Experimental Study on the Detectability of Man-in-the-Middle Attacks for Cloud Applications

Cheng-Yu Cheng, Edward Colbert, and Hang Liu
Department of Electrical Engineering and Computer Science
The Catholic University of America
Washington, DC, USA
{chengc, colberte, liuh}@cua.edu

***Abstract*—Man-in-the-Middle (MITM) attacks can significantly compromise the security of the Internet and cloud computing applications, where an attacker intercepts the packets transmitted between the clients and servers over the network to steal confidential information and/or change the packets. It is essential and challenging to detect MITM attacks. In this paper, we build a virtual network testbed to emulate a real-world cloud environment and study the detection of MITM attacks. We consider an MITM detection approach that utilizes network packet analysis and machine learning techniques to measure the changes in the packet Round-Trip Time (RTT) between a client and a server. Specifically, we use the machine learning algorithms in TensorFlow to analyze the RTT data collected on the testbed to determine the detectability of MITM attacks. If the attacker's link speed is much higher than the client's link speed in an access network, e.g. an attacker connecting the network through a wired Ethernet and a normal client connecting the network through a wireless link, it would be difficult to discern the RTT difference with and without the MITM attacks. We are able to deduce a threshold below which the MITM attacks can be detected based on the RTT difference with a certain accuracy. Our experiments show the detectability accuracy becomes lower.**

***Index Terms*—Man-in-the-middle attacks, packet round-trip-time analysis, machine learning.**

I. Introduction and Motivation

In this paper, we explore detectability of adversarial attacks on network connections. With the ever-changing nature of computer networks there are many types of network attacks. In Distributed Denial of Service (DDoS) attacks, an attacker breaks into different computers (called zombies), and then controls all these zombies to send a large number of network packets to a victim server. By executing this attack, one can consume the server's finite resources and affect the availability of the services [1]. Other attacks such as SQL injection, cross-site scripting and SSL attacks affect different aspects of computer networks.

In particular, we focus on an attack known as the Man-in-the-Middle (MITM) attack. This attack offers the ability to intercept the connection between two network devices. By executing this attack, one can eavesdrop on the traffic and steal confidential information such as bank account number, Social Security number, passwords, or anything that is being transferred in this network connection. The MITM attack is an active attack. It utilizes several network attack methods such as Address Resolution Protocol (ARP) spoofing, Dynamic Host Configuration Protocol (DHCP) spoofing or port stealing [2]. For ARP spoofing, the MITM attack is executed on the media access control (MAC) layer by poisoning the ARP cache of a local network switch or router. A detection method for ARP spoofing type of attack was proposed by [3] by measuring the Round Trip Time (RTT) of network packets. However, for fast networks such as land-based Ethernet network links or for the anticipated 5th Generation cellular links, the additional delay may not be measurable by the user.

In this paper, we want to explore the impact of attacker's link speed to the detection method that we adopted from [3] for detecting MITM attack. The contributions of this paper is that we created a virtual network testbed to simulate a real-world network topology, and conducted a MITM attack within the virtual network. To make it more realistic, we generated background traffic by setting up a web server and using web browser automation to automatically fetch the websites. RTT measurements between client and server were collected by sending ICMP packets and calculating the time that elapsed. In order to have more comprehensive experimental results, we introduced a link with variable speed into the network. We compared the RTT result using Tensorflow Deep Neuron Network (DNN) classification to test the accuracy of the detection method. The result shows that the smaller difference in connection speed between client's link speed and attacker's link speed, the detection accuracy will be the worst. In other word, if the attacker's link speed is much higher than the client's link speed, then the proposed MITM detection method may have a lower chance to detect the presence of MITM attack.

The remainder of this paper is organized as follow. In Section 2, we will talk about some related research and previous work. Section 3 introduces our experiment configuration and the deployment of our testbed. Section 4, we detail the method for detecting MITM attack and the way to adopt machine learning to test the detectability. Section 5 presents our conclusions and future work.

II. Related Work

In this section, we will introduce several MITM attack detection mechanisms and briefly explain how they work.

