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

Using a Fuzzy Logic Controller to Thwart Data Link Layer Attacks in Ethernet Networks

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

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

***Abstract*— [1] Nowadays data networks represent the most common communication environment for transfer of data, voice or image. Such popularity led network users to becoming more vulnerable to network attacks and intrusions. Data link layer attacks, ex. ARP poisoning, is considered to be one of these dangerous attacks. ARP poisoning attack is a technique used to attack an Ethernet network. It may allow an attacker to sniff network traffic or stop the traffic altogether. In this paper, we use a Fuzzy Logic controller to thwart Data Link layer attacks in ethernet networks (ARP poisoning). Each host in the network is assigned certain dynamic characteristics. Then a Fuzzy Logic controller is used to combine these characteristics keeping in mind the synergy between them. The output of the controller decides if the host is trusted or not. Moreover, we use a stateful ARP cache, instead of the traditional stateless ARP cache.**

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

I. Introduction

Network attacks, comprised of different techniques exploiting the vulnerabilities of the Internet in general and the TCP/IP/ARP protocols in particular, steadily continue to grow in sophistication, using a very large set of tools and techniques. At the Data Link layer, most of these attacks rely on the use of the ARP poisoning attack. This attack consists of maliciously modifying the association between an IP address and its corresponding MAC address in the ARP cache.

In this paper we propose a solution to the ARP poisoning problem based on extending the ARP protocol. We propose a stateful ARP cache instead of using the actual stateless ARP cache. In addition, when a host receives more than one ARP replies for a given ARP request, we propose a new Fuzzy logic approach to identify the correct ARP reply before updating the corresponding ARP cache.

The rest of the paper is organized as following: Section II provides an overview of the related work done in this area. Section III discusses common attacks based on ARP poisoning. Section IV discusses the proposed prevention mechanism. Section V discusses the Fuzzy logic controller. Section VI concludes the paper and presents future research directions.

II. Related Work

A possible technique to protect an ARP cache from the ARP cache poisoning attack is to use static entries in the ARP cache. That is, the entries in the ARP cache cannot be updated by ARP request and reply packets, and do not expire. But, this can provide a wrong believe of security under some operating systems (OSs), such as Windows 2000 and Sun Solaris 5.9. In fact, those OSs marks static entries in their ARP caches, but authorize their updates by ARP request and reply packets.

Port security is another mechanism for tackling the problem. This is often suggested as an effective protection against ARP poisoning, but it is not. If the attacker does not spoof its own MAC address, it can poison the two victims' caches without letting the switch interfere with the poisoning process.

Some kernel patches exist that try to defend against ARP poisoning. "Anticap" [2] does not update the ARP cache when an ARP reply carries a different MAC address for a given IP from then one already in cache and will issue a kernel alert that someone is trying to poison the ARP cache. Such a solution is against ARP definition itself, since it drops legal gratuitous ARP. All these solutions [2], [3], [4] have the same problem. If the malicious ARP reply is sent before the real one is put in the cache, for a real request, the victim caches the wrong reply and discards the real one. A race condition exists between the attacker and the victim. When the first ARP request is broadcast, both the victim and the attacker receive the message. The first one who replies will take over the other forever.

Solutions such as a centralized ARP cache, a DHCP server or a secure server [5] broadcasting ARP information, as they are deployed in IP over ATM networks [6], have not been considered as the attacker could spoof the source of the broadcast and poison the whole LAN. In [5], a secure server is connected to the Ethernet and two protocols are used: an invite-accept protocol and a request-reply protocol. Each computer connected to the Ethernet can use the invite-accept protocol to periodically record its IP address and its hardware address in the database of the secure server. A race condition exists between the attacker and the server. The attacker may impersonate the server activities and send false ARP replies.

In [7], a detection technique has been proposed based on the use of trap ICMP ping packets. The technique detects only the hosts performing MiM attacks. The drawback of the technique is that it uses the same techniques used by malicious hosts to redirect their victim traffic. That is, the corruption the ARP caches of the suspected hosts. In addition, the technique

injects considerable additional network traffic while looking for malicious hosts.

