A Man-in-the-Middle Attack Against OpenDayLight SDN Controller

Michael Brooks Baijian Yang
Department of Computer and
Information Technology
Purdue University
West Lafayette, IN 47906
{*brooks23, byang*}@purdue.edu

ABSTRACT
In this paper, we investigated the vulnerabilities surrounding software-defined networking (SDN). Specifically, we examined the vulnerabilities of OpenDayLight SDN controller. Among all the possible attack vectors, a man-in-the-middle attack using ARP poisoning was successfully launched to intercept the traffic between a client and the OpenDayLight controller. Details of the experiment method, procedures and results were described in this manuscript. To the best of our knowledge, this is the first successful practical attempt to penetrate an SDN controller and be able to capture login credentials of the controller. The significance of this attack should not be taken lightly; once the SDN controller is under the control of the adversary, there will be no security at all for the entire network governed by this controller.

Categories and Subject Descriptors
D.4.6 [**Security and Protection**]: *Invasive software*
C.2.0 [**Computer-Communication Networks**]: *General: Security and Protection*

General Terms
Performance, Design, Reliability, Experimentation, Security, Verification.

Keywords
SDN, Open Daylight, Man in the middle, Security

1. INTRODUCTION
With any new technology, there will be risks. Whether that new technology thrives or not is based largely on whether its potential outweighs the consequences of those risks. In the world of computer networking, these risks can be magnified due to the nature of the environment. The networks we build cannot simply be fast or optimized in 2015, they must also be secure. As consumers begin to trust more and more sensitive data to Cloud networks, and the Internet in general, security must become one of the determining factors in whether a new technology lives or dies.

One such new technology is software-defined networking (SDN). The ideas and concepts SDN introduces to the networking world are desirable, to say the least. While SDN would not make old networking methods obsolete, it would begin to usher in a new age of networking should it become fully adopted by the IT world. As it stands, SDN looks to become an $8 billion market by the year 2018 [1]. The concerning part in this is that it is still widely accepted that there are serious security flaws in SDN [2][3][4]. Most notably, the controller in an SDN is centralized, which works well for SDN, but also provides a central attack vector. If the controller is compromised in an SDN, it is likely that the entire network will be compromised along with it. This is simply the nature of SDN, and no solution has been set forward yet at this time.

It is not the intention of this work to solve the security flaws of SDN controllers. Instead, it looks to solidify the fact that there is a problem with a controller through the means of a man in the middle (MitM) attack. If successful, a MitM attack could be devastating for an SDN given the nature of MitM [5]. Ultimately, the point of this paper rests in two questions:

- Can a man-in-the-middle attack be executed successfully on an SDN controller?
- Can a man-in-the-middle attack be executed on an SDN controller in such a way that it compromises the entire network?

The rest of the paper is organized as follows. In Section2, key concepts and terms, such as SDN, SDN security, ODL, MITM will be visited and explained. The methodology and the setup of our expletory efforts are then described in Section 3, followed by the results and discussions in Section 4. Potential counter measures are then highlighted in Section 5. Finally paper concludes in Section 6 with recommendations and future work.

2. LITERATURE REVIEW
2.1 SDN Overview
It is important first to understand exactly the term Software-Defined Networking (SDN) we are investigating in this work. Computer networking hardware involves devices like routers, switches, and access points. Currently, these devices provide functions that belong to both data plane and control plane: a device contains the hardware that moves the data forward (data

RIIT'15, September 30-October 03, 2015, Chicago, IL, USA
 ISBN 978-1-4503-3836-3/15/09…$15.00
DOI: http://dx.doi.org/10.1145/2808062.2808073

plane) and also contains the software to make the forwarding decisions (control plane). The drawbacks of the current network paradigm include, but not are not limited to, high costs, slow innovations, limited network topology awareness, and the fact that they are difficult to manage at a large scale. It also suggests that a network professional must learn each different proprietary OS of network devices of different brands and configure each device separately [6].

