An Automated approach for Preventing ARP Spoofing Attack using Static ARP Entries

Ahmed M.AbdelSalam
Information Technology Dept.
Faculty of Computers and Information, Menofia University
Menofia, Egypt

Wail S.Elkilani
Computer Systems Dept.
Faculty of Computers and Information, Ain Shams University
Cairo, Egypt

Khalid M.Amin
Information Technology Dept.
Faculty of Computers and Information, Menofia University
Menofia, Egypt

***Abstract*—ARP spoofing is the most dangerous attack that threats LANs, this attack comes from the way the ARP protocol works, since it is a stateless protocol. The ARP spoofing attack may be used to launch either denial of service (DoS) attacks or Man in the middle (MITM) attacks. Using static ARP entries is considered the most effective way to prevent ARP spoofing. Yet, ARP spoofing mitigation methods depending on static ARP have major drawbacks. In this paper, we propose a scalable technique to prevent ARP spoofing attacks, which automatically configures static ARP entries. Every host in the local network will have a protected non-spoofed ARP cache. The technique operates in both static and DHCP based addressing schemes, and Scalability of the technique allows protecting of a large number of users without any overhead on the administrator. Performance study of the technique has been conducted using a real network. The measurement results have shown that the client needs no more than one millisecond to register itself for a protected ARP cache. The results also shown that the server can a block any attacker in just few microsecond under heavy traffic.**

Keyword—component; layer two attacks; ARP spoofing; ARP cache poisoning; Static ARP entries

I. Introduction

The evolving of computer networks, and the variety of its services and applications, has increased the users need for LANs [1] and the security of LANs also become a more concern. An essential part of successful communication between users within LAN is the Address Resolution Protocol (ARP) [2]. ARP is specified in RFC 826 [3] to allow hosts to resolve network layer address (IP) to datalink layer address (MAC) [3]. Although the importance of ARP protocol for communication in LAN, it formulates the most dangerous attacks threating LANs. ARP spoofing or ARP cache poisoning are the two main attacks threating the ARP protocol operations [4].

ARP Spoofing is a hacking technique to send fake ARP request or ARP reply, ARP spoofing problem comes from the way the ARP protocol works [5]. Since the ARP protocol is a stateless protocol that receives and processes ARP replies without issuing ARP request [6], the ARP cache can be infected with records that contain wrong mappings of IP-MAC addresses. ARP spoofing can be used to launch one of two different attack categories [7]: Denial of Service (DoS) attacks or Man in the Middle (MITM) attacks

Several solutions have been proposed to mitigate the ARP spoofing, but each has its limitations [7]. The solutions have been classified into five different categories [8]:

- Modifying ARP using cryptographic techniques

These solutions add some cryptographic features to the ARP protocol, but will not be compatible with the standard ARP and affect the protocol performance.

- Kernel-based patching

The technique adds a patch to the operating system kernel in order to prevent ARP spoofing attacks, but the problem is that not all operating systems can be patched and it may become incompatible with the standard ARP protocol.

- Securing switch Ports

Use the switch port security or Dynamic ARP inspection (DAI) option to prevent ARP spoofing. However its ability of preventing ARP spoofing easily, the cost of implementing such solution may not be acceptable by most of the organizations.

- ARP spoof detection & protection software

Programs or tools developed to prevent ARP spoofing attacks, but the experimental results have shown there ineffectiveness in protection.

- Manually configuring static ARP entries

The most basic and effective way to prevent ARP spoofing [1] [6] [9] is adding static ARP entries at each host. However this solution cannot be easily managed and cannot scale well specially in organizations that have large number of users and require a heavy workload on the network administrator.

In this paper, a scalable technique to prevent ARP spoofing attacks, which automatically configures static ARP entries is proposed. It overcomes the problems of the solutions of the techniques that use static entries. The remaining part of the paper is organized as follows: Section II surveys background and related work. Section III shows the details of the proposed method. The experimental results are discussed in section IV. Finally section V concludes the paper.

II. Background and related work

A. Address resolution Protocol

ARP is specified in RFC 826 [3] as a protocol that provides dynamic mapping from an IP address to the corresponding MAC address to grant successful communication between users within LAN. ARP messages are classified as request and reply message. When a user has packet to transmit, it will send a broadcast ARP request asking about the MAC address for a certain IP. The machine, recognizing the IP address as its own address, returns an ARP reply containing its MAC address. The mapping will be saved in the device ARP cache [2].

B. *ARP cache*

ARP cache is a table of recently resolved IP addresses and their corresponding MAC addresses. The ARP cache is checked first before sending an ARP Request frame. ARP cache entries can be dynamic or static [4].

- ***Static ARP cache entries:*** are permanent and manually added records using a TCP-IP utility. Static ARP cache entries are used to provide ARP requests for commonly used local IP addresses. The problem with static ARP entries is that they have to be manually updated when network interface equipment changes [10].
- ***Dynamic ARP cache entries:*** are entries learned by ARP protocol and have a time-out value associated with them to remove entries from the cache after a specified period of time [10].

