Two Methods for Active Detection and Prevention of Sophisticated ARP-Poisoning Man-in-the-Middle Attacks on Switched Ethernet LANs

Kenan Kalajdzic, Center for Computing Education, Bosnia and Herzegovina

Ahmed Patel, Universiti Kebangsaan Malaysia, Malaysia, and Kingston University, UK

Mona Taghavi, Universiti Kebangsaan Malaysia, Malaysia

ABSTRACT

This paper describes two novel methods for active detection and prevention of ARP-poisoning-based Man-in-the-Middle (MitM) attacks on switched Ethernet LANs. As a stateless and inherently insecure protocol, ARP has been used as a relatively simple means to launch Denial-of-Service (DoS) and MitM attacks on local networks and multiple solutions have been proposed to detect and prevent these types of attacks. MitM attacks are particularly dangerous, because they allow an attacker to monitor network traffic and break the integrity of data being sent over the network. The authors introduce backwards compatible techniques to prevent ARP poisoning and deal with sophisticated stealth MitM programs.

Keywords: *ARP Poisoning, Digital Forensics, Intrusion Detection & Prevention, Man-in-the-Middle Attacks, Protocols, Security*

INTRODUCTION

Although every machine on the Internet has one (or more) IP (Internet Protocol) addresses, these cannot be used for sending and receiving packets at the hardware level. IP addresses are administratively assigned logical addresses and are thus not understood by the network hardware. Nowadays, most computers are attached to a Local Area Network (LAN) through a network interface card (NIC) that only understands physical addresses. For instance, every Ethernet NIC ever manufactured comes equipped with a 48-bit physical Ethernet address. In order to avoid address conflicts, manufacturers of Ethernet NICs are assigned unique blocks of physical addresses by a central address allocation authority to ensure that no

DOI: 10.4018/jdcf.2011070104

two NICs will ever have the same address. NICs send and receive frames based solely on 48-bit Ethernet addresses, without any knowledge of the IP protocol.

Network applications, on the other hand, use IP addresses for communication, so a fundamental question now arises: How does an IP address get mapped to the physical address, such as an Ethernet address? The protocol which gives an answer to this question is called ARP (Address Resolution Protocol) and defined in RFC 826 (Plummer, 1982). It is implemented and run in almost every machine as an essential component of communication in *open* wide and local area networks to ensure unique identification of the network interface cards such as those encountered in Ethernet LAN environments. ARP provides a mechanism to translate logical network addresses into physical Media Access Control (MAC) addresses which are required for the exchange of packets on a local area network.

ARP is a stateless protocol designed without security in mind, which makes it an ideal means for launching DoS and MitM attacks on a LAN. By sending spoofed MAC addresses in ARP reply packets, a malicious host can poison the ARP cache of other hosts on the local network and thereby easily redirect network traffic.

To mitigate the danger of ARP-based attacks on local networks, multiple techniques have been proposed to detect and prevent attacks by malicious hosts. Detection of ARP poisoning is usually performed by specialized network tools, such as *arpwatch* (LBNL Network Research Group), or Intrusion Detection Systems. Carnut and Gondim (2003) and Trabelsi and Shuaib (2007) proposed delegating the detection to specialized detection or test stations with digital forensic capabilities.

For prevention of ARP-based attacks, a simple solution consists of using static ARP entries in the ARP cache. This solution, however, does not scale well especially in heterogeneous networks with dynamic IP addressing. Other solutions include use of cryptography for authenticating ARP traffic (Bruschi, Ornaghi, & Rosti, 2003; Goyal & Tripathy, 2005; Limmaneewichid & Lilakiatsakun, 2011; Lootah, Enck, & McDaniel, 2007), artificial intelligence (Trabelsi & El-Hajj, 2007), or hardware support for dynamic ARP inspection (Cisco Systems, 2009; Ortega, Marcos, Chiang, & Abad, 2009).

