This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2007 proceedings.

Preventing ARP Attacks using a Fuzzy-Based Stateful ARP Cache

Zouheir Trabelsi
College of Information Technology
UAE University
United Arab Emirates
Email: trabelsi@uaeu.ac.ae

Wassim El-Hajj
College of Information Technology
UAE University
United Arab Emirates
Email: welhajj@uaeu.ac.ae

Abstract— **ARP cache poisoning is considered to be one of the easiest and dangerous attacks in local area networks. This paper proposes a solution to the ARP poisoning problem by extending the current ARP protocol implementation. Instead of the traditional stateless ARP cache, we use a stateful ARP cache in order to manage and secure the ARP cache. We also use a novel Fuzzy Logic approach to differentiate between normal and malicious ARP replies. The Fuzzy Logic controller uses a dynamically populated data base that adapts to network changes. The limits of the current approaches are discussed and analyzed.**

Keywords: ARP cache poisoning, Man-in-the-Middle (MiM) attack, Denial of Service (DoS) attack, Cloning attack, stateful ARP cache, Fuzzy Logic.

I. Introduction

Local Area Networks (LAN) use ARP, the Address Resolution Protocol, to resolve IP addresses into hardware, or MAC (Medium Access Controllers), addresses [1]. The LAN's hosts keep caches of resolved addresses, called the ARP caches. ARP resolution is invoked when a new IP address has to be resolved or an entry in the ARP cache expires. ARP has proved to work well under regular circumstances, but it was not designed to cope with malicious hosts performing ARP cache poisoning or spoofing attacks. The ARP poisoning attacks are often used as part of other serious attacks: Man-in-the-Middle (MiM) attack, and Denial of Service (DoS) attack. With a MiM attack, traffic between two hosts is redirected to a third host, which is usually the attacker's host. This attack allows the attacker to sniff the traffic exchanged between the two victim hosts. With DoS attack, a target host is denied from communicating with other hosts. This paper proposes a solution to the ARP poisoning problem by extending the existing ARP protocol. The new extension includes (1) statefull ARP cache, (2) Fuzzy Logic controller, (3) cross layer design, and (4) adaptive database manipulation. The limits of current approaches are discussed.

The rest of the paper is organized as following: Section II provides some background about ARP attacks. Section III provides an overview of the related work done in this area. Section IV discusses the proposed approaches. Section V concludes the paper and presents future research directions.

II. Background: ARP Cache Poisoning and ARP Spoofing

When a host adds an incorrect <IP, MAC> mapping to its ARP cache, this is known as ARP cache poisoning or ARP spoofing. The last terminology refers to the fact that an attacker uses fake or "spoofed" ARP packets to poison an ARP cache. In an ARP cache poisoning attack, the attacker sends ARP replies or requests with fake <IP, MAC> mappings, in an attempt to poison the ARP caches of other hosts on the LAN. Based on our experience's results in [2], skilled attackers use mostly ARP requests to poison their target ARP caches, since ARP requests can always corrupt any ARP cache even if the sender's IP address is not in the target ARP cache. The ARP poisoning attacks are often used as part of other serious attacks:

- *DoS attacks*: An attacker can poison an ARP cache of a host with a fake <IP, MAC> pairing so that every packet that host sends is sent to a fake host, or to the attacker's host instead of its real destination. In the latter case, the attacker blocks the communication from the host being attacked.

- *Host impersonation*: Instead of just dropping the packets received from the host being attacked, the attacker can respond, impersonating any host in the network.

- *MiM attacks*: By spoofing two hosts in the network at the same time, an attacker can silently sit in between the hosts so that they think they are communicating with each other. Then, the attacker is able to listen to the traffic sent in both directions. With a MiM attack, the attacker can gain access to sensitive information (e.g. passwords, emails' contents) or he/she can even modify the data being sent, compromising the data's integrity.

- *Cloning attack (MAC spoofing attack)*: In this attack, the malicious host changes its IP and MAC to become identical to those of the victim host. Once this change is done, there will be two hosts in the network with the same IP and MAC addresses. For the victim host, this situation will cause some network disconnection troubles and a DoS situation.