978-1-7281-3101-6/19/$31.00 ©2019 IEEE
DOI 10.1109/CloudSummit47114.2019.00015

Since the most popular MITM attack approach in Local Area Network (LAN) is ARP Spoofing, majority of the detection methods are focusing on detecting ARP Spoofing types of MITM attack. A method was proposed to detect MITM attacks using machine learning algorithm [4]. They collected MITM attack network traffic on a real production network and provided a feature subset algorithm to select which parts of packet should be compared along with other packets for detecting MITM packets. A combination of supervised and semi-supervised machine learning method was used to build the detection model. The result shows that with the selected fields of IP identification number, TCP sequence number and TCP flags, the True Positive Rate (TNR) of the model is 0.9113.

A detection method was proposed by Ziqian et al. [3] using their client-side application that set a threshold for measured round trip time (RTT) and received signal strength (RSS) without MITM attack scenario and do analysis on the measured RTT and RSS to detect and further locate the MITM attacks. Since spoofed packets travel additional links to and from the attacker machine and this introduces some delay in the connection. By setting a detection threshold for the RTT and monitoring the delay from client-base application, one is often able to detect the presence of a MITM attack. If the delay is longer than the normal baseline, it will be considered as MITM attack. The framework will activate the locating mechanism to locate the physical machine by applying the measured RSS to machine learning model and predict the location.

Several MITM attack detection methods are calculating difference in delay to illustrate the abnormal behavior occurring in the connection. In this paper, we mainly focusing on testing the detectability of the method that use delay as their decision-making factor. We adopt the concept from [3] using Ping application to collect the RTT for each packet. And also collect the RTT for each packet under MITM attack. Finally, we adopt the concept from [4] using supervised machine learning algorithm to further test the detectability of the RTT detection method.

III. Experiment Configuration

In this section, we present the virtual network used this experiment. We deployed the virtual network on VMware ESXi vSphere Server (ESXi) [6] which can contain many virtual machines. We used VyOS virtual router operating systems [7] and Ubuntu operating systems [8] to simulate a realistic computer network. We built an Apache Web Server [9] using one of the Ubuntu computers and let all other computers perform common web browsing to simulate a background traffic.

A. Virtual Network

1) Network Infrastructure: ESXi is software that provides a virtualization layer which can abstract specific hardware resources from the physical host, such as processor, storage, memory, and networking interface cards. ESXi also provides several network infrastructures which allow the creation of different Virtual Local Area Networks (VLANs). VLANs are the logical segmentation of workgroups that the nodes in the same group can communicate with each other. One of the infrastructures is virtual switches which have the same packet-switching functionality as a physical switch. In addition, the virtual switches also have flexible capabilities that can connect the virtual machines to both virtual network and the physical network. We configured unique ESXi port groups and keep the configuration of logical rules and policy to the virtual ports for each virtual switch, so that the switches effectively operate as VLANs for each network segment. A diagram of our virtual network is shown in Fig. 1.

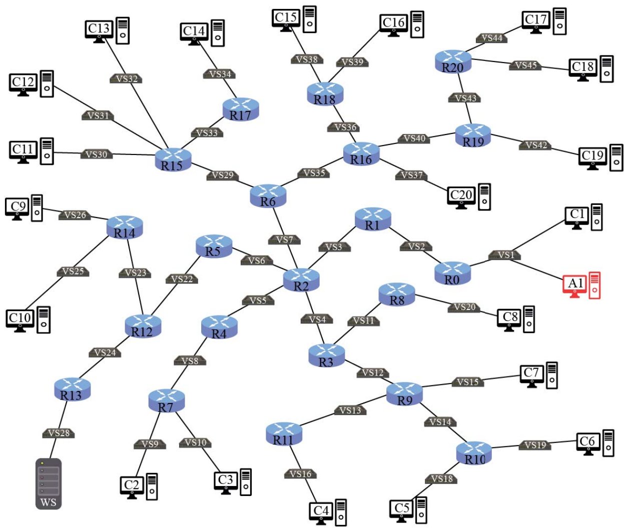

Fig. 1. Virtual Network Diagram, contains 20 clients, 45 virtual switches, 21 VyOS virtual routers, 1 webserver and 1 attacker

We connected the 45 virtual switches (Fig. 1, labeled as VS1...VS45) to 21 VyOS virtual router (Fig 1, labeled as R0...R20). VyOS virtual router software has the advantages that it is open source, highly configurable, easily scaling into larger network and most important of all, capable of running on a virtual machine, which makes it a good choice of this experiment. We adopted Routing Information Protocol (RIP) as the routing protocol. Every router that uses RIP will update its routing table periodically with its routing table through multitasking and exchanging routing information. The routing information that received from neighbor routers will always be trusted by host router itself.

