Defensive Cyber Operations in a Software-Defined Network

Thomas Parker, Jeremy Jones, Joshua Mayberry, Gassium Chanman, Zachary Staples, Murali Tummala, John McEachen

Department of Electrical and Computer Engineering
Naval Postgraduate School, Monterey, CA
{tcparke1, jnjones, jcmayber, pchanman, zhstaple, mtummala, mceachen}@nps.edu

Abstract

Software-defined networks (SDNs) offer network defenders the opportunity to choose from a variety of protection techniques in response to different threats. In contrast, traditional network architectures often lack the flexibility to implement threat-specific security controls. This research was conducted on a hardware SDN test bed running custom security applications to demonstrate techniques that support a network administrator’s requirement to observe, orient, decide, and act upon suspicious activities in the network. Additionally, SDN application development was used to teach undergraduate and graduate students offensive and defensive cyber techniques and tactics.

1. Introduction

Software-defined networks (SDNs) are being adopted by industry because they increase performance and reduce cost. For example, SDNs have been implemented in Google and Facebook data centers because they have shown performance increases reaching upwards of 95% link utilization [1]. SDN also reduces cost because the complexity of the routing hardware has been removed. For example, control algorithms that were once built into application-specific integrated circuits (ASICs) in hardware across the network have been moved to the centralized controller [2]. These algorithms are implemented in software, which further reduces cost and allows for rapid innovation to improve performance or close security gaps. Centralized control makes SDNs a cost effective choice because it also provides network defenders with a richer set of data to analyze, and a dynamic tool to defend the network. However, more data and a rapid network configuration tool do not by default result in better security.

Better security is the result of a comprehensive strategy built on a proven conceptual framework for victory in a competitive domain. The observe, orient, decide and act (OODA) loop, as shown in Fig. 1, is a foundational construct for comparing the cycle of two adversaries engaged in a competitive endeavor where one competitor will emerge victorious and the other will fail. Originally pioneered by a fighter pilot in the United States Air Force, Colonel John Boyd, it is now a common conceptual framework in military operations; it is built around the idea that two competing entities must collect information about the situation, synthesize that information into understanding, then make a decision, and act faster than an adversary in implementing their course of action [3]. It has been shown that the competitor who can more quickly and effectively proceed through this sequence will outperform a slower, less efficient adversary [4].

In Defensive Cyber Operations (DCO), we must be able to detect an intrusion, orient to the potential risk posed by the intrusion, decide how to respond, and then quickly implement that decision before the intruder can complete their intended action. The primary objective of the DCO operator is to ensure the security of the network and the information stored and carried on that network. In other words, the goal is to break the cyber kill chain as defined in [17]. Secondarily, the DCO operator will collect intelligence to better understand the adversary. To achieve these goals, the defender must close his OODA loop faster than the adversary. We advocate the use of SDNs in DCO because the information and network control options available to the SDN controller allow a defender to move quickly and effectively through each step of the OODA loop [4] [19].

Figure 1. The defender’s and the adversary’s OODA loops are connected because there is a single network that is being observed and acted upon. The orient
and decide phases are independent for each [3].

In this paper, we describe the architecture of a SDN, and we describe the applications and framework of the SDN controller. Later in the paper we present five sample applications that are demonstrations of DCO in a SDN. The paper is organized as follows. The importance of setting the tempo of operations in DCO is detailed in section 2. In section 3, we describe software-defined networks and the software architecture. The applications designed for the SDN and their uses are described in Section 4. Our approach to teaching cyber operations with the use of a SDN is described in Section 5. Our conclusion and planned future work are reviewed in Section 6.

1. Controlling the Tempo of Operations

The first step in outperforming an adversary is to control the tempo of observations within our own network. As depicted in Fig. 1, the adversary and the defender are both observing and acting within the same physical and logical space. Each may have different tools to observe and act, but we must recognize that the adversary must act within the defenders’ physical and logical boundaries, which allows the defender to set the speed at which observations are made and actions are executed. This could provide an advantage to the defender if he can exploit this advantage.

To gain the advantage, we define two observation speeds in our network with the intent that we proactively set the desired speed required to observe network activity in a multiple hypothesis situation. For example, an anomalous outflow of data from the network could, in one hypothesis, be an indication of malicious data exfiltration, or in a countering hypothesis, it could be a legitimate transfer of information to a colleague or customer. It is customary to build network security devices to work at the full speed of the interface it is deployed upon. The novelty of our approach to network observation is that we recommend throttling specific anomalous flows to tilt the advantage toward network defenders.