In [8], a secure version of the ARP protocol that provides protection against ARP poisoning is proposed. Each host has public/private key pair certified by a local trusted party on the LAN, which acts as a Certification Authority. While this version of the ARP protocol solves the problem of the authentication of the ARP packets, but its implementation and management is not a practical task mainly in large networks. In addition, solutions based on public/private keys add a big overhead to the network specially because encryption using public/private keys is known to be very slow and the key size is very large.

A number of research work proposed protection mechanisms against the MiM attack, but at the application layer [9], [10], [11], [12], [13], [14]. However, our work in this paper is concerned with the detection of the MiM attack at the Data link layer.

III. Networks attacks based on fake ARP packets

This section describes the well-known networks attacks based on the use of fake ARP packets. There are three main attacks, namely: *MiM* attack, *DoS* attack, and cloning attack. All these attacks are based on the corruption of the ARP caches of the target hosts.

A. Corruption of ARP Caches

The corruption of the ARP caches (called ARP cache poisoning attack) is the malicious act, by a host in a LAN, of introducing a spurious IP address to MAC address mapping in another host's ARP cache. This can be done by manipulating directly the ARP cache of a target host, independently of the ARP messages sent by the target host. To do that, the malicious host can either add a new fake entry in the target host's ARP cache, or update an already existing entry by fake IP and MAC addresses.

In [7], an experiment has been conducted on a number of OSs. The experiment showed that when using ARP reply messages, the ARP cache poisoning attack becomes difficult to realize against most OSs. However, it remains indeed possible when using ARP request messages.

B. Man-in-the-Middle and Denial of Service Attacks in a Switched LAN Network

In a switched LAN network, all packets sent by a host will be received only by the destination host, unless it is a broadcast packet.

Using fake ARP messages, there are several ways that sniffing can be performed in a switched network, a MiM attack is the most common one. This attack consists of re-routing (redirecting) the network traffic between two target hosts to a malicious host. Then, the malicious host will forward the received packets to the original destination, so that the communication between the two target hosts will not be interrupted and the two hosts' users will not notice that their traffic is being sniffed by a malicious user.

In this kind of an attack, the malicious user first enables his host's IP packet routing, in order to become a router and be able to forward the redirected packets. Then, using an ARP cache poisoning attack, the malicious user corrupts the ARP caches of the two target hosts, in order to force the two hosts to forward all their packets to him. It is important to notice that if the malicious host corrupts the ARP caches of the two target hosts without enabling its IP packet routing, then the two hosts will not be able to exchange packets and it will be a DoS attack. For more information on how MiM attack is performed, you can refer to [7].

C. Cloning Attack

In this attack, the malicious host changes its IP and MAC to become the same as that of the victim host. It is believed that only IP address can be changed in a given host. However, actually, with tools like Mac Changer, any user can change its MAC address. Once this change is done, there will be two hosts in the network with the same IP and MAC addresses. For Wireless and switched LAN networks, this situation will cause some network disconnection troubles and a DoS situation.

IV. Prevention Mechanism

The following sections describe a mechanism based on a stateful ARP cache and Fuzzy Logic that prevents any malicious host that is using fake ARP packets to perform MiM attack or DoS attack. We assume that the hosts in the network are mainly: servers (Web, FTP, etc.), routers, printers, personal computer, and laptops. In addition, we assume that the hosts in the network have different level of trustiness. That is, some hosts can potentially perform attacks, others are unlikely to perform any attack, and finally others are in between.

A. Stateful ARP cache

To prevent ARP attacks, the ARP cache is implemented so that it is a stateful cache. The main differences between the current stateless ARP cache and the proposed stateful ARP cache are:

1) If a host receives an ARP reply, then 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 and ARP Gratious. It is important to mention that all the tested OSs update their ARP caches once they receive ARP requests.

