Enabling Defensive Deception by Leveraging Software Defined Networks

Ilias Belalis*, Georgios Kavallieratos†, Vasileios Gkioulos†, Georgios Spathoulas †
*Department of Computer Science and Biomedical Informatics
University of Thessaly
Lamia, Greece
ibelalis@uth.gr
†Department of Information Security and Communications Technology
Norwegian University of Science and Technology
Gjøvik, Norway
{georgios.kavallieratos, vasileios.gkioulos, georgios.spathoulas}@ntnu.no

***Abstract*—Computer networks are critical for modern society and protecting their security is of high importance. Due to their increasing size and complexity providing the required cyber security counter measures has become a very difficult task. One of the most recent approaches is to employ defensive deception techniques, in order to provide to the attacker a false perception about the protected network and thus increase the effort that is needed to carry on a successful intrusion. In this paper we present a comprehensive literature review and a comparison of existing SDN based defensive deception methods. Additionally, we propose a novel deception mechanism that combines moving target and honeypots approaches and carry out extensive tests of its functionality.**

***Index Terms*—network security, SDN, deception, defense, moving target, honeypots**

I. Introduction

Nowadays, the complexity of computer networks and our dependence on them are continuously increasing, also projected as increasing system interconnections and interdependencies [1]. Networking becomes imperative, even across previously isolated domains such as critical infrastructures, due to fundamental data flows that are essential for accomplishing novel functionalities and business models. Nevertheless, this practice enables additional attack vectors, not only from novel attacks but also from those that are considered common within ICT. A plethora of cyber-attacks, such as Distributed Denial of Service (DDoS) is reported daily. These attacks target computer networks to compromise devices and execute malicious actions [2]. Although various works in the literature aim to mitigate such attacks [3], [4] the complexity and heterogeneity of the networks impede the implementation of traditional techniques.

Software-Defined Networks (SDN) are a novel technology aiming to facilitate the management and the programming of large-scale networks. Notably, the control and data planes are grouped in traditional networks. In a more flexible approach, the SDN technology separates the control plane from the data plane and enables the programmability of the network. Thus, the routers and the switches can be programmed via the control plane and therefore, the network management and evolution are better facilitated. Furthermore, computer network challenges such as resilience, scalability, performance, security and dependability can also be addressed by SDN technology.

In this article, we argue that the combination of existing defensive deception techniques and the dynamicity provided by the SDN technology can be utilised for enhancing the security, the robustness and the scalability of contemporary computer networks. Furthermore, the dynamic reprogramming of the network and the monitoring of the data flows are impractical by leveraging existing defensive deception techniques such as honeypots and Moving Target Defense (MTD). On the other hand, SDN allows overcoming such limitations and enables the implementation of defensive deception techniques.

The contribution of this work is twofold:

- A comprehensive survey of existing defensive deception techniques on SDN technology is provided. The analysis is focused on the most prominent techniques considering five distinct properties.
- A Defensive Deception mechanism based on SDN technology is presented. This mechanism aims at misleading malicious activities in a computer network by presenting a virtual network topology and hiding the real network along with possible vulnerabilities that attackers can exploit. Furthermore, by using this Defensive Deception mechanism defenders are able to track malicious activities in time and respond before adversaries are able to succeed in their attacks.

The rest of the paper is structured as follows : Section II presents related work while Section III discusses SDN technology and defensive deception techniques in general. The two main sections of the paper are Section IV which thoroughly analyses and compares current state of the art methods for SDN based defensive deception and Section V which presents the proposed novel defensive deception mechanism. Finally, Section VI presents the results obtained through experiments and Section VII discusses our conclusions.

 ISBN: 978-1-61208-796-2

II. Related work

Various approaches exist in the literature analyzing the application of defensive deception techniques in contemporary systems and networks. Hoffman et al. in [5] proposed a framework to identify potential attacks and defence mechanisms in repudiation systems. Furthermore, Jajodia et al. in [6] analyzed the implementation of the moving target defence technique in order to protect modern computer networks. Nevertheless, the implementation of MTD on SDN by leveraging its capabilities is not considered. Furthermore, Lei et al. in [7] conducted a survey on different moving target defence techniques. Through their work, they analyzed the design principles and system architecture of MTD. Ward et al. in [8] provide an overview of different cyber moving target techniques, their threat models and their technical details. However, none of the previous works has focused on the MTD implementation on SDN. Additionally, the SDN implementations and capabilities have been studied in the literature. Specifically, Rowshanrad et al. in [9] analyzed the different SDN application and different southbound interfaces, also discussing potential SDN applications on Cloud, wireless, and mobile networks. In addition, a comprehensive survey for the SDN has been conducted by Kreutz et al. in [1]. Although various surveys examined the SDN technology, none of them has considered it in combination with the implementation of defensive deception techniques in their architectures and how such techniques affect the security in contemporary networks. Further, the application of defensive deception techniques has been studied extensively [10]–[21]. These are analyzed in detail in Section IV towards the identification of the most appropriate defensive deception mechanism explained and applied in Sections V and VI respectively.

III. Background

In this section, the basic notions on SDN technology and defensive deception techniques are presented.

A. SDN Technology

Traditional networks consist of a three-layer architecture; (i) Data plane, (ii) control plane and (iii) management plane where the last two layers are to some extent considered as one [1]. In particular, the data plane consists of the necessary network devices that are responsible for data forwarding. Further, the control plane contains network protocols that are used by network devices while the management plane consists of the software services which enable the remote monitoring and configuration of the network. Various management and configuration challenges arise from this architectural paradigm since deploying elaborate policies and adjusting the network in case of faults and changes can become cumbersome.

SDN proposes a different network architecture where different abstraction layers facilitate network management and configuration. The motivation behind SDN technology is to differentiate the control from the data plane. Particularly, SDN technology was proposed a couple of years ago, where both academia and industry were trying to build programmable networks. Two distinct approaches have been proposed in [22], [23] for active networks: (1) programmable switches and (2) capsules. Further, different approaches have arisen regarding programmable networks [24]–[27]. Recent initiatives such as ForCES [28], OpenFlow [29] and POF [30] have been described in [1]. These approaches propose the separation of the control plane from the data plane without significant adjustments to the network's infrastructure. Furthermore, one of the major approaches in the field of network virtualization was presented in [31] in the Tempest project. In particular, this project proposed the concept of switches in ATM networks in order to facilitate network management, allowing the ATM networks to communicate under the same physical resources. Further, different projects such as Planet lab [32], MBone [33], GENI [34], and VINI [35] proposed architectures for virtual network topologies. SDN technology arose by the OpenFlow work at Stanford University, CA, USA [36].

