this method rapidly generates numerous disguised hosts and ports. The implementation of this approach on an FPGA hardware platform allows for the evaluation of its effectiveness. In simulated environments, the method demonstrates remarkable efficiency in constructing deceptive environments, with a policy query time of approximately 50ns. Transitioning to real-world network scenarios, the prototype system extends the time required for attackers to identify genuine hosts by a factor of 4.5. Moreover, the average delay time of the prototype system in processing 64-byte data packets is approximately 5.68us, showcasing consistent performance across various deception strategies. Crucially, the system's overhead remains minimal, effectively confounding and deterring attackers while increasing the complexity and cost associated with mounting successful attacks.**

***Index Terms*—cyber deception defense, deception environment construction, FPGA, performance evaluation**

I. Introduction

The rapid proliferation of software and hardware vulnerabilities, coupled with the surge in malicious programs, underscores the critical state of cybersecurity today [1]. The emergence of APTs has further exacerbated the situation, as APT organizations exploit zero-day vulnerabilities and automated attack frameworks to persistently infiltrate and target specific entities until their objectives are achieved [2]. Conventional network defense strategies, including firewalls, intrusion detection systems, data encryption, malware detection [3], and situational awareness [4], rely on known parameters, rendering their protective measures static, passive, and reactive, thereby struggling to effectively counter sophisticated attacks. The deterministic, static, and uniform nature of existing systems has created an imbalance favoring attackers in the cyberspace domain, granting them advantages in time, space, and resources over defenders.

In response to this pressing challenge, both academia and industry have increasingly turned their attention to the development of cyber deception defense technologies. Since 2015, Cyber Deception Defense has been recognized by Gartner as a leading innovation in cybersecurity defense. Publications such as "Cyber Deception" [5] and "Autonomous Cyber Deception" [6] have delved into the realm of cyber deception, elucidating its principles and applications. Cyber deception defense involves the strategic deployment of deceptive tactics within a network environment to disrupt attackers' perceptions, leading them to cease harmful actions or inadvertently aid defenders, thereby identifying, delaying or blocking attacks, and increasing system security.

Presently, cyber defense mechanisms like honeypots predominantly rely on software-based implementations, deploying virtual machine services to lure and observe malicious activities. However, challenges persist in terms of deployment complexity, policy management, and inherent security vulnerabilities. Field Programmable Gate Array (FPGA) technology, known for its parallel processing capabilities and task concurrency, offers a unique opportunity to enhance cyber deception defense strategies. This paper introduces a novel approach to cyber deception defense leveraging FPGA technology, serving as a valuable complement to host-based deception tactics like honeypots. Key advantages of this method include:

Efficient Deception Environment: By manipulating data packets at the network layer, the deception environment is established with greater efficiency and reduced system resource overhead compared to traditional honeypot approaches involving virtual machines and services.

Enhanced Processing Speed: Leveraging FPGA's processing power enables rapid and efficient handling of network data packets, significantly outperforming software-based solutions in terms of speed and responsiveness.

Improved Security: Modifying data packets at the network layer to create deceptive traffic and logging activities through the FPGA chip ensures a higher level of security. Even if attackers compromise the host, the recorded logs remain inaccessible. In contrast, software-based solutions like honeypots rely on host-side agents for traffic monitoring and logging, making them vulnerable to data breaches when the host is compromised.

II. Related Works

In recent years, new network attacks such as APT have continued to emerge, and traditional defense methods can't effectively respond. Researchers have begun to explore solutions in the field of deception defense. With the support of new technologies such as game theory, artificial intelligence, SDN, and virtualization, deception defense has gone far beyond honeypot defense. It can through various deception methods such as device layer deception, network layer deception, application layer deception, and data layer deception, build a high-interaction deception environment, and implement active deception and traceability countermeasures.

Moving Target Defense (MTD) [7] aims to create a dynamic, randomized defense strategy that confuses attackers, shifting the advantage away from them. Gao Chungang et al. [8] developed an MTD-enhanced network defense system using SDN, changing IP addresses randomly to confuse attackers and boost defense efficiency. While multiple parameter changes in MTD can enhance defense, they may also interfere with each other, affecting system stability. Balancing defense effectiveness with system operation is crucial in implementing MTD successfully.

Network Address Translation (NAT) alters the network topology by modifying the mapping between network addresses and hardware devices to conceal the actual device. It offers user transparency, easy deployment, and protection against unidentified attacks. However, NAT introduces system overhead that could impact service performance. Wang Kai et al. [9] introduced a spoofing defense approach utilizing dynamic domain names and address transformations to heighten attack complexity. By generating continuously changing domain names and network addresses, this method raises the difficulty level for potential attackers.