C. *ARP spoofing*

ARP is a stateless protocol that uses ARP replies to update ARP cache using wrong or spoofed mappings [11]. As mentioned before, ARP spoofing can be used to launch [7] either one of the following attack categories:

- ***Man in the Middle (MITM):*** An attacker deceives both ends of communication and fills their ARP cache with wrong IP-MAC mapping. As a matter of fact, it inserts itself between the two ends of communication. Hence, it will gain a copy of every bit sent between them.
- ***Denial of service (DoS):*** An attacker fills the ARP cache of victim with wrong IP-MAC mapping, so every packet sent from victim will be sent to the wrong MAC.

D. *Related Work*

As mentioned previously, solutions attempting to prevent ARP spoofing attack using the static ARP cache entries are very efficient. Yet this category of solutions has some major problems [7] [8]: (1) overhead required for manual configuration of static entries, (2) Limited scalability for large networks, and (3) Ability to work in static and DHCP based networks.

In the following, we will survey several methods belonging to this category along with their drawbacks.

The DAPS (Dynamic ARP spoof Protection System) technique suggested in [8] is a solution to ARP spoofing that snoops DHCP packets and use them as vaccines. Yet this technique doesn't scale well for those network that use static IP addressing scheme and also vaccines will be invalid if DHCP starvation attack occurs.

In [12], the NIDPS (Network Intrusion Detection and Prevention System) technique is suggested have a server collecting IP-MAC mappings from users using small agents. These mappings will be then used as static ARP entries to correct any wrong mapping detected. However, agents aren't authenticated to the server. Moreover, it detects only attacks from its LAN segment. Also, the server examines every packet going in or out the LAN segment. Finally, it waits for the attack to occur and then try to solve it.

Xiangning et al. [13] has proposed a technique that expands the snort preprocessors plug-ins by adding an ARP detection module. The proposed technique doesn't scale well in large networks due to the need of manual configuration of the static mappings at the server. It also doesn't work in DHCP based networks.

A solution to ARP spoofing using a server is proposed in [14]. The server will get mappings for the network users from the DHCP server. It replies also to ARP requests. Unfortunately, this solution works only in DHCP networks. Also, it is not compatible with the standard ARP. Moreover, if DHCP starvation occurs, all the server information will be invalid.

Ai-zeng Qian [15] proposed a technique to prevent ARP spoofing by using static ARP entries but the technique still doesn't work with dynamic networks using DHCP addressing. The administrator must assign all IP addresses along with their MAC to the server so it will be not visible for large scale network.

A method is suggested in [16] to solve ARP spoofing problem using snort IDS and static ARP entries. Yet, it still needs the administrator to add the static mappings manually. Also, it works only in static networks.

Table 1 compares the proposed algorithm with the previous solutions, the comparison criteria includes if it works in DHCP and static networks, ability to prevent attacks, scalability, and if manual or automatic configuration of the static entries is used.

TABLE I. COMPARISON BETWEEN THE PROPOSED ALOGORITHM AND PREVIOUS SOLUTIONS

Technique	***Static***	***DHCP***	***Prevention***	***Scalability***	***Automatic***
Proposed	✓	✓	✓	✓	✓
DAPS [7]		✓	✓		✓
NIDPS [12]	✓	✓		✓	✓
Xiangning [13]	✓		✓		
Ortega [14]		✓	✓		✓
Ai-zeng [16]	✓		✓		
Xiangdong [17]	✓		✓		

III. THE PROPOSED METHOD

The proposed technique is a client-server protocol that prevents ARP spoofing by automatically configuring static ARP entries. The protocol works in both static and DHCP networks.

Moreover, it can work in large-scale networks without any overhead on the administrator. In addition, the technique doesn't require special hardware to be deployed, as any host can work as ARP server.

The protocol proposed defines three different messages:

A. *Register Message**:** is a unicast message sent from the client to the server. It contains its IP and MAC address. Also it includes a hashed authentication key.*

B. *Update Message: is a broadcast notification message sent from the server to all users in the network indicating that a new user has entered the network. It also contains the IP and MAC address of that new user.*

C. *Register Response Message**:** is a unicast message sent from the server to the new user. It contains all static ARP entries of users successfully registered at the server.*

The protocol also defines two different entities:

*a) ARP Client**:** is a software installed on user's machines. It fulfills the following*

- Automatically get the IP and MAC address of the user and use them to send register message to the server.
- Receive update and register response messages from the server.
- Verify that update or register response messages received are coming from a trusted server.
- Use the IP and MAC pairs received in the update or register response message to add static ARP entries to the user ARP cache.

*b) ARP Server**:** is a server software that can be installed on any device in the network. It can also be installed on a dedicated server, and has the following functions:*

- Receive register messages from the ARP clients.
- Verify that the message is coming from a trusted user.
- Make use of the IP and MAC pairs encapsulated within the register message to create a list of trusted users in the network.
- Send broadcast update message to notify them that a new user has come to the network.
- Send register response message to the new users.
- Take the proper action regarding users who try to violate the protocol security rules.

The proposed protocol defines two different algorithms for the client and server in order to prevent the ARP spoofing attack

1) Client Algorithm

The client algorithm described in Algorithm 1 adds static entry for the server in the client ARP cache to avoid the rogue server threat. Furthermore, it obtains the user IP and MAC address automatically to make the user has no opportunity to send fake information to the server.