2) Network Elements: The attacker of this experiment (Fig. 1, labeled as A1) was deployed at virtual switch number zero using a computer running Kali Linux [10]. Kali Linux is a pen-test operating system that has many tools and flexible frameworks for security researcher to utilize, such as vulnerability assessment, penetration testing and in this paper, launching a Man-in-the-Middle attack. Kali Linux provides a tool called ARPspoof [11], which has an ability to intercept the conversation between two nodes by maliciously modifying the

Address Resolution Protocol (ARP) cache of a nearby network device. In our experiments the attacker A1 modified the ARP table in network device R0.

To simulate real-world background traffic, an Apache Web Server version 2.4.18 (Fig. 1, labeled as WS) was introduced into this virtual network. We built up the Apache Web Server on an Ubuntu virtual machine, and used it to provide website services to the clients. We make use of Hypertext Markup Language (HTML) to create twelve different websites. Each website contained a picture and a sentence. We also configured 20 web clients (Fig. 1, labeled as C1...C20) running Ubuntu 16.04 to fetch these websites.

B. Simulation of Web User Background Traffic

In order to make the virtual network more realistic, we utilized the Selenium Browsers Automation (Selenium) [13] software to enable automated browsing. We used one of the tools called Selenium-WebDriver, which we controlled using a command-line python script. In the Python automation script, we stored the URL of twelve websites that we had created in the WS, and we closed the cache, cookie and history of Firefox by configuring the Firefox profile through the Selenium-WebDriver, so that every website request will go through the virtual network to the WS and will not fetch directly from the cache. In addition, all 20 Ubuntu clients were running the same Python script, but were randomly fetching different websites and set to fetch websites with a random time interval between 1 and 3.5 seconds. The traffic rate of website requests from all 20 clients was approximately 150 packets per second, measured from the WS.

IV. Experiment

We simulated a MITM attack by adopting the method of [3] to detect MITM attack. We collected the RTT using the Linux Ping application and compared the RTT variance between normal traffic transmission, and the intercepted transmission. We conduct seven experiments with different setting in order to provide comprehensive results to support the assumption. In addition, we use Tensorflow to analyse the collected RTT to find out the impact of attacker's link speed with regard to the detectability of MITM attack.

A. Experiment Description

After setting up all the background traffic, we want to investigate the relationship between attacker's connection speed and victim's connection speed and the accuracy of detecting MITM attack. We deployed the MITM attack between C1 and R0 (Fig. 1) by launching an ARPspoofing from A1. After the ARP table was modified, C1 will believe that A1 is its default gateway, and R0 will believe that A1 is the one that sending the website fetching request. Every traffic that going through between C1 and R0 will be intercepted by A1, the MITM attacker. To detect the presence of MITM attack, we adopted the method of [3] by collecting the RTT from Ping application and comparing the RTT variance between normal traffic transmission, and the intercepted transmission between C1 and WS. With all the background traffic turning on, we collected two datasets for the first case and labeled it as CASE-1. The first dataset is using Ping application to send ICMP packets to WS without the presence of MITM attack between C1 and R0. We collected the RTT by letting C1 send 5000 ICMP packets to WS, waiting for the reply from WS. The Ping application will record the RTT for each packet and store all the RTT including the measurement of minimum, average, maximum and standard deviation of RTT. The second dataset is having the same setting as above, but have the MITM attack deployed between C1 and R0. With an aim of knowing the relationship between the connection speed and the detectability of MITM attack, all the run-time connection speed to the R0 was calculated by iPerf [14]. iPerf is an open source tool that can measure the bandwidth of a wired or wireless connection within the WLAN/LAN network. We use iPerf to measure the bandwidth of C1 to R0 and A1 to R0 to represent their connection speed. After using iPerf to measure each connection speed, we calculated the ratio of A1's link speed to C1's link speed and used this value to represent the relativity of client and attacker. These values will also be used to indicate the difference in link speed between C1 to R0 and A1 to R0 for Tensorflow analysis in the next section.