III. Related Work

Specialized tools like Arpwatch [3] can be used to detect suspicious ARP traffic. The main problem is that it depends on the network administrator being able to differentiate between non-malicious events and ARP cache poisoning attacks, and also on his/her ability to take appropriate and timely measures when an attack occurs. Intrusion Detection Systems (IDSs) like Snort [4] are usually able to detect ARP attacks. The main problem with IDSs is that they tend to generate a high number of false positives. Also, their ability to detect ARP poisoning is limited [5], as they may not be able to detect all of the forms of the attack.

Carnut et al. [6] proposed an architecture for detecting ARP spoofing attacks on switched networks. Their experiments showed that the architecture was very good at detecting ARP attacks without generating false positives. However, attackers could hide behind volume traffic to remain undetected for reasonably long periods.

ARP-Guard [7] uses a sensor-based architecture to detect and localize several internal network attacks, including ARP attacks. The management system alerts administrators in case an ARP attack is detected from the analysis of the information received from the LAN and SNMP sensors.

MAC spoofing attacks can be detected by sending an Inverse ARP (InARP) [8] request for a MAC address. The response can be used to determine if a computer is performing cloning [9] (if and only if the computer being cloned has not been DoSed or shutdown). This is a very limited solution as it only detects this type of ARP attacks.

Cloning attack can be prevented using a feature available on many modern switches called port security [10]. This solution is very efficient, but does nothing to prevent other types of ARP attacks that do not require the cloning of a MAC address [9].

Tripunitara et al. [11] proposed a middleware approach to asynchronous and backward compatible detection and prevention of ARP cache poisoning attacks. This schema does not prevent/detect attacks in which the host being spoofed is down or being DoSed.

A simple and effective way to prevent ARP attacks is to use static entries in the ARP cache. This solution has two disadvantages: (1) it does not work in dynamic environments (e.g. networks set up to use DHCP), and (2) it does not scale well, as it would be very cumbersome for the network administrator to deploy and update these tables throughout the network. Furthermore, some operating systems (such as Windows) may accept dynamic ARP replies and updates for static entries [5].

Anticap [12] is a kernel patch for various UNIX-based operating systems that aims at preventing ARP poisoning attacks by rejecting ARP updates that contain a MAC address different from the current table entry for that IP address. This solution works in static environments, but does not work in dynamic (DHCP-enabled) networks, and is available for a limited number of operating systems.

Gouda et al. [13] proposed an architecture for resolving IP addresses into hardware addresses over an Ethernet. The architecture consists of a secure server connected to the network and two protocols used to communicate with the server: an invite-accept protocol and a request-reply protocol. The disadvantage of this solution is that the secure server represents a single point of failure in the network, and becomes an obvious target for DoS attacks.

Several solutions that involve cryptography to authenticate the origin of ARP packets have been proposed [14], [15], [16], [17]. S-ARP [14] is a backward compatible extension to ARP that relies on public-key cryptography to authenticate ARP replies. For this solution to be implemented in a LAN, every host to be secured should be modified to use S-ARP instead of ARP. Additionally, there must be a certification authority, called the AKD, that is contacted to obtain the public key of a host so that replies can be authenticated by verifying the appended signature. A drawback of this scheme is that the AKD constitutes a single point of failure in the network.

TARP [15] implements security by distributing centrally issued secure <IP, MAC> address mapping attestations (called tickets) through existing ARP messages. TARP is backward compatible with ARP, but as S-ARP, is susceptible to replay attacks during a small window of vulnerability.

Goyal et al. [16] proposed a new architecture for secure address resolution. Their system is based on a Merkle hash tree, a trusted node on the network (TN) and a broadcast authentication protocol (e.g. Tesla). This solution has the advantage that no symmetric/asymmetric cryptography operations are required. A major drawback of this solution is that it is not backward compatible with ARP and it can be very inefficient in highly dynamic networks. Goyal et al. [17] also proposed a modification to S-ARP based on the combination of digital signatures and one time passwords based on hash chains to authenticate ARP <IP, MAC> mappings. Their scheme is based on the same architecture as S-ARP, but its clever use of cryptography allows it to be significantly faster.

Some high-end Cisco switches have a new feature called Dynamic ARP Inspection [18]. This feature allows the switch to drop ARP packets with invalid <IP,MAC> address bindings [10]. This scheme promises to be a very effective solution to the problem of ARP attacks, but thorough tests need to be performed to confirm if in fact it is able to prevent all types of ARP attacks.