Algorithm 1: *ARP_Client*

```
Step 1: Add static ARP entry for the server.
Step 2: Automaitcally get user IP and MAC address
Step 3: Formulate the Register Message
Step 4: Send the register message to the server.
Step 5: Listen to updates from the server
Step 6: if message received from the server then
            Extract the source IP
            if  source IP = Server IP then
                Extract Key, IP, MAC
                if received key is correct then
                    if similar MAC in ARP Cache then
                        Delete this record
                    else
                        Add static ARP entry using extracted IP
                        and  MAC address
                        Return to step 5
                    end if
                else
                    Discard the message
                    Return to step 5
                end if
            else
                Discard the Message
                Return to step 5
            end if
        else
            Return to step 5
        end if
```

The algorithm checks the source IP address of the received message to be sure that it is coming from the trusted server. It only accepts the IP and MAC addresses encapsulated in the message if the key is correct.

In order to work in DHCP networks where IP and MAC mappings are frequently changing, the algorithm searches for the MAC encapsulated in the message. If matched map is found, it will be changed to the new mapping. Otherwise a new mapping will be added.

Finally if any of the conditions are not met, the algorithm will discard the message and return to listen for another message from the server.

2) Server Algorithm

The server algorithm, described in Algorithm 2, listens to incoming register messages from the clients, checks the hash code to be sure that the message is coming from a trusted host. Users are given only three trials to send the correct hash code. If it fails to send the correct hash code within the three trials, the server will block this user.

The blocking action depends on the addressing scheme being used, for DHCP networks, the MAC address of the user will be added to DHCP deny list. Hence, it will not be able to obtain IP configuration from the DHCP server again, for static networks, the server will prevent traffic from this user to reach the server by obstructing its IP address.

Algorithm 2: *ARP_Server*

```
Step 1: Add static entry for itself
Step 2: Listen to users Register Messages
Step 3: if register message received from user then
            if the hash code matched then
                if wrong trials less than 3 then
                    if similar MAC in ARP cache then
                            update this record
                    else
                            delete this record
                    end if
                    send update message to all users
                    send register response message to the new user
                    return to step 2
                else
                    discard the message
                    return to step 2
                end if
            else if it has wrong previous wrong trials then
                Increment wrong trials for that MAC
                if wrong trials equal 3 then
                    Add to DHCP deny list Or Block this IP
                end if
                Discard the message
                Return to step 2
            else
                Add to sucpicious list
                Discard the message
                Return to step 2
            end if
          else
            Return to step 2
        end if
```

If user wrong trials have reached three or more, its traffic will be discarded even if it gets the correct hash code, and the administrator is the only one who has the ability to remove it from the block list. It is to be noted that we block undesirable users on the network layer instead of the usual blocking criteria using TCP. This enables the protocol proposed to stop DoS attacks completely.

If the key is correct and the number of wrong trials doesn't reach the threshold, the server will search its ARP cache for matching between MAC address encapsulated in the register message received and MAC address in ARP cache. This gives the algorithm the ability to work with DHCP based networks. In turn, it prevents an intruder having the hash code to spoof all ARP cache entries. As a matter of fact, it can only spoof one at a time. If it tries to spoof another one the old spoofed entry will be deleted.

In case a user tries to register with wrong hash code for the first time, it will be inserted in the suspicious users list. When its wrong trials reaches three, it will be moved to the blocked list.

User who has successfully registered at the server will receive a register response message contains the IP and MAC addresses of all successfully registered users to add them as static ARP entries. Moreover, all other users will receive an update message contains IP and MAC address of the new user to add it as a static entry in their ARP cache.

Using the client and server algorithms, every user in the network will have its ARP cache filled with static ARP entries for all other users in the network. Hence, it will not suffer from the ARP spoofing problem again. And everything is done automatically without any overload on the administrator; this gives the algorithm a greater scalability.

IV. EXPERIMENTAL RESULTS

Experimental measurement has been chosen to evaluate the performance of the proposed algorithm. The faculty of computers and information, Menofia University (Menofia is one of the districts of Egypt) network, shown in fig.1, is used to conduct the measurements. The network consists of three separate LANs. Each LAN ends with an edge switch. The LANs are connected through a core switch.

LAN 1 consists of 19 users and an ARP server. LAN 2 consists of 13 users and an application server offering web browsing, FTP, and mail services. LAN 3 is a network of 16 users and an Asterisk VOIP server for voice over IP calls between network users.

All PCs are core i5 processor with 4 GB of RAM. The edge switches are cisco catalyst 2960 switch with 24 ports. The core switch is cisco catalyst 4006 switch. Also the wireshark software is used in measuring the values.

The response time metric has been chosen to evaluate the performance of the algorithm. The response time is measured at the different stages of the algorithm and at both sides of the protocol (client and server) taking in mind the different parameters affecting the response time values. These measures speed, reliability, and robustness of the algorithm. Hence, it proves the algorithm efficiency.