We have developed two methods for detection and prevention of ARP-poisoning-based MitM attacks. For simplicity and convenience, we call these Method1 and Method2, respectively. Our motivation was to find ways to cope with increasingly sophisticated MitM attack tools, while still maintaining backward compatibility with existing ARP implementations. Our methods feature several important advantages compared to the aforementioned approaches:

- We avoid the use of specialized computers as helpers in the attack detection process. While these solutions may be among the simplest to implement, delegating detection to a particular test computer or LAN switch makes them a single point of failure. Our methods also do not rely on special network devices, but address detection and prevention of ARP poisoning in the most common and usual network settings.
- Our methods do not use cryptography. Despite the fact that cryptographic functions generally help in preventing ARP poisoning, they require a special infrastructure and modifications of various components in the entire network. With our methods, it is possible to implement detection and prevention of ARP poisoning on any host in the network independently of other computers.
- Instead of relying on artificial intelligence and heuristics in detecting ARP poisoning through anomaly analysis, both our methods make use of active IP probing, which helps in an unambiguous detection of Man-in-the-Middle attacks.

METHOD1 uses certain techniques proposed by Trabelsi and Shuaib (2007), but brought about several improvements in the approach to detection. Instead of relying on a test host to detect potential attacks, each host performs detection by itself. This eliminates the need for a test host, which is a single point of failure, and makes it possible to extend METHOD1 to perform distributed and coordinated detection with multiple hosts. Moreover, with METHOD1 detection is triggered by a reception of one or more ARP replies and targets only the hosts which send these replies, instead of scanning the whole network in the search of potential attackers.

METHOD2 addresses limitations of METHOD1 in dealing with sophisticated MitM attack tools. It relies on a novel technique for detection of MitM attacks on switched Ethernet LANs through modification of the switch CAM table in a way which makes the detection transparent to the MitM host.

METHOD1: REVERSE ARP POISONING WITH ACTIVE IP PROBING

METHOD1 consists of the following two steps:

1. *Reverse ARP poisoning* – A host implementing reverse ARP poisoning sends an ARP reply as a response to every ARP reply it receives from other hosts. The purpose of this reverse ARP reply is to poison the ARP cache of attacking hosts.
2. *Active IP probing* – Active IP probing is then used to differentiate between legitimate hosts and MitM hosts. This step consists of sending a single IP packet to the host from which the initial ARP reply was received by analyzing the response. For simplicity, in this document we use probe packets containing simple ICMP echo requests, even though it may generally be more reliable to use TCP or UDP instead of ICMP.

The best way to illustrate the workings of METHOD1 is to see it in action. For this purpose, we use two common scenarios.

In the first scenario, we analyze the packet exchange in the case of a legitimate host sending an ARP reply. The second scenario will then show how METHOD1 behaves when a MitM host attempts to carry out an ARP poisoning attack.

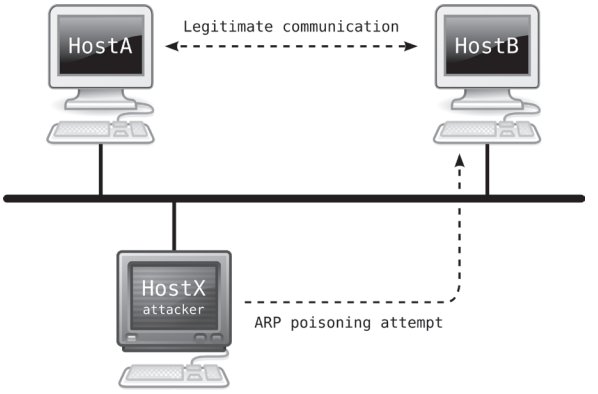
Figure 1 is used as a reference for both scenarios. We assume that all three hosts, HostA, HostB and HostX, are on the same Ethernet LAN. Furthermore, HostA and HostB are legitimate hosts and HostX is a MitM attacker. Also, HostA uses a regular implementation of ARP, as found in modern operating systems. HostB, on the other hand, implements METHOD1, thus handles ARP traffic in a different way. This is described in more detail in a subsequent section.