Terminal Information Jumping involves both parties in network communication synchronously altering their communication network parameters according to a specific rule, disrupting the attack process and enabling active protection. Network parameters like IP addresses, ports, and protocols are typically modified. Tang Xiucun et al. [10] introduced a port address hopping method based on random time intervals, facilitating node synchronization through an SDN controller.

Wu Jiangxing et al. [11] introduced Cyber Mimic Defense (CMD), a concept leveraging heterogeneity, diversity, and dynamics to alter the similarity, unity, and certainty of the target system. CMD employs a heterogeneous redundant multi-mode ruling mechanism that involves multiple diverse entities processing external requests collectively to detect and thwart attacks. However, challenges such as cost and deployment complexities arise due to the redundancy associated with this approach.

The Modern Honey Network (MHN) project [12] focuses on developing a structured deception defense system utilizing honeypots. MHN utilizes numerous honeypot terminals as sensors to gather attack information, streamlines honeypot deployment, and accommodates various honeypot software as data collectors. This setup enables effective defense against unfamiliar network attacks.

Huang Chen et al. [13] have introduced a method for constructing deception services in the realm of intelligent cyber deception defense. This approach aims to enhance the automation level of building a deceptive environment. It involves utilizing machine learning techniques to adjust parameters and create models based on the feature data of the target service. Ultimately, this process results in the creation of a deceptive service that closely resembles the target service.

Pilla et al. [14] have provided a summary of current artificial intelligence-driven deception defense methods. Their focus lies on the utilization of machine learning algorithms like Naive Bayesian, Decision Tree, Random Forest, Support Vector Machine, Genetic Algorithm, as well as deep learning algorithms such as Artificial Neural Network, Recurrent Neural Network, Deep Neural Networks, and Reinforcement Learning to implement defensive deception strategies. They also highlight the constraints of existing research and outline potential future development directions in this field.

III. Method Design

The Cyber Deception Defense based on FPGA (CDD-FPGA) leverages hardware concurrency and pipeline processing to effectively manage network data packets, swiftly create highly authentic disguised hosts and ports, and replicate genuine host fingerprint information. This approach aims to blur the lines between real and fake elements. When an attacker launches a scan, they will interact with the disguised node, triggering an alert. These methods can significantly enhance intranet attack detection capabilities while maintaining lower system overhead and heightened security levels.

A. System Structure

The system architecture, as depicted in Fig. 1, showcases the deception module primarily comprising the control and management CPU alongside the FPGA. The main function of the CPU is to configure the FPGA and manage deception resources. The FPGA part mainly completes the processing of network data packets and the generation of alarm events. Analysts configure policies into the CPU through the deception policy interface, which in turn controls the FPGA to produce convincingly realistic disguised hosts. The responding host is tasked with handling data packets directed at the masquerading hosts. As attackers navigate laterally within the intranet, upon interaction with the generated disguised host and port, the attack traffic is directed to the application emulation module for a response. Simultaneously, the attack analysis module records logs and network packets discreetly, without alerting the attacker.

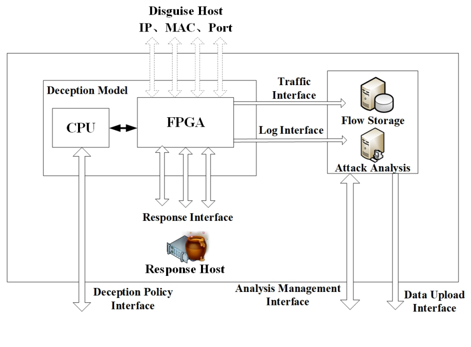

Fig. 1. The diagram of system architecture

The processing flow inside the FPGA is shown in Fig. 2. First, data packet is cached in the packet receiving module, which serves as the basis for subsequent data packet modifications; Then, the message header fields are extracted in packet analysis module. In policy query module, match the deception policy table to clarify how to modify the attack packet; in session restoration module, match the deception session table to clarify how to restore the response packet; finally complete the modification of data packet based on the cached message in packet modification and sending module.