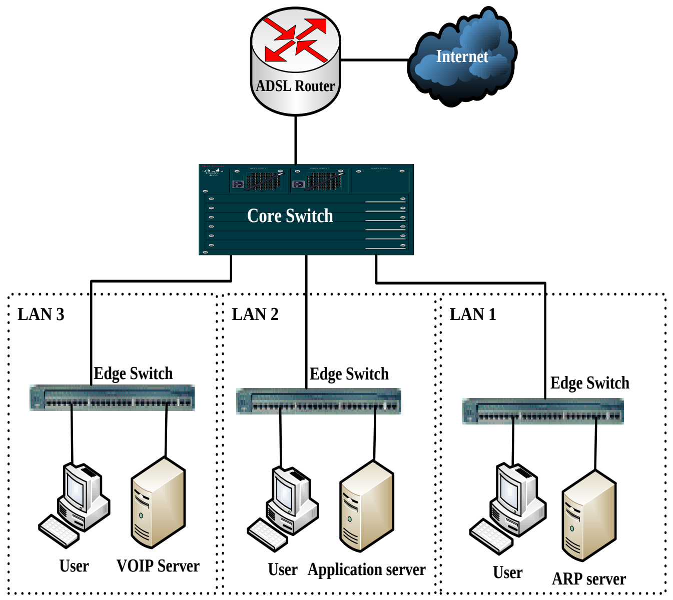
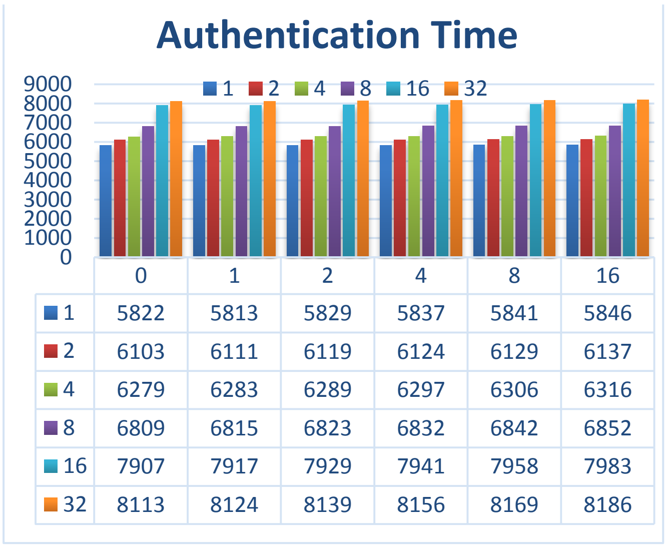


Fig. 1. Faculty of computers and Information, Menofia university network

A. *Server Side Measures*

1) *Authentication time (T_{auth})*

It represents the amount of time needed to authenticate a trusted user. Fig.2 shows the authentication time values. The X Axis represents the number of simultaneous attackers trying to authenticate using wrong key. The Y Axis represents the authentication time values in nanoseconds. The different colors represent the number of users trying to authenticate at the same time.

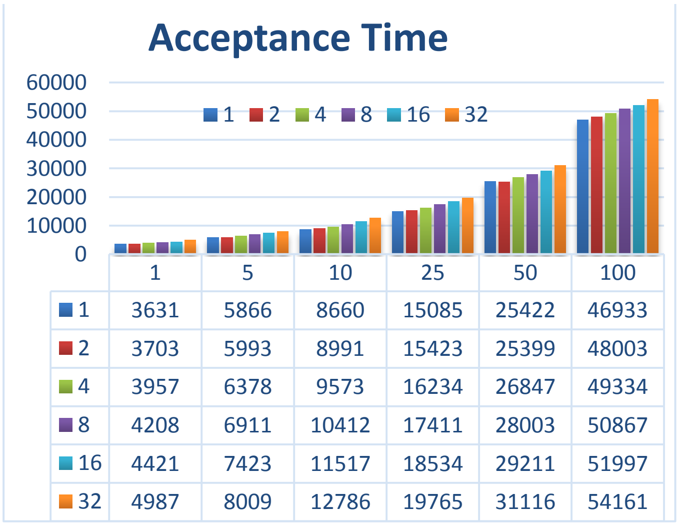


	0	1	2	4	8	16
🟦 1	5822	5813	5829	5837	5841	5846
🟥 2	6103	6111	6119	6124	6129	6137
🟩 4	6279	6283	6289	6297	6306	6316
🟪 8	6809	6815	6823	6832	6842	6852
🟦 16	7907	7917	7929	7941	7958	7983
🟧 32	8113	8124	8139	8156	8169	8186

Fig. 2. Authentication Time ($\boldsymbol{T_{auth}}$) in nanoseconds versus number of attackers for the different number of users

2) Acceptance time ($\boldsymbol{T_{acc}}$)

It represents the time spent by the server to accept an authenticated user. Fig.3 shows the acceptance time values. The X Axis represents the total number of ARP cache records. The Y Axis represents the acceptance time values in nanoseconds. The different colors represent the number of users trying to authenticate at the same time.