SDN, though not a totally new concept, is revolutionary because it separates the control plane and the data plane. This separation suggests that off-the-shelf commodities can be easily implemented to be the heavy duty workers in the enterprise environment. Together with the ever popular open source SDN controllers, organizations can easily plug in and customize appropriate middle ware modules, such as custom IDS/IPS systems. In the control plane, one of the major benefits of SDN controllers is the awareness of the network topology: more in-depth knowledge of who is connected to who at what capacity not only provides means for traffic engineering but also offers additional ways to fight network attacks such as DDoS [15].

From the perspective of network administrations, the network management will be different too. Instead of connecting and logging into each router and switch to manually configure each separate interface, VLAN, and routing protocol, the SDN approach enables configuration to be done from a central control server. That ability alone makes SDN appealing. In addition to basic router configurations, SDN allows for the creation of flow tables, which let you essentially control anything related to the data traveling on your SDN [6].

2.2 OpenDaylight (ODL) Controller

Within SDN, there are several different variations of the controller. This work does not mean to single out one specific SDN controller against the penetration test. The reason ODL was chosen for the study was mainly because it is open source and it is built for the enterprise environment, which means it has greater impact to the progression of SDN controllers. In addition, ODL is easy to install and test in the lab environment. It also allows fast deployment from the lab environment to the production environment.

In particular, we investigated the Helium release, which was the newest release we could get at the time of experiment. ODL defines itself as "an open platform for network programmability to enable SDN and NFV for networks at any size and scale [7]." One interesting note is that ODL does not come with any security plug-ins preinstalled. In fact, the only security defense it really provides is something called Defense4All, which runs on a machine separate from the controller and "communicates with OpenDaylight Controller via the ODC north-bound REST API [8]." The main purpose of Defense4All is DDOS prevention.

2.3 SDN Vulnerabilities

There have been many papers written already about possible vulnerabilities in SDN. In [2], Kruetz summarized seven attack vectors including forged traffic flows, attacks on switches, attacks on control plane communications, attacks on controllers, attacks on applications, attacks on administrative stations and the lack of trusted resources for forensics and remediation. These attack vectors are visualized respectively in Fig. 1.

In this work, we focus on two threat vectors in our investigation. Listed as threat vector three and vector four in [2], "attacks on control plane communications" and "attacks on and vulnerabilities in controllers" were examined and investigated in our lab environment (the other attack vectors were not pertinent to this experiment). Both of these vulnerabilities are specific to SDN, which makes them particularly interesting to study in that they cannot be observed through testing on a normal enterprise network. They also magnify why controller security is such a huge issue in SDN.

- With vector three, there is a huge issue in that communication is not secured between the controller and SDN devices. This allows for exploration of said communications, which can have devastating effects on the network. Most notably, this could allow for DoS or MitM attacks to occur.
- Vector four is the main threat here, though. Any vulnerability or security fault in the controller itself compromises the entirety of the network. If the controller is taken over by a malicious entity, there is no actual limit on what that person could do to the network. This should be a huge red flag. As of yet, there is no concrete solution to fix this issue in SDN.

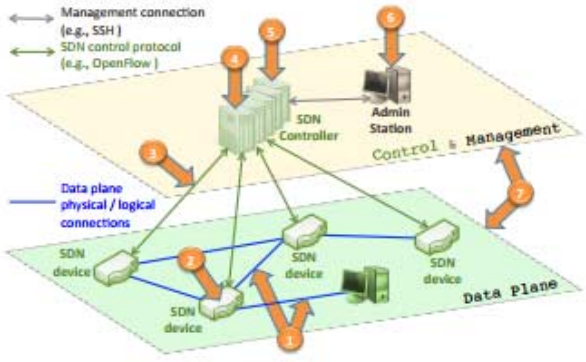

Figure 1. SDN vulnerability map [2]