The above experimental procedures, with virtual network's websites fetching turning on, sending ICMP packets from C1 to WS, and collecting two RTT datasets of without and under the MITM attack, were used for all the experiments. However, With the aim of having more comprehensive results of the effect of link speed to the detectability of MITM attack, for second to seventh experiments we moved R0 router, C1 client and A1 attacker to a physical network, and changing R0 to be a physical wireless router. The virtual switch inside the ESXi has the ability that can connect the virtual machines with physical network. So, we used this feature to connect a NETGEAR access point (Fig. 2, labeled as AP) to the virtual switch (Fig. 2, labeled as VS2), and set the access point with the same configuration as VyOS, running RIP to route the packets. We use NETGEAR N300 Wi-Fi Router, which has 802.11g and 802.11n protocol to manipulate for wireless connection within these experiments. The client (C1) was changed from virtual machine to a physical personal computer running Ubuntu 16.04.2 LTS, with the same configuration as the original C1 in ESXi. The attacker (A1) was also changed to a physical personal computer running Kali Linux, with ARPspoof tool installed. Both client and attacker have wired and wireless network interface that allow us to utilize for different network scenarios.

For the second experiment, we connected C1 to AP with an Ethernet CAT 5 cable which has the connection speed up to 100 Mbps and connected A1 to AP with wireless signal. Fig. 2 shows the network scenario for CASE-2. The datasets that have been collected for this experiment were labeled as CASE-2. For the third experiment (CASE-3), we connected both C1 to AP and A1 to AP with wireless connection. Fig. 3 shows the network scenario for CASE-3. The third datasets that we generated from this experiment were labeled as CASE-

3. Lastly, for the fourth experiment (CASE-4), we connect C1 to AP with wireless connection and connect A1 to AP with an Ethernet cable. The datasets were labeled as CASE-4. Fig. 4 shows the network scenario for CASE-4. In these network scenarios, the AP was set with 802.11n protocol for all the wireless connection which means it has up to 150 Mbps link speed in this NETGEAR N3000 Wi-Fi router.

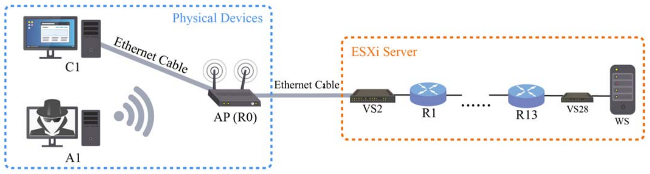

Fig. 2. Network Architecture of CASE-2 and CASE2a

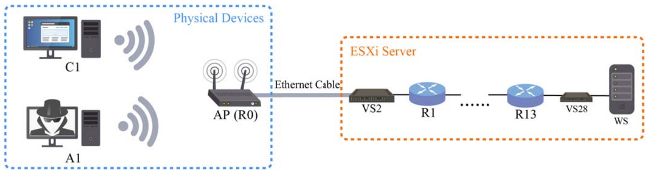

Fig. 3. Network Architecture of CASE-3 and CASE3a

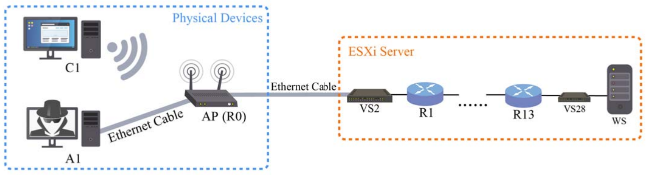

Fig. 4. Network Architecture of CASE-4 and CASE4a

In order to have much more comprehensive results to support the assumption, we also configure the wireless protocol of AP from 802.11n to 802.11g and implement three more experiments. By changing the wireless connection protocol of the AP, it will limit the transmission speed of the packets. To be concise, the 802.11g protocol provides the highest of 54 Mbps bandwidth for the wireless connection in this NETGEAR N3000 Wi-Fi router. In addition, the delay that was introduced to the connection by the attacker might also be different. As a result, we can have more experimental results to show the relationship between the connection speed and the detectability of MITM attack from different connection speed. The experiments were labeled as the following. The network scenario from Fig. 2 was adopted for CASE-2a, Fig. 3 was adopted for CASE-3a and finally, Fig. 4 was adopted for CASE-4a. For CASE-2a to CASE-4a, we also utilized iPerf to measure all the connection speed from C1 to AP and A1 to AP. Table 1 shows the measurement result of connection speed and difference between 2 link speeds for all the cases.

B. *Experiment Result and Comparison*

After collecting all the datasets for different network scenarios. We first compared these results by manual analysis and secondly, used Tensorflow machine learning technology to classified these data. We then use the result accuracy to represent the detectability of MITM attack within different network scenarios.