According to Kreutz et al. in [1], the forwarding state of the data plane is managed by a remotely controlled plane in SDN architecture. Further, eight different layers have been developed in the SDN architecture. These are listed below:

- Network Infrastructure: Different physical systems are installed which are responsible for forwarding packets.
- Southbound Interface: The connections between control and forwarding elements, which is one of the major SDN's advantages, are represented.
- Network Hypervisor: Using contemporary virtualization tools, SDN can virtualize machines and typologies to share the same hardware components.
- Network Operating Systems – NOS: Programmed APIs which provide services to the programmers in order to facilitate the network management. Through NOS, programmers are able to control APIs to generate the network configuration according to specific policies.
- Northbound Interface: A software system which promotes the application's portability and interoperability among the different control platforms.
- Language-based Virtualization: Different programming languages such as Pyretic and Splendid can be used in order to virtualize network topologies to enable the SDN's interoperability.
- Programming Languages: High-level programming languages are used by programmers in order to configure an SDN topology.
- Network Applications: The application in this layer implements the control logic for the SDN. Namely, such systems compile the commands which must be implemented in the data plane.

B. Defensive deception techniques

The development and application of the SDN has been studied extensively in the literature [37]–[39]. However, different security techniques have been developed that could be applied in such infrastructures in order to ensure their operations. Defensive deception techniques are able to increase security and dependability of Software Defined Networks.

In the following section, we will present and describe such techniques.

Fraunholz et al. in [40] conducted a comprehensive survey of deception technologies. Notably, different security mechanisms have been categorized considering the application layer of each technique; (i) Network, (ii) System, and (iii) Data layer. Furthermore, Han et al. in [41] examined deception techniques in computer security and categorized existing works according to the unit of deception, the layer where deception is applied, the goal of the deception solution, and the deployment mode.

This work focuses on network-based deception technologies in order to examine their application to the SDN. According to [41], network-based deception technologies aim to mitigate three threat categories: network fingerprinting, eavesdropping, and infiltration and attack propagation. To this end, ten different techniques have been identified. These are listed below:

- Network Tarpit: This technique focuses on sticky connections aiming to slow or stall automated network scanning in order to confuse attackers.
- Traffic forging: This technique increases the traffic flow in the network to slow down adversary's actions.
- Deceptive topology: This technique aims to distort network topology through traffic forging to slow the attacker.
- OS obfuscation: Through this technique, a mimic of the network behaviour of fake operating systems is created in order to deceive potential attackers.
- Honeytokens: Honeytokens consist of honey passwords, honey URL parameters, database honeytokens and honey permissions.
- Deceptive attack graphs: This technique uses attack graph representations to lead adversaries to follow a rouge attack path in order to distract them from their real targets.
- Deceptive simulation: The simulation enables the monitoring of the network topology and the creation of false attack targets in order to deceive adversaries.
- Decoy services: The defender share fake protocol messages, respond delays and crafted error messages in order to delay the attacker.
- Moving Target Defense: MTD is an asymmetric situation which keeps moving the attack surface of protected systems through dynamic shifting, which can be controlled and managed by the administrator [42].
- Honeypots: Honeypots are built to intentionally expose vulnerable resources to attackers by emulating or simulating systems such as databases, servers and file systems and services such as authentication [13].

Existing works considering defensive deception techniques for computer networks are depicted in Table I, while the classification has been adopted from [41]. As can be seen, various approaches for defensive deception have been applied in traditional computer networks.

TABLE I
DEFENSIVE DECEPTION TECHNIQUES.

Reference	Technique
[43]–[45]	Network Tarpit
[46]	Traffic forging
[47]	Deceptive Technology
[48]	OS obfuscation
[49], [50]	Honeytokens
[51], [52]	Deceptive attack graph
[53]	Decoy services
[54]	Deceptive simulation
[20]	Moving Target Defense
[13]	Honeypots

IV. DEFENSIVE DECEPTION TECHNIQUES ON SDN IMPLEMENTATIONS

The application of defensive deception techniques by leveraging the SDN technology has been extensively studied in the literature. The research papers analyzed in this work are identified by searching in the ACM Digital Library, Science Direct, Scopus, IEEE Xplore and Semantic Scholar databases with appropriate keywords. For the selection of the articles we consider the criterion that the proposed approach should be exclusively related to defensive deception techniques on SDN implementations. These approaches are analyzed below considering their core elements such as the *used systems*, *defensive deception technique*, and the *outcome/results*.

Achleitner et al. in [10] simulate network topologies based on SDN in order to deceive potential attackers targeting the network. The core element in this deception technique is the Reconnaissance Deception System – RDS. The RDS is a reconnaissance deception system which aims to defend the network from potential adversaries. By leveraging a software-defined network virtual network topologies, including its physical components, are simulated. In particular, SDN is responsible to dynamically generate flow rules, analyze flow statistics of the switch rules in order to identify malicious activity and steer and control network traffic by generating rules upon the arrival of the packet.

Zhao et al. in [11] propose a decoy chain deployment (DCD) method based on SDN and NFV as a defensive technique against penetration attacks. The decoy chain consists of a sequence of virtual machines which could operate as decoy switches, middleboxes or terminal hosts. As long as an attacker runs into a decoy chain will continue to penetrate decoy nods without any intrusion to the network. Therefore, the adversary's malicious actions are slowed down, and sensitive targets are protected, while at all times, the SDN controller monitors the security status of the whole network.

Kyung et al. in [12] designed an SDN-based honeynet to globally monitor all internal traffic with the help of the SDN controller. An advance honeynet named HoneyProxy has been proposed to improve data capture capabilities by leveraging the SDN controller. Hence, honeynet provides more flexibility in terms of network access management. Particularly, a honeyproxy using SDN controller monitors the data flows over the network and performs the necessary intervention to the proxy

 ISBN: 978-1-61208-796-2

servers when a honeypot is compromised. Although honeypots constitute one of the most prominent defensive deception techniques, their implementation based on SDN technology is limited.

Han et al. in [13] proposed an SDN-based intelligent honeynet in order to attract attackers and learn about their scope, tactic and behaviour. By leveraging SDN technologies, honeynets can avoid fingerprinting attacks. Namely, HoneyMix leverages the SDN technology in different layers of its architecture toward a more efficient and effective data control. The programmability of the SDN offers a dynamic reconfiguration of the network rules depending on each situation. Furthermore, the SDN switch allows the direct connection among honeypots and hence, increases the protection from honeypot fingerprinting attacks.

Chowdhary et al. in [14] propose a MTD technique based on shuffle strategy using an SDN controller in order to dynamically reconfigure the network and hence, make harder for an attacker to understand the network topology. SDN is prefered due to its ability to reconfigure a cloud-based network topology continuously. In this work, a shuffle based MTD technique has been deployed where the port number for each service or the IP address of a VM are continuously changing.