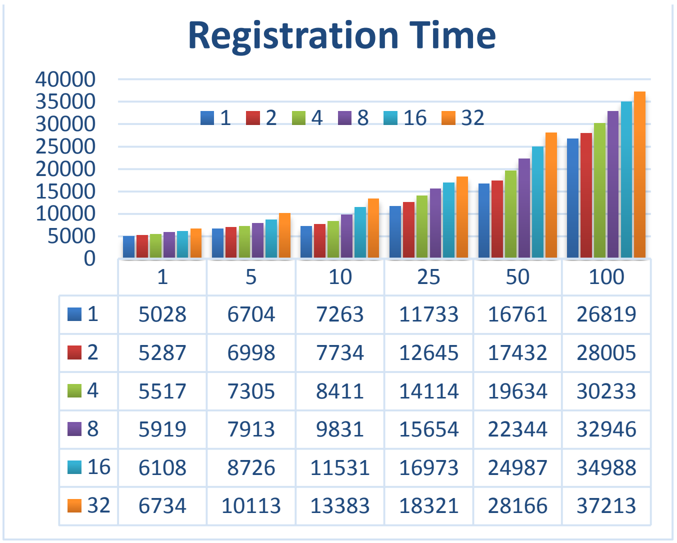


	1	5	10	25	50	100
🟦 1	3631	5866	8660	15085	25422	46933
🟥 2	3703	5993	8991	15423	25399	48003
🟩 4	3957	6378	9573	16234	26847	49334
🟪 8	4208	6911	10412	17411	28003	50867
🟦 16	4421	7423	11517	18534	29211	51997
🟧 32	4987	8009	12786	19765	31116	54161

Fig. 3. Acceptance Time ($\boldsymbol{T_{acc}}$) in nanoseconds versus number of ARP cache records for different number of users

3) Registration time ($\boldsymbol{T_r}$)

It represents the time taken by the server to add an entry for a new user in the ARP cache. Fig.4 shows the registration time values. The X-Axis represents the total number of ARP cache records. The Y-Axis represents the registration time values in nanoseconds. The different colors represent the number of users trying to register themselves at the same time.



	1	5	10	25	50	100
🟦 1	5028	6704	7263	11733	16761	26819
🟥 2	5287	6998	7734	12645	17432	28005
🟩 4	5517	7305	8411	14114	19634	30233
🟪 8	5919	7913	9831	15654	22344	32946
🟦 16	6108	8726	11531	16973	24987	34988
🟧 32	6734	10113	13383	18321	28166	37213

Fig. 4. Registration Time ($\boldsymbol{T_r}$) in nanoseconds versus number of ARP cache records for the different number of users

4) Update time ($\boldsymbol{T_u}$)

It represents the time needed by the server to notify all network users that a new user has entered the network. Fig.5 shows the update time values. The X Axis represents number of simultaneous users trying to communicate with the server. The Y Axis represents the update time values in nanoseconds.

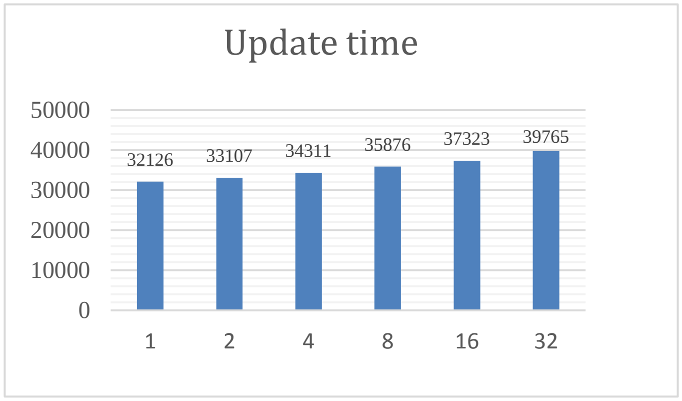
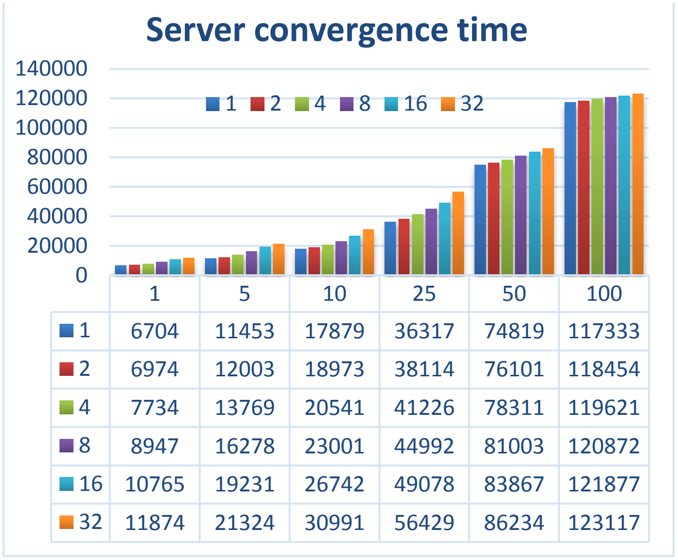


Fig. 5. Update Time ($\boldsymbol{T_u}$) in nanoseconds versus the number of users

5) Server Convergence time ($\boldsymbol{T_{Con}}$)

It represents the time taken by the server to send to the new user its ARP cache as a register response message. Fig.6 shows the server convergence time values. The X Axis represents the total number of ARP cache records. The Y Axis represents the server convergence time values in nanoseconds. The different colors represent the number of users trying to communicate with the server at the same time.