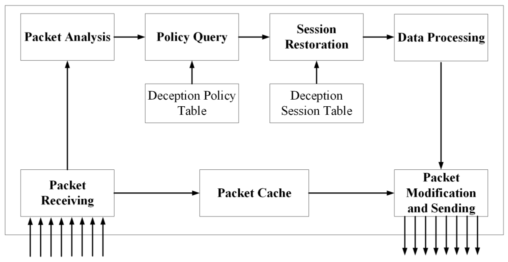

Fig. 2. The processing flow chart in FPGA

The communication protocol of this method is shown in Fig. 3, including attack hosts *(AMAC, AIP, APORT)*, disguise hosts *(VMAC, VIP, VPORT)* and response hosts *(RMAC, RIP, RPORT)*. The specific communication steps are as follows:

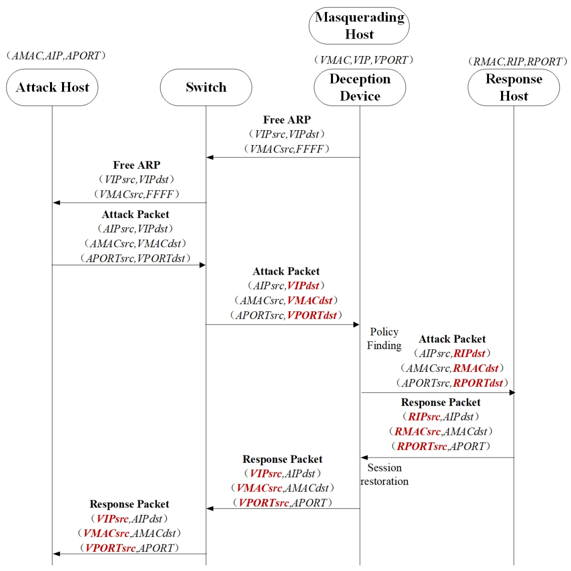

Fig. 3. Communication protocol chart

- Deception device connected to the network, it will send ARP packet according to deception policy, data packet format is *(VIPsrc, VMACsrc→VIPdst, FFFF)*.
- After receiving APR packet, the switch updates MAC address table to match the masquerade host to the port.
- Attacker obtains address of masquerade host, launches attack packets *(AIPsrc, AMACsrc, APORTsrc→VIPdst, VMACdst, VPORTdst)*.
- The switch updates MAC address table to match the attack host to the port.
- Deception device matches deception policy, and modifies data packet according to the strategy. Modified data packet is *(AIPsrc, AMACsrc, APORTsrc→RIPdst, RMACdst, RPORTdst)*, so that packets sent to masquerade host are forwarded to the response host.
- The packets returned by response host *(RIPsrc, RMACsrc, RPORTsrc→AIPdst, AMACdst, APORTdst)* are changed into *(VIPsrc, VMACsrc, VPORTsrc→AIPdst, AMACdst, APORTdst)* and sent to attacker after session restoration.

B. Deception Policy Query

Content Addressable Memory (CAM) is a special kind of memory in FPGA. The deception policy query technology based on CAM can realize rapid retrieval of deception policies, including two parts: deception policy write and match.

a) Deception Policy Write:: Deception policy write is mainly to complete the bitwise storage of policy in the CAM to prepare for fast matching. The algorithm principle is as follows: First, extract the masquerade IP, Port, and protocol from the deception policy as the KEY value, divide the 60-bit KEY value into 6 pieces, and use it as the address bus of SRAM, and combine the 6 SRAM into 1 CAM. For the ith deception policy, write 1 at the ith column data bus corresponding to 6 SRAM address buses respectively, and write the strategy CONTENT into the off-chip SRAM at the same time. This method can save the FPGA chip space as much as possible on the basis of ensuring the high-speed query of the strategy.