B. Stateful ARP cache updates

If a host generates an ARP request, then an entry in the stateful ARP cache is created, with the status of "Waiting". For example, if host A sends a broadcast ARP request looking for the MAC address of host B, figure 1 shows the entry created in the stateful ARP cache of host A.

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

IP address	MAC address	Status
IP_B	*00:00:00:00:00:00*	*Waiting*

Fig. 1. Example of an entry in the stateful ARP cache

in the network. It is important to notice that if host A receives more that one ARP reply, it means that more than one host have replied to the ARP request sent by host A. Therefore, among the hosts that generated the ARP replies, only one host is an honest host, which is host B in our example. The others are probably malicious hosts, using fake ARP replies to corrupt the ARP cache of host A.

C. Detection mechanism

Let's assume that host A has generated an ARP request looking for the MAC address of host B. Two possible situations may occur: (1) If within the timeout, host A receives only one ARP reply, then host A updates its cache (figure 2(a)). (2) If within the timeout, host A receives more than one ARP reply, then there are malicious hosts that generated fake ARP replies (figure 2(b)). Hence, host A should identify which ARP reply is coming from host B, before any update of host A's ARP cache occurs. The following section describes a mechanism used to identify the most trusted ARP reply that will be used to update the ARP cache. However, first we introduce the different types of ARP replies that host A may receive.

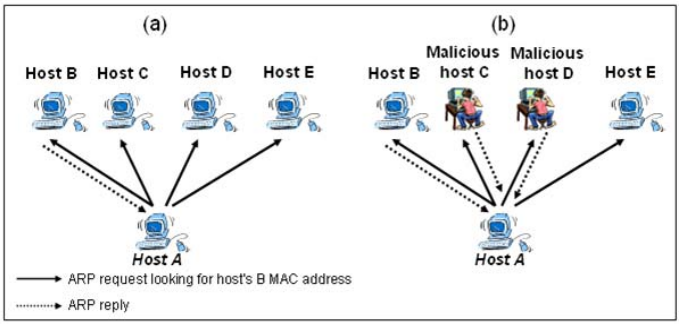

Fig. 2. Possible sources of the ARP replies

1) Types of the ARP replies: The types of the possible ARP replies that host A may receive from host B and the malicious hosts are:

- Type 1 is an ARP reply that host B will generate once a broadcast ARP request is received. Such an ARP reply does not include fake IP and MAC addresses, and provides host A with the correct MAC address of host B.
- Type 2 is a fake ARP reply generated by a malicious host to do MiM attack. For example, if host C is such a malicious host, then the contents of the fake ARP reply fields are shown in figure 3.

  By generating such a fake ARP reply, the malicious host C wants to tell host A, that the MAC address of host B is the MAC address of host C. Therefore, all the traffic sent by the host A to the host B will go first to the host C. Then, host C forwards the received traffic to the real destination host, which is the host B.
- Type 3 is a fake ARP reply generated by a malicious host to do DoS attack. For example if host D is such a malicious host, then the contents of the fake ARP reply fields are shown in figure 4.

Ethernet header
Source MAC address = **Any source MAC**
Destination MAC address = **Host A's MAC**
Ethernet Type (=0x0806 for ARP message)
ARP message header
Source IP address = **Host B's IP**
Source MAC address = **Host C's MAC**
Destination IP address = **Host A's IP**
Destination MAC address = **Host A's MAC**
Operation code: **2 (for ARP reply)**

Fig. 3. ARP reply from malicious host C

Ethernet header
Source MAC address = **Any source MAC**
Destination MAC address = **Host A's MAC**
Ethernet Type (=0x0806 for ARP message)
ARP message header
Source IP address = **Host B's IP**
Source MAC address = **Nonexistent/fake MAC**
Destination IP address = **Host A's IP**
Destination MAC address = **Host A's MAC**
Operation code: **2 (for ARP reply)**

Fig. 4. ARP reply from malicious host D

By generating such a fake ARP reply, the malicious host D wants to tell to host A, that the MAC address of host B is a nonexistent/fake MAC address. Therefore, all the traffic sent by the host A to the host B will go to a nonexistent host; so that host A will not be able to communicate with host B. This is a DoS attack, since host A is denied from communicating with host B.