Kampanakis et al. in [15] studied the application of the SDN technology in network-based MTD techniques. The use of SDN in MTD techniques aims to obfuscate the attacker's actions. SDN implementations improve the defence against port scanning either TCP port scan or UDP and ICMP scans. The proposed implementation clarifies that the SDN could open fault ports which are related to actual services in the network and hence confuse the attacker consistently.

Steinberger et al. in [16] discuss the combination of MTD and SDN in order to reduce the effects of a large-scale cyber-attack. The static configuration of traditional computer networks jeopardizes the infrastructure since attackers can reconnaissance the network and choose the most effective attack in order to cause damage to the network. SDN and MTD aim to address the issues above due to the scalability of SDN. The MTD technique is implemented by leveraging the carrier-grade SDN network operating systems, named ONOS. Two MTD techniques are used in this defensive solution: (i) the network-level MTD, and (ii) the host-level MTD. The former is based on BGP routes and multiple routers while the latter performs IP hopping in order to set up a honeypot.

Makanju et al. propose an Evolutionary Computation (EC) technique for MTD in combination with the SDN technology in [17]. Particularly, EC algorithms have been designed in order to search large spaces for optimal solutions efficiently. The MTD technique facilitates the deployment of a new configuration for the network, considering the network status indicators and the intrusion alerts.

Debroy et al. in [18] proposed an implementation of MTD technique in an SDN in cloud infrastructure. An adequate VM location is proposed by using the SDN controller which directs OpenFlow switches over the cloud infrastructure. The MTD technique's goal is to allow the proactive migration of target nodes in a VM. Additionally, the control module initiates the migration process through the migration initiator module, and thus, the clients are rerouted to a VM using OpenFlow switches. The intrusion detection module ensures the migration process by detecting DoS attacks.

TABLE II
SDN IMPLEMENTATIONS USING DEFENSIVE DECEPTION TECHNIQUES.

SDN	Defensive Deception Techniques
[10]	Deceptive Technologies
[11]	Decoy Services
[12], [13]	Honeypots, Honeytokens
[14]–[17], [17]–[21]	Moving Target Defense

Jafarian et al. in [19] developed a MTD architecture using an SDN controller in order to change the IP addresses of each host dynamically. The OpenFlow Random Host Mutation (OF-RHM) technique assigns a random virtual IP which is translated to/from the real IP of the host in order to overcome the static configuration of the traditional computer networks. The aim of the technique is to protect the network's topology for stealthy scanning, worm propagation and other scanning based attacks.

Macfarland et al. in [20] proposed a MTD application using SDN technology In particular, their technique aims to service unmodified clients while avoiding scalability limitations. An SDN controller provides to defenders the ability to distinguish trustworthy and untrustworthy clients by using pre-shared keys, cryptographic MACs, or embedding passwords into hostnames. The anonymity and unlinkability are ensured by utilizing the SDN controller since it prevents the revocation of the flows across movements over the network.

Aydeger et al. in [21] proposed a defensive mechanism for Crossfire DDoS attacks. By leveraging the SDN technology and MTD technique, the dynamic reconfiguration of the network environment is achieved. The SDN controller enables the management of the traffic flow over the network and is capable of protecting the network from both proactive and reactive attacks. However, this work focuses particularly on crossfire attacks. The defensive deception mechanism consists of four inter-related SDN modules; (i) ICMP monitoring, (ii) Traceroute profiling, (iii) Route mutation, and (iv) Congestion-link monitoring.

Table II depicts current SDN applications considering defensive deception techniques. It can be noticed that defense deception techniques in SDN is immature state since only four out of ten techniques have been applied.

Table IV depicts twelve of the existing techniques considering the five properties of defensive deception techniques. These properties have been adopted from [55] and are depicted in Table III.

Particularly, five out of twelve implementations fulfilled four properties while seven met three out of five properties. We can conclude that MTD techniques aim to increase the attacker's workload and uncertainty. On the other hand, techniques such as honeypots or honeyproxies facilitate the identification of the attack before adversaries succeed. Moreover, five out

 ISBN: 978-1-61208-796-2

TABLE III
DEFENSIVE DECEPTION TECHNIQUES PROPERTIES.

Property	Description
P1	Increase the attacker's workload
P2	Allow defenders to better track attacks and respond before adversaries succeed
P3	Exhaust adversary's resources
P4	Increase the sophistication required for an attack
P5	Increase the attacker's uncertainty

TABLE IV
COMPARISON OF ANALYZED IMPLEMENTATIONS.

Reference	P1	P2	P3	P4	P5
[10]	✓	✓			✓
[11]	✓	✓	✓		
[12]		✓		✓	
[13]	✓	✓		✓	✓
[14]	✓		✓	✓	✓
[15]			✓	✓	✓
[16]	✓	✓	✓		
[17]	✓		✓	✓	✓
[18]	✓	✓		✓	✓
[19]	✓			✓	✓
[20]	✓			✓	✓
[21]	✓	✓		✓	✓
12	10	7	5	9	9

of twelve analyzed studies aim to exhaust the adversary's resources. Although SDN technology can meet the aforementioned defensive deception properties, the fulfilment of all five properties is challenging. To the best of our knowledge, there are not implementations that consider all these properties.

V. DEFENSIVE DECEPTION MECHANISM

Due to the static nature of computer networks, intruders are able to detect their structure and identify vulnerabilities which they can then exploit through advanced attacks. The attackers initially examine a target networks, in order to identify hosts, open ports and map the network topology and to find known or unknown (zero-day) vulnerabilities that will enable them to continue their attack.

The approach proposed herein, aims at misleading such malicious activities by presenting a virtual network topology while it also hides the real network along with possible vulnerabilities that attackers can exploit. Presenting a virtual network falsifies all the information an intruder collects from examining a network and delays the rate of identification of actual vulnerable servers. Delaying the intruder is critical as the extra time given before the actual attack is executed, can facilitate the detection of the intruder and the timely reaction for protecting the network.

In the threat model of this work, it is assumed that the intruder is trying to locate the computers on the network and collect as much information as possible about each computer in order to continue the attack. The main purpose of our deception mechanism is to face those malicious network activities regardless of whether they come from a compromised or a non-compromised host.

Important part of the deception mechanism is the virtual network composition in order to delay and mislead intruders from locating real and possibly vulnerable computers. Furthermore, each host has a different view of the virtual network, so the Deception mechanism is independent of the compromised computer.