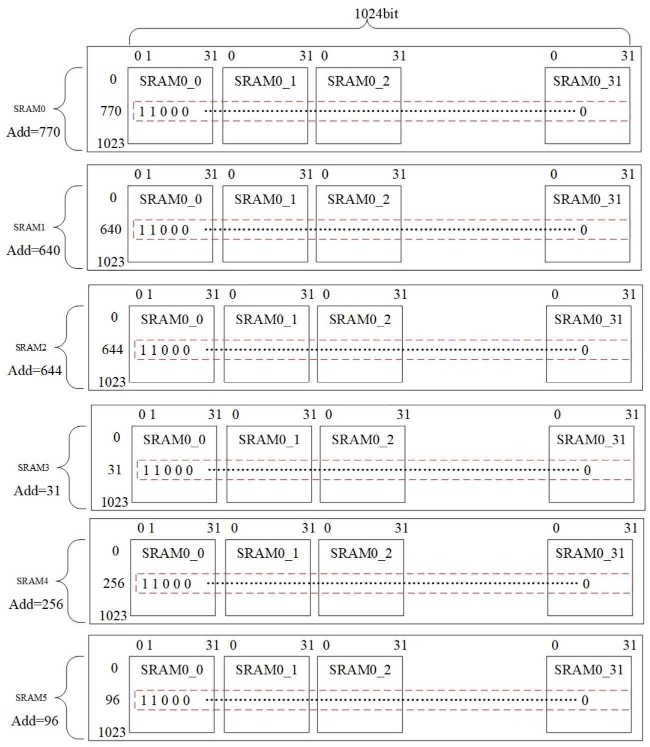

Fig. 4. Communication protocol chart

For ease understanding, the entire writing process is described below by taking two deception strategies as examples. The KEY value of the first policy is *(192.168.10.16,8000,6)*, and 6 SRAM addresses are 770, 640, 644, 31, 256, and 96. The KEY value of the second policy is *(192.168.10.16,8001,6)*, and 6 SRAM addresses obtained are 770, 640, 644, 31, 260, and 96. Only the 5th SRAM address bus value is different. The deception policy query principle is shown in Fig. 4. After obtaining the addresses of 6 SRAMs, enter the CAM writing process. For the first strategy, the data in the first column of the 6 SRAM addresses are assigned 1, and all others are assigned 0. The only difference between the second strategy is 5th address changed from 256 to 260, so the second column of address 260 is assigned 1, and the other address are the same. The cheating policy writing algorithm is as follows.

Algorithm 1 Deception policy write algorithm

```
Input: Deception policy (VIP, VPORT, Protocol; RIP, VMAC, MAC, VPORT)
Output: deception policy KEY is written to CAM, and CONTENT is written to off-chip RAM
 1: CAM consists of 6*1024*1024
 2: Configure M deception policy (VIP, VPORT, Protocal, RIP, VMAC, RMAC, RPORT)
 3: for all each $S_i, i = 1, ..., M$ do
 4:     Extract the 32bit VIP in policy
 5:     Extract the 16bit VPort in policy
 6:     Extract the 8bit Protocal in policy
 7:     key = VIP, VPORT, Protocal
 8:     KEY = key, 0, 0, 0, 0
 9:     KEY divided into 6 segments by bit as address of CAM
10:     if $row == key \& col == I$ then
11:         Assign 1
12:     else
13:         Assign 0
14:         Policy CONTENT is written to off-chip RAM
15:     end if
16: end for
```

Algorithm 2 Deception policy match algorithm

```
Input: Packet (DIP, DPORT, Protocol; SIP, SPORT)
Output: deception policy (KEY, CONTENT)
 1: CAM consists of 6*1024*1024
 2: for all each $P_j, j = 1, ..., N$ do
 3:     Extract the 32bit DIP in policy
 4:     Extract the 16bit DPort in policy
 5:     Extract the 8bit Protocal in policy
 6:     key = DIP, DPORT, Protocal
 7:     KEY = key, 0, 0, 0, 0
 8:     KEY divided into 6 segments by bit as address of CAM
 9:     Sum= sum (CAM1 & ... & CAM6)
10:     if $Sum > 0$ then
11:         policy match succeed
12:         take out CAM colomn i where the value is not 0
13:     else
14:         policy match fail
15:     end if
16: end for
17: take out policy content and modify the packet
```

b) Deception Policy Match:: The main purpose of deception policy matching is to quickly query the policy stored in the CAM after receiving the attack packet, and prepare for modifying data packet. The main principles of the algorithm are as follows: First, extract the destination IP, destination port and protocol type from the data packet to form the KEY value. Then divide the 60-bit KEY value into 6 blocks, as SRAM address, query the CAM to obtain the values corresponding to the 6 addresses, and perform a bitwise AND summation operation. If the result is greater than 0, it indicates that policy matching is successful, and take out corresponding number i where the value that is not 0 as off-chip RAM index, and the complete cyber deception strategy can be obtained through the index. Finally, according to deception strategy, the modification of the data packet is completed, and the data packet originally sent to the masquerade host is forwarded to the response host. The matching algorithm is as follows.

C. Deception session restoration

Because the number of masquerade hosts are large and the number of response hosts are small, the relationship between masquerade hosts and response hosts are often many-to-one. How to accurately restore the data packets returned by the responding host to the fake host before modification is a difficult problem that must be solved. Aiming at this problem, this paper proposes a deceptive session restoration technology based on Hash algorithm, including two parts: deception session write and restoration.

a) Deception session Write:: Deception session write is mainly to complete the storage of deception sessions and prepare for subsequent restoration of deception sessions. The main principles of the algorithm are as follows: First, both masquerade host and response host information are written into the deception session table. Then, extract source IP, source port, response host IP, response host port, protocol type as KEY value. Finally, the KEY is used as the RAM address after Hash calculation. If it is empty, session will be written. If it is not empty, it means that a Hash conflict has occurred, and the session will be discarded. Note that the KEY value here uses the IP and port of the responding host, otherwise the restoration deception session cannot be completed, because there is no information about the masquerade host in the data packet returned by the response host.