2) Detection process: The following sections describe the detection process used to detect which ARP reply came from the honest host. To cover all possible cases, we assume that host A receives three ARP replies of *Type 1, Type 2 and Type 3*, from host B, host C, and host D, respectively. In addition, we assume that host C wants to perform MiM attack, and host D wants to perform DoS attack.

In the ARP header of each received ARP reply, there is the MAC address of the host that sent the ARP reply, which is supposed to be the MAC address of host B. However, in the previous section, we showed that that MAC address can be one of the following three addresses: MAC address of host B, a nonexistent/fake MAC address, or MAC address of a malicious host (for example: host C). This depends on the type of attacks that the malicious host wants to perform.

For each received ARP reply, host A will generate a unicast ARP request, looking for the MAC address of host B. Since host A received three ARP replies, then three unicast ARP requests will be sent to host B, host C and a nonexistent host:

Unicast ARP request sent to host B: Host A sends a unicast ARP request to host B looking for its MAC address. The MAC address of host B is extracted from the ARP reply received from host B. The contents of the fields of the unicast ARP request are shown in figure 5.

Ethernet header
Source MAC address = **Host A's MAC**
Destination MAC address = **Host B's MAC**
Ethernet Type (=0x0806 for ARP message)
ARP message header
Source IP address = **Host A's IP**
Source MAC address = **Host A's MAC**
Destination IP address = **Host B's IP**
Destination MAC address = **00:00:00:00:00:00**
Operation code: **1 (for ARP request)**

Fig. 5. Unicast ARP request sent to host B

Host B will reply by an ARP reply which includes its MAC address. This ARP reply will be used by host A to verify that the MAC address of host B in the received ARP reply is similar to the one in the first ARP reply received from host B. If they are similar, then the MAC address in the ARP reply will be classified as a possible MAC address of host B.

Unicast ARP request sent to host D: Host A sends a unicast ARP request to a nonexistent host looking for the MAC address of host B. The MAC address of the destination nonexistent host in the Ethernet header is extracted from the ARP reply received from host D. The contents of the fields of the unicast ARP request are the same as the ones shown in figure 5 except for the Destination MAC address field in the Ethernet header, which will be equal to "Nonexistent MAC address". No host will reply since the destination MAC in the Ethernet header of the unicast ARP request is a nonexistent MAC address. Host A will wait a timeout and then concludes that no host in the network has such a nonexistent MAC address. Consequently host B cannot have such a nonexistent MAC address. The received ARP reply from host D will be ignored.

Unicast ARP request sent to host C: Host A sends a unicast ARP request to host C looking for the MAC address of host B. The MAC address of the destination host in the Ethernet header is extracted from the ARP reply received from host C, which is the MAC address of host C. The contents of the fields of the unicast ARP request are the same as the ones shown in figure 5 except for the Destination MAC address field in the Ethernet header, which will be equal to "Host C's MAC address". Here, two cases may occur:

1) Host C will not generate any ARP reply, since the destination IP address in the ARP header of the unicast ARP request is the IP address of host B. Host A will wait a timeout and no ARP reply will come from host C. Then host A concludes that host C is a malicious host that attempting to perform MiM attack on host A.
2) Host C will generate a fake ARP reply which looks like it has been generated by host B. By this ARP reply, host C is still attempting to mislead host A by telling that host B has as MAC address the MAC address of host C. In such a case, host A will assume that host C can be the real host, and consequently the MAC address in the ARP reply (which is the MAC address of host C) will be classified as a possible MAC address of host B.