1) Manual Analysis: After collecting the RTT results for all the cases, we did a manual analysis for all the cases. Table 2 shows statistics for the RTT measurements result for all the cases. For CASE-1, we can find out that there is only a slight difference in RTT between without the MITM attack and under the attack. The difference in average RTT is 0.114 millisecond, and for the difference in minimum RTT is only 0.004 millisecond. This can indicate that in the virtual network scenario, because the attacker's link speed is fast enough that it didn't introduce much delay to the attack. Therefore, it might be a little harder for the client to find out the presence of the attacker by analysing the RTT measurement.

For CASE-2, CASE-3 and CASE-4, we can see that the average packet RTT of CASE-2 without the presence of MITM attack is 1.062 millisecond and the average packet RTT under the MITM attack is 10.302 millisecond. The result shows that the MITM attacker introduced a huge delay to the link between C1 and AP. In addition, for CASE-3, the average packet RTT without the attack is 4.854 millisecond and with the attack is 17.257 millisecond. The difference in average RTT of CASE-3 is smaller than CASE-2 because the difference in connection speed between C1 and A1 at CASE-3 is also smaller than CASE-2. As for CASE-4, we can find out that the difference in average RTT is only 0.497 millisecond, which also provides an evidence that if the attacker's link speed is getting much higher than the client, it might influent the RTT detection method.

For CASE-2a, CASE-3a and CASE-4a, we can know from Table 2 that the measurement RTT is similar to CASE-2 to CASE-4 respectively. Except that the packet loss and the average RTTs of these cases are all higher than CASE-2 to CASE-4. The reason is that when we changed the wireless connection protocol from 802.11n to 802.11g, it also reduced the channel bandwidth, causing a higher rate of packet loss.

2) Tensorflow Analysis: We used machine learning technology to test the detection method by adopting a classification model using Google Tensorflow Python API [15]. We utilized a sample code of supervised classification model from Tensorflow, and modified it to test the detectability of the MITM attack detection method. For the Tensorflow analysis, we separated the datasets into training and testing data for each case. We've trained different classifier for each case to test the classifying accuracy respectively. This accuracy then be used to represent the detectability of the MITM attack. For the propose of supervised learning, we pre-processed each case's datasets into formal input data. Take CASE-1 as an example. We tagged each RTT result from the first dataset with an "OFF" label to specify that this is the result without the presence of MITM attack. Same thing with the second dataset, we tagged each RTT result with an "ON" to indicate

TABLE I
MEASUREMENT RESULT OF WIRELESS CONNECTION SPEED AND WIRED CONNECTION SPEED

Case	*Type of Connection to the AP*	*C1 Connection Speed (Mbps)*	*A1 Connection Speed (Mbps)*	*Ratio of A1's Connection Speed to C1's Connection Speed*
Case-1	Both Virtual Wired	7245	7224	0.99
Case-2	C1 wired, A1 wireless	93.3	33.2	0.355
Case-3	C1 wireless, A1 wireless	29.5	29.9	1.013
Case-4	C1 wireless, A1 wired	31.7	91.2	2.87
Case-2a	C1 wired, A1 wireless	94.4	14.8	0.15
Case-3a	C1 wireless, A1 wireless	15.5	17.6	1.13
Case-4a	C1 wireless, A1 wired	14.6	93.8	6.4

*Protocol of wireless access point: 802.11n for CASE-2 to CASE-4, 802.11g for CASE-2a to CASE-4a

TABLE II
MEASUREMENT RESULT OF PACKETS RTT

Case	**Dataset**	*Average RTT*	*Minimum RTT*	*Maximum RTT*	*Packets Loss*
Case-1	Attack OFF	0.639 ms	0.299 ms	7.703 ms	0 packet
	Attack ON	0.753 ms	0.303 ms	4.935 ms	0 packet
Case-2	Attack OFF	1.062 ms	0.457 ms	4.276 ms	0 packet
	Attack ON	10.302 ms	3.246 ms	877.476 ms	24 packet
Case-3	Attack OFF	4.854 ms	1.213 ms	202.864 ms	2 packet
	Attack ON	17.257 ms	3.566 ms	605.234 ms	6 packet
Case-4	Attack OFF	4.854 ms	1.213 ms	202.864 ms	2 packet
	Attack ON	5.351 ms	1.808 ms	380.688 ms	0 packet
Case-2a	Attack OFF	1.052 ms	0.446 ms	3.601 ms	0 packet
	Attack ON	20.785 ms	4.258 ms	231.837 ms	134 packet
Case-3a	Attack OFF	5.557 ms	1.052 ms	509.582 ms	5 packet
	Attack ON	22.466 ms	3.697 ms	991.334 ms	209 packet
Case-4a	Attack OFF	5.557 ms	1.052 ms	509.582 ms	5 packet
	Attack ON	7.180 ms	2.072 ms	284.951 ms	11 packet