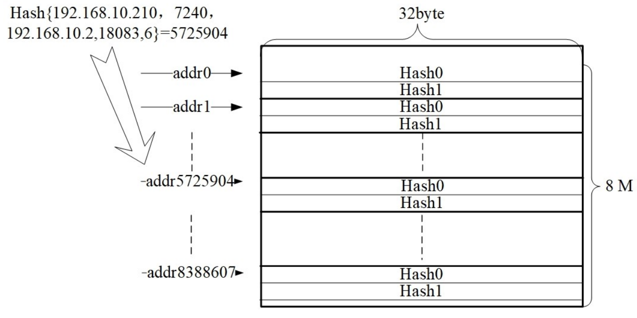

Fig. 5. Schematic diagram of deception session restoration

The principle of deception session write is shown in Fig. 5. Taking a set of data as an example to illustrate the process of writing session table. the KEY value of the session table *(SIP:192.168.10.210, Sport:724, DIP:192.168.10.2, Dport:18083, Protocol:6)*. Firstly, the Hash operation is performed on the KEY value, and the output address value is 5725904. The process of Hash calculation will inevitably produce certain conflicts. The principle of conflict handling is that if the two Hash buckets corresponding to the address are free, the fraudulent session content will be written, otherwise it will be discarded automatically. At this time, the address of 5725904 is empty and will be written automatically. Deception session write algorithm is as follows.

Algorithm 3 Deception session write algorithm

```
Input: Session information (AIP, APORT, RIP, RPORT, Protocol; VIP,VPORT,VMAC,RMAC, Time)
Output: deception session table
 1: Session storage is 8byte*32byte
 2: for all each $P_i, i = 1, ..., M$ do
 3:     if packet match strategy success then
 4:         Extract AIP, APORT, RIP, RPORT, Protocal
 5:         key= AIP, APORT, RIP, RPORT, Protocal
 6:         KEY=Hash(key) as RAM address
 7:     end if
 8:     Query whether the RAM address is empty
 9:     if Hash[0] is empty then
10:         Write session information in Hash[0]
11:     else
12:         if Hash[1] is empty then
13:             Write session information in Hash[1]
14:         else
15:             Hash conflict, session discarded
16:         end if
17:     end if
18: end for
19: Deception session information has been written
```

Algorithm 4 Deception session restoration algorithm

```
Input: Response packet (RIP, RPORT, AIP, APORT, Protocol, Time)
Output: deception session restoration is over
 1: Session storage is 8byte*32byte
 2: for all each $P_j, j = 1, ..., N$ do
 3:     Extract RIP, RPORT, AIP, APORT, Protocal
 4:     key= AIP, APORT, RIP, RPORT, Protocal
 5:     key= RIP, RPORT, AIP, APORT, Protocal
 6:     key=Reverse(key)
 7:     KEY=Hash(key) as RAM address
 8:     Query whether the RAM address is empty
 9:     if Hash[0] is not empty then
10:         Extract Hash[0]
11:     else
12:         if Hash[1] is not empty then
13:             Extract Hash[1]
14:         else
15:             Finding failed, session discarded
16:         end if
17:     end if
18:     Compare (KEY, Hash[0], Hash[0])
19:     Get correct deception session information
20: end for
21: Modify packet according to result, send it to attack
```

b) Deception session restoration:: Deception session write is mainly to complete the storage of deception sessions and prepare for subsequent restoration of deception sessions. The main principles of the algorithm are as follows: First, both masquerade host and response host information are written into the deception session table. Then, extract source IP, source port, response host IP, response host port, protocol type as KEY value. Finally, the KEY is used as the RAM address after Hash calculation. If it is empty, session will be written. If it is not empty, it means that a Hash conflict has occurred, and the session will be discarded. Note that the KEY value here uses the IP and port of the responding host, otherwise the restoration deception session cannot be completed, because there is no information about the masquerade host in the data packet returned by the response host.

Because deception session restoration must only be able to replace the response packet under the premise of querying the session information, otherwise it will be discarded directly. This mechanism determines that even if the attacker compromises the response host, it cannot use the response host as a springboard to initiate active detection. This is one of the key points for the cyber deception defense technology based on FPGA to ensure its own security. Deception session restoration algorithm is as follows.

IV. Experiment

n this section, we will evaluate the effectiveness and system overhead of the FPGA-based cyber deception defense method through simulation experiments, real network experiments, and performance testing experiment.

A. Simulation Experiments

We build VCS and Verdi simulation test environment on the CentOS7 operating system. In the simulation experiment, the deception strategy is to transform the masquerading port of the masquerading host to the response port of the responding host. It can be seen from the simulation results, the deception policy query of CDD-FPGA is completed within 5 clock cycles (100M clocks), and the time-consuming is about 50ns, achieving high-speed matching, which proves that the method has a high efficiency in constructing deception environment. In order to verify the deception environment construction ability of this method, we compared this method with Dynamic Virtual Network Topology (DVNT) methods in reference [8], CDD-FPGA can transform more network parameters as shown in Tab. I.