When host A identifies more than one possible MAC address for host B, a fuzzy logic controller is used to identify which MAC address is probably the real MAC address of host B. The selected MAC address will be used later by host A to update its ARP cache.

V. Fuzzy Logic controller

Recall from section IV that we are trying to detect/prevent MiM and DoS attacks on a switched network by preventing any attempt to update the ARP cache with fake *IP/MAC* entries. Section IV provided us with a list of MAC addresses, one of which is the real MAC address of host *B*. For example, the list might include the MACs of $\{B, C, E\}$. But the question remains, which MAC is the real MAC address of host B i.e. whom shell *A* trust: *B*, *C*, or *E*?

Each host in the system creates a small database in order to store some information about the hosts it communicates with. For example if host *A* can communicate with hosts $\{B, C, E\}$, *A* stores in its database two values for each host. The two values that correspond to a certain host describe the security level of that host. The two values are: Trust Level (*TL*) and Importance (Im). *TL* indicates the trust level of the system for example, highly trusted or not trusted at all. *Im* indicates the importance of the system 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. Figure 6 shows an example of the database that host *A* creates. It says, for instance, that host B has a low trust level with high importance i.e. it might be an attacker, but it is a router that connects the LAN to the Internet. Note that, when we mention the word host, we mean the MAC address which identifies that host.

MAC	Trust Level	Importance
B	low	high
C	high	high
E	normal	low

Fig. 6. Database stored in host *A* indicating the trust level and importance of hosts *B*, *C*, and *E*

Our strategy is to combine these two values in order to decide whether the host is a trusted host or not. For example, if host *A* detects that either *B* or *C* is an attacker, it can check the combined value of *TL* and *Im* of each host and then decide on which one to trust. Because these parameters have different

units and their values can be defined in ranges, we use fuzzy logic to express the effect of their interaction. In this section, we explain the fuzzy logic controller in details. [15] gives a good overview of fuzzy logic and its uses.

A. Security Controller

One of the major contributions of our mechanism is to decide whether a certain host is an attacker or not. A host is characterized by its trust level (*TL*) and importance (*Im*). A secure and trusted host is one with high *TL* and high *Im*. Otherwise, the host is less secure. We design a fuzzy logic controller to aggregate *TL* and *Im* keeping in mind the synergy between them. The controller produces an output that indicates the security level (*SL*) of the host. The higher *SL*, the more secure the host is. Next, we explain the membership functions and the rules that form the fuzzy logic controller in details.

1) Importance Membership Function: Im is represented by 3 triangular membership functions as shown in figure 7. The triangular membership function is specified by three parameters (a, b, c) as follows:

$$triangular(x : a, b, c) = \begin{cases} 0 & x < a \\ (x-a)/(b-a) & a \leq x \leq b \\ (c-x)/(c-b) & b \leq x \leq c \\ 0 & x > c \end{cases} \quad (1)$$

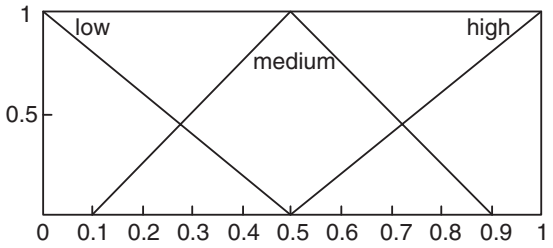

Fig. 7. Importance membership function

The x-axis represents the value of *Im*. *Im* can be any value between 0 and 1. The higher the value of *Im*, the more important the host is. For example, a laptop might have $Im = 0.2$, a server might have $Im = 0.5$, a router might have $Im = 0.8$, etc. Three triangular membership functions are used to represent *Im*: "low", "medium", and "high". If $0 < Im < 0.1$, the system is considered to have *low* importance. If $0.1 < Im < 0.5$, the system is considered to have *low* importance with a certain degree and *medium* importance with a certain degree. If $0.5 < Im < 0.9$, the system is considered to have *medium* importance with a certain degree and *high* importance with a certain degree. If $0.9 < Im < 1$, the system is considered to have *high* importance. Note that, in this context Importance and Availability can mean the same thing.

2) Trust Level Membership Function: TL is represented by 5 Gaussian membership functions as shown in figure 8. The Gaussian membership function depends on two parameters σ and c as given by:

$$gaussian(x : \sigma, c) = e^{\frac{-(x-c)^2}{2\sigma^2}} \quad (2)$$

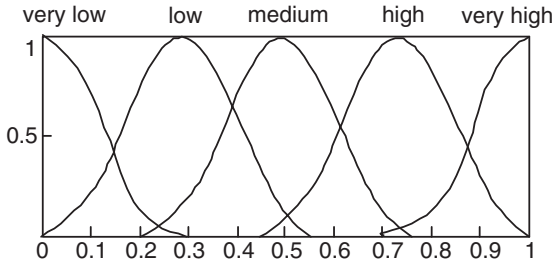

Fig. 8. Trust Level membership function

The x-axis represents the value of *TL*. *TL* can be any value between 0 and 1. The higher the value of *TL*, the more trusted the host is. Initially, the network administrator can assign a *TL* value for every host in the network. Otherwise, he can set $TL = 0.5$ to all hosts in the network; 0.5 is considered to be the default value. The value of TL is dynamic i.e. it decreases based on whether the host attempts to conduct any malicious activity. Five gaussian membership functions are used to represent *TL*: "very low", "low", "medium", "high", and "very high". Any value of *TL* matches two membership functions with a certain degree. If $0.8 < TL < 1$, the system is considered to have *high* trust with a certain degree and *very high* trust with a certain degree.

3) Security Level Membership Function: SL is the output of the controller and it is represented by 3 trapezoidal membership functions as shown in figure 9. A Trapezoidal membership function is specified by four parameters (a, b, c, d) as follows:

$$trapezoid(x : a, b, c, d) = \begin{cases} 0 & x < a \\ (x-a)/(b-a) & a \leq x \leq b \\ 1 & b \leq x \leq c \\ (d-x)/(d-c) & c \leq x \leq d \\ 0 & x \geq d \end{cases} \quad (3)$$

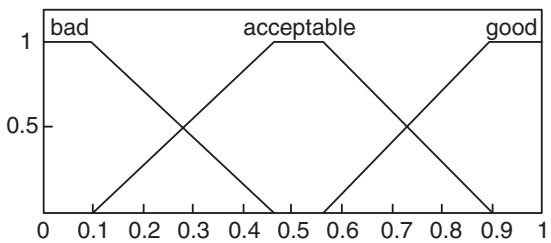

Fig. 9. Security Level membership function

The x-axis represents the value of *SL*. *SL* can be any value between 0 and 1. The higher the value of *SL*, the more secure the host is. Hosts with low *SL* are considered to be potential attackers. Three trapezoidal membership functions are used to

represent *SL*: "bad", "acceptable", and "good". For example, if $0 < SL < 0.1$, the host is considered to have *bad* security and it can be a potential attacker. A smooth transition is made between the various membership functions. The output of the fuzzy logic controller (*SL*) depends mainly on the rules that govern the system. The fuzzy logic rules are discussed next.

4) Fuzzy Logic Rules: The fuzzy logic rules present the heart of the controller. In general, the input to an if-then rule is the current value for the input variable (in this case, *Im* and *TL*) and the output is an entire fuzzy set (in this case, *SL*). This set will later be defuzzified, assigning one value to the output [16].

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

Table I presents nine if-then rules used to combine *Im* and *TL* keeping in mind the synergy between them. A certain input of *TL* and *Im* will be mapped to the membership functions {*very low, low, medium, high, very high*} and {*low, medium, high*}, respectively. Each host in the network uses the fuzzy logic controller to evaluate the level of security that other hosts possess.