IV. Stateful ARP cache and Fuzzy Logic Controller based Prevention Mechanism

The following sections describe a mechanism based on a stateful ARP cache and a Fuzzy Logic controller to prevent malicious hosts from corrupting other hosts' ARP caches with fake entries.

We assume that the hosts in the network are mainly: servers (Web, FTP, etc.), routers, printers, and computers. In addition, we assume that the hosts in the network have different level of trustiness and importance. The trustiness level of a host depends on its potential to perform attacks on other hosts in the network. Some hosts can potentially perform attacks (Common network users for examples), others are unlikely to generate any attack (Network routers and servers for examples), and finally others are between. The importance level of a given host A to a host B depends on the percentage of the number of packets exchanged between them compared to the total number of packets that host B exchanged within a period of time.

A. Stateful ARP cache

The proposed prevention mechanism is based on the use of a stateful ARP cache. When host A generates an ARP request to get the MAC address of host B, an entry is created in its stateful ARP cache, with the status of "Waiting". Host A waits for an ARP reply, within a predefined timeout. If an ARP reply comes, then host A waits another timeout in order to collect other possible ARP replies sent by other hosts in the network. Note that if host A receives more that one ARP reply, then this means that most likely more than one host has replied. Therefore, among those hosts, only one host is an honest host, which is host B. The others are probably malicious hosts, performing ARP cache poisoning attack to corrupt the ARP cache of host A.

The main differences between the current stateless ARP cache and the proposed stateful ARP cache are:

1) When a host receives an ARP reply, the current stateless ARP cache will update the corresponding entry if it exists already in the ARP cache. However, the stateful ARP cache will not update the corresponding entry unless an ARP request has been generated before for that entry, even if the entry exists already in the cache.
2) The stateful ARP cache will not update its entries using ARP requests. It is important to mention that all the tested OSs update their ARP caches once they receive ARP requests [2]. By doing this, ARP cache will be better protected from ARP cache poisoning attack

B. Types of possible ARP replies

The types of the possible ARP replies that host A may receive depends on the type of the attack (MiM, DoS, or Cloning) that the malicious host intends to perform on host A.

If within a timeout, host A receives only one ARP reply, then we can assume that host B has generated the ARP reply, which does not include fake IP and MAC addresses. In this case, host A updates its ARP cache, and changes the status of the entry corresponding to the host B's IP address to "*Resolved*". The content of the non fake ARP reply packet generated by host B is shown in figure 1.

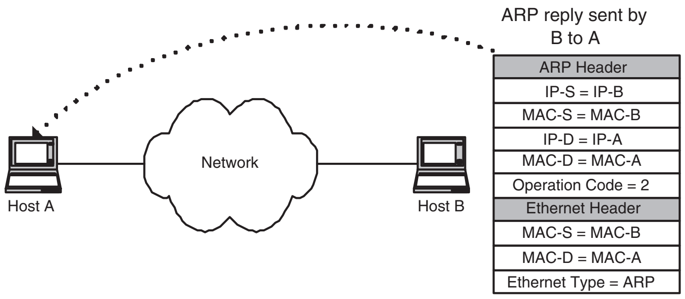


Fig. 1. Host B sends a legal ARP reply to host A via the network

If within the timeout, host A receives more than one ARP reply, then we can assume that one packet came from host B and the remaining packets came from malicious hosts. We assume that host C is a malicious host. Depending on the type of the attack host C intends to perform on host A, the following is the contents of the possible ARP reply packets that host C may generate:

1) In case of a *MiM* attack, two possible ARP reply packets can be generated by host C. Figure 2(a) presents a case where host C inserts its MAC address as the source MAC address. Figure 2(b) presents the case where host C hides its MAC address in order to avoid any potential detection, and inserts a fake MAC address.
2) In case of a DoS attack, three possible ARP reply packets can be generated by host C as shown in figure 3. Figure 3(a) shows host C inserting its MAC address as the source MAC address. Figure 3(b) shows host C hiding its MAC address in order to avoid any potential detection, and inserting a fake MAC address (x). The source MAC address (x) in the Ethernet header is similar to the source MAC address in the ARP header (x). Figure 3(c) shows how host C hides its MAC address in order to avoid any potential detection, and inserts a fake MAC address (x). The source MAC address (x) in the Ethernet header is not similar to the source MAC address in the ARP header (y).
3) In case of a Cloning attack, two identical ARP reply packets are generated, one by host B and the other by host C. The contents of the two packets are the same (figure 1).