TABLE I
TRANSFORMABLE PARAMETER COMPARISON TABLE OF CDD-FPGA AND DVNT

Different Methods	**Transformable Network Parameters Name**					
	SIP	***DIP***	***SPORT***	***DPORT***	***SMAC***	***DMAC***
CDD-FPGA	√	√	√	√	√	√
DVNT	×	×	×	×	×	×

B. Validity Test Experiment

The experimental system is shown in Fig. 6. There are 10 real hosts in the network. We use the deception defense method to build 20 masquerade hosts. The masquerade hosts are evenly distributed in the network, and the responding hosts are connected to deception devices. In the experiment, we used NMAP to simulate the attacker, and recorded the time required for the attacker to obtain all 10 real host information in the two scenarios of No Cyber Deception Defense (NO CDD) and CDD-FPGA, to test the effectiveness of CDD-FPGA. To ensure the accuracy and reliability of the experimental results, 100 scanning and detection experiments were repeated in the two scenarios, and the shortest time, the longest time, and the average time of detecting the real host were recorded for comparative analysis.

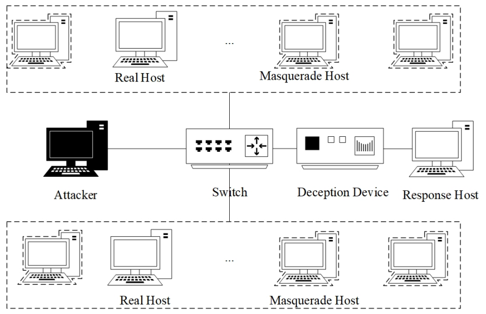

Fig. 6. Schematic diagram of the experimental system

Fig. 7 is a curve for the probability of attacker finding the real host under two scenarios NO CDD and CDD-FPGA. Analyzing the experimental results, in the NO CDD scenario, the probability of the attacker finding the real host within 400 seconds is 90%, while in the CDD-FPGA scenario, it takes 1800S, which is 4.5 times of NO CDD scenario. At the same time, in NO CDD scenario, it takes 500 seconds for the attacker to find the real host with 100% probability, while in the CDD-FPGA scenario, it takes 2200 seconds, which is also about 4.5 times of NO CDE scenario. Experimental results show that by deploying a network deception environment, the attack action can be significantly delayed, and the time for the attacker to obtain real host vulnerability information through scanning and detection is greatly extended.

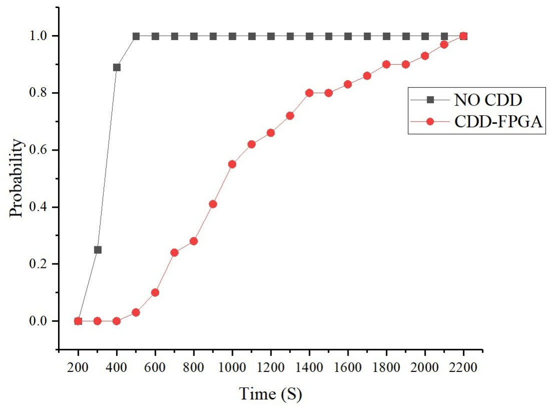

Fig. 7. Comparison of the probability of finding real host in two scenarios

C. Performance Testing

In order to further verify the performance of CDD-FPGA, we use the IXIA Perfect Storm One 10G flow tester to test the CDD-FPGA prototype system according to the RFC2544 method. The traffic tester sends data packets with lengths of 64, 128, 256, 512, 1024, 1280, and 1518 bytes. Packet rates are 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000Mbps. Packets of different lengths and rates flowed through the spoofed prototype system and back to the traffic tester. System delay time is tested in four scenarios: network cable direct connection, access prototype system without deception strategy, access prototype system with 1 deception strategy, and access prototype system with 100 deception strategies. The experimental results are shown in the Tab. II

The following conclusions can be drawn from Fig. 8. Firstly, the average delay time is about 1.888us in the scenario of direct network connection. Secondly, in the scenario where the network cable is directly connected, as the length of the data packet increases, the average delay time is almost unchanged; After connecting to the prototype system, because the device processes the header of the data packet, the delay will change as the length of the data packet changes. From the experimental results, it can be seen when processing small data packets of 64bytes, the average delay of the device is the smallest, about 5.68us, and when processing large data packets of 1518bytes, the average delay of the device is the largest, about 24.8us. Comparing the experimental results with DVNT methods in reference [8], CDD-FPGA has obvious advantages over DVNT as shown in Tab. III