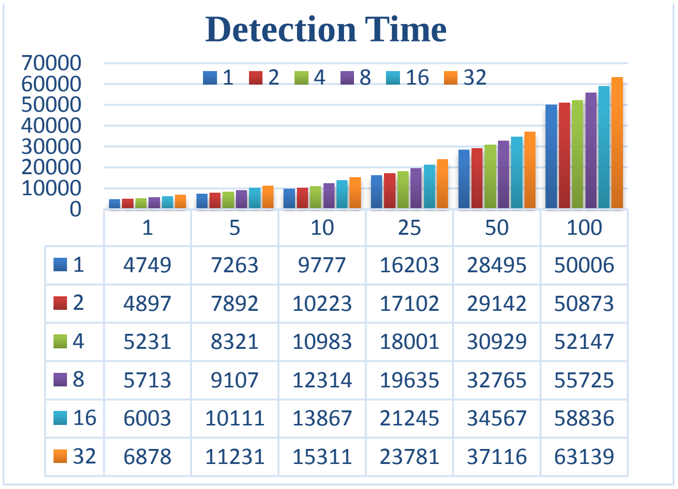
	1	5	10	25	50	100
🟦1	6704	11453	17879	36317	74819	117333
🟥2	6974	12003	18973	38114	76101	118454
🟩4	7734	13769	20541	41226	78311	119621
🟪8	8947	16278	23001	44992	81003	120872
🟦16	10765	19231	26742	49078	83867	121877
🟧32	11874	21324	30991	56429	86234	123117

Fig. 6. Server convergence Time (T_{Con}) in nanoseconds versus number of ARP cache records for the different number of users

6) Detection time (T_d)

It represents the time in nanoseconds spent by the server to detect a new attacking user and adds it to the suspicious users list, Fig.7 shows the detection time values.

The X Axis represents the number of suspicious user's list records. The Y Axis represents the detection time values in nanoseconds. The different colors represent the number of users trying to communicate with the server at the same time.



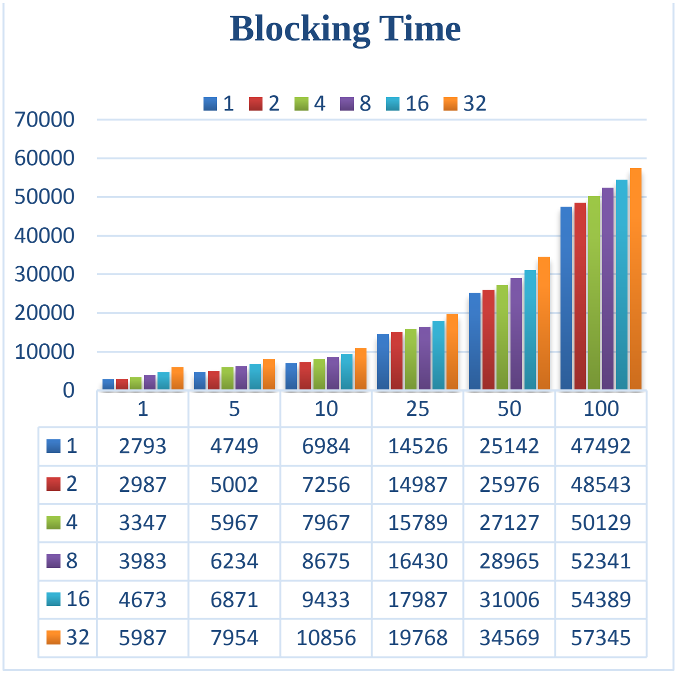
	1	5	10	25	50	100
🟦1	4749	7263	9777	16203	28495	50006
🟥2	4897	7892	10223	17102	29142	50873
🟩4	5231	8321	10983	18001	30929	52147
🟪8	5713	9107	12314	19635	32765	55725
🟦16	6003	10111	13867	21245	34567	58836
🟧32	6878	11231	15311	23781	37116	63139

Fig. 7. Detection Time (T_d) in nanoseconds versus number of record in suspicious users list for the different number of users

7) Blocking time (T_b)

It represents the time spent by the server to add a user to the blocked users list. Fig.8 shows the blocking time values. The X Axis represents the total number of records in the suspicious users list.

The Y Axis represents the detection time values in nanoseconds. The different colors represent the number of users trying to communicate with the server at the same time.



	1	5	10	25	50	100
🟦1	2793	4749	6984	14526	25142	47492
🟥2	2987	5002	7256	14987	25976	48543
🟩4	3347	5967	7967	15789	27127	50129
🟪8	3983	6234	8675	16430	28965	52341
🟦16	4673	6871	9433	17987	31006	54389
🟧32	5987	7954	10856	19768	34569	57345

Fig. 8. Blocking Time (T_b) in nanoseconds versus number of suspicious users list records for different number of users

It can be noted from Fig 2 to 8 that for any measured time ($T_{auth}, T_{acc}, T_r, T_u, T_{con}, T_d, T_b$), the time needed per user is nearly constant for any number of users for the same number of records.