Rule 1 says that if a host has a *very low* trust level, then its *security level* is *bad*. This means that the host might be an attacker and should not be trusted. Rule 2 says that if a host has a *low* trust level with *high* importance, then the *security level* of that host is *acceptable*. An example of such a scenario is a router having a *low* trust level. Since a router is a very important piece of a network, even though it might have a *low* trust level we do not cut the communication with it. We give it an *acceptable* security level. Rule 9 indicates that a host with *very high* trust level is secure no matter what the importance of that host is. The rest of the rules can be explained using similar reasoning. In general, as the trust level of a certain host increases, its security level increases as well. Figure 10 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*).

VI. CONCLUSION

In this paper we proposed a solution to the ARP poisoning problem by extending the ARP protocol. We use a Fuzzy Logic approach to prevent some dangerous Data Link layer attacks, such as the ARP poisoning attack. We start by using simple but intelligent techniques to catch naive or non-experienced intruders. Then we design a Fuzzy Logic controller that combines the various characteristics of the hosts. The output

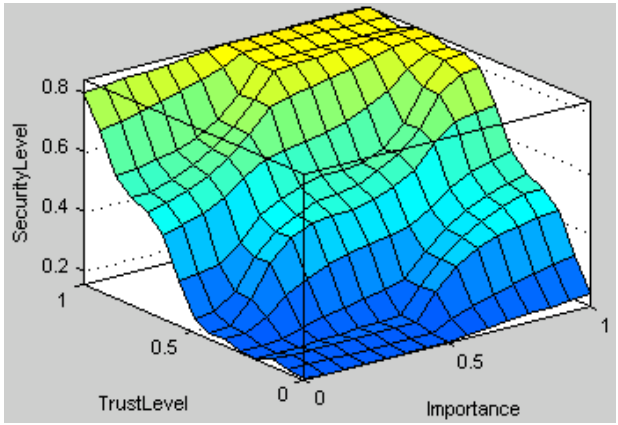

Fig. 10. Surface associated with the Security controller

of the controller decides whether the host is trusted or not. Moreover, we updated the way ARP cache is structured; we changed the ARP cache to become stateful ARP cache. We believe that our proposed approach will efficiently catch all malicious attempts to alter the ARP cache.

REFERENCES

[1] Wasim El-Hajj, D. Kountanis, A. Al-Fuqaha, and Hani Harbi. Optimal hierarchical energy efficient design for manets. In *International Wireless Communications and Mobile Computing Conference*, pages 287–292, Vancouver, British Columbia, Canada, July 2006.

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

[3] I. Teterin. Antidote. http://online.securityfocus.com/archive/1/299929.

[4] 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.

[5] 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.

[6] M. Laubach. Classical ip and arp over atm. RFC 1577, 1994.

[7] 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.

[8] 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.

[9] R. Oppliger, R. Hausser, and D. Basin. Ssl/tls session-aware user authentication - or how to effectively thwart the man-in-the-middle. *Computer Communications, Elsevier, Article in Press*, 2006.

[10] R. L. Rivest and A. Shamir. How to expose an eavesdropper. *Communications of the ACM*, 27(4):393–395, 1984.

[11] S. M. Bellovin and M. Merritt. An attack on the interlock protocol when used for authentication. *IEEE Transactions on Information Theory*, 40(1), 1994.

[12] M. Jakobsson and S. Myers. Stealth attacks and delayed password disclosure, 2005. http://www.informatics.indiana.edu/markus/stealth-attacks.htm/.

[13] B. Kaliski and M. Nystrom. Authentication: risk vs. readiness, challenges and solutions, October 6 2004. http://www.rsasecurity.com/rsalabs/staff/bios/bkaliski/publications/other/kaliski-authentication-risk-readi- ness-bits-2004.ppt./.

[14] N. Asokan, V. Niemi, and K. Nyberg. Man-in-the-middle in tunneled authentication protocols. In *Proceedings of the International Workshop on Security Protocols*, pages 15–24, 2003.

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

[16] MathWorks. Fuzzy logic toolbox.