Scenario 1: Legitimate ARP Reply

In this scenario, HostA sends a legitimate ARP reply to HostB. We can follow the exchange of packets generated as METHOD1 is employed:

1. HostA sends an ARP reply packet to HostB. Since this is a legitimate ARP reply, it contains the mapping between HostA_IP and HostA_MAC.
2. HostB executes the first step of METHOD1, and immediately sends an ARP reply back to HostA attempting to poison its ARP cache. In this ARP reply HostB maps HostA_IP to HostB_MAC. Since, HostA is the owner of HostA_IP, it simply drops this ARP reply with the invalid mapping.
3. HostB then continues to the second step of METHOD1 and sends an ICMP echo request packet addressed to HostA_IP with HostA_MAC as the destination MAC address in the Ethernet frame header.
4. HostA receives the ICMP echo request and responds to HostB with an ICMP echo reply. For HostB this is an indicator that the reverse ARP poisoning attempt was unsuccessful and that the ARP reply sent by HostA is a legitimate one.

Figure 1. An ARP poisoning attack on a switched LAN. HostA and HostB are legitimate hosts on this network, and HostX is an intruder attempting a Man-in-the-Middle attack by means of poisoning the ARP caches of HostA and HostB.



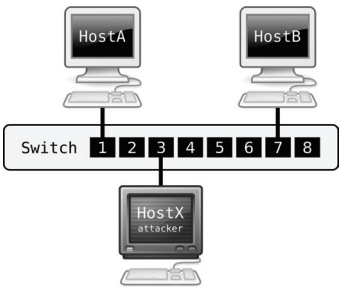
5. As a result, HostB stores the mapping HostA_IP↔HostA_MAC in its ARP cache.

Scenario 2: ARP Poisoning Attempt

In this scenario the attacking host, HostX attempts to poison the ARP cache of HostB in order to impersonate HostA. This should allow the attacker to hijack all traffic going from HostB to HostA. Since HostB implements METHOD1, the exchange of packets in this case will be as follows:

1. The first packet is an ARP reply sent from HostX to HostB. This ARP reply contains the mapping between HostA_IP and HostX_MAC. If HostB had a regular implementation of ARP, it would accept this ARP reply and store the incorrect mapping in its ARP cache. From that point on, HostB would deliver all network traffic destined to HostA_IP to HostX's network interface.
2. Nevertheless, HostB handles ARP traffic in compliance with METHOD1, so instead of blindly accepting the ARP reply from HostX, HostB begins the detection procedure by sending a reverse ARP reply to HostX. This ARP reply contains the mapping between HostA_IP and HostB_MAC. Assuming that the attacking host (i.e., HostX) uses an unmodified implementation of ARP, the reply sent by HostB will poison its ARP cache.
3. HostB proceeds with the MitM detection by delivering an ICMP echo request packet, destined to HostA_IP, to HostX's network interface (by using HostX_MAC as the destination in the Ethernet frame header).
4. HostX is acting as a MitM attacker, so it attempts to forward this ICMP echo request packet to HostA. However, HostX's ARP cache has previously been poisoned by HostB; HostX delivers the probe packet to HostB's MAC address. This effectively means that the same packet sent in the previous step by HostB will be returned to it by HostX. The detection of a duplicate packet is a clear indicator for HostB that reverse ARP poisoning was successful and that HostX is a MitM attacker.
5. HostB thus drops the initial ARP reply sent by HostX. Since at this point an intrusion attempt has been detected, HostB can generate a real-time intrusion alert and log

Figure 2. Physical connection of HostA, HostB and HostX in our LAN



the intrusion attempt for the purpose of a future digital forensic investigation.

METHOD2: ACTIVE IP PROBING WITH CAM TABLE POISONING

METHOD1 works well for detection of MitM computer systems which rely on the operating system built-in routing and ARP functions. There are, however, much more sophisticated MitM programs, which take full control over packet forwarding. This allows these programs to disguise themselves very well in order to evade detection. One popular program which falls into this category is the well-known *Cain & Abel* (Montoro).