Therefore, our model includes two nominal categories of network flow speed. The first flow control is colloquially called network speed and refers to an interface sending and receiving data at the maximum speed possible. Observation at this speed is achieved by designing autonomy into the network controller, which allowed it to conduct sophisticated network analysis at line speed and then notify network defenders of decisions through a set of monitoring criteria [5]. Network speed is the criteria most appliances have been designed to meet in both traditional networks and SDNs.

Implementation of state-of-the-art research in deep packet inspection and other data and traffic analysis techniques will improve the capability for network controllers to autonomously reason against rich sets of network data. However, for every advance in network defense it is prudent to assume a new counterattack will be developed, and that this continuous cycle of attack-defend-counterattack will always present DCO professionals with ambiguous situations in which their network autonomy is indecisive.

Therefore, we present a second class of flow control called human speed. This is a deliberate attempt to overcome existing limitations in autonomy. In cases where there is ambiguity in the autonomous capability of the controller to classify network activity, a decision by a human must be made. All too often nefarious activity has progressed on a network and human analysis of the indicators of compromise was not conducted until the post-incident response. We demonstrate a technique whereby a SDN, with properly configured security applications, can implement connection specific flow control and automatically increase the data collection against a questionable flow while simultaneously notifying DCO professionals to the potential risk. By throttling the connection to a trickle of bits the potential adversary’s network connection is kept alive and can be analyzed in real time thereby tilting an advantage to the defenders during the observation phase of the OODA loop.

When autonomous decisions are clear, the centralized controller is able to take defensive measures at network speeds and is able to close the OODA loop faster than a human can. However, the human DCO professional’s ability to deal with new situations is preserved because the network controller can slow the network’s flow speed to allow a human to decide if an activity is or is not malicious.

2. SDN Controller

A SDN is a network that has a centralized controller managing network traffic [6]. Unlike a traditional network, a centralized controller manages all traffic in the network. Each SDN switch in the network receives rules to instruct them how to behave. Each rule has a match field and an action field [7]. No traffic is allowed to move in the network unless the controller has installed a rule that allows that traffic. Once the flow rule is installed, all traffic
associated with that flow will be acted upon as defined by the rule’s action. These matches and actions can be exceptionally granular to control specific types of traffic or specific devices or they can be applied more generally to account for several types of traffic or groups of addresses.

2.1 SDN Heirarchy

The architecture of our implementation is shown in Fig. 2. It is based on the generic hierarchy of SDNs where the network hosts are connected to SDN switches, which are in turn connected to and controlled by a SDN controller. The devices in our network ranged from voice over internet protocol (VoIP) phones to standard desktop personal computers. The switches are connected to the SDN controller, which includes both the Ryu framework and our applications [8]. The Ryu framework is the network operating system (OS) that we have selected for our test bed [9]. The network OS is software that abstracts the switching hardware from the applications. The applications make requests of Ryu to issue commands and receive data from the switches via the OpenFlow protocol [7].

There are two keys to DCO in SDNs. First, the SDN controller is in complete control of all traffic in the network, which allows it to rapidly implement actions at any location in the network [19]. Second, the controller is able to monitor the network in detail [19]. SDN switches keep counters for the number of packets and bytes through each physical port as well as how many packets and bytes have matched a particular flow [7].

Using these monitoring features and crafting detailed matches in the rule set, the controller is able to monitor traffic by application, internet protocol (IP) address, or media access control (MAC) address or some combination of these and additional fields as detailed in the OpenFlow protocol [7]. SDNs have network sensors built into the architecture and protocol; in contrast to traditional networks, there is no need to add additional hardware or software network appliances to gather this information.

These monitoring functions allow SDN applications to observe the network in unprecedented detail. Based on these observations, the applications can then orient the controller to the current state of the network. In other words, the detailed analysis of traffic patterns, identification of hosts on the network and the topology as a whole describe the state of the network. Analysis of all of that data is centralized at one location within a SDN to conduct DCO.