2.4 Man-in-the-Middle Attacks on SDN

The attack that we focus on in this paper is a Man-in-the-Middle (MitM) attack. The basic premise behind a MitM attack is that, somehow, a third party inserts his or herself between two endpoints in a network, normally a client and that client's gateway. In this way, all traffic from the client destined for the Internet must now travel through the third party [5]. This allows the attacker to inspect all traffic sent from the client, possibly gathering things like login credentials to sensitive websites. MitM attacks could exist at all seven layers the OSI networking model. More practically, they are often found at layer 2 by means of ARP

poisoning, at layer 5 by means of session hijacking, and at layer 7 by taking advantages of security flaws of network applications.

The idea of a MitM attack taking place on an SDN controller is one that could cause serious issues for SDN moving forward. In fact, there have been successful attempts at perpetrating such an attack as demonstrated in [3] with their MitM attack. However, in that experiment, it was assumed that the host was already compromised. The premise of a MitM attack is that it should only be deemed successful if you can accomplish the first step of compromising the host, and only then doing packet-sniffing; if you cannot first compromise the host (i.e. controller), then the MitM attack is essentially nullified.

The MitM attack described in this work however does not assume the host needs to be compromised and the consequences of the attack are simply devastating: the attacker will be able to sniff the login credentials to potentially take full control of the entire network.

3. METHODOLOGY

In this experiment, it was assumed that the controller would be hosted in a remote-office setting, defined in [4] as particularly vulnerable as it would often be less focused on security procedures. The attack executed in this experiment could still be possible in a more secure, enterprise environment; however, it is far less likely given the necessity to be on the same network segment as the controller.

3.1 Experiment Setup

In this experiment, three physical machines were used. Fig. 2 details the separate machines and how they were connected.

One machine, running CentOS 6.4, hosted the ODL Helium controller. In this instance, ODL was installed with all default features. However, one extra feature (L2 Switch Component) was installed to allow for access to the ODL DLUX web interface [9].

The second machine hosted an instance of Kali Linux installed on a USB drive. No special features were added to Kali for this experiment. The only tool used within Kali that was *ettercap*. This one tool was used to: 1) initially perform an ARP-cache poisoning attack (the most common version of a MitM attack) on the controller; 2) perform traffic sniffing; 3) attempt to capture login credentials to the DLUX web interface.

The third machine was a neutral machine running Windows 7 used to access the DLUX web interface from outside the LAN segment hosting the ODL controller.

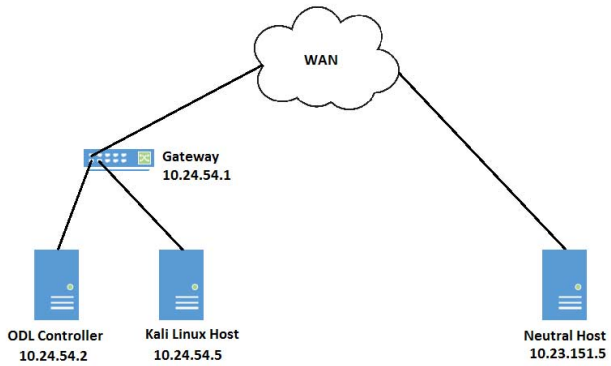

Figure 2. Experiment setup

No SDN switches were used in this experiment. It was deemed unnecessary as the experiment focused solely on whether or not the controller itself could be compromised from the outside, and not through switch traffic. Refer to the MitM attack performed in [3] for an attack done using switch traffic.