The deception mechanism consists of four essential elements:

- an **SDN Controller** responsible for dynamically creating and managing the flow rules in order to direct and control network traffic
- a **Packet Handler** responsible for handling network packets and for simulating specific virtual network resources
- a **Virtual Network Generator** that contains a description of the virtual network components and their connectivity
- as well as **a Honeypot Server** responsible for the services that honeypots will provide to the attacker after a port scanning

When a packet arrives at a switch, the SDN Controller applies a flow rule according to the virtual network topology. Thus, the controller forwards ARP, ICMP, UDP packets to the Packet Handler while it forwards TCP packets to the destination host. When the Packet Handler receives a packet, it creates a response packet according to the virtual topology and sends it back to the source. The source of the packet may be classified as an intruder according to at least one of the following criteria:

- A packet is destined for a honeypot;
- Multiple SYN messages are sent from one source to multiple destination ports of another network host.

The proposed defence deception mechanism has been implemented in the Mininet, which is a SDN Simulator and the POX SDN controller has been used. As mentioned above, the proposed defensive deception mechanism consists of four basic components the functionality of which is analyzed as follows:

SDN Controller is responsible for dynamically creating and managing the flow rules of switches. It is also responsible for creating and periodically updating the virtual network topology. The SDN Controller is able to create flow rules for routing ARP, ICMP, UDP and TCP packets.

As regards the ARP packets, when the switch receives an ARP request it forwards it to the Packet Handler. Then, the Packet Handler creates an ARP reply based on the virtual topology and sends it to the switch. The switch in turn forwards the ARP reply to the switch port from which it received the ARP request.

For ICMP packets, when the switch receives an ECHO request it forwards it to the Packet Handler. Then, the Packet Handler creates an ECHO reply based on the virtual topology and sends it to the switch. The switch in turn forwards the ECHO reply to the switch port from which it received the ECHO request.

For UDP packets, the packet destination port is first checked. If the destination port is 53 then the switch should

forward the DNS query to the Packet Handler. Then, the Packet Handler will create a reply based on the virtual topology and send it back to the switch. The switch in turn must forward the reply to the switch port from which it received the DNS query. If the destination port is between 33434 and 33523, that is, the traceroute command has been used; the switch should forward the request to the Packet Handler. Then, the Packet Handler will create a reply based on the virtual topology and send it to the switch. The reply could be ICMP time exceeded or ICMP destination/port unreachable. The switch in turn must forward the reply to the switch port from which it received the request.

In the case of TCP packets, when a switch receives a TCP packet it will forward it to its destination given that the source of the packet has not been identified as an intruder from the SDN controller. A host can be identified as an intruder by the SDN Controller in two cases. The first case is if a host interacts with a honeypot through ping or traceroute commands. The second case is if a host makes a port scanning attempt against another host on the network. In order to detect port scanning attempts, the number of SYN messages sent from a host along with the corresponding destination ports are examined. In the case that a host is classified as an intruder, its network activity is forwarded to the Honeypot Server. Actually, when the switch receives a TCP packet from such a host it replaces its destination IP and MAC addresses with those of the Honeypot Server. Then, in the Honeypot Server response the switch replaces the source IP and MAC addresses with the original addresses and forward it to the switch port where the intruder is connected. The main reason that TCP packets are forwarded to the Honeypot Server instead of the Packet Handler is because in that case we can increase the level of deception for the intruder by using a honeypot. The Honeypot server allows for more interaction with the intruder, which in the case of complex TCP connections can end up to with the intruder having a largely false perception of the network.

Packet Handler is responsible for generating and sending packets, according to the virtual topology created by the virtual topology generator, when this is required. Packet Handler has been implemented in Python while the Scapy Framework has also been used. The role of the packet handler is taken up by one of the hosts created in Mininet. Thus, the packets will be forwarded by the switch to the Packet Handler, which in turn will generate a response and send it back to the switch.

Honeypot Server is responsible for generating the virtual set of services for each host, that will be provided to the attacker after a port scanning attempt. As mentioned above, it is not allowed to an intruder to perform a port scanning on the actual network and the network traffic will be routed to a honeypot. Thus, the need arises to present a different set of services for each scanned computer. One of the hosts created in the Mininet has the role of Honeypot Server. Honeypot Server has also been implemented in Python and starts services on the computer on which is running. Also, it should be mentioned that these services are periodically updated.

Virtual topology generator is responsible for creating the virtual topology as well as periodically updating the virtual IP and MAC addresses of the Mininet hosts and honeypots. This generator creates a text file that is accessible from the SDN controller and Packet Handler. Each line of this file corresponds to a component of the deception mechanism. Specifically, there are four types of entities, the Packet Handler, the hosts, the fake-routers and the routes.

The Packet Handler is unique to the network and is connected at port 1 of the switch. In addition, the text file contains information about the real IP and MAC addresses and about the virtual IP and MAC addresses.

As regards hosts, they can either be Mininet hosts or honeypots. In addition, the text file contains information about real IP and MAC addresses, virtual IP and MAC addresses as well as the switch port in which is connected the Packet Handler. If the entity is a Mininet host, then there is information about the port that is connected to the switch. If the entity is a honeypot, there is information about the port that the Honeypot Server is connected to the switch. From the information contained in host-type rows, the Packet Handler can respond to ARP, ICMP and UDP requests. The SDN controller knows the IP addresses of honeypots in order to designate a host as an intruder. Also, there is information on routing TCP packets.

The fake-router rows contain information about virtual routers that the deception mechanism will use to deceive the traceroute command. This information is about the router interface, the virtual IP and MAC addresses as well as the switch port that the Packet Handler is connected to.

Route rows contain information about virtual routes from one host to another. As well as, fake routers are used to mislead the traceroute command. This information is about a source host, a destination host, and the intermediate hops which are the fake router's interfaces mentioned above.

As mentioned above, one of the functions of the virtual topology generator is to periodically update the IP and MAC addresses in order to increase the attacker's uncertainty about the target host. However, there is a limitation regarding the termination of the TCP connections by changing the IP and MAC addresses. The number of subnets created by this mechanism is static and the optimal way to create a virtual topology given a real topology is a future track of research.

In order to present the functionality of the Defensive Deception Mechanism components as a whole, two different scenarios are described, one for an attacker scanning hosts in a network and a second one for an attacker scanning services on a host.

In the first scenario, we assume that an attacker will interact with a host on the network via the ping command. Thus, the switch will receive an ARP request which will be forwarded to the SDN Controller. The SDN Controller will send two flow rules to the switch. According to the first flow rule the switch will forward the ARP request to the Packet Handler. In turn, Packet Handler will generate an ARP reply and send it to the switch. According to the second flow rule the switch will forward the ARP reply to the switch port from which it received the ARP request. After that, an ECHO request will

 ISBN: 978-1-61208-796-2

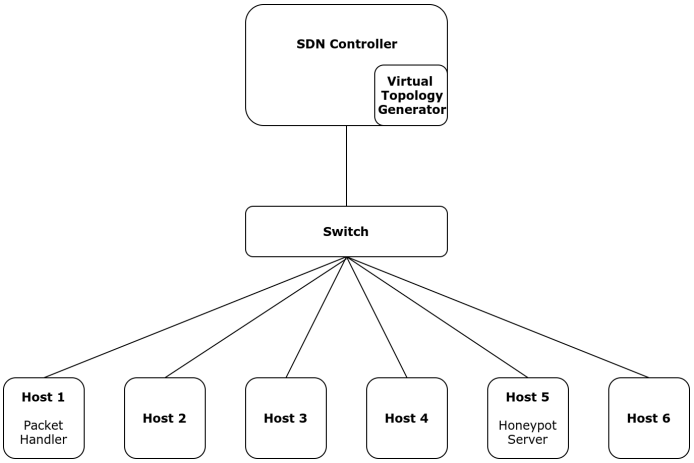

Fig. 1. Topology of implementation

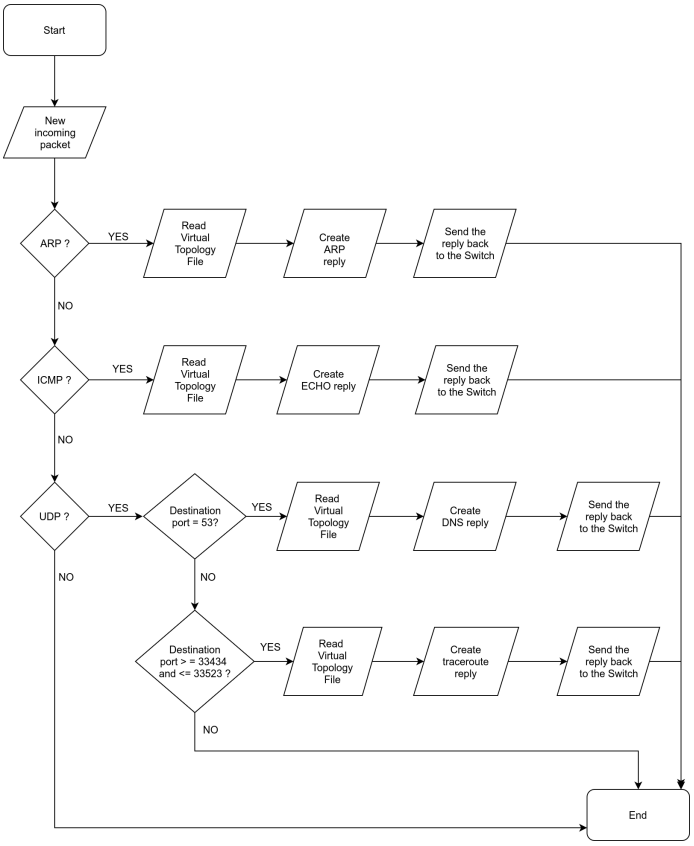

Fig. 2. Packet handler flowchart

be sent from the attacker's computer. The switch will forward it to the SDN Controller and the SDN Controller will create two flow rules. According to the first flow rule, the switch will forward the ECHO request to the Packet Handler. In turn, Packet Handler will generate an ECHO reply and send it back to the switch. According to the second flow rule the switch will forward the ECHO reply to the switch port from which it received the ECHO request. Figure 4 depicts the sequence diagram for the aforementioned scenario.

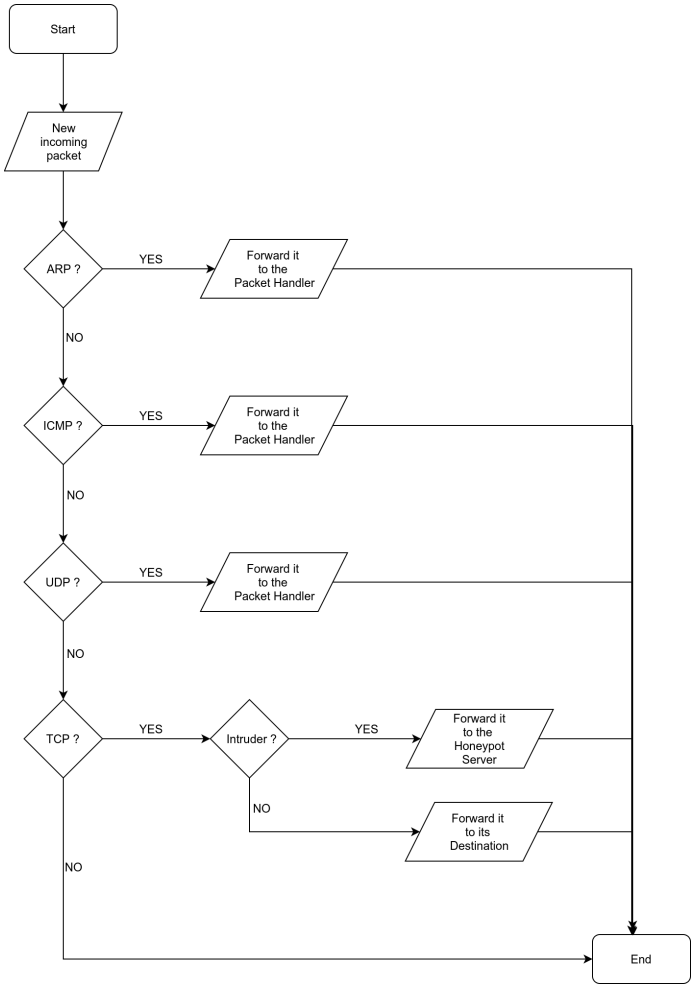

Fig. 3. SDN controller flowchart

In the second scenario, we assume that the attacker will interact with a host on the network through the nmap program. Based on the function of the nmap program, several SYN-type messages will be sent from the attacker's computer to many host's ports. All of these messages will be forwarded from the switch to the SDN Controller, and once the number of those SYN messages surpasses a specified limit for a multitude of different ports then the SDN Controller will create two flow rules. According to the first flow rule, when the switch receives a TCP packet, from the attacker's computer, it replaces its destination IP and MAC addresses with those of the Honeypot Server. According to the second flow rule, in the Honeypot Server response the switch replaces the source IP and MAC addresses with the original addresses and forward it to the switch port where the attacker is connected. Figure 5 depicts the sequence diagram for the second scenario.

VI. RESULTS

In this section, the effectiveness of the deception defense mechanism discussed above is examined. For this purpose the nmap port scan tool was used, in order to scan a network protected by the proposed defensive deception mechanism.

 ISBN: 978-1-61208-796-2

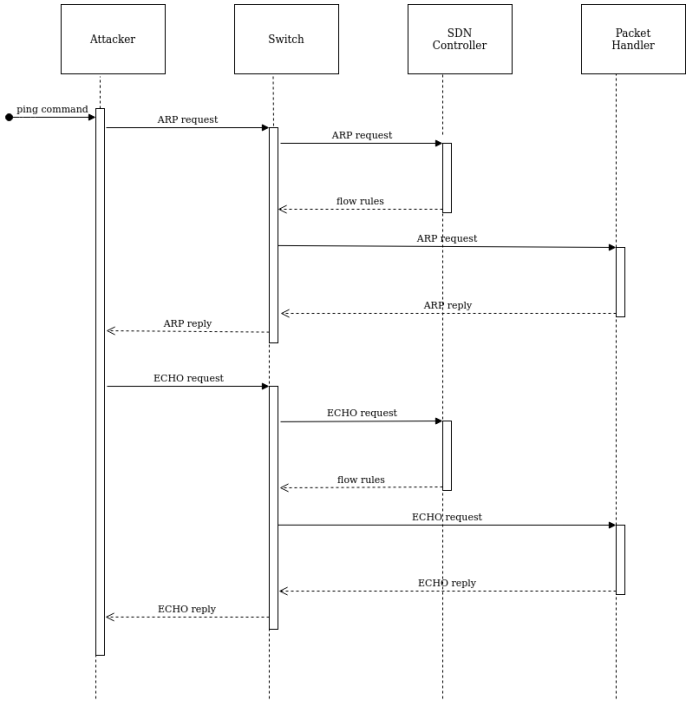

Fig. 4. Ping command sequence diagram

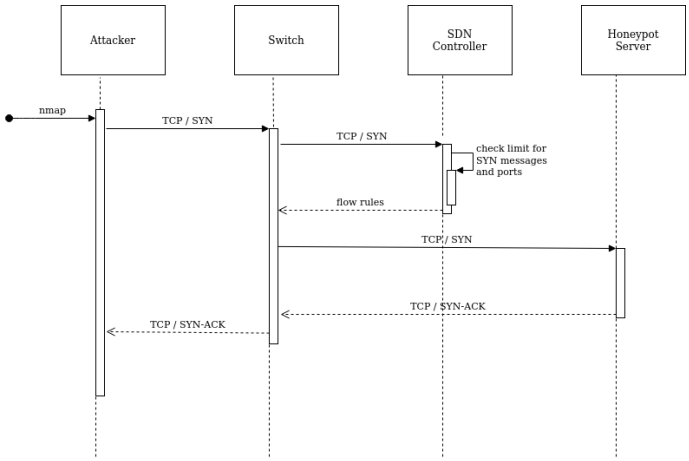

Fig. 5. Nmap sequence diagram

The results will be based on the defensive deception techniques properties presented in Table III.

Initially, an SDN network consisting of a switch and six computers was created in Mininet. The defensive deception mechanism created three sub-networks, each consisting of eight honeypots and two real computers. In total there are twenty four honeypots and six real computers. The ping sweep function of the nmap tool will be used to locate the computers running on the network. The results of the scans on the three subnets as well as the time taken for each scan are presented in Figure 6 and Figure 7.

In these scans it was observed that we can increase the number of computers in a network by increasing the number

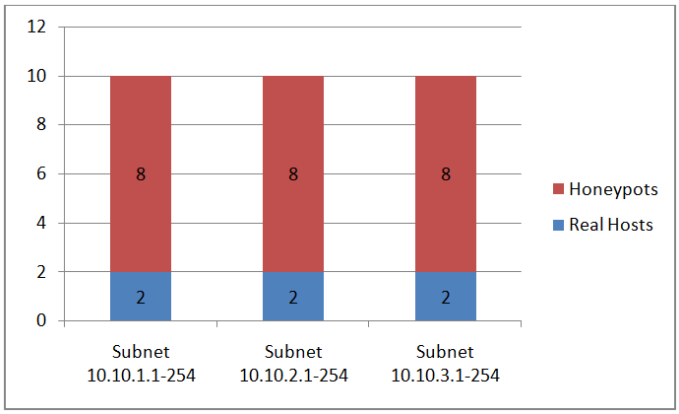

Fig. 6. Nmap ping sweep results (computers per subnet)

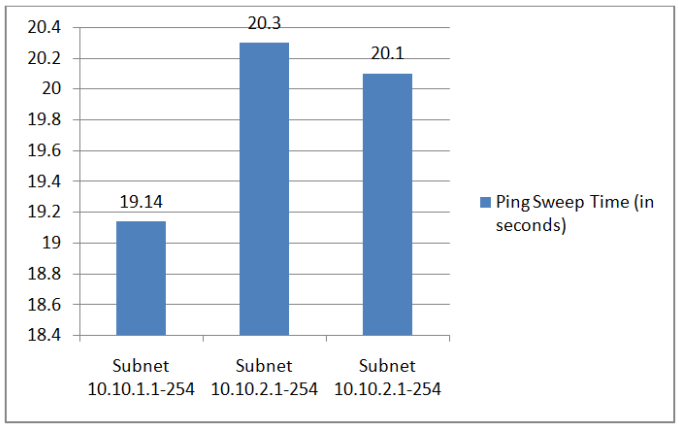

Fig. 7. Nmap ping sweep time (in seconds)

of honeypots. Thus, we increase the workload (P1) and uncertainty (P5) of the attacker. Also, we are able to track attacks and respond before attackers succeed (P2).

Each of the network computers that responded to the ping sweep will then be examined separately. For this reason, the nmap tool will be used again to identify the services that each computer hosts. The following graphs, Figure 8 and Figure 9, show the number of services that each computer runs and the time required for each scan.

In these scans it has been observed that we can vary the number of services running on a computer. Thus, we increase the workload (P1) and uncertainty (P5) of the attacker. Also, We are able to detect attacks and respond before attackers succeed (P2).

VII. Conclusions

As computer networks become more complex and attacks against those become more sophisticated, the mitigation efficiency of passive static countermeasures is going to decline. In order to be able to cope up with protecting modern computer networks from attackers we are required to come up with more sophisticated solutions such as SDN based deceptive defense techniques, that can delay and reveal the attacker.

 ISBN: 978-1-61208-796-2

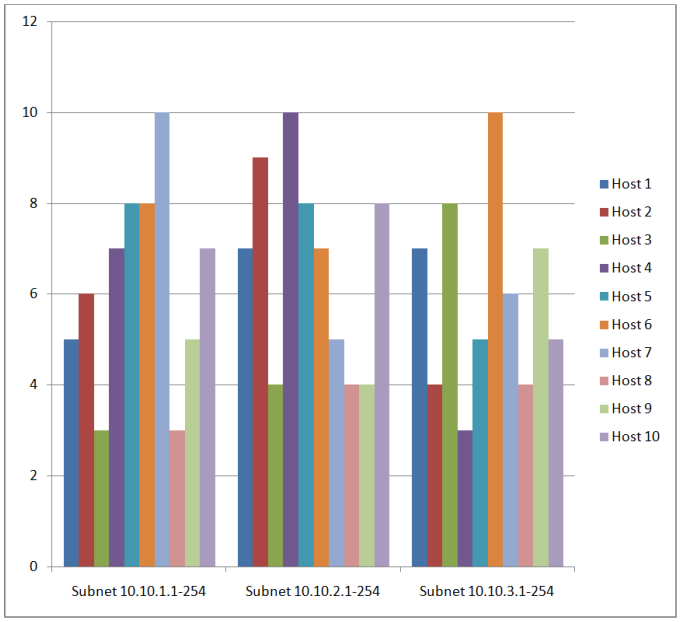

Fig. 8. Nmap scan results (open ports per computer)

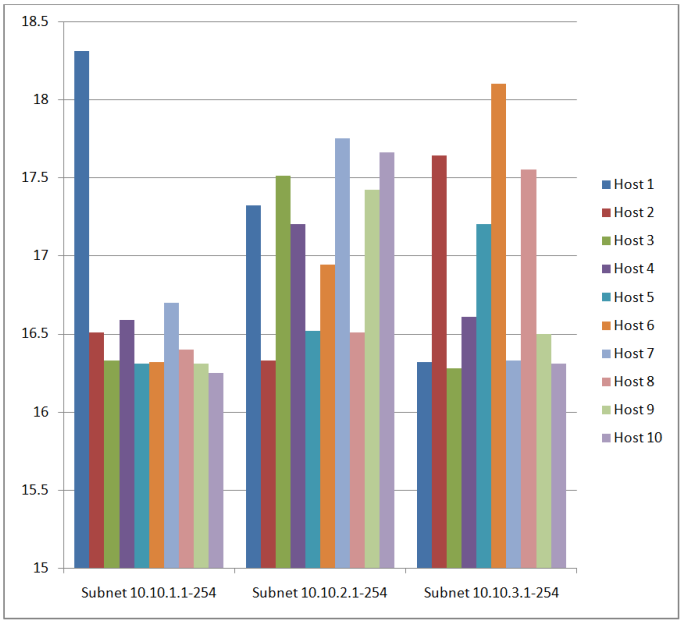

Fig. 9. Nmap scan time (in seconds)

Our comprehensive literature review showed that there are multiple efforts to set up such deceptive defense mechanisms based on SDN and the results are promising. The different methods have been analysed with respect to their main properties according to Table III.

Furthermore, we introduce a novel defensive deception mechanism that leverages the capabilities provided by SDN technologies. The proposed mechanism combines components from multiple defensive deception techniques as identified in the examined prior studies (see Table II), namely Deceptive Technologies, Honeypots, and Moving Target Defense. Finally, we have implemented the proposed mechanism and evaluated its capacity to satisfy the defensive deception mechanisms properties (see Table III), and more specifically to increase the attacker's workload, to allow defenders to better track attacks and respond before adversaries succeed, and to increase the attacker's uncertainty.

As future work, the proposed system is going to be enhanced both in terms of detection capabilities and prevention mechanisms. Detection is going to be expanded through the use of machine learning techniques. Prevention is going to be revised, to enable deception techniques to adapt to each protected network in terms of size and type of connected hosts or devices.

REFERENCES

[1] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2014.

[2] C. Layne, "Cyber attacks against critical infrastructure," Ph.D. dissertation, Utica College, 2017.

[3] D. Wenda and D. Ning, "A honeypot detection method based on characteristic analysis and environment detection," in *2011 International Conference in Electrics, Communication and Automatic Control Proceedings*. Springer, 2012, pp. 201–206.

[4] O. Schneider and N. Giller, "Method for mitigation of cyber attacks on industrial control systems," Jul. 3 2018, uS Patent 10,015,188.

[5] K. Hoffman, D. Zage, and C. Nita-Rotaru, "A survey of attack and defense techniques for reputation systems," *ACM Computing Surveys (CSUR)*, vol. 42, no. 1, pp. 1–31, 2009.

[6] S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, *Moving target defense: creating asymmetric uncertainty for cyber threats*. Springer Science & Business Media, 2011, vol. 54.

[7] C. Lei, H.-Q. Zhang, J.-L. Tan, Y.-C. Zhang, and X.-H. Liu, "Moving target defense techniques: A survey," *Security and Communication Networks*, vol. 2018, 2018.

[8] B. C. Ward, S. R. Gomez, R. Skowyra, D. Bigelow, J. Martin, J. Landry, and H. Okhravi, "Survey of cyber moving targets second edition," MIT Lincoln Laboratory Lexington United States, Tech. Rep., 2018.

[9] S. Rowshanrad, S. Namvarasl, V. Abdi, M. Hajizadeh, and M. Keshtgary, "A survey on sdn, the future of networking," *Journal of Advanced Computer Science & Technology*, vol. 3, no. 2, pp. 232–248, 2014.

[10] S. Achleitner, T. F. La Porta, P. McDaniel, S. Sugrim, S. V. Krishnamurthy, and R. Chadha, "Deceiving network reconnaissance using sdn-based virtual topologies," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 1098–1112, 2017.

[11] Q. Zhao, C. Zhang, and Z. Zhao, "A decoy chain deployment method based on sdn and nfv against penetration attack," *PloS one*, vol. 12, no. 12, 2017.

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

[13] W. Han, Z. Zhao, A. Doupé, and G.-J. Ahn, "Honeymix: Toward sdn-based intelligent honeynet," in *Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, 2016, pp. 1–6.

[14] A. Chowdhary, A. Alshamrani, D. Huang, and H. Liang, "Mtd analysis and evaluation framework in software defined network (mason)," in *Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, 2018, pp. 43–48.

[15] P. Kampanakis, H. Perros, and T. Beyene, "Sdn-based solutions for moving target defense network protection," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*. IEEE, 2014, pp. 1–6.

 ISBN: 978-1-61208-796-2

[16] J. Steinberger, B. Kuhnert, C. Dietz, L. Ball, A. Sperotto, H. Baier, A. Pras, and G. Dreo, "Ddos defense using mtd and sdn," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2018, pp. 1–9.

[17] A. Makanju, A. N. Zincir-Heywood, and S. Kiyomoto, "On evolutionary computation for moving target defense in software defined networks," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2017, pp. 287–288.

[18] S. Debroy, P. Calyam, M. Nguyen, A. Stage, and V. Georgiev, "Frequency-minimal moving target defense using software-defined networking," in *2016 international conference on computing, networking and communications (ICNC)*. IEEE, 2016, pp. 1–6.

[19] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow random host mutation: transparent moving target defense using software defined networking," in *Proceedings of the first workshop on Hot topics in software defined networks*, 2012, pp. 127–132.

[20] D. C. MacFarland and C. A. Shue, "The sdn shuffle: creating a moving-target defense using host-based software-defined networking," in *Proceedings of the Second ACM Workshop on Moving Target Defense*, 2015, pp. 37–41.

[21] A. Aydeger, N. Saputro, K. Akkaya, and M. Rahman, "Mitigating crossfire attacks using sdn-based moving target defense," in *2016 IEEE 41st Conference on Local Computer Networks (LCN)*. IEEE, 2016, pp. 627–630.

[22] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden, "A survey of active network research," *IEEE communications Magazine*, vol. 35, no. 1, pp. 80–86, 1997.

[23] N. Feamster, J. Rexford, and E. Zegura, "The road to sdn," *Queue*, vol. 11, no. 12, pp. 20–40, 2013.

[24] A. T. Campbell, I. Katzela, K. Miki, and J. Vicente, "Open signaling for atm, internet and mobile networks (opensig'98)," *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 1, pp. 97–108, 1999.

[25] A. Doria, F. Hellstrand, K. Sundell, and T. Worster, "General switch management protocol (gsmp) v3," 2002.

[26] J. Biswas, A. A. Lazar, J.-F. Huard, K. Lim, S. Mahjoub, L.-F. Pau, M. Suzuki, S. Torstensson, W. Wang, and S. Weinstein, "The ieee p1520 standards initiative for programmable network interfaces," *IEEE Communications Magazine*, vol. 36, no. 10, pp. 64–70, 1998.

[27] M. Casado, T. Garfinkel, A. Akella, M. J. Freedman, D. Boneh, N. McKeown, and S. Shenker, "Sane: A protection architecture for enterprise networks." in *USENIX Security Symposium*, vol. 49, 2006, p. 50.

[28] A. Doria, J. H. Salim, R. Haas, H. M. Khosravi, W. Wang, L. Dong, R. Gopal, and J. M. Halpern, "Forwarding and control element separation (forces) protocol specification." *RFC*, vol. 5810, pp. 1–124, 2010.

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

[30] H. Song, "Protocol-oblivious forwarding: Unleash the power of sdn through a future-proof forwarding plane," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, 2013, pp. 127–132.

[31] J. E. Van der Merwe, S. Rooney, L. Leslie, and S. Crosby, "The tempest-a practical framework for network programmability," *IEEE network*, vol. 12, no. 3, pp. 20–28, 1998.

[32] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: an overlay testbed for broad-coverage services," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 3–12, 2003.

[33] M. R. Macedonia and D. P. Brutzman, "Mbone provides audio and video across the internet," *Computer*, vol. 27, no. 4, pp. 30–36, 1994.

[34] L. L. Peterson, T. Anderson, D. Blumenthal, D. Casey, D. Clark, D. Estrin, J. Evans, D. Raychaudhuri, M. Reiter, J. Rexford *et al.*, "Geni design principles," *Computer*, vol. 39, no. 9, pp. 102–105, 2006.

[35] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "In vini veritas: realistic and controlled network experimentation," in *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, 2006, pp. 3–14.

[36] S. Ortiz, "Software-defined networking: On the verge of a breakthrough?" *Computer*, no. 7, pp. 10–12, 2013.

[37] S. Shin, Y. Song, T. Lee, S. Lee, J. Chung, P. Porras, V. Yegneswaran, J. Noh, and B. B. Kang, "Rosemary: A robust, secure, and high-performance network operating system," in *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, 2014, pp. 78–89.

[38] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, "A security enforcement kernel for openflow networks," in *Proceedings of the first workshop on Hot topics in software defined networks*, 2012, pp. 121–126.

[39] K. Benton, L. J. Camp, and C. Small, "Openflow vulnerability assessment," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, 2013, pp. 151–152.

[40] D. Fraunholz, S. D. Anton, C. Lipps, D. Reti, D. Krohmer, F. Pohl, M. Tammen, and H. D. Schotten, "Demystifying deception technology: A survey," *arXiv preprint arXiv:1804.06196*, 2018.

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

[42] G.-l. Cai, B.-s. Wang, W. Hu, and T.-z. Wang, "Moving target defense: state of the art and characteristics," *Frontiers of Information Technology & Electronic Engineering*, vol. 17, no. 11, pp. 1122–1153, 2016.

[43] T. Liston, "Labrea:"sticky" honeypot and ids," 2001.

[44] K. Borders, L. Falk, and A. Prakash, "Openfire: Using deception to reduce network attacks," in *2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops-SecureComm 2007*. IEEE, 2007, pp. 224–233.

[45] L. Shing, "An improved tarpit for network deception," Naval Postgraduate School Monterey United States, Tech. Rep., 2016.

[46] E. Le Malécot, "Mitibox: camouflage and deception for network scan mitigation," in *Proceedings of the 4th USENIX conference on Hot topics in security*. USENIX Association, 2009, pp. 4–4.

[47] S. T. Trassare, "A technique for presenting a deceptive dynamic network topology." Naval Postgraduate School Monterey United States, Tech. Rep., 2013.

[48] M. Smart, G. R. Malan, and F. Jahanian, "Defeating tcp/ip stack fingerprinting." in *Usenix Security Symposium*, 2000.

[49] B. M. Bowen, V. P. Kemerlis, P. Prabhu, A. D. Keromytis, and S. J. Stolfo, "Automating the injection of believable decoys to detect snooping," in *Proceedings of the third ACM conference on Wireless network security*, 2010, pp. 81–86.

[50] S. Chakravarty, G. Portokalidis, M. Polychronakis, and A. D. Keromytis, "Detecting traffic snooping in tor using decoys," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2011, pp. 222–241.

[51] F. Cohen, I. Marin, J. Sappington, C. Stewart, and E. Thomas, "Red teaming experiments with deception technologies," *IA Newsletter*, 2001.

[52] F. Cohen and D. Koike, "Leading attackers through attack graphs with deceptions," *Computers & Security*, vol. 22, no. 5, pp. 402–411, 2003.

[53] N. Provos *et al.*, "A virtual honeypot framework." in *USENIX Security Symposium*, vol. 173, no. 2004, 2004, pp. 1–14.

[54] J. L. Rrushi, "An exploration of defensive deception in industrial communication networks," *International Journal of Critical Infrastructure Protection*, vol. 4, no. 2, pp. 66–75, 2011.

[55] F. Cohen, "The use of deception techniques: Honeypots and decoys," *Handbook of Information Security*, vol. 3, no. 1, pp. 646–655, 2006.

 ISBN: 978-1-61208-796-2