2.2 Control and Monitoring Applications

We chose to implement five DCO applications. These five support one or more phases of the OODA loop. The DCO apps included in this research are:

- **Fingerprint** - a host identification application, using active and passive techniques [10] [11] [19]
- **Throttle** - an application for implementing human speed traffic flow limiting [12]
- **Redirect** - an application to redirect browser traffic with DNS poisoning [11] [19]
- **Black Hole** - application for blocking all traffic from a host [12] [19]
- **Poison** - an application to execute data corruption on outbound flows [13]

Fingerprint was used to determine which hosts were on the network, where they were located, and what type of operating system they were running. Next, Throttle was developed to demonstrate the ability to adapt the orientation speed of the network to its human operators. Finally, Redirect, Black Hole and Poison were developed to represent the broad variety of response tools network administrators could use to prevent malicious behavior in their network.

3. SDN Applications

The SDN controller requires custom applications to implement advanced DCO as described in this paper. There are no native security analysis tools in the Ryu OS [9]. The applications built and tested for this paper do not represent the full spectrum of capabilities desired in a production network, but rather they were developed to demonstrate the versatility of SDNs in retaking the advantage at multiple points within the OODA loop.
3.1 Fingerprint Application – Observe Network Users

Ultimately, we are more concerned about the human beings using our networks than we are about the devices they use to access the network. However, the cumulative knowledge of the device MAC address, where it physically connects to the network and other data allow us to build a unique fingerprint of devices that can be cross-correlated to the actual humans who need to use the network for its intended purpose. For a secure network, the network defender must be able to identify all of the network devices [19].

For example, let’s assume that an authorized network user, Bob, had his username and password stolen through a successful spear phishing campaign, and a network attacker subsequently attempted to log into the network using that stolen information. The traditional login system on the network would let the attacker into the network based on Bob’s stolen credentials, but a fully fingerprinted controller could detect the access anomaly in many ways.

In this extended example, assume that Bob logs into the network from his desktop in the office. Unless the attacker replicates the MAC address, clones the client OS, and logs in from the same network port then our fingerprint will detect the anomaly. In this example, the attacker would have to defeat our physical security in addition to the stealing Bob’s credentials. Therefore, the Fingerprint application gives us a much better observation of our network.

Fingerprint is implemented by detecting dynamic host configuration protocol (DHCP) traffic and creates a fingerprint database [10]. This application is a passive tool that the network operator could use to monitor the hosts on the network without exposing that this mapping and fingerprinting is occurring. This application passively listens to all DHCP traffic on the network. Since nearly every device will send DHCP traffic to connect to the network, the controller is made aware of all devices that request dynamic IP addresses. For those that do not request a dynamic IP address, an active scan of the network is conducted periodically to find hosts that are using static IP addresses. A fingerprint is obtained by analyzing the fields of the DHCP packets and matching those fields to known differences between operating systems [10]. Different devices such as phones, laptops, and printers will request different options and will send different information based on the needs of the device. For instance, Polycom VoIP phones include option 66 in their DHCP request because it is expecting the IP address of the boot server to use [14]. This requirement is unique to VoIP phones.

Fingerbank.org contains a database that contains much of the information required by Fingerprint to identify a host’s operating system. For the purposes of our research, this database was converted to an .xml file. The fingerprint is based on the device’s MAC address, IP address, switch location, port number, host name, and OS. Certain devices will provide more detailed information such as the vendor part number. This information will be included in the fingerprint if it is available. As the DHCP packets are analyzed by the application, it saves the fingerprints of each device for later comparison to see if the fingerprint has changed. In addition, this information is made available to the DCO applications. Table 1 shows the information that is saved in each fingerprint and two example fingerprints.

### Table 1. Example Fingerprints

<table>
<thead>
<tr>
<th>Device</th>
<th>Hostname</th>
<th>IP</th>
<th>MAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>VoIP Phone</td>
<td>N/A</td>
<td>10.10.20.17</td>
<td>00-04-F2-15-00-3C</td>
</tr>
<tr>
<td>Raspberry Pi</td>
<td>raspberry</td>
<td>10.10.13.13</td>
<td>88.77.06.04.31</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Device</th>
<th>Switch</th>
<th>Physical Port</th>
<th>Option 60</th>
<th>OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>VoIP Phone</td>
<td>9</td>
<td>33</td>
<td>66</td>
<td>Polycom SoundPointIP SoundPointIP SoundPointIP SIP: 301.2345-11900.010A SIP: 21.1.01170 6.9.10.02433 BR: 4.1 20037/11-Aug-08 16:49</td>
</tr>
<tr>
<td>Raspberry Pi</td>
<td>1</td>
<td>5</td>
<td>None</td>
<td>Debian</td>
</tr>
</tbody>
</table>

In some cases, devices may not request an IP address via DHCP, which requires the controller to conduct an active scan of the network IP space. This capability was included in the Fingerprint application. The controller crafts packets and transmits those into the network to actively scan the IP space. To gather additional information, the Fingerprint application can also conduct a port scan of discovered devices. When the controller transmits, it selects a random source IP and MAC address to aid in anonymizing the true source of the active scan.