*Attack OFF represent the normal traffics, Attack ON represent the intercepted traffics..
*Each dataset contains 5000 ICMP packets.

the presence of MITM attack. Next, we randomly select 4000 RTT results from the first dataset and combine it with 4000 RTT results that have been randomly selected from second dataset and store the combination as a CSV file, which will be used as a training data. The remaining 2000 results from both first and second datasets are combined into a second CSV file, which will be used as a testing file.

Next, we built three layers Deep Neuron Network (DNN) with 10, 20, 10 units respectively within the classifier and set the training step to 10,000 steps. The DNN classifier will be trained from the 8000 packets dataset, and then it will be utilized to classify the testing dataset to see the packet is at the class of without or under the MITM attack ("OFF" or "ON"). We use this accuracy result to represent the detectability of the MITM attack which was detected by RTT measurement method. The accuracy result shows that for CASE-1, the classification accuracy is 0.701, meaning that 70% of the 2000 testing packets were classified correctly.

For CASE-2, which is wired C1 and wireless A1, the testing accuracy of the model is 1.00, which means that the 2000 testing packets have all been classified correctly through the classifier. This is because the delay that was introduced to the link by the attacker is long enough for the classifier to figure out whether the packet is without or under the attack. For CASE-3, wireless connection on both C1 and A1, the classifying accuracy of the model is 0.8735, which also indicates that with an identical link speed, it is a little harder for the classifier to identify the presence of the attack compare to CASE-2. And for CASE-4, since the attacker's link speed is much higher than the client, the classification accuracy is only 0.518 which means the classifier can only recognize half of the RTT record. Fig. 5 indicates the accuracy with respect to the ratio of A1's connection speed to C1's connection speed for all the Tensorflow analysis results. From Fig. 5, we can see that our assumption was confirmed. When the attacker's link speed is getting higher than the client's link speed (Fig. 5, X-Axis toward right), the RTT-base detection method will have lower chance to detect the presence of MITM attack.

For wireless connection, we can find out from Table 2 that there were some delays caused by environmental effects that would appear as anomalies. For example, the maximum RTT of CASE-3a is 509.582 ms without the presence of MITM attack. And the effects that appear in CASE-2a to CASE-4a are really significant. The explanation is that these scenarios were using 802.11g protocol, which has lower bandwidth and weaker signal frequency. To illustrate this, we present sample data in graphical form. Fig. 6 was generated from two sample datasets without MITM attack and it also shows different amount of anomalies between two different wireless protocol. This kind of delay would appear similar to a delay of a MITM attack which may cause some miss-classifications, in other word reduced the accuracy of the classification result. In our current Tensorflow analysis, we assumed that the machine learning classifiers are intelligent enough to figure out what kind of RTT results are considered anomalous and it would not affect the accuracy.

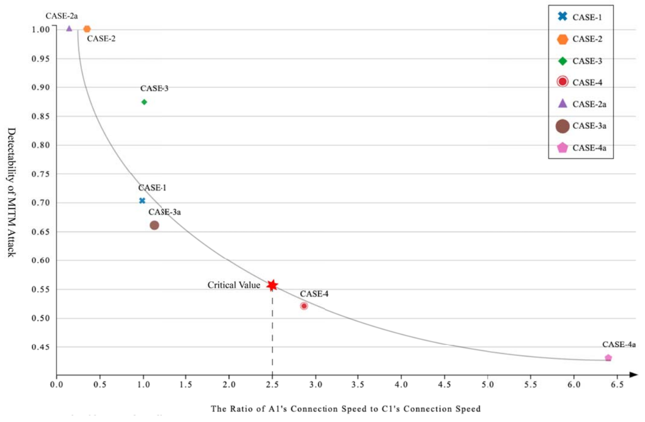

Fig. 5. Relationship between link speed and Man-in-the-Middle attack detectability