B. Client side measures

1) Client Acceptance time (TC_{acc})

It represents the time needed by the client to be sure that the server accepted its register request message. It will be calculated using equation (1):

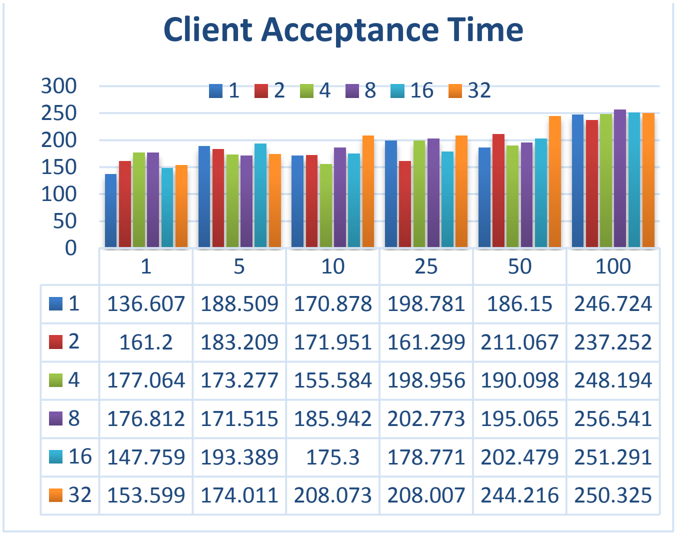
$$TC_{acc} = RTT + T_{auth} + T_{acc} + T_r + T_u \quad (1)$$

Where TC_{acc} is the client acceptance time, RTT is the round trip time, T_{auth} is the server authentication time, T_{acc} is the sever acceptance time, T_r is the server registration time, and T_u is the server update time.

The RTT value depends of the nature of traffic workload. The users are divided into four groups and every group of users is using one or more of the services: mail, file transfer, VoIP call, or web browsing. The workload depends on the number of services used by every group. The results were taken for 3 different types of workloads: light, normal, and heavy workload. Light workload represents one service usage. Two services are considered for normal workload and three for heavy workload [17].

a) Light traffic workload

Fig.9 shows the Client acceptance time values in light traffic workload, The X Axis represents the number of records in Server ARP cache. The Y Axis represents the Client acceptance time in Microseconds, the different colors represent the number of users trying to communicate with the server at the same time.

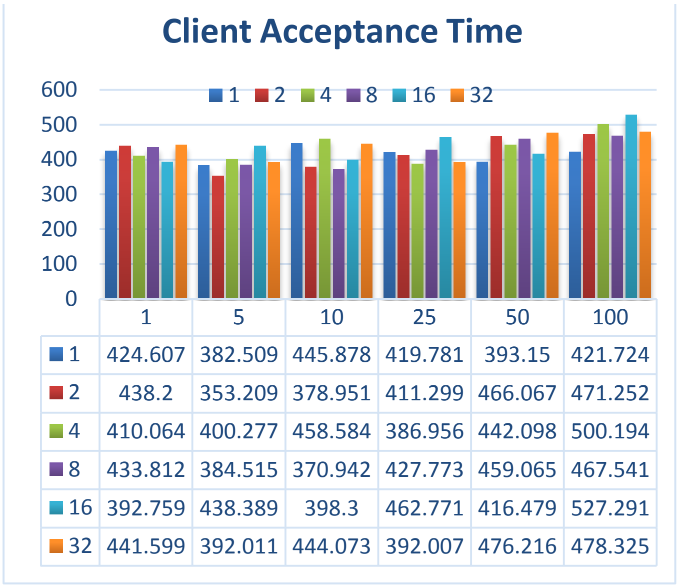


	1	5	10	25	50	100
🟦1	136.607	188.509	170.878	198.781	186.15	246.724
🟥2	161.2	183.209	171.951	161.299	211.067	237.252
🟩4	177.064	173.277	155.584	198.956	190.098	248.194
🟪8	176.812	171.515	185.942	202.773	195.065	256.541
🟦16	147.759	193.389	175.3	178.771	202.479	251.291
🟧32	153.599	174.011	208.073	208.007	244.216	250.325

Fig. 9. Client acceptance Time ($\boldsymbol{TC_{acc}}$) in nanoseconds versus number ARP cache records for different number of users in light worload

b) Normal workload

Fig.11 shows the Client acceptance time values in normal traffic workload, The X Axis represents number of record in Server ARP cache The Y Axis represents the client acceptance time values in microseconds, and the different colors represent the number of users trying to communicate with the server at the same time.