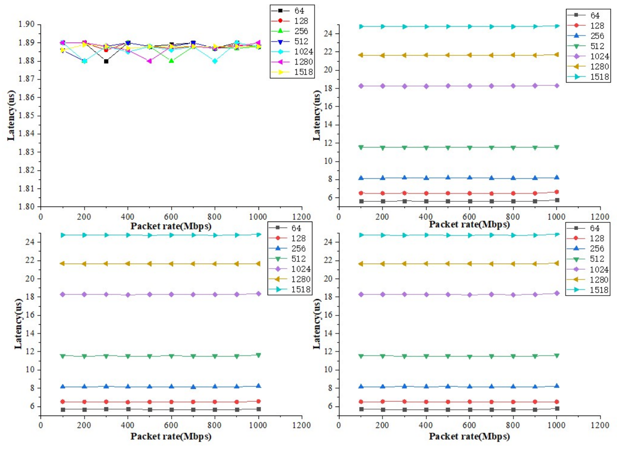

Fig. 8. Average latency with packet rate in different scenarios

Finally, the variation of prototype system average delay with the number of deception strategies is tested. In the three scenarios of no deception strategy, 1 deception strategy, and 100 deception strategies, the average delay of the system is almost the same. This is because CDD-FPGA uses a hardware pipeline processing method to process data, and the advantage of parallel processing determines that the system performance will not decrease with the increase in the number of configured deception strategies. However, in the software deception defense scheme, the system performance will decrease when the number of configured deception strategies continues to increase inevitably.

V. Conclusion and Future Work

Cyber deception defense, as a novel active defense mechanism, has garnered significant attention from both academia and industry since its inception. Research in this field has provided valuable insights across various domains. Addressing the challenge of effectively defending during the scanning detection stage, this paper introduces the CDD-FPGA method.

TABLE II
DELAY TIME IN DIFFERENT SCENARIOS

Scenes	Packet Length (bytes)	Packet Rate(Mbps)										
		100	*200*	*300*	*400*	*500*	600	700	800	900	1000	Average
Network cable direct connection	64	1.89	1.89	1.88	1.89	1.888	1.889	1.89	1.887	1.89	1.888	1.888
	128	1.89	1.89	1.886	1.89	1.888	1.887	1.888	1.887	1.889	1.888	1.888
	256	1.89	1.88	1.888	1.89	1.888	1.88	1.888	1.887	1.887	1.888	1.887
	512	1.886	1.88	1.888	1.89	1.888	1.888	1.89	1.887	1.888	1.888	1.887
	1024	1.89	1.88	1.888	1.885	1.888	1.886	1.888	1.88	1.89	1.888	1.886
	1280	1.89	1.89	1.888	1.886	1.88	1.888	1.888	1.887	1.888	1.89	1.888
	1518	1.886	1.889	1.888	1.887	1.888	1.888	1.888	1.888	1.888	1.888	1.888
Access prototype system without deception strategy	64	5.66	5.63	5.67	5.63	5.66	5.63	5.64	5.63	5.64	5.76	5.655
	128	6.52	6.49	6.52	6.49	6.5	6.49	6.46	6.49	6.49	6.65	6.51
	256	8.16	8.16	8.19	8.16	8.19	8.2	8.16	8.16	8.16	8.22	8.176
	512	11.56	11.52	11.55	11.52	11.55	11.52	11.55	11.52	11.55	11.58	11.542
	1024	18.27	18.27	18.24	18.24	18.27	18.27	18.27	18.27	18.3	18.3	18.27
	1280	21.66	21.6	21.63	21.66	21.64	21.66	21.63	21.66	21.63	21.7	21.647
	1518	24.8	24.8	24.76	24.77	24.8	24.8	24.77	24.77	24.8	24.86	24.793
Access prototype system with 1 strategy	64	5.69	5.69	5.7	5.7	5.66	5.66	5.66	5.63	5.66	5.73	5.678
	128	6.53	6.5	6.5	6.46	6.49	6.49	6.49	6.49	6.49	6.56	6.5
	256	8.16	8.16	8.19	8.16	8.16	8.16	8.12	8.16	8.16	8.22	8.165
	512	11.55	11.52	11.55	11.52	11.52	11.55	11.52	11.52	11.52	11.64	11.541
	1024	18.27	18.28	18.27	18.24	18.27	18.27	18.27	18.27	18.27	18.37	18.278
	1280	21.67	21.63	21.66	21.66	21.64	21.63	21.63	21.63	21.63	21.66	21.644
	1518	24.8	24.8	24.8	24.8	24.76	24.8	24.8	24.76	24.8	24.86	24.798
Access prototype system with 100 strategies	64	5.7	5.67	5.67	5.66	5.66	5.67	5.67	5.66	5.66	5.76	5.678
	128	6.52	6.53	6.53	6.49	6.49	6.52	6.49	6.49	6.52	6.5	6.508
	256	8.16	8.16	8.19	8.16	8.19	8.16	8.16	8.16	8.13	8.22	8.169
	512	11.55	11.55	11.52	11.52	11.52	11.49	11.52	11.52	11.52	11.61	11.532
	1024	18.28	18.27	18.3	18.27	18.27	18.24	18.27	18.24	18.27	18.4	18.281
	1280	21.63	21.63	21.66	21.66	21.66	21.66	21.63	21.63	21.63	21.7	21.649
	1518	24.8	24.77	24.76	24.8	24.76	24.76	24.8	24.76	24.77	24.89	24.787