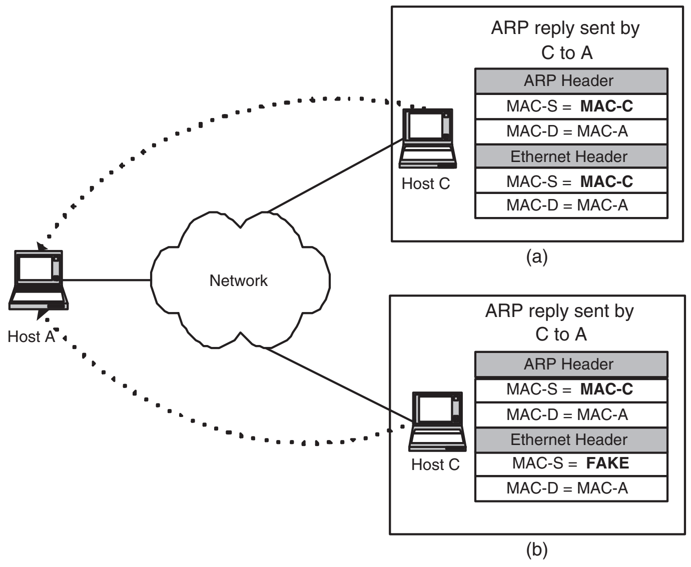


Fig. 2. Host *C* performing a MiM attack against host *A*. (a) and (b) present two strategies for the attack

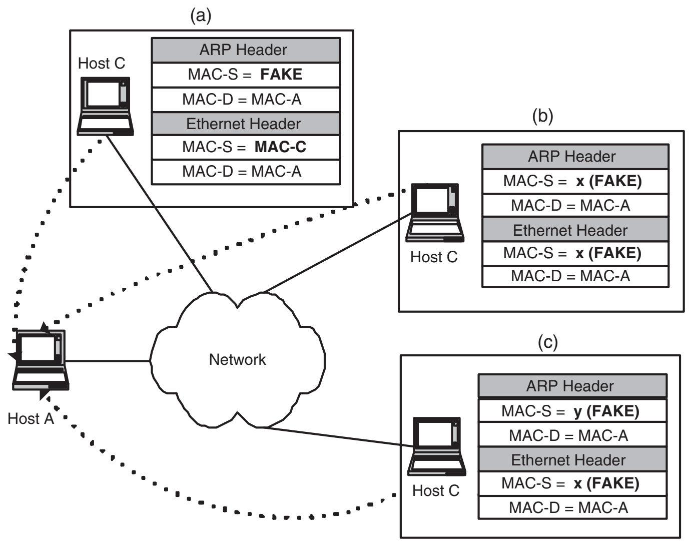


Fig. 3. Host *C* performing a DoS attack against host *A*. (a), (b), and (c) present three different strategies for the attack

C. Prevention mechanism

Depending on the nature of the received ARP reply packets, host A uses a mechanism based on Fuzzy Logic, to select the most trusted packet, among the received ones, that will be used to update host A's ARP cache.

1) Cross Layer Controller: The current ARP protocol implementation does not make any cross layer control between the ARP layer and the Ethernet layer to verify whether or not the source MAC addresses in these 2 layers are similar. An extension of the implementation of the ARP protocol would proceed to do this verification before accepting any ARP reply packet. Any ARP packet, whose source MAC addresses in the Ethernet layer is different from the source MAC addresses in the ARP header, is considered a fake packet and must be discarded.

Consequently, the packets shown in figures 2(b), 3(a), and 3(c) will be discarded by host A since the source MAC address in the Ethernet header is different from the source MAC address in the ARP header in those packets.

- In case of a MiM attack, the packets shown in figures 2(a) or 2(b) may be received by host A, in addition to the non fake ARP reply packet received from host B (figure 1). However, the cross layer controller will discard the packet shown in 2(b). Only the packet shown in 2(a) is accepted by host A for further processing. Therefore, host A can accept the packets shown in figures 1 and 2(a).