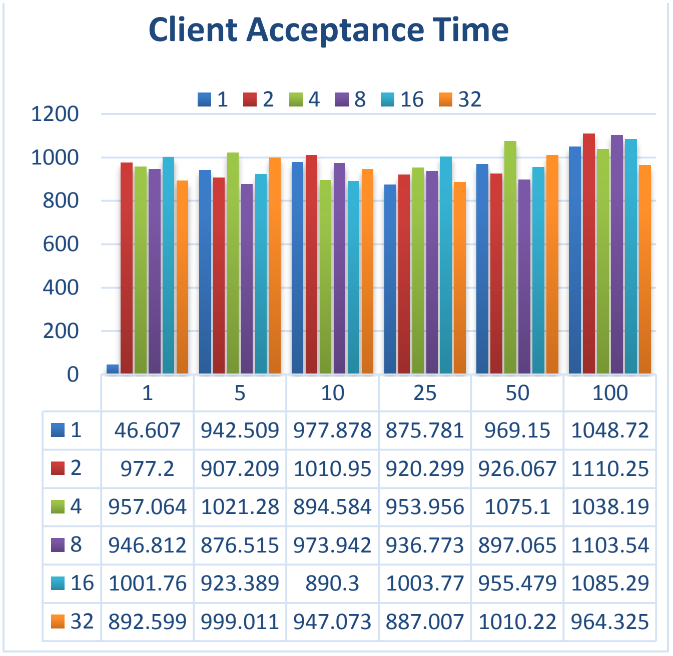


	1	5	10	25	50	100
🟦1	424.607	382.509	445.878	419.781	393.15	421.724
🟥2	438.2	353.209	378.951	411.299	466.067	471.252
🟩4	410.064	400.277	458.584	386.956	442.098	500.194
🟪8	433.812	384.515	370.942	427.773	459.065	467.541
🟦16	392.759	438.389	398.3	462.771	416.479	527.291
🟧32	441.599	392.011	444.073	392.007	476.216	478.325

Fig. 10. Client acceptance Time ($\boldsymbol{TC_{acc}}$) in nanoseconds versus number ARP cache records for different number of users in normal worload

c) Heavy traffic workload

Fig.11 shows the Client acceptance time values in Heavy traffic workload. The X Axis represents the number of records in the server ARP cache. The Y Axis represents the client acceptance time values in microseconds. The different colors represent the number of users trying to communicate with the server at the same time.

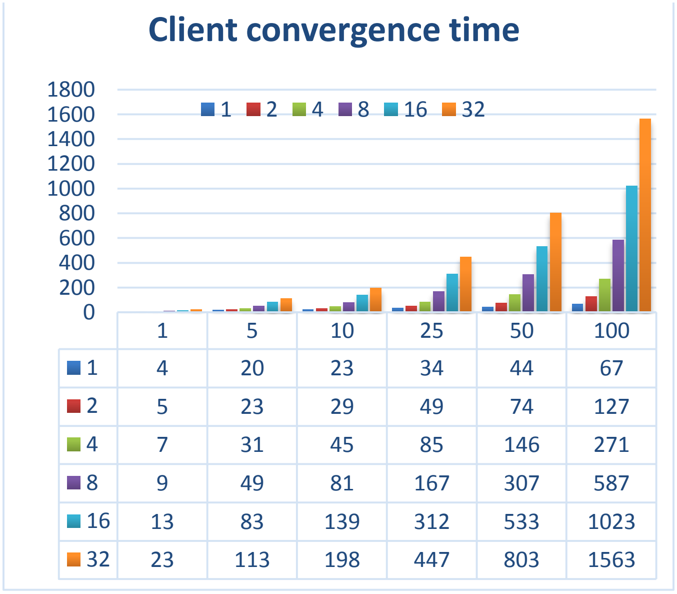


	1	5	10	25	50	100
🟦1	46.607	942.509	977.878	875.781	969.15	1048.72
🟥2	977.2	907.209	1010.95	920.299	926.067	1110.25
🟩4	957.064	1021.28	894.584	953.956	1075.1	1038.19
🟪8	946.812	876.515	973.942	936.773	897.065	1103.54
🟦16	1001.76	923.389	890.3	1003.77	955.479	1085.29
🟧32	892.599	999.011	947.073	887.007	1010.22	964.325

Fig. 11. Client acceptance Time ($\boldsymbol{TC_{acc}}$) in nanoseconds versus number ARP cache records for different number of users in heay worload

2) Client convergence time ($\boldsymbol{Tc_{Con}}$)

It represents the amount of time needed by the client to process the register response message and add static ARP entries using IP and MAC pairs encapsulated within the message. Fig.12 shows the client convergence time values. The X Axis represents the total number IP and Mac pairs encapsulated in the register response message. The Y Axis represents the client convergence time values in nanoseconds. The different colors represent number of users trying to communicate with the server at the same time.



	1	5	10	25	50	100
🟦1	4	20	23	34	44	67
🟥2	5	23	29	49	74	127
🟩4	7	31	45	85	146	271
🟪8	9	49	81	167	307	587
🟦16	13	83	139	312	533	1023
🟧32	23	113	198	447	803	1563

Fig. 12. Client Convergence Time ($\boldsymbol{Tc_{Con}}$) in nanoseconds versus number of ARP cache records for the different number of users

V. Conclusion

In this paper, a solution to the problem of ARP spoofing has been proposed, the solution is an automatic and scalable method of configuring static ARP entries instead of manually configuring. The solution solves the main problems related to this category of solutions Usage of static entries, automation, scalability, manageability, prevention, and cost are the main features of the proposed method. The proposed method has defined two separate algorithms, one for the client, and the other for the server. Experimental evaluation was conducted on the LAN network of the faculty of computers and information, Menofia university of Egypt. The response time metric is used to evaluate the algorithm. The values of the response time were measured at the different stages of the algorithm. Also different types of traffic workloads were used during the measuring the response to show the effect volume of traffic on the response time values. The results prove how fast and accurate the proposed algorithm is since any new user needs less than one millisecond to be safe from ARP problem for heavy workloads.

References

[1] Yafeng Xu and Shuwen Sun , “The study on the college campus network ARP deception defense," 2010 2nd International Conference on Future Computer and Communication (ICFCC), 3(1), pp. 465-467, May 2010.

[2] R. W. Stevens. TCP/IP Illustrated, