Proceedings of the 52nd Hawaii International Conference on System Sciences | 2019

Cyber Deception Architecture: Covert Attack Reconnaissance Using a Safe SDN Approach

Toru Shimanaka
Fujitsu System Integration Laboratories
shimanaka.tohru@jp.fujitsu.com

Ryusuke Masuoka
Fujitsu System Integration Laboratories
masuoka.ryusuke@jp.fujitsu.com

Brian Hay
Hume Center
Virginia Tech
brianhay@vt.edu

Abstract

Significant valuable information can be determined by observing attackers in action. These observations provide significant insight into the attacker's TTPs and motivations. It is challenging to continue observations when attackers breach operational networks. This paper describes a deception network methodology that redirects traffic from the compromised Operational Network (O-Net) to an identically configured Deception Network (D-Net) minimizing any further compromise of operational data and assets, while also allowing the tactics, techniques, and procedures of the attacker to be studied. To keep the adversary oblivious to the transfer from the O-Net to the D-Net, we employ a sophisticated and unique packet rewriting technique using Software Defined Networking (SDN) technology that builds on two other strategies. This paper discusses the foundational strategies and introduces a new strategy that improves behavior for our described scenarios. We then provide some preliminary test results and suggest topics for further research.

1. Introduction

1.1. Background

An adversary who conducts Advanced Persistent Threat (APT) cyber attacks is often a nation state or an organization backed by significant resources and with purposes beyond monetary gain (e.g. data theft, establishing long-term presence, etc.). As such, their attacks are targeted and very sophisticated. They are determined to penetrate the target's well-protected networks and can maintain an undetected presence in the network for a long period of time [1]. In addition, they are likely to come back in, using alternate attack vectors, even if some (or all) of their previous activities are discovered and mitigated.

They frequently follow the Cyber Kill Chain Methodology [2]; conducting an extensive survey of their target organization and developing malware or attack methodologies customized for the target's environment before they start their attacks. Then the adversary uses spear phishing, watering holes, supply chain attacks, insiders, and/or other techniques to deliver their payload and build a beachhead inside the target's network. When successful, a malicious backdoor program can be installed on one of the compromised devices in the network to build a remote operation environment connected to their external C2 (Command and Control) server. The adversary is then able to conduct reconnaissance on the compromised network to find additional targets and/or discover where sensitive information is stored, often moving laterally within the target environment to reach more strategic positions. During such time, the adversary generally takes steps to avoid detection and uses legitimate tools and commands as much as possible so that it is difficult to discern the adversary's activities from legitimate ones. Depending on the overall objective, they might exfiltrate sensitive data and take actions to cover their tracks, leave false flags and indicators, or maintain their stealthy existence within the target network.

Many cybersecurity textbooks and industry best-practices dictate that when a compromised PC is discovered, it should be disconnected or quarantined from the network to prevent further damage. However, when dealing with APT attacks, this procedure is not always the best approach, as it often results in a loss of valuable information about the attack and the adversary. Even if you identify and stop the intrusion once, the adversary could learn from their failure and be very likely to come back again using more sophisticated tools and techniques which may be more challenging to detect. After detecting a suspected APT attack in progress, it can be used as an opportunity to apply cyber deception [3,

URI: https://hdl.handle.net/10125/60166
ISBN: 978-0-9981331-2-6

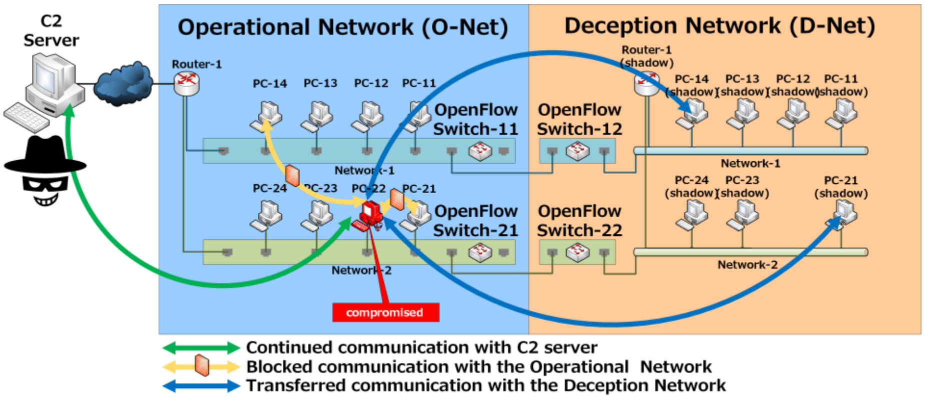


Figure 1. Allow the adversary uninterrupted remote-control of the compromised PC from the C2 server while transferring the network activities of the compromised PC from the O-Net to the D-Net

4] to obtain intelligence on the adversary, identify their TTPs, understand their purposes and intentions, and potentially, to keep them complacent with their current TTPs and delay the development and use of more sophisticated tools and tactics.

1.2. Challenges

Deploying cyber deception (vs. immediately shutting down an intrusion and patching the system), however, is a potentially dangerous reaction as we are allowing the adversary to continue the attack. In addition, the Cyber Deception campaign needs to be conducted covertly so that the adversary does not notice what is going on and alter behavior. When operating a Deception Network, it is important to both contain and observe the attack in real time and do so safely and covertly. More specifically, we need to accomplish the followings:

1) Switch communications between the compromised host(s) and endpoints on the Operational Network to corresponding endpoints on the Deception Network without any adverse side effects.
2) Maintain the session between the compromised host(s) and the C2 server out on the Internet through the process described in (1) above.
3) Ensure that these defensive operations do not provide the adversary with observable effects that could alert them to the cyber deceptive activities.

These are the challenges that this article and our technical solution address.

1.3. Core Concept

The most important objective of all is that the adversary does not notice that we are conducting a cyber deception operation. To achieve this, we deceive the adversary into believing they are maintaining control of the compromised host on the Operational Network (O-Net) from their command and control (C2) server. In addition, it should also appear to the adversary that they are communicating with the other network nodes (PCs and servers) on the O-Net, through the compromised host, without any observable differences in behavior during the changeover to the Deception Network (D-Net). This deception effect is accomplished by ensuring the D-Net is configured nearly identically to the O-Net. (Figure 1). Once the adversary resides within the D-Net, we can monitor all activity in a safe environment, allowing normal operations to continue on the O-Net.

To achieve this effect, we employ Software Defined Networking (SDN) technologies (See Sec. 2 for details). Each O-Net subnet and the corresponding D-Net subnet are connected through two OpenFlow switches as shown in Figure 1. We give the D-Net the same network configuration and each endpoint (e.g. workstations, servers, and routers) uses the same IP addresses, name, and roles/functions as the corresponding endpoint on the O-Net. The primary differences between the O-Net and the D-Net are (1) the MAC addresses of endpoints and, (2) there is only non-sensitive or fake information on the D-Net. Information accessed or stolen from the D-Net does not impact operations and may be selected or created so as to deliberately misinform the adversary.

Whenever a compromise is detected on the O-Net, we start rewriting the packet information flowing between the compromised host and other endpoints on the O-Net, resulting in each flow being directed into the D-Net. As we do not know in advance which host may become compromised, it is necessary to rewrite packets dynamically. For that purpose, we picked Software Defined Networking (SDN) technology, specifically OpenFlow-enabled switches and a corresponding OpenFlow controller (Ryu). Flow tables of OpenFlow are used extensively to match and process packets to enable necessary packet rewriting.

When rewriting packets using flow tables to deceive the adversary, we must consider a few objectives. Communications between the compromised hosts and the endpoints on the O-Net need to be switched to ones the corresponding D-Net hosts without any noticeable effect. In addition, communication between the compromised host and the C2 server outside of the target's organization needs to continue uninterrupted. With the naive "match packet IP address, then rewrite its MAC address" (strategy #1), you can transfer UDP packets from the O-Net to the D-Net, but TCP communication cannot be established as the ARP information on the endpoint on D-Net does not get updated. Since TCP communication fails to establish, strategy #1 is not an acceptable strategy.

With "match packet MAC address and ARP packets, then rewrite its MAC information" (strategy #2), this enables TCP communication within the O-Net subnet and the D-Net subnet where the compromised host resides. However, it cannot sustain communications between the compromised host and both the C2 server and the other D-Net subnets at the same time. This is because the packets to the C2 server and the packets to the other D-Net subnets require them to be sent through the different routers respectively.

To fully achieve our goals, strategy #3, which employs both strategy #2 and the new "match packet network IP address, then switch port accordingly", is utilized. This allows for TCP communications between the compromised host and the O-Net to be transferred to the D-Net without noticeable effects and allows for TCP communications between the compromised host and the C2 server to continue uninterrupted. Table 1 summarizes the three strategies.

1.4. Structure of this paper

The remainder of this paper is organized as follows. Section 2 provides a brief introduction to SDN and OpenFlow. Section 3 describes work related to this research. Section 4 describes the architecture and implementation of the proposed deception technique. Section 5 describes evaluation results of the proposed technique. Section 6 concludes the paper and provides some avenues for future research.

Table 1 . Packet rewriting strategies

Strategy	Description	Comments
#1	match packet IP address, then rewrite its MAC address	Naive, works only for UDP packets
#2	match packet MAC address and ARP packets, then rewrite its MAC information	Works for TCP within the subnets, but not for comm. with both the C2 servers and other D-Net subnets at the same time
#3	strategy #2 + match packet network IP address, then switch port accordingly	This solution works for all internal and external communications

2. SDN and OpenFlow

Software Defined Networking (SDN) is an architecture that dynamically controls the network with software. OpenFlow [5] is one of the SDN implementations, and its standardization is advanced by the Open Network Foundation (ONF). OpenFlow has the following features:

1) *Separation of control plane and data plane*

There are two functions for switches: to communicate with other switches to determine how network traffic should be forwarded; and to then actually forward (or drop) packets accordingly. The former occurs in the control plane, and the latter in the data plane. For legacy switches, those two functions happen in the same place, namely within the switch. For OpenFlow, those two planes are separated with the control plane activities being moved to an external OpenFlow Controller, which dictates traffic forwarding rules to the switch in the form of flow table entries. The data plane remains on the switch and utilizes the controller-provided flow tables to make the necessary forwarding decisions.

2) *Flexible packet processing*

Flow tables enable flexible packet processing. The OpenFlow Controller adds, removes, or modifies entries in the flow tables of its associated switches. These rules can not only cause the switch to forward or drop packets, but also result in packets that are rewritten on the fly or sent to a specific set of output ports. The rule(s) applied to a given packet are selected based on matching fields between the rule and incoming packet.

In this research, we created an OpenFlow controller using the Ryu [6] framework. In addition, we implemented a REST API to configure the flow tables using Northbound APIs of the OpenFlow Controller. (Northbound APIs are APIs to control an OpenFlow Controller from an application.) We used the Open vSwitch [7] as our OpenFlow switch implementation, although the techniques are not specific to that particular switch and could be applied to a variety of other OpenFlow enabled devices.

3. Related work

There have been several attempts to covertly observe cyber attacks before. We describe four major approaches and compare them with our approach:

3.1. Sandbox

A sandbox such as Cuckoo [8] is a type of malicious software analysis system. It simulates physical operating systems in a virtual environment. The sandbox executes or opens potentially malicious artifacts (ex. codes and documents) in an isolated environment and observes the resulting behavior of the system. This approach can be effective at observing the types of activity applied in the exploitation stage of the kill chain, but are less useful if one wants to observe advanced post-compromise activity such as lateral movement and practically no use to understand adversary's intentions. Sandboxes do have limitations, which include anti-sandboxing mechanisms in malware itself (e.g., timeouts before malicious activity begins, detection of system artifacts that are typically found in sandboxes, and detection of human behavior or recent activity which is often absent in sandboxes) [9]. Our focus is to observe how a human adversary performs his/her attack after a successful malware infection and has established a beachhead.

3.2. Honeypot

A honeypot is a decoy computer system designed to look like a legitimate system an adversary will want to break into while, unbeknownst to the adversary, they are being covertly observed [10]. A honeypot is generally deployed on the perimeter of the organization's network such as an Internet facing server. They can also be placed throughout an organization's network, but it requires the adversary to be lured to the honeypot through the Operational Network (O-Net), and it can be a dangerous and high-risk process. We transfer the attack to the Deception Network (D-Net), which is a type of high-interaction honeypot within the organization as soon as a compromise on one of endpoints is detected. This does not require luring the adversary through the O-Net and the adversary cannot access the O-Net once the attack is contained within the D-Net.

3.3. Moving Target Defense (MTD)

Vincent E. Urias et al. proposed the Moving Target Defense (MTD), whose aim is to increase attack difficulty [11] by dynamically changing the targeted network. This method differs from our purpose of observing attacks safely and covertly. Though they prepare the Deception Network (D-Net) with the same configuration as the Operational Network (O-Net) to contain the attack, they create the O-Net in a fully realized virtual environment. Our architecture consists of an O-Net consisting of actual physical PCs, servers and network equipment and the D-Net built in a virtual environment. Although we usually use a D-Net in a virtual environment, the D-Net can be physical as well.

3.4. Deception on Operational Networks

Recent cyber deception technologies are interwoven directly into the Operational Networks (O-Nets) for detection, diversion, resource depletion, uncertainty, and intelligence purposes.

Reconnaissance Deception System (RDS) was proposed in [12] to delay or thwart malicious network reconnaissance. This is done through providing the adversary a different virtual network view at the assignment of a new DHCP lease by virtually blowing up a single subnet into a multitude of virtual subnets with hosts on the original subnet scattered randomly among them along with honeypots. The network topology can appear different for the adversary every time the new virtual network view is provided. This is confusing, but not stealthy nor fit for our intelligence purpose. From the technical perspective, its deception happens within the scope of a single subnet and does not involve more sophisticated packet rewriting strategies like strategy #3 in Table 1.

Shadow Networks [13, 14] is a solution that leverages the advantages of both low- and high-interaction honeypots. It projects (connects through virtual switches) many low-interaction honeypots onto the O-Net. When an attacker probes into one of low-interaction honeypots, a high-interaction honeypot can be swapped in to take its place. When a connection is attempted from the one of the honeypots to a physical computer, a host emulator can step in to take the place of the physical computer. In this system, SDN is used to prevent collisions between the duplicated IP addresses. To realize this, Shadow Networks changes the destination to another one within the O-Net (likely within the same subnet) through relatively straight-forward packet manipulations and does not involve more sophisticated packet rewriting strategies like strategy #3 in Table 1.

4. Architecture and Implementation

We propose a Cyber Deception Architecture consisting of a network configuration and an attack transferring mechanism to transfer network communications from the O-Net to the D-Net, using the OpenFlow technologies. In this section, we describe its architecture, with a focus on how the transfer mechanism is implemented by using OpenFlow.

4.1. Network Configuration

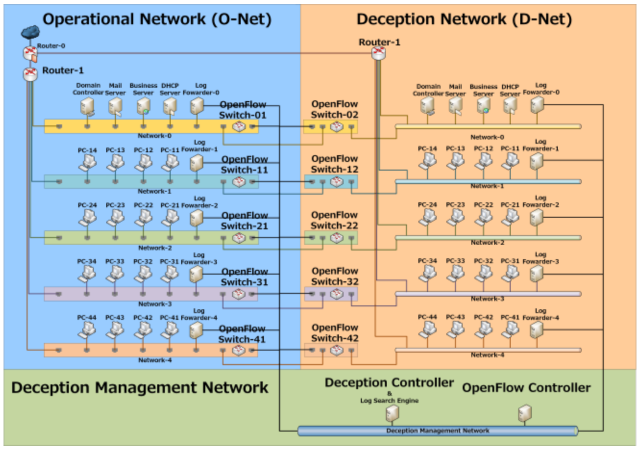


Figure 2. O-Net, D-Net, and Deception Management Network

Figure 2 shows an example network configuration we use to illustrate our architecture in this paper. The two primary components of our architecture are an Operational Network (O-Net) and a Deception Network (D-Net) configured to be identical to the O-Net. For our research, we typically use a physical O-Net and a virtual D-Net that uses a single physical server. However, our proposed technique would be extendable to any combination of a physical/virtual/hybrid O-Net and a physical/virtual/hybrid D-Net. To control the operation, there is also a Deception Management Network.

To avoid alerting the adversary that the attack has been transferred from the O-Net to the D-Net, each endpoint on the D-Net has the same IP address as the corresponding endpoint on the O-Net.

The O-Net has an OpenFlow switch as an access switch for each subnet. The O-Net connects to the corresponding subnet on the D-Net through the OpenFlow switch and the corresponding OpenFlow switch on the D-Net. Initially, the flow tables of the two OpenFlow switches are set to block any communication between the O-Net and the D-net.

4.2. Attack Transfer Mechanism

The deception operation of the attack transfer is performed by the Deception Controller on the Deception Management Network, by controlling the OpenFlow Controller, which, in turn, controls the OpenFlow Switches. The Attack Transferring Mechanism uses the steps shown in Figure 3 to facilitate a safe and covert attack transfer. Each step is described in detail in the following section.

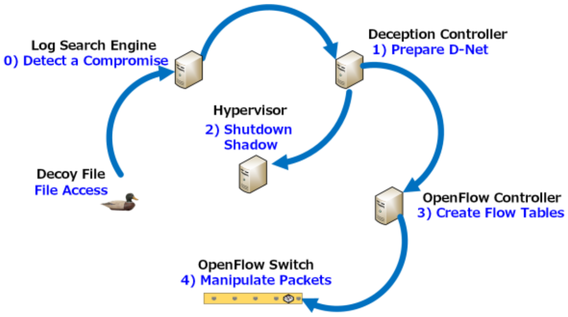


Figure 3. Attack Transfer Mechanism

0) **Detect a Compromise**. [This step is out of scope of this paper] As an example, the Log Search Engine detects a compromise on the O-Net through an adversary's access of a decoy file. The Log Search Engine then sends a compromise alert to the Deception Controller.
1) **Prepare D-Net**. Upon receiving the compromise alert, the Deception Controller creates and executes a script to (Step 2) instruct the hypervisor to turn off the shadow (corresponding endpoint on the D-Net) of the compromised PC, and (Step 3) instruct the OpenFlow Controller using the REST API to set the flow tables to the OpenFlow Switches to transfer the

communications of the compromised PC to the D-Net.

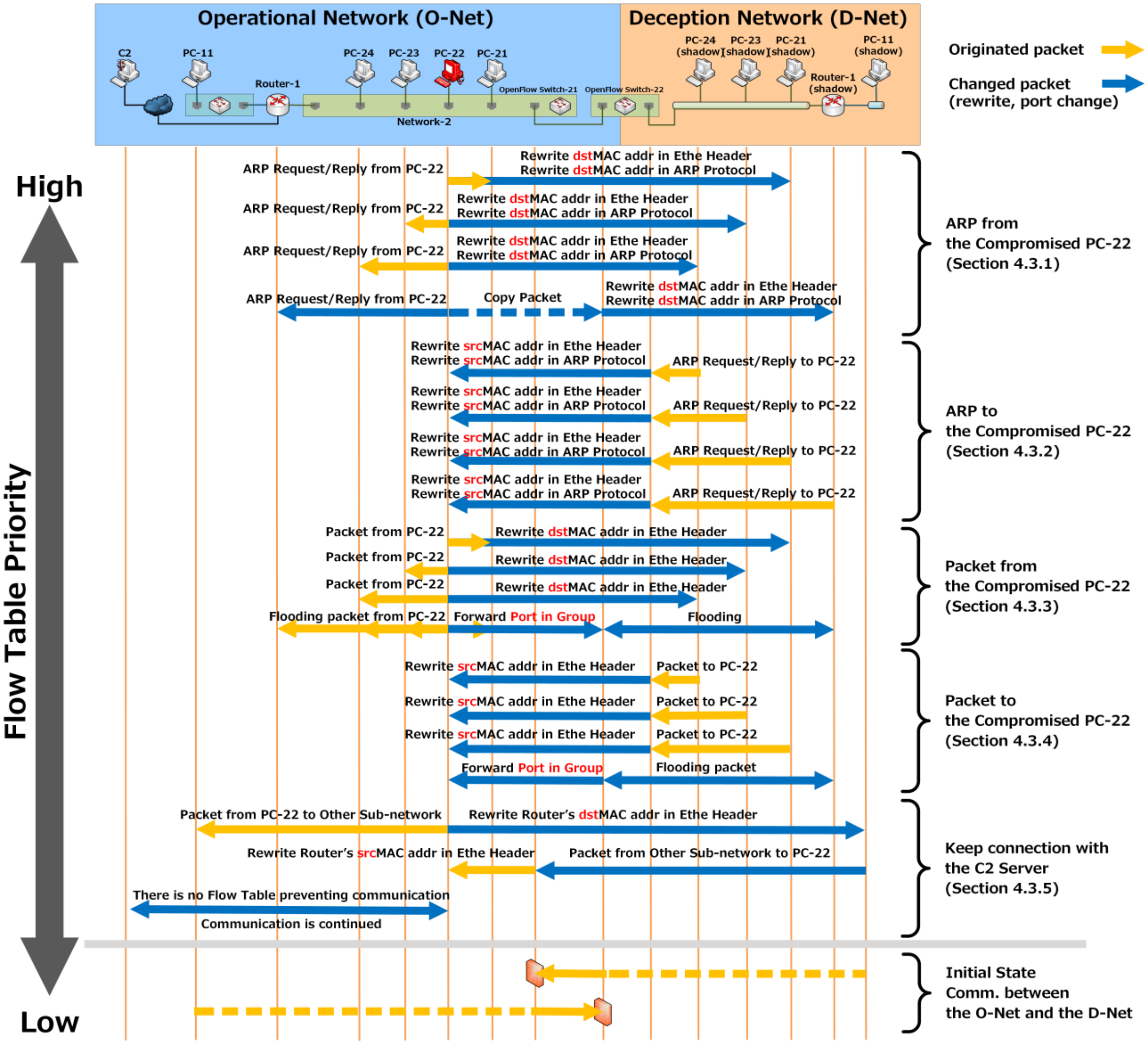
2) **Shutdown Shadow**. When instructed by the Deception Controller, the hypervisor turns off the shadow of the compromised PC on the D-Net. This step is necessary because the compromised PC transferred to the D-Net, not its shadow, interacts with other endpoints on the D-Net.
3) **Create Flow Tables**. When instructed by the Deception Controller, the OpenFlow Controller sets the flow tables customized for each OpenFlow Switch to match, rewrite, and change the output ports of, packets.
4) **Manipulate Packets**. The OpenFlow Switches stores the flow tables provided by the OpenFlow Controller and starts matching, rewriting, and changing the output ports of, packets accordingly.

After the completion of these steps, the OpenFlow switches work in coordination to transfer the communications between the compromised PC and the O-Net to the ones between the compromised PC and the D-Net while the session between the compromised PC and the C2 server is maintained.

The following section describes the packet manipulation by the flow tables used to achieve this.

4.3. Packet Manipulation by the Flow Tables

We implement our sophisticated and unique packet rewriting strategy #3 (Table 1) using the flow tables.



Operational Network (O-Net)
Deception Network (D-Net)
C2
PC-11
Router-1
PC-24
PC-23
PC-22
PC-21
OpenFlow Switch-21
OpenFlow Switch-22
Network-2
PC-24 (shadow)
PC-23 (shadow)
PC-21 (shadow)
Router-1 (shadow)
PC-11 (shadow)
Originated packet
Changed packet (rewrite, port change)
High
Flow Table Priority
Low
ARP Request/Reply from PC-22
Rewrite dstMAC addr in Ethe Header
Rewrite dstMAC addr in ARP Protocol
ARP Request/Reply from PC-22
Rewrite dstMAC addr in Ethe Header
Rewrite dstMAC addr in ARP Protocol
ARP Request/Reply from PC-22
Rewrite dstMAC addr in Ethe Header
Rewrite dstMAC addr in ARP Protocol
ARP Request/Reply from PC-22
Copy Packet
Rewrite dstMAC addr in Ethe Header
Rewrite dstMAC addr in ARP Protocol
ARP from the Compromised PC-22 (Section 4.3.1)
Rewrite srcMAC addr in Ethe Header
Rewrite srcMAC addr in ARP Protocol
ARP Request/Reply to PC-22
Rewrite srcMAC addr in Ethe Header
Rewrite srcMAC addr in ARP Protocol
ARP Request/Reply to PC-22
Rewrite srcMAC addr in Ethe Header
Rewrite srcMAC addr in ARP Protocol
ARP Request/Reply to PC-22
Rewrite srcMAC addr in Ethe Header
Rewrite srcMAC addr in ARP Protocol
ARP Request/Reply to PC-22
ARP to the Compromised PC-22 (Section 4.3.2)
Packet from PC-22
Rewrite dstMAC addr in Ethe Header
Packet from PC-22
Rewrite dstMAC addr in Ethe Header
Packet from PC-22
Rewrite dstMAC addr in Ethe Header
Flooding packet from PC-22
Forward Port in Group
Flooding
Packet from the Compromised PC-22 (Section 4.3.3)
Rewrite srcMAC addr in Ethe Header
Packet to PC-22
Rewrite srcMAC addr in Ethe Header
Packet to PC-22
Rewrite srcMAC addr in Ethe Header
Packet to PC-22
Forward Port in Group
Flooding packet
Packet to the Compromised PC-22 (Section 4.3.4)
Packet from PC-22 to Other Sub-network
Rewrite Router's dstMAC addr in Ethe Header
Rewrite Router's srcMAC addr in Ethe Header
Packet from Other Sub-network to PC-22
There is no Flow Table preventing communication
Communication is continued
Keep connection with the C2 Server (Section 4.3.5)
Initial State Comm. between the O-Net and the D-Net

Figure 4. Packet Manipulation by the Flow Tables

As a reminder, strategy #3 is “match packet MAC address and ARP packets, then rewrite its MAC information” and “match packet network IP address, then switch port accordingly” combined. There are three separate packet manipulations in the strategy #3.

A) Match ARP packets, then rewrite their MAC information and change output ports
B) Match packets by their MAC addresses, then rewrite the packet MAC addresses and change output ports
C) Match packets by their network IP addresses, then switch output ports accordingly

As mentioned in the Introduction, manipulation (A) is necessary. ARP packets need to be rewritten to ensure that TCP communications between the compromised PC and the endpoint on the D-Net are established. This is essential as ARP is used to associate the MAC address with the IP address. When an endpoint X on the D-Net attempts to send a packet to the IP address of the compromised PC, X uses an ARP request packet to determine the MAC address for the compromised PC’s IP address. The ARP request packet reaches the compromised PC, but the ARP response packet will be sent to the endpoint on the O-Net corresponding to X. Therefore, X will never find the MAC address of the compromised PC, failing to communicate to the compromised PC. When the ARP packets, including the ARP response packets from the compromised PC, are rewritten as in (A), X can determine the MAC address of the compromised PC.

With manipulation (B) along with (A), packets to/from the compromised PC are sent to/from the endpoints on the D-Net, thus the communications of the compromised PC are successfully transferred from the O-Net to the D-Net.

However, to ensure that communications of the compromised PC beyond the subnet of the O-Net and the corresponding D-Net subnet work correctly, manipulation (C) is necessary. Without (C), the communication between the compromised PC and the C2 server, hosted outside the network and the communication between the compromised PC and other subnets could not be maintained simultaneously. This is due to the fact that the router for the communication of the compromised PC to/from the C2 server and the router for the communications of the compromised PC to/from the endpoints on the other subnets on the D-Net are different. With (C), the packet is sent out from the appropriate port depending on the network IP address of the packet.

Figure 4 shows in detail how the mechanism (deception architecture) works in the following subsections. At the top of figure 4 are the subnets (Network-2 of the O-Net and D-Net), PC-21, PC-22, PC-23, and PC-24 on the subnet Network-2 of the O-Net. Router-1, and PC-11 on the subnet Network-1 of the O-Net. PC-22 is the compromised PC. There is a C2 server operated by the adversary somewhere on the Internet. The OpenFlow Switch-21 is the access switch of the subnet Network-2. An endpoint on the D-Net has the same name as the corresponding endpoint on the O-Net. (We use ’ in this paragraph as a substitute for “shadow” to represent the D-Net version of an O-Net endpoint.) Therefore, there are PC-21’, PC-23’, and PC-24’ on subnet Network-2’ of the D-Net. Router-1’, and PC-11’ on the subnet Network-1’ of the D-Net. The OpenFlow Switch-22 bridges the subnet Network-2 and the subnet Network-2’. PC-22’, the shadow of the compromised PC-22, has already been removed by the Deception Controller and the hypervisor before this transfer process is initiated.

4.3.1 ARP from the compromised PC

To make the compromised PC-22 logically belong to the D-Net, ARP request and ARP reply packets from the compromised PC are rewritten. The destination MAC address in the Ether header and the destination MAC address (Target Hardware Address) of the ARP are rewritten from the MAC address of the endpoint on the O-Net to the MAC address of its shadow (the corresponding endpoint) on the D-Net. The output port for the packet is changed to the one for the OpenFlow Switch-22.

4.3.2 ARP to the compromised PC

Corresponding to Section 4.3.1, we also need to control the ARP packets sent from the D-Net's endpoint as if the compromised PC is on the same subnet. The source MAC address in the Ether header and the source MAC address (Sender Hardware

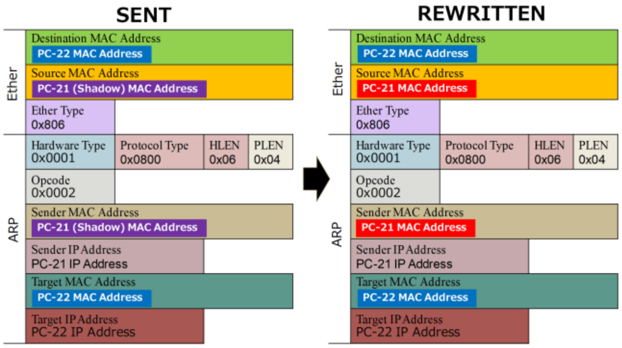


Figure 5. Rewriting an ARP Reply from PC-21 (shadow) to the compromised PC-22

Address) of the ARP (see Figure 5) addressed to the compromised PC-22 from an endpoint on the D-Net are rewritten to the MAC address of the corresponding endpoint on the O-Net. The output port for the packet is changed to the one for the OpenFlow Switch-21.

4.3.3 Packet from the compromised PC

The ARP tables of the endpoints on the O-Net and D-Net are bridged coherently through the activity described in Section 4.3.1. To send a packet other than ARP, the destination MAC address in the Ether header of the packet is rewritten to the MAC address of the shadow on the D-Net. The output port for the packet is changed to the OpenFlow Switch-22. Since the priority of the flow tables for this operation is lower than the flow tables for rewriting the ARP packet, this applies to packets other than ARP.

Broadcast packets from the compromised PC-22 must be sent to the endpoints on the D-Net instead of those on the O-Net. To achieve this, we use the OpenFlow Group function. Broadcast packets from PC-22 are sent to the group that combines the connection port of PC-22 and the output port of OpenFlow Switch-22. With this configuration, broadcast packets are sent only to the endpoints on the D-Net.

4.3.4 Packet to the compromised PC

As discussed in Section 4.3.3, packets from the endpoints on the D-Net PC need to be rewritten so that the compromised PC appears to be on the same subnet. The source MAC address in the Ether header addressed to the compromised PC-22 from the endpoint on the D-Net is rewritten to the MAC address of the corresponding endpoint on the O-Net. The output port for the packet is changed to the one for OpenFlow Switch-21. In addition, we need to forward broadcast packets from the endpoint on the same subnet of the D-Net to the compromised PC-22. By sending this packet to the group described in Section 4.3.3, these broadcast packets reach only PC-22 on the O-Net and no other endpoints on the O-Net.

4.3.5 Maintaining the connection with C2 server

To maintain the connection between the compromised PC and the C2 server out on the Internet, the packets going out to the C2 server (and other endpoints outside of the corporate network) need to be handled differently from the packets going to the other subnets within the corporate network. For this purpose, the destination MAC address of the packet from the compromised PC-22 addressed to another subnet within the corporate network is rewritten to the MAC address of the Router-1 (shadow) on the D-Net and the output port is changed to the one for OpenFlow Switch-22. The source MAC address of Router-1 (shadow) in the Ether header of the packets form another D-Net subnet is rewritten to the MAC address of Router-1 of the O-Net and the output port is changed to the one for OpenFlow Switch-21.

No changes are required for the packets between the compromised PC and the C2 server and all communication between them continue uninterrupted through Router-1.

5. Evaluation

5.1. Strategies #1 and #2 Tests

We confirmed the network behaviors of strategies #1 and #2 in Table 1. For strategy #1, we used a simple setup of three PCs connected to an OpenFlow switch in a virtualized environment. The TCP communication attempts between the compromised PC and the endpoint on the D-Net were not successfully established due to the lack of ARP packet rewriting.

For strategy #2, we used a smaller version of the environment as described in Subsection 5.2. Communications between the compromised PC and the endpoints on the D-Net were observed to be working properly. Even though communications between the compromised PC and the C2 server out on the Internet were maintained, the communications to the endpoints on other subnets of the O-Net failed to be transferred to the corresponding subnets of the D-Net. Instead, the latter communication was sent to the subnets of the O-Net. This is not a desirable result and introduces additional risk to the O-Net.

5.2. Test Methodology

To test our solution (strategy #3), two servers were connected by a switch. One server provides a virtualized environment for the O-Net (including OpenFlow Switches) and the C2 server on the Internet, and another server provides a virtualized environment for the D-Net (including OpenFlow Switches) and the Deception Management Network. Both the O-Net and the D-Net have six subnets and 31 endpoints as shown in Figure 6.

For purposes of evaluation, attacks were conducted on the compromised PC from the C2 server. We

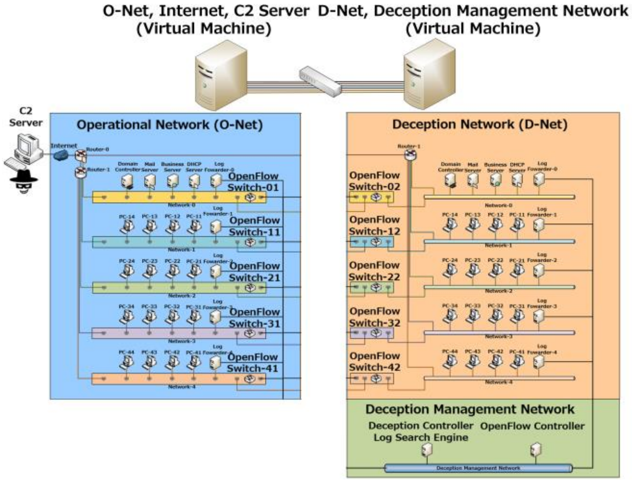


Figure 6. Evaluation Environment

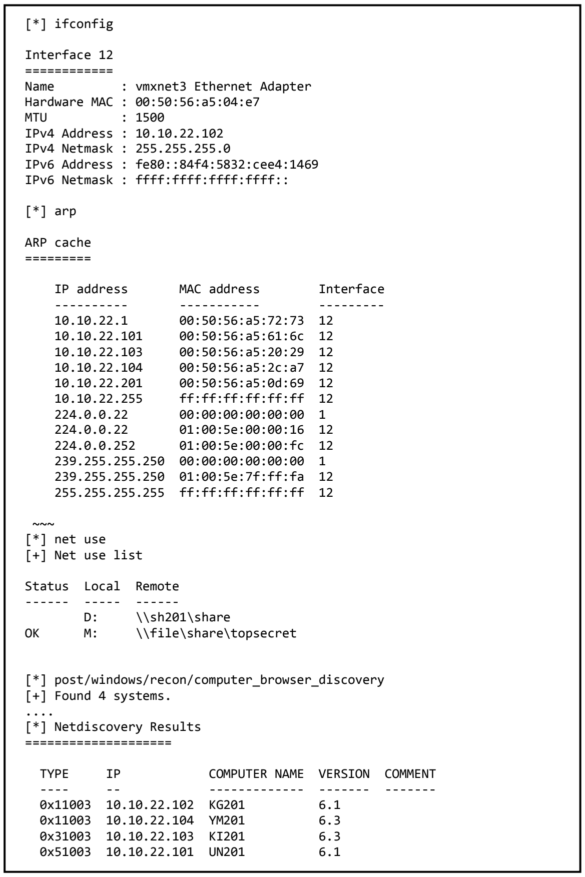
observed from the adversary's point of view (i.e. from the C2 server) to try to identify any differences detected before and after the attack transfer from the O-Net to the D-Net. In addition, we watched other negative triggers (like session termination) during the transfer.

We automated the following post-compromise activities selected from common attacks and let the sequence run on the compromised PC from the C2 server's console before and after the transfer. The script consists primarily of network-related commands.

1) sysinfo: Get system and user information
2) idletime: Get the time since user's last operation
3) getpid: Get the process ID of the current attack process
4) ps: List current processes
5) post/windows/gather/enum_services: Get names and configuration info of services
6) getproxy: Get information about proxy
7) post/windows/gather/credentials/enum_cred_store: Get credentials of IE
8) post/multi/gather/thunderbird_creds: Get credentials of Thunderbird
9) ls -lR C:\\Users\\: List the users
10) post/windows/gather/enum_ie: Get IE browser history
11) post/windows/gather/dumplinks2: List files recently used
12) post/windows/gather/enum_applications: List installed applications
13) net user / net localgroup: Get information about local accounts, groups and administrators
14) net share: Get shared folder information
15) show_mount: List currently attached mounts
16) ifconfig: Get network interface information (MAC address, IP address)
17) arp: Get the ARP table
18) route: Get the routing table
19) netstat: Get connection information
20) net use: Get SMB remote connection information
21) post/windows/recon/computer_browser_discovery: List neighbor computers

5.3. Evaluation result

From the adversarial perspective (both on the compromised host and on the C2 server) the automated attack script in Section 5.2 produced identical results before and after the transfer from the O-Net to the D-Net. The session between the compromised PC and the C2 server remained uninterrupted during and after the transfer. In addition, no negative triggers were observed. Figure 7 shows the shortened output of the script.



```
[*] ifconfig

Interface 12
============
Name         : vmxnet3 Ethernet Adapter
Hardware MAC : 00:50:56:a5:04:e7
MTU          : 1500
IPv4 Address : 10.10.22.102
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::84f4:5832:cee4:1469
IPv6 Netmask : ffff:ffff:ffff:ffff::

[*] arp

ARP cache
=========

    IP address       MAC address        Interface
    ----------       -----------        ---------
    10.10.22.1       00:50:56:a5:72:73  12
    10.10.22.101     00:50:56:a5:61:6c  12
    10.10.22.103     00:50:56:a5:20:29  12
    10.10.22.104     00:50:56:a5:2c:a7  12
    10.10.22.201     00:50:56:a5:0d:69  12
    10.10.22.255     ff:ff:ff:ff:ff:ff  12
    224.0.0.22       00:00:00:00:00:00  1
    224.0.0.22       01:00:5e:00:00:16  12
    224.0.0.252      01:00:5e:00:00:fc  12
    239.255.255.250  00:00:00:00:00:00  1
    239.255.255.250  01:00:5e:7f:ff:fa  12
    255.255.255.255  ff:ff:ff:ff:ff:ff  12

 ~~~
[*] net use
[+] Net use list

Status  Local  Remote
------  -----  ------
        D:     \\sh201\share
OK      M:     \\file\share\topsecret


[*] post/windows/recon/computer_browser_discovery
[+] Found 4 systems.
....
[*] Netdiscovery Results
====================

  TYPE     IP            COMPUTER NAME  VERSION  COMMENT
  ----     --            -------------  -------  -------
  0x11003  10.10.22.102  KG201          6.1
  0x11003  10.10.22.104  YM201          6.3
  0x31003  10.10.22.103  KI201          6.3
  0x51003  10.10.22.101  UN201          6.1
```

Figure 7. Logs collected by adversary

6. Conclusion

Our objective is to contain and observe an APT-like attack safely and covertly so that we can monitor adversarial behavior in real time to understand their TTPs, purposes, and intentions. To achieve this objective, we propose creating and deploying a Deception Network (D-Net) with the same network topology and endpoints, using the same IP addresses of the corresponding endpoints on the Operational Network (O-Net). When we detect a compromise on the O-Net, we transfer the communications between the compromised PC and the O-Net endpoints to the ones between the compromised PC and the D-Net while keeping the communications between the compromised PC and the C2 server intact. That is achieved by using OpenFlow's flow tables for matching and rewriting packets. We have confirmed that we can contain the compromised PC without the adversary observing any difference before and after the cyber deception and that the session remained intact during and after the transfer.

The focus of the paper, the attack transferring mechanism through SDN is important but is still just one piece of the whole Cyber Deception puzzle and it needs to be incorporated into the entire cyber deception operation. We have combined the mechanism with carefully crafted honey tokens on O-Net and intelligence gathering in D-Net so that as soon as the attacker touches a honey token, it triggers the attack-transferring mechanism automatically, leading to endpoint and network intelligence gathering on D-Net. We tested this system in cyber war games and it worked seamlessly and successfully deceived the red team for many hours until the game ended.

For future research and analysis, we will continue our empirical evaluations of the technology and work to develop scientific and objective evaluation methods, to continue to refine the technology. For technology refinement, we plan to implement and evaluate our cyber deception architecture for IPv6 using our already implemented Northbound API for matching and rewriting the Neighbor Discovery Protocol (NDP) of IPv6, the ARP equivalent of IPv4.

Potential flaws revealing the deception include network latency changes and server content continuity before and after the switch from the O-Net to the D-Net. In follow-on research, we will test network latency changes in various real and virtualized configurations. In more and more virtualized operational networks, however, network latency may not provide significant clues for the adversary to determine if he or she is in a deceptive environment or not. Server content continuity is an important consideration in balancing maintenance and other costs, and realism of the deceptive environment and would require innovative solutions, which is a focus on our follow-on research.

7. References

[1] FireEye, “M-Trends 2018”, Available at https://www.fireeye.com/blog/threat-research/2018/04/m-trends-2018.html

[2] E. M. Hutchins, M. J. Cloppert and R. M. Amin, “Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains”, Leading Issues in Information Warfare & Security Research, vol. 1, pp. 80, 2011.

[3] Kristin E. Heckman, et. al., “Cyber Denial, Deception and Counter Deception: A Framework for Supporting Active Cyber Defense,” Springer, 2015.

[4] Cliff Wang and Zhou Lu, "Cyber Deception: Overview and the Road Ahead," pp. 80-83, IEEE Security & Privacy, March/April 2018.

[5] Open vSwitch, “Open vSwitch: Production quality, multilayer open virtual switch.” Available at: http://www.openvswitch.org/

[6] Nippon Telegraph and Telephone Corporation, “Ryu Network Operating System”, Available at: http://osrg.github.com/ryu

[7] Open Networking Foundation, “OpenFlow Switch Specification Version, 1.3.0,” June 25, 2012, Available at: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf

[8] Cuckoo Sandbox, Available at: http://www.cuckoosandbox.org/

[9] Dilshan Keragala, “Detecting Malware and Sandbox Evasion Techniques”, SANS Institute InfoSec Reading Room Detecting Malware and Sandbox Evasion Techniques, The SANS Institute, 2016.

[10] Brian Scottberg, William Yurcik, and David Doss, “Internet Honeypots: Protection or Entrapment?”, Proceedings of International Symposium on Technology and Society, August 2002, pp. 387-391.

[11] Vincent E. Urias, William M.S. Stout, “Computer Network Deception as a Moving Target Defense”, 2015 International Carnahan Conference on Security Technology (ICCST), September 2015.

[12] "Deceiving Network Reconnaissance Using SDN-Based Virtual Topologies", IEEE Transactions on Network and Service Management, Vol. 14, No. 4, December 2017, pp. 1098-1112.

[13] Steven M. Silva, et. al., *US Patent No. US9021092B2*, 2015.

[14] Chad O. Hughes, et. al., *US Patent No. US8978102B2*, 2015.