Cain & Abel does not rely on the ARP and routing functions of the operating system, but instead maintains its own mappings between IP addresses and MAC addresses. The program utilizes these private mappings when forwarding frames between hosts on the network. This makes it insusceptible to reverse ARP poisoning, which is the basis of METHOD1.

In order to be able to detect any MitM host, regardless of the way it handles routing of packets between other hosts in the network, we need to influence flow of packets in a way which is beyond control of the MitM host.

In the following paragraphs, we utilize an alternative method to achieve this, which we call METHOD2 for brevity and simplicity.Figure 2 will serve as a reference for our description of METHOD2.

During normal operation of the switch, its Content-Addressable Memory (CAM) table contains the mappings shown in Table 1. (Note that in order to minimize switching latency, Ethernet switches store the mappings between MAC addresses and switch ports in a table inside a special CAM).

We again assume that HostX wants to redirect traffic between HostA and HostB through the use of ARP poisoning. HostA uses a regular implementation of TCP/IP, including ARP, and HostB employs METHOD2. We can now follow the use of METHOD2 through the following flow of events:

1. HostX sends an ARP reply to HostB. This ARP reply contains the mapping between HostX_MAC and HostA_IP.
2. Before entering this mapping into its ARP cache, HostB begins executing METHOD2, whose first step is broadcasting of an ARP request for HostA_IP.
3. As a result of this ARP request, HostB receives two replies with two different MAC address mappings for HostA_IP: one reply comes from HostA with HostA_MAC and the other is from the attacker with HostX_MAC. METHOD2 does not require these two replies to reach HostB in any particular order.
4. The reception of two different MAC addresses for a single IP address is a first indicator for HostB that one of them comes from a MitM attacker. Thus, HostB continues with the next step of METHOD2, which

Table 1. Switch CAM table during normal operation

MAC Address	Port
HostA_MAC	1
HostB_MAC	7
HostX_MAC	3

is sending multiple ICMP echo request packets out its network interface. All these packets carry HostB_IP as the source IP address and HostA_IP as the destination IP address in their IP header. However, their Ethernet frame header may contain one of the following two combinations of MAC addresses:

(a) HostX_MAC is the destination MAC address and HostA_MAC is the source MAC address,
(b) HostA_MAC is the destination MAC address and HostX_MAC is the source MAC address.

5. To understand the purpose of using these two MAC address combinations, let us analyze what happens when HostB sends two ICMP echo request packets addressed as in 4a and 4b, respectively:
   (a) The frame, addressed as specified in 4a leaves HostB and enters the switch through port #7. Based on the entries in its CAM table (Table 1), the switch forwards the frame to HostX through port #3. Meanwhile, since the frame with source MAC address HostA_MAC entered the switch through port #7, the switch updates its CAM table with a new mapping for HostA_MAC so the CAM table now has the contents shown in Table 2. HostX receives the frame, looks up the destination IP address, and forwards the frame immediately towards HostA, specifying HostA_MAC as the destination MAC address. Once this frame reaches the switch, two possibilities exist:
      i. If the switch CAM table still contains the mapping between HostA_MAC and port #7, the switch will forward the frame out through port #7. HostB receives its own ICMP echo request packet, which is an indicator that HostX attempted to forward this frame to HostA. This means that HostX is not the real owner of HostA_IP, but a MitM attacker.
      ii. If, in the meantime, HostA sent some network traffic through switch port #1, the original mapping of HostA_MAC to port #1 in the CAM table of the switch will have been restored. In this case, the switch forwards the ICMP echo request through port #1 to HostA, and HostA responds by sending an ICMP echo reply packet back to HostB. In this case HostB cannot conclude with certainty that HostX forwarded the frame to HostA.
   (b) The frame, addressed as specified in 4b enters the switch through port #7, and switch forwards it through port #1 to HostA. Since the source machine's MAC address of this frame is HostX_MAC, the switch maps HostX_MAC to port #7 in its CAM table. Table 3 shows the new mapping.

When HostA receives the ICMP echo request packet, it builds a response in form of an ICMP echo reply packet with source IP address HostA_IP and destination IP address HostB_IP:

i. Assuming that HostA's ARP cache has been previously poisoned by HostX, the

Table 2. Switch CAM table after HostB sends a frame from HostA_MAC to HostX_MAC

MAC Address	Port
HostA_MAC	7
HostB_MAC	7
HostX_MAC	3

Table 3. Switch CAM table after HostB sends a frame from HostX_MAC to HostA_MAC

MAC Address	Port
HostA_MAC	1
HostB_MAC	7
HostX_MAC	7

response packet will be sent in a frame addressed to HostX_MAC. If the contents of the CAM table has not been modified in the meantime (i.e., they are still as shown in Table 3), the switch will deliver this frame through port #7 to HostB. If, on the other hand, HostX generated some network traffic while HostA was preparing the response, the CAM table will have returned to its original state (Table 1). Thus, the switch will send the response packet from HostA to HostX through port #3. Because HostX is a MitM host, it will forward the response to HostB.

ii. If the ARP cache of HostA has not been modified, it will contain a correct mapping between HostB_IP and HostB_MAC. Therefore, the ICMP reply packet from HostA will be sent to HostB_MAC and delivered by the switch through port #7 to HostB.

We see that, in either case, using the MAC address combination given in 4b results in an ICMP echo reply packet being sent to HostB. In other words, it is not possible, that in the given scenario an ICMP echo request packet with source MAC address HostX_MAC and destination MAC address HostA_MAC gets delivered back to HostB.

On the other hand, the combination of source and destination MAC addresses as specified in 4a, it is possible for the original ICMP echo request packet to be delivered back to HostB (see 5(a) i), simultaneously, HostB may receive an ICMP echo reply from HostA (see 5(a) ii). The latter case cannot generally be distinguished from the case described in 5b, which uses frames addressed as in 4b.

Therefore, we must ensure that a host implementing Method2 (in our case, HostB) quickly sends multiple ICMP echo request packets with both combinations of source and destination MAC addresses given in 4a and 4b. To identify the MitM host it suffices for HostB to receive only one of its own ICMP echo request packets back through its network interface.

Method2 alters the CAM table of the switch so that some frames destined to HostA are delivered to HostB (Table 2). To restore the original mapping of HostA_MAC to port #1 (Table 1), HostB may broadcast an ARP request for HostA_IP. This would force HostA to return an ARP reply, and thereby assist the network switch to realign or reassociate its MAC address with port #1.

For Method2 to work, HostB's network card must be put into a promiscuous mode, so it can collect the hijacked frame which HostX attempts to forward to HostA. (Note that when

a network card operates in promiscuous mode, it accepts all traffic and passes it to the central processing unit, even if this traffic is not addressed to that particular network card.) Another important assumption is that HostA was not subject to DoS attacks, thus, it was able to respond to our ARP requests free of obstruction.

RESULTS

We ran multiple tests on a switched Ethernet LAN to test the effectiveness of METHOD1 and METHOD2 in detecting ARP-poisoning-based MitM attacks. In all these tests our setup was as depicted by Figure 2. HostA and HostB were running Windows XP and Linux respectively, and the operating system of HostX changed as required by the tests. Using several common tools, we performed MitM attacks from HostX, attempting to poison the ARP cache of HostA and HostB. The role of HostB was to detect these attack attempts by employing METHOD1 and METHOD2.

Detecting *Ettercap* and *dsniff* with METHOD1

In the first test HostX (running Backtrack Linux) performed attacks against ARP cache of HostA and HostB using two mainstream attack tools, *Ettercap* (Ornaghi & Valleri) and *arpspoof* with *dsniff* (Song, n. d.).

HostB was set up to perform attack detection with METHOD1. Since *Ettercap* and *dsniff* rely on the operating system built-in ARP and routing functions, we were able to successfully perform reverse ARP poisoning and detect all the attacks through active IP probing (i.e., METHOD1) with 100% accuracy.

Detecting *Cain & Abel* with METHOD1

For the purpose of this test we booted HostX into Windows XP and launched multiple MitM attacks against HostA and HostB using *Cain & Abel*. This time, however, HostB failed to detect any of our attacks. As was expected, that *Cain & Abel* uses its own IP-to-MAC address mappings when forwarding packets, thus, bypassing the detection process.

Detecting *Cain & Abel* with METHOD2

As discussed previously, when using METHOD2 HostB poisons the CAM table of the switch in order to capture the frame which HostX attempts to forward towards HostA. This is not a big problem when HostA is idle. If, however, HostA is actively communicating, this creates a race between HostA and HostB. Depending on the rate at which HostA sends out packets into the network, it may be more or less difficult for HostB to win the race and hijack the packet required for detection of the MitM attack.

To test the effectiveness of METHOD2, we set up HostA to send thousands of packets per second into the network and measured the attack detection ratio, whereby

$$Detection\ ratio = \frac{Number\ of\ successful\ detections}{Total\ number\ of\ probes\ sent}$$

During these tests, HostB was sending either single probe packets or series of 3, 5, 7, 9, 11, 13 or 15 packets per probe. The results of our measurements are summarized in Figures 3 and 4.

We notice that the success of detection depends on the number of packets sent in a single probe. The rather low detection ratio of 30% for single-packet probes was doubled by sending three packets in each probe. Further increases in number of packets per probe to five, seven and nine raised the detection ratio to 80%, 90% and 97% respectively.

Even though the detection ratio of 97% for nine-packet probes is very high and offers high reliability in detection of the MitM attacks, we continued our experiments with 11, 13 and 15 packets per probe attempting to further increase the detection ratio. As Figure 4 illustrates, we have reached a ratio of nearly 100% (in fact, 99.8%) with probes containing 13 packets.

Figure 3. Success in detection of Cain & Abel with Method2

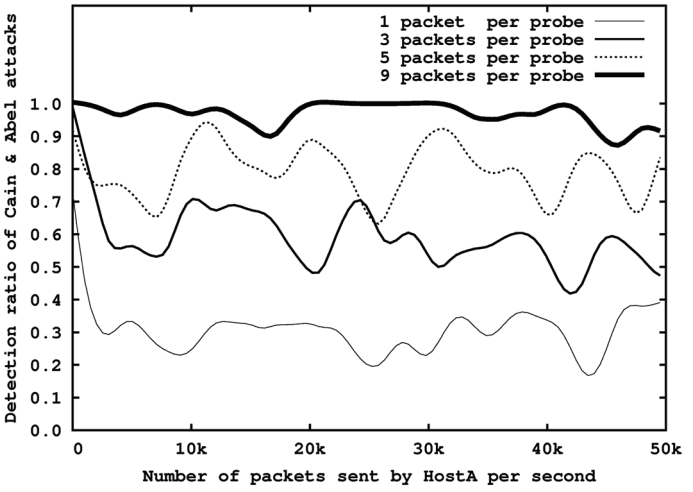
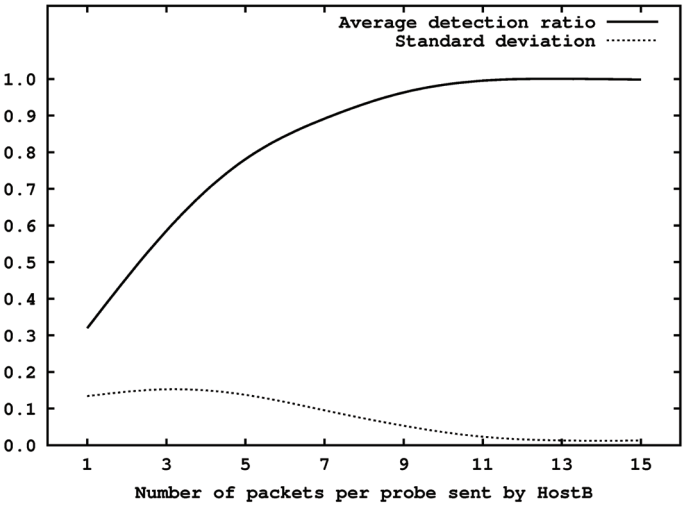