TABLE III
LATENCY BETWEEN CDD-FPGA AND DVNT

Different Methods	Latency In Different Packet Sizes				
	64KB	***128KB***	***256KB***	***512KB***	***1024KB***
CDD-FPGA	5.7us	6.5us	8.2us	11.5us	18.3us
DVNT	80us	96us	118us	196us	368us

The effectiveness of this method is assessed through simulation experiments, real network trials, and performance testing. Results indicate that CDD-FPGA can establish a virtual network topology, obfuscate the attacker's detection efforts, notably prolong the time required for attackers to scan vulnerabilities in real hosts, and demonstrate high performance in constructing network deception environments with minimal delay.

Moving forward, the focus will shift towards enhancing the development of dynamic cyber deception environments. This will involve integrating cutting-edge technologies such as artificial intelligence and machine learning to elevate the intelligence quotient of cyber deception defense systems. Future research endeavors will concentrate on intelligent deception environment creation, traceability countermeasures for deception, assessment of deception defense efficacy, and the integration of chip-level cyber deception defense capabilities. These areas are poised to be the leading research directions in cyber deception defense, promising extensive growth and application prospects for the field.

Acknowledgment

The work was supported by the National Key Research and Development Program, Program Number: 2022YFC3320600.

References

[1] CNCERT, "2020 china internet network security report," https://www.cert.org.cn/publish/main/upload/File/2020%20Annual%20Report.pdf, 2021.

[2] J. Esteves, E. Ramalho, and G. De Haro, "To improve cybersecurity, think like a hacker," *MIT Sloan Management Review*, 2017.

[3] Y. Liu, B. Liu, Z. Zhao, C. Liu, X. Wang, and X. Wu, "Powershell malware detection method based on features combination," *Journal of Cyber Security*, vol. 6, no. 1, pp. 40–53, 2021.

[4] H. Yang, Z. Zhang, and L. Zhang, "Network security situation assessment based on deep weighted feature learning," *Journal of Cyber Security*, vol. 7, no. 4, pp. 32–43, 2022.

[5] S. Jajodia, V. Subrahmanian, V. Swarup, and C. Wang, "Cyber deception," *Springer*, vol. 1, pp. 2–1, 2016.

[6] E. Al-Shaer, J. Wei, W. Kevin, and C. Wang, "Autonomous cyber deception," *Springer*, 2019.

[7] G. Cai, B. Wang, T. Wang, Y. Luo, X. Wang, and X. Cui, "Research and development of moving target defense technology," *Journal of Computer Research and Development*, vol. 53, no. 5, pp. 968–987, 2016.

[8] C. Gao, Y. Wang, X. Xiong, and W. Zhao, "Mtdcd: an mtd enhanced cyber deception defense system," in *2021 IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, vol. 4. IEEE, 2021, pp. 1412–1417.

[9] K. Wang, X. Chen, and Y. Zhu, "Random domain name and address mutation (rdam) for thwarting reconnaissance attacks," *PloS one*, vol. 12, no. 5, p. e0177111, 2017.

[10] X. Tang, L. Zhang, Y. Kong, and H. Xu, "Transparent synchronization and random interval based sdn address and port hopping scheme," *Application Research of Computers*, vol. 33, no. 12, pp. 3774–3779, 2016.

[11] J. Wu, "Meaning and vision of mimic computing and mimic security defense," *Telecommunications Science*, vol. 30, no. 7, p. 2, 2014.

[12] M. Weijie, "Research and implementation of intranet threat capture system based on honeypot," Master's thesis, Jiangsu University of Science and Technology, 2021.

[13] C. Huang, J. Han, X. Zhang, and J. Liu, "Automatic identification of honeypot server using machine learning techniques," *Security and Communication Networks*, vol. 2019, pp. 1–8, 2019.

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