3.2 Kali Linux Tools

Only one tool found in Kali Linux was used during this experiment. The most useful of the tools proved to be *ettercap*. Kali provides several ways to successfully execute a MitM attack through ARP poisoning, but *ettercap* seemed to be the easiest method for doing so. *ettercap* provided means for traffic capturing and password sniffing as well. For this experiment, it proved to be an extremely powerful and useful tool.

ettercap was edited in several ways to facilitate this experiment. First, *etter.conf* had to be edited to allow for packet forwarding. If this had not been done, as soon as the MitM attack started executing the attack target, i.e. the SDN controller, would instantly lose its network connection. This was not feasible for this experiment where the aim was to both capture network traffic to and from the controller, as well as capturing login credentials to a web application.

etter.conf also had to be edited to specifically sniff specific ports in order to capture login credentials. Normally, *ettercap* only sniffs TCP port 80 for HTTP traffic. This is logical as nearly all HTTP traffic is transferred using that port number. However, it is possible to transfer HTTP traffic over a different port if specificed by the traffic origin. This is the case with ODL and its DLUX web interface. Instead of using port 80, it used the ports 8080 and 8181. *etter.conf* was therefore edited to sniff HTTP traffic on these ports as well.

3.3 ODL Controller Setup

For this experiment, nothing particularly out of the ordinary was configured on the controller. The controller, Open Daylight Helium (their newest release) was downloaded and installed on top of a machine running CentOS 6.4. Once installed, one package was added to ODL to enable the DLUX web interface, that package being the L2 Switch component. After basic installation and installation of the L2 Switch component, the DLUX web interface could be accessed from the neutral host.

4. RESULTS

As has been previously stated, the two goals of this experiment were to capture traffic, and if traffic could be captured, also use that traffic to compromise the controller in some manner. The manner chosen here was to attempt to capture the login credentials to the DLUX web interface. Both of these goals were met successfully.

4.1 Traffic Capture

Through ARP poisoning, all traffic destined for and originating from the ODL controller had to first pass through the Kali Linux box. This allowed for the capture of that traffic. Fig. 3 shows some of that traffic relating to the DLUX web interface on the controller.

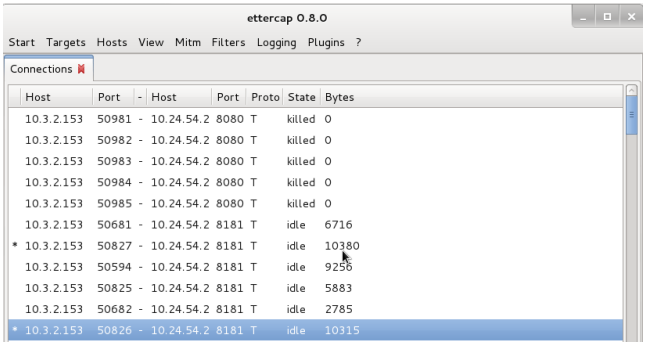

Figure 3. ODL controller traffic

In Fig. 3, it should be noted that 10.3.2.151 corresponds to the neutral host initiating the connection on the DLUX web interface, while 10.24.54.2 corresponds to the ODL controller.

From this information, it was determined that DLUX was not using TCP port 80 for its HTTP traffic, instead using ports 8080 and 8181. It should also be noted that the DLUX web interface used HTTP to communicate with the controller, not HTTPS. This information was further used to sniff login credentials for the DLUX web interface.

Armed with the knowledge of which ports ODL and DLUX were using for HTTP traffic, *etter.conf* was configured to sniff these ports, along with port 80 for HTTP traffic. Fig. 4 shows this configuration.

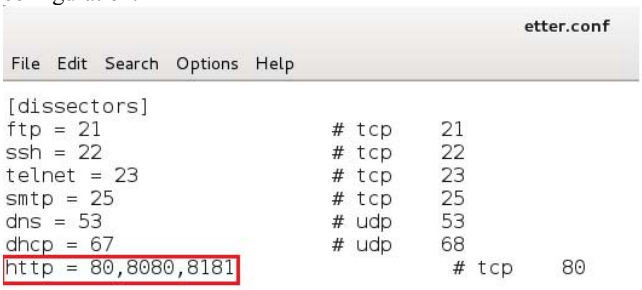

Figure 4: etter.conf port configuration

4.2 Credential Capture

With *etter.conf* configured properly, login credentials could be sniffed using the same methods for traffic capture [10]. Fig. 5 shows these login credentials successfully captured.

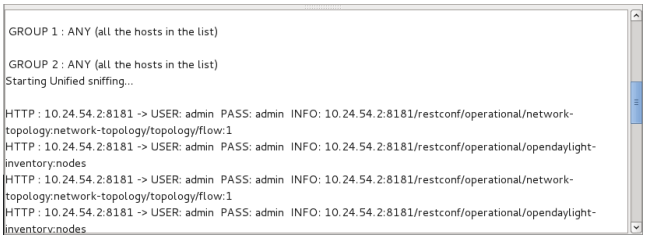

Figure 5. Credential capture

With this login information, the DLUX web interface can be accessed. From this, the network can be manipulated and changed. While this provides additional functionality for the controller, it can also be seen as a weakness if it can be exploited. If an attacker can gain access to the web interface, it could be disastrous for the SDN as a whole.

5. DEFENSE MEASURES

Perhaps the most alarming part when dealing with MitM attacks is that they are very hard to detect once they've already occurred. Therefore, it's best to detect and prevent MitM attacks before they've been successfully executed. Thankfully, a number of tools have come out of the past few years to help deal with and detect ARP poisoning attacks like the one used in this experiment.

There have been a number of tools developed to help dealing with ARP poisoning attacks as they can be particularly devastating if carried out successfully. Some of the better tools include Snort, ArpAlert, and ArpwatchNG [11][12][13]. In the cases of ArpAlert and ArpwatchNG, the tools can be installed directly onto the Linux host that the controller is also installed on.

ArpAlert is pretty simple in its operations. It looks at all of the ARP traffic on a specific interface, and then makes sure the MAC address to IP address mapping matches a preconfigured, static list of mappings. If the mapping is not present in the preconfigured list, the program then runs a predefined script to handle the possible attack taking place.

ArpwatchNG is similar to ArpAlert, in that it monitors the MAC addresses being used on the network. These MAC addresses are then written into a file with timestamps and change notifications. No actions are carried out on behalf of the program though, unlike ArpAlert; it is up to the network administrator to notice anything malicious in the file tracking the MAC addresses being used.

Snort is a bit more involved a method than the previous two. Snort is an intrusion prevention system (IPS), designed to detect a multitude of attacks. One of the tools included in Snort is an ARP Spoof Preprocessor [14], which performs the same actions as ArpAlert, but on a larger scale (i.e. watches your entire network for ARP attacks). Using Snort, or a similar IPS, is probably the best chance one would have of detecting the attacks used in this experiment, and preventing them from occurring. This would involve setting up an IPS (something most enterprise networks should already have), and configuring it with an interface on the same network as the SDN controller. Note that Snort was not used and configured at all in the experiment described in the paper. It is simply speculative that it would provide the best defense at detecting and preventing the ARP attack used.

Another defense strategy one can use is to always use a cryptographically sound method for mutual authentications of both communication parties whenever it is feasible. In particular, we are surprised that the OpenDayLight User Experience (DLUX) component that carries user login information was implemented over HTTP, not HTTPS. This suggests software developers should ban HTTP for mission critical tasks at all costs and system administrators should enforce an HTTPS only policy for all critical network traffic.

6. CONCLUSIONS

SDN is a rapidly growing technology. The potential it provides to the networking world is enormous. However, as with any new technology, there are issues with SDN. This paper looked at some of the security issues with SDN, focusing closely on the issues surrounding the controller.

In a software-defined network, the controller stands as a central point of failure for if it can be compromised. This paper shows that it is possible in at least one aspect. Through a man-in-the-middle attack, it is possible to capture enough information to

compromise the controller. With the controller compromised, the network can be manipulated in any way the attacker chooses.

Some considerations should be made here, though. First, the DLUX web interface is specific to OpenDaylight. This project focused solely on the ODL Helium controller, thus can only make statements about that distribution. Second, the DLUX interface is not installed by default in ODL. It had to be enabled by installed a separate component. The DLUX web interface simply provides some extra functionality to ODL by providing visualizations for the network configuration. It is not necessary in ODL for SDN operations.

However, this project assumes that ODL would have DLUX enabled, and shows the vulnerabilities if it is enabled. Currently, ODL has the possibility for DDoS protection through the installation of the Defense4All plug-in. However, that plug-in does not provide MitM attack protection. In fact, protecting against ARP poisoning can be very difficult, and there aren't a lot of methods for successfully preventing it. One method is to statically set the ARP value for the gateway on the controller machine. This paper and experiment did not explore prevention methods, though, so cannot account for the validity of this method in stopping the MitM attack performed herein.

Controller vulnerabilities are a serious issue that SDN developers must address in the coming months and years. Without a totally secure controller, SDN adoption will likely be deterred in industry settings.

It is expected that vulnerabilities like the one described in this paper will have been dealt with in some manner in the near future. However the lessons we learned from this project, once again, proved that a system is as secure as its weakest link. In the case of this investigation, the DLUX feature, as it currently stands, is the weakest spot in the implementation. Though this is not a must-have component in ODL architecture, the compromise of DLUX could potentially generate significant harm to the overall system. This reminds us, caution must be taken whenever a new component needs to be added to system. And the security should be considered when the system is designed, not an after-thought or a patch.

In the future, we would like to employ more applied approaches to further investigate the vulnerabilities that are specific to the SDN controllers and architecture. Recommendations and best practices will be made from our future study to help the industry safe guard their future network infrastructure.

7. REFERENCES

[1] Ramel, D. (2014). Explosive Growth Forecast for Software-Defined Networking. Virtualization Review. https://virtualizationreview.com/articles/2014/08/21/sdn-growth-forecast.aspx

[2] Kreutz, D., Ramos, F., and Verissimo, P. (2013). Towards Secure and Dependable Software-Defined Networks. Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking (pp. 55-60).

[3] Hong, S., Xu, L., Wang, H., and Gu, G. (2015). Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures. NDSS 2015.

[4] Benton, K., Camp, L. J., & Small, C. (2013). Openflow vulnerability assessment. In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking (pp. 151-152). ACM.

[5] Callegati, F., Cerroni, W., & Ramilli, M. (2009). Man-in-the-Middle Attack to the HTTPS Protocol. IEEE Security and Privacy, 7(1), 78-81.

[6] McKeown, N. (2009). Software-defined networking. INFOCOM keynote talk, 17(2), 30-32.

[7] (2015). Open Daylight. http://www.opendaylight.org/software

[8] (2014). Defense4All:User Guide. https://wiki.opendaylight.org/view/Defense4All:User_Guide

[9] (2014). Open Daylight Installation Guide. https://www.opendaylight.org/sites/opendaylight/files/bk-install-guide-20141002.pdf

[10] Encarnacion, L. (2014). Perform A Man In The Middle Attack With Kali Linux & Ettercap. http://lewiscomputerhowto.blogspot.com/2014/03/perform-man-in-middle-attack-with-kali.html

[11] (2008). ArpAlert. http://www.arpalert.org/arpalert.html

[12] (2012). Arpwatch-NG-VLAN. https://github.com/SgtMalicious/Arpwatch-NG-VLAN

[13] (2015). Snort. https://www.snort.org/

[14] Esler, Joel. (2012). ARP Spoof Preprocessor. http://manual.snort.org/node151.html

[15] Nick Feamster, Jennifer Rexford, and Ellen Zegura. 2014. The road to SDN: an intellectual history of programmable networks. *SIGCOMM Computer. Rev.* 44, 2 (April 2014), 87-98. DOI= http://doi.acm.org/10.1145/2602204.2602219