- In case of a DoS attack, the packets shown in figures 3(a), 3(b), and 3(c) may be received by host A, in addition to the non fake ARP reply packet received from host B (figure 1). However, the cross layer controller will discard the packets shown in figures 3(a) and 3(c). Only the packet shown in figure 3(b) is accepted by host A for further processing. Therefore, host A can accept the packets shown in figures 1 and 3(b).

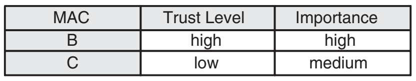
- If host A receives two similar ARP reply packets that look like coming from host B. Therefore, it is most likely that a malicious host is performing the cloning attack. Consequently, the two packets will be ignored and host A will not update its ARP cache.

Based on the discussion presented above, host A has the potential to accept the normal packet or the malicious packet. In the following sections, we present a novel Fuzzy Logic approach that makes a decision on which packet to choose. With a very high percentage, the controller will accept the honest packet and reject the malicious one.

2) Fuzzy logic Control: During the network setup time, each host in the network collects some information about the other hosts it is communicating with. Based on the scenarios we discussed in section IV-B, host *A* collects some information about hosts *B* and *C*. This information include 2 numerical values describing the Trust Level (*TL*) and Importance (*Im*) of each host. *TL* indicates the trust level of the host for example, highly trusted or not trusted at all. *Im* indicates the importance of the host for example, a laptop is less important than an internal server and an internal server is less important than a router that connects hosts to the internet.

The collected information is stored locally on host A. Later on, it will be used to classify certain hosts as attackers or honests. Figure 4 shows an example of a database populated on host *A*. It says, for instance, that host B has a high trust level with high importance i.e. it might be a secure router. Note that, when we mention the word host, we mean the

MAC address which identifies that host.



MAC	Trust Level	Importance
B	high	high
C	low	medium

Fig. 4. Database stored in host *A* indicating the trust level and importance of hosts *B* and *C*

It is important to note that the values of *TL* and *Im* are completely *dynamic* and they *adapt* to the network changes. Later in this section, we explain in details how these values are obtained and maintained.

Going back to the scenarios discussed in section IV-B, host *A* might receive multiple different ARP replies for a single ARP request. Therefore, one of the hosts sending the ARP replies is an attacker (either *B* or *C*). To decide which host is the attacker, we designed a Fuzzy Logic controller that considers a single host and combines its *TL* and *Im* values to produce a single value. So the Fuzzy Logic controller running on host *A* assigns one value for host *B* and another for host *C*. The host having the lowest value is, with a high percentage, the attacker and its packet is dropped. [19] gives a good overview on designing Fuzzy Logic controllers and their advantages.

The designed Fuzzy Logic controller aggregates *TL* and *Im* keeping in mind the synergy between them. It produces an output that indicates the security level (*SL*) of the host. The higher *SL*, the more secure the host is. Since the controller accepts 2 values as input and produces one value as output, it is composed of three membership functions. Importance Membership Function (*Im*) is represented by 3 triangular membership functions. *Im* can be any value between 0 and 1. The higher the value of *Im*, the more important the host is. Note that, in this context Importance and Availability can mean the same thing. Trust Level Membership Function (*TL*) is represented by 5 Gaussian membership functions. *TL* can be any value between 0 and 1. The higher the value of *TL*, the more trusted the host is. The value of *TL* is dynamic i.e. it increases or decreases based on whether the host attempts to conduct any malicious activity (discussed later). Security Level Membership Function (*SL*) is the output of the controller and it is represented by 3 trapezoidal membership functions. Hosts with low *SL* are considered to be potential attackers.

SL is obtained as a result of some delicate interaction between the *TL* and *Im* membership functions. This interaction is based on a set of Fuzzy Logic rules that present the heart of the controller. Table I presents the nine if-then rules used to combine *Im* and *TL* keeping in mind the synergy between them. Each host in the network uses the fuzzy logic controller to evaluate the level of security that other hosts possess. Figure 5 presents the surface associated with the fuzzy system we just described. It presents the output (*SL*) of the Fuzzy Logic controller given any values of the input variables (*TL* and *Im*). More details on the controller design can be found in [20].