Figure 4. Average detection ratio of Cain & Abel attacks and the corresponding standard deviation, which are calculated for probes with number of packets ranging from 1 to 15



At the same time, Figure 4 shows the standard deviation, which drops as the number of packets per probe is increased. This decline of the value of standard deviation indicates that a greater number of packets per probe mean or imply higher reliability in detecting the attacks. For thirteen-packet probes, the value of the standard deviation lies around 1.3%, which under the consideration of the aforementioned average detection ratio of 99.8% means a de-facto guaranteed detection.

It is also obvious that the detection ratio is not dependent on the rate at which HostA sends packets into the network. If we ignore the variations in the value of the detection ratio, which exist due to a stochastic nature of real-time

network communication, we can consider all four curves in Figure 3 as constants.

DISCUSSION AND CONCLUSION

In this paper we described two novel methods for detection and prevention of ARP-based MitM attacks on switched Ethernet LANs. Both methods worked as extensions to the ARP protocol which did not interfere with normal ARP operation. Therefore, both these methods can co-exist on the same LAN with regular ARP implementations and were thus suitable for incremental deployment. We have shown examples of such co-existence in experiments in which one host (HostB) used either Method1 or Method2, while another host (HostA) used default implementation of ARP as provided by the operating system.

Both the theoretical discussion and results of our experiments have revealed certain limitations of both proposed methods.

We have also shown that the biggest limitation of Method1 is its inability to handle detection of MitM attack tools which use their own IP-to-MAC address mappings for forwarding packets (e.g., *Cain & Abel*). Even though Method2 solved this problem, other factors exist which may limit its effectiveness.

In the third step of the detection process with Method2, we assumed that HostB receives ARP replies for HostA_IP from both HostA and HostX. While this is generally the case, HostX might as well launch a DoS attack against HostA, preventing it from successfully delivering its ARP reply to HostB. This way only HostX's ARP reply would reach HostB, rendering Method2 useless.

The results of our experiments have shown that the effectiveness of Method2 depends on the number of packets sent in a single probe. Sending too many probe packets, however, may cause disruption in traffic flow towards HostA, due to the fact that HostB temporarily hijacks all LAN traffic destined to HostA_MAC. This problem could be solved by storing the hijacked packets in a queue on HostB and delivering them back to HostA after the probe.

Even though both our methods can be used to identify and prevent ARP poisoning attacks, an ultimate solution to the problem of ARP insecurity can only be provided through an improved version of the ARP protocol, which would be backwards compatible and would allow for an incremental implementation.Abad and Bonilla (2007) defined an ideal solution for prevention of ARP-based attacks, which may be the first step towards reaching this goal.

REFERENCES

Abad, C., & Bonilla, R. (2007). An analysis on the schemes for detecting and preventing ARP cache poisoning attacks. In *Proceedings of the 27th International Conference on Distributed Computing Systems Workshops*, Toronto, ON, Canada (p. 60).

Bruschi, D., Ornaghi, A., & Rosti, E. (2003). S-ARP: A secure address resolution protocol. In *Proceedings of the 19th Annual Computer Security Applications Conference*, Las Vegas, NV (p. 66).

Carnut, M., & Gondim, J. (2003). ARP spoofing detection on switched Ethernet networks: A feasibility study. In *Proceedings of the 5th Symposium on Security in Informatics*.

Cisco Systems. (2009). *Configuring dynamic ARP inspection: Catalyst 6500 series switch Cisco IOS software configuration guide, release 12.2(18) SXF and rebuilds and earlier releases* (pp. 1–22). San Jose, CA: Cisco Systems.