Fingerprint data is provided to the Controller to use in its decision-making. A variety of DCO actions, some autonomous and some implemented by network security professionals, can be taken in response to anomalous fingerprints. Our Fingerprint application is an incomplete solution to validating authorized users, but a good example to demonstrate how an SDN allows a strategic approach to developing custom solutions for the observation phase of the OODA loop.
3.2 Throttle Application - Orient and Decide at the Correct Speed

Actions within a network take place at a volume and speed that exceeds a human’s capacity to process the inputs and make decisions. Over one-hundred years ago, we abandoned human telephone operators for automated switching and the pace of connecting and switching network traffic has been increasing ever since. Traditionally, if a network user is properly connected and authenticated then the network provides service at the maximum speed possible, consistent with quality of service standards to balance priority traffic and priority users across the network. We are advocating that DCO professionals reconsider the notion of providing the best possible service in order to slow anomalous network events down to human speed. Fig. 3 shows measured data rates of the targeted device. The data rate starts at approximately 8.9 Mbps and is reduced to 10 kbps and then is returned to its previous value. This actions was completely controlled by the SDN controller without human intervention.

![Figure 3. Throttle limited the data rate of a specific device for a specified amount of time to allow human intervention.](image)

3.3 Black Hole, Redirect, and Poison Applications - Take Decisive Action

After observation, orientation and decision, an action must be taken. The foregone conclusion of the OODA loop is that competitors have valid courses of action from which to choose. Without a menu of tailored and proportionate actions, the investment in orientation, observation and decision-making would be for naught. Therefore, we implemented three DCO applications to demonstrate potential options that might be resident in a full DCO suite.

It is the intent of our work to make some actions available to the autonomous decision-making system operating in the controller and to reserve some actions for a human to authorize. However, it should be noted that the application-based architecture of the Ryu framework is agnostic to whether the controller initiates actions or a network defender targets an action at an intruder. Therefore, the actions listed here are purely representative of total control of the network exerted by the SDN controller.

3.3.1 Black Hole. When contemplating potential actions to take against an intruder, the seemingly obvious choice was to take away the user’s ability to conduct any further network activity. The Black Hole application achieves this end state. Once activated, the controller initiates a rule that any packet destined to or received from the suspect device is immediately dropped. Once initiated, the Black Hole application isolates that client and the human operating it from every other network device. Packets initiated by the client are dropped by every switch in the network. The downside to this brute force approach is that the attacker will immediately notice that his access has been taken away. In many cases, this type of obvious quarantine would be appropriate, but against an Advanced Persistent Threat (APT), network defenders may want an opportunity to implement a more nuanced action in which the APT’s tactics, techniques and procedures (TTPs) can continue to be observed and added to our understanding of the adversary [18].

3.3.2 Redirect. This application is a first attempt to move a suspected intruder into a controlled environment where defenders can study their TTPs and gain a knowledge advantage. Once implemented against a fingerprinted device, the Redirect
application sends all DNS requests from the targeted device through the controller, which resolves each request to an address in a duplicate page in a controlled VM. The process described is shown in Fig. 4. Additional monitoring tools may be loaded on the fingerprinted client, and detailed logs of activity are retained for analysis.

The intent of Redirect is to be subtle whereas Black Hole is intentionally overt. With this subtlety, an attacker may not perceive that they are identified and their malware activity can be observed. This long term observation may reveal the IP addresses of more command and control servers and could lead to intelligence related to classifying the entire campaign. For example, if appropriate sanitized resources were provided in the learning environment, it is possible to ascertain whether an intruder chooses to copy and archive credit card data or is more interested in espionage because they aggregate and archive data on personnel records and intellectual property.

Figure 4. Redirect application flow chart showing the process described above.

Our implementation of Redirect looked only at capturing and redirecting DNS, but could be expanded to include moving the suspected device into a complete virtual environment with clients and services designed to ascertain a full array of intelligence activity on the intruder.

3.3.3 Poison. Finally, we wanted a subtle way to prevent data exfiltration while still allowing defenders to learn about the threat. Our Poison application gives us this blend of protection and continued observation and orientation. Additionally, Poison demonstrates the power of the SDN controller to impact traffic flowing through its network.

Specifically, our implementation of Poison works against outbound connections using the File Transfer Protocol (FTP). In several recent high profile hacks [15] [16], FTP has been used as the final tool by the adversary to exfiltrate data off a victim’s network. Assuming that we can identify these anomalous outbound flows, we would like to protect the outbound information and yet continue to learn about the attacker’s infrastructure. Poison does this by altering the FTP packets of a fingerprinted connection by flipping bits in the payload portion of the TCP packet, correcting the checksum and sending the packet to the destination. At the destination, the checksum will be correct and the packet will be accepted. However, when the adversary attempts to decrypt the packet, the file will be useless because of the flipped bits in the payload. The intent is that the APT-style attacker will then try another command and control server thus giving us more information about their infrastructure and/or logging in to directly interact with their malware thus giving us more information about the extent of the threat network.

Like the other DCO applications, our implementation of Poison is an incomplete solution, but was chosen to demonstrate a SDN controller’s capability to act as a man-in-the-middle to change the data inside packets in addition to controlling the routing decision for those packets.

4. Teaching Cyber Operations

The test bed that was used to demonstrate these applications was built at the Naval Postgraduate School (NPS). The applications were written by undergraduate students from the United States Naval Academy (USNA) and graduate students from the NPS. The midshipmen were interns at NPS as part of their summer training cycle. This is the first time that USNA and NPS students had conducted joint research in cyber operations.

The test bed used by these students was disconnected from NPS’s network and the larger Internet. They were able to learn in an environment where they could not interfere with any devices or traffic that was not present on the test network. This allowed them to explore and learn how an SDN works without fear of breaking network policies enforced by the NPS network administrators.

SDNs are an ideal tool to help students learn networking and cyber operations. To develop the applications described above, the students first had to learn the detailed implementation of each protocol. For instance, to parse a DHCP packet and determine what information is important within the packet, the students learned in detail how devices are dynamically assigned IP addresses.

Fundamental to SDNs is the rule development by the controller. The flow rules, which are installed in the switches, have both a match field and action field.
To implement the DCO applications, the students had to develop code that automatically installed flow rules when new packets were sent to the controller. The process of learning how to route packets in a network is not a built in function of the Ryu OS. The students had to develop the application to route the packets in the network. This requires a fundamental understanding of addressing at layers 2 and 3 of the OSI model [13], and it also requires a detailed understanding of how routes are develop in a network.

Because SDNs require this level of detailed understanding to develop applications, it is an effective tool to use when teaching networking and cyber operations. Another particular benefit of the NPS test bed is that there is no virtualization of the switches or hosts. Therefore, students have access to a training network built on physical switches with live hosts. The students were able to develop the ideas, write the scripts, and test each application. In the process, they learned the nuisances and difficulties of managing physical devices.

5. Conclusions

This project demonstrated a strategic approach to Defensive Cyber Operations based on the proven conceptual framework of the OODA loop. We began by showing advantages that can be gained through SDN applications in the observe phase. We looked at how to use the SDN controller’s capability to gather a robust set of client data to develop unique fingerprints for each client on the network. The sophistication of this identification technique could be extended to include behavioral components including time and day of the week to further refine the capability to detect anomalous behavior. After detecting an anomaly, we showed a novel approach to flow control against a specific device to inject packets in the network. This requires a fundamental understanding to route packets in a network.

We demonstrated a SDN can be used as an effective means to teach networking and cyber operations. In the test bed developed for this application, students can learn the protocols that are fundamental to the Internet and how to exploit them to successfully defend a network. The deep understanding required to implement these demonstration applications is fundamental to learning how to effectively conduct cyber operations.

In closing, it is important to highlight that a successful competitor makes several loops around the OODA framework continually learning and targeting their adversary before taking final decisive action that ends the engagement. Our proposed applications, first, attempt to blunt the impact of an adversary’s progress inside our network, and then continue observing and orienting to take subsequent follow-on actions to decisively end the adversary’s cyber campaign.

6. References