TABLE I

FUZZY LOGIC CONTROLLER RULES

Rule Number	Trust Level	Importance	Security Level
1	very low		bad
2	low	high	acceptable
3	low	*not* high	bad
4	medium	low	bad
5	medium	medium	acceptable
6	medium	high	good
7	high	low	acceptable
8	high	*not* low	good
9	very high		good

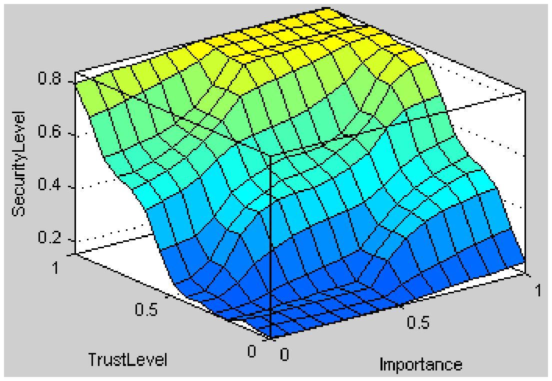


Fig. 5. Surface associated with the Security controller

When host *A* suspects that it is under attack, it consults the Fuzzy Logic controller to evaluate the security level of the suspected hosts. Host A will then be able to make a decision on which host to trust; it trusts the host having the highest *SL*. The major issue that remains is how to assign the values of *TL* and *Im* i.e. how to fill the table shown in figure 4.

3) Database Population: When the network setup starts, each host in the network populates a table like the one shown in figure 4. Assume that host *A* wants to populate its own table. When host *A* connects to the network, it starts keeping track of the total number of packets sent from its network interface card (NIC), let this number be *T*. Host *A* also keeps track of the number of packets sent to individual hosts, let this number be $T_{hostname}$. For example, let the total number of packets sent by host *A* be $T = 10000$, let the number of packets sent from host *A* to host *B* be $T_B = 8000$, and let the number of packets sent from host *A* to host *C* be $T_C = 2000$. With respect to host *A*, the importance (*Im*) of host B is calculated by dividing T_B over T, i.e. $Im_B = \frac{T_B}{T}$. Same for host C; $Im_C = \frac{T_C}{T}$. In general Im is calculated according to the following formula:

$$Im = \frac{T}{T_{hostname}} \tag{1}$$

Our logic behind using the percentage of communication to reflect the host importance is derived from the fact that more communication is usually done with the crucial hosts in the network such as routers, servers, etc. The process of collecting such information is performed on regular bases. When host *A* wants to decide on whether a certain host is malicious, it only uses the latest statistics; for example, those done in the last 5 minutes of the communication.

MAC	Trust Level (TL)	Importance (Im)
B	0.5	0.8
C	0.5	0.2

Fig. 6. An example of the database stored in host *A* indicating the trust level and importance of hosts *B* and *C*

When deciding on a value for the trust level (*TL*), each host in the network populates its table by assigning an initial fixed value of 0.5 to all hosts it is communicating with. So, host A sets $TL_B = 0.5$ and $TL_C = 0.5$. Figure 6 shows an example of the database stored on host *A*. The value of *TL* is dynamic and adaptive. It has the potential to *increases linearly* and *decreases exponentially*. If a certain host discovers that another host is an attacker, it decreases the attacker's trust level exponentially. Otherwise, it increases it linearly. Figure 7 shows the pseudocode of *TL* increase and decrease.

```
Linear increase

loop
      TL = TL + 1
until TL >= 10
```

```
Exponential decrease

dif = 1
loop
      TL = TL - dif
      dif = dif * 2
until TL <= 0
```

Fig. 7. Pseudocode for the linear increase and exponential decrease of the host's trust level

The following example discusses the scenarios presented in section IV-B where host *A* wants to decide on whether to accept packets from host *B* or host *C*. Host *A* uses the values in its database (figure 6) as input to the Fuzzy Logic controller. The controller returns $SL_B = 0.66$ and $SL_C = 0.34$, so host *A* decides to consider host *C* as an attacker. It then decreases the trust level of *C* to *0.4*. If C was caught again doing malicious activities, its trust level will decrease to *0.2*. It decreases exponentially until it becomes less than 0 and as a result host *C* will be denied any communication with host *A*.