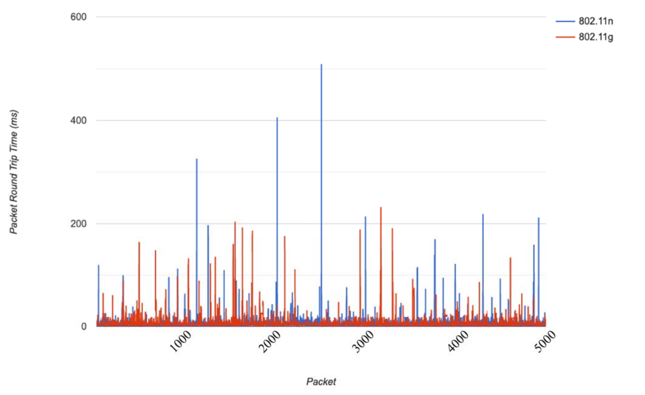

Fig. 6. Sample datasets of C1 sending 5000 ICMP packets to WS with different wireless protocol

V. Conclusion and Future Work

We test the detectability of using round-trip time to identify Man-in-the-Middle attack between two network devices in different network scenarios by analysing the measured RTT. In addition, we also assume that for the faster network interface, the delay that was introduced to the connection by attacker may not be measurable by the users. To assess the detectability of the method, we simulate a realistic network testbed using VMware ESXi vSphere Server for the deployment of detection method, and analysing the RTT results by manual calculation and Tensorflow machine learning classification. We generated background traffic to make the network more realistic. In order to cover different results for different network scenarios, we also combine ESXi virtual server with physical wireless access point to present different types of network connection. If the attacker's link speed is faster than the client, it is really hard to tell the different through the RTT result. And from the Tensorflow analysis, the classification results also provide indication of difference in connection speed could impact the detectability of the RTT detection method. Specifically, when the detectability accuracy became lower than 0.55 (corresponding to when the attacker link speed is 2.5 times faster than the client link speed), the detection method could not detect the attack correctly. In the worst case (attacker's link speed is faster than client's link speed), the detection accuracy is only 0.43 when using Tensorflow analysis to test the detectability of MITM attack using RTT-base method. Despite we use a physical wireless access point to deploy several experiments and utilizing two wireless protocols to represent different connection speed, the result did not cover all the existing network speed range. Also, the physical wireless access point did not provide very stable connecting signal which introduced some anomaly to the RTT measurement. More experiments still need to be done with much more stable wireless access point or any other substitute which can provide different range of connection speeds.

References

[1] Wu, Y., Tseng, H., Yang, W. and Jan, R. (2011), "DDoS detection and traceback with decision tree and grey relational analysis, " International Journal of Ad Hoc and Ubiquitous Computing, 7(2), p.121.
[2] Calvert, C., Khoshgoftaar, T., Najafabadi, M. and Kemp, C. (2017), "A Procedure for Collecting and Labeling Man-in-the-Middle Attack Traffic, " International Journal of Reliability, Quality and Safety Engineering, 24(01), p.1750002.
[3] Dong, Z., Espejo, R., Wan, Y. and Zhuang, W. (2015), "Detecting and Locating Man-in-the-Middle Attacks in Fixed Wireless Networks, " Journal of Computing and Information Technology, 23(4), p.283.
[4] Najafabadi, M. M. (2017), "Machine Learning Algorithms for the Analysis and Detection of Network Attacks, " Florida Atlantic University.
[5] Vallivaara, V. A., Sailio, M., and Halunen, K. (2014, March), "Detecting man-in-the-middle attacks on non-mobile systems, " In Proceedings of the 4th ACM conference on Data and application security and privacy (pp. 131-134). ACM.
[6] VMware ESXi vSphere Server. https://www.vmware.com/products/esxi-and-esx.html
[7] VyOS Network OS. https://vyos.io/
[8] Ubuntu. https://www.ubuntu.com/
[9] Apache. https://httpd.apache.org/
[10] Kali Linux. https://www.kali.org/
[11] ARPspoof. https://linux.die.net/man/8/arpspoof
[12] OpenSSL. https://www.openssl.org/
[13] Selenium Browser Automation. https://www.seleniumhq.org/
[14] iPerf. https://iperf.fr/
[15] Google TensorFlow. https://www.tensorflow.org/