Goyal, V., & Tripathy, R. (2005). An efficient solution to the ARP cache poisoning problem. In *Proceedings of the 10th Australasian Conference on Information Security and Privacy* (pp. 40-51).

LBNL Network Research Group. (n. d.). *arpwatch: The ethernet monitor program; for keeping track of Ethernet/IP address pairings*. Retrieved from ftp://ftp.ee.lbl.gov/arpwatch.tar.gz

Limmaneewichid, P., & Lilakiatsakun, W. (2011). P-ARP: A novel enhanced authentication scheme for securing ARP. In *Proceedings of the International Conference on Telecommunication Technology and Applications* (pp. 83-87).

Lootah, W., Enck, W., & McDaniel, P. (2007). TARP: Ticket-based address resolution protocol. *Computer Networks*, *51*(15), 4322–4337. doi:10.1016/j.comnet.2007.05.007

Montoro, M. (n.d.). *Cain & Abel*. Retrieved from http://www.oxid.it/cain.html

Ornaghi, A., & Valleri, M. (n.d.). *Ettercap*. Retrieved from http://ettercap.sourceforge.net/

Ortega, A. P., Marcos, X. E., Chiang, L. D., & Abad, C. L. (2009). Preventing ARP cache poisoning attacks: A proof of concept using OpenWrt. In *Proceedings of the Network Operations and Management Symposium*, Punta del Este, Uruguay (pp. 1-9).

Plummer, D. (1982). *RFC-826: An ethernet address resolution protocol*. Retrieved from http://www.ietf.org/rfc/rfc826.txt

Song, D. (n. d.). *dsniff*. Retrieved from http://monkey.org/~dugsong/dsniff/

Trabelsi, Z., & El-Hajj, W. (2007). Preventing ARP attacks using a fuzzy-based stateful ARP cache. In *Proceedings of the IEEE International Conference on Communications* (pp. 1355-1360).

Trabelsi, Z., & Shuaib, K. (2007). NIS04-4: Man in the middle intrusion detection. In *Proceedings of the Global Telecommunications Conference* (pp. 1-6).

Kenan Kalajdzic received his BSc and MSc degrees in Electrical Engineering from the Sarajevo Faculty of Electrical Engineering, Bosnia and Herzegovina. His interests span a wide range of topics in the area of operating systems, computer networks and computer security. He is currently working as a lecturer at the Center for Computing Education in Sarajevo, and as an external visiting researcher with Prof. Ahmed Patel at the Universiti Kebangsaan Malaysia.

Ahmed Patel received his MSc and PhD degrees in Computer Science from Trinity College Dublin (TCD), specializing in the design, implementation and performance analysis of packet switched networks. He is a Lecturer and Consultant in ICT and Computer Science. He is a Visiting Professor at Kingston University in the UK and currently lecturing at Universiti Kebangsaan Malaysia. His research interests span topics concerning high-speed computer networking and application standards, network security, forensic computing, autonomic computing, heterogeneous distributed computer systems and distributed search engines and systems for the Web. He has published well over two hundred technical and scientific papers and co-authored two books on computer network security and one book on group communications, co-edited a book distributed search systems for the Internet. He is a member of the Editorial Advisory Board of the following International Journals: (i) Computer Communications, (ii) Computer Standards & Interfaces, (iii) Digital Investigations, (iv) Cyber Criminology, and (v) Forensic Computer Science.

Mona Taghavi received her BSc degree in Information Technology from Parand Islamic Azad University of Iran in 2007. Besides her involvement in several Iranian national ICT research projects, she had worked for an IT consulting and project managing company which was responsible for overseeing the Supreme Council of Information and Communication Technology (SCICT) of Iran programme. She was responsible for preparing some of the technical reports for this programme. Currently, she is pursuing her M.Sc. in Management Information Systems and Network Security at Universiti Kebangsaan Malaysia and undertaking research in cooperation with Prof. Dr. Ahmed Patel in advanced secure Web-based information systems and mashup technologies.