V. Conclusion

In this paper we proposed a solution to the ARP poisoning problem by extending the existing ARP protocol. The new extension includes (1) statefull ARP cache, (2) Fuzzy Logic controller, (3) cross layer design, and (4) adaptive database manipulation. The statefull ARP cache assists in the decision making of updating the ARP cache, especially when some updates intend to poison the cache. The Fuzzy Logic controller is used to aggregate some important hosts' properties in order to detect malicious hosts. The cross layer design is used as a very fast way to check some of the naive ARP attacks. And the adaptive database is mainly used to keep the decision making dynamic and up to date. We are now in the process of technically testing the approaches discussed in this paper. We expect the results to be very promising where many ARP attacks will be caught and stopped.

References

[1] David C. Plummer. An ethernet address resolution protocol-converting network protocol to 48 bit ethernet address for transmission on ethernet hardware. RFC-826, November 1982.

[2] Zouheir Trabelsi and Khaled Shuaib. Man-in-the-middle intrusion detection. In *IEEE GlobeCom - Network Security Systems Symposium, (IEEE GlobeCom 2006)*, San Franscisco, California, USA, 27 November - 1 December 2006.

[3] Arpwatch. ftp://ftp.ee.lbl.gov/arpwatch.tar.gz.

[4] Snort Project. Snort:the open source network intrusion detection system. http://www.snort.org/.

[5] T. Demuth and A. Leitner. Arp spoofing and poisoning: Traffic tricks. *Linux Magazine*, 56:26 – 31, July 2005.

[6] M. Carnut and J. Gondim. Arp spoofing detection on switched ethernet networks: A feasibility study. In *Proceedings of the 5th Simposio Seguranca em Informatica*, November 2003.

[7] Arp-guard. http://arp-guard.com.

[8] T. Bradley, C. Brown, and A. Malis. Inverse address resolution protocol. RFC 2390, September 1998.

[9] S. Whalen. An introduction to arp spoofing. *2600: The Hacker Quarterly*, 18(3), Fall 2001. http://www.node99.org/projects/arpspoof/arpspoof.pdf.

[10] C. Schluting. Configure your catalyst for a more secure layer 2, January 2005. http://www.enterprisenetworkingplanet.com/netsecur/article.php/3462211.

[11] M. V. Tripunitara and P. Dutta. A middleware approach to asynchronous and backward compatible detection and prevention of arp cache poisoning. In *Proc. 15th Annual Computer Security Application Conference (ACSAC)*, pages 303–309, 1999.

[12] M. Barnaba. Anticap, 2003. http://cvs.antifork.org/cvsweb.cgi/anticap.

[13] Mohamed G. Gouda and Chin-Tser Huang. A secure address resolution protocol. *The International Journal of Computer and Telecommunications Networking*, 41(1):57 – 71, January 2003.

[14] D. Bruschi, A. Ornaghi, and E. Rosti. S-arp: a secure address resolution protocol. In *19th Annual Computer Security Applications Conference (ACSAC '03)*, page 66, 2003.

[15] W. Lootah, W. Enck, and P. McDaniel. Tarp: Ticket-based address resolution protocol. In *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC '05)*, December 2005.

[16] V. Goyal, V. Kumar, , and M. Singh. A new architecture for address resolution, 2005. http://www.itbhu.ac.in/departments/comp/crypto/mayank.htm (unpublished).

[17] V. Goyal and A. Abraham. An efficient solution to the arp cache poisoning problem. In *Proceedings of the 10th Australasian Conference on Information Security and Privacy (ACISP '05), published in Lecture Notes in Computer Science (LNCS 3574)*, pages 40–51, July 2005.

[18] Cisco Systems. Configuring dynamic arp inspection. Catalyst 6500 Series Switch Cisco IOS Sofware Configuration Guide, Release 12.2SX, 2006.

[19] Wasim El-Hajj. *A Distributed Hierarchical Energy-Efficient Scheme for Large Scale Mobile Wireless Ad Hoc Networks*. PhD thesis, Western Michigan University, 2006.

[20] Wassim El-Hajj and Zouheir Trabelsi. Using a fuzzy logic controller to thwart data link layer attacks in ethernet networks. In *Proceedings of IEEE Wireless Communication and Networking Conference (IEEE WCNC'07)*, Hong Kong, China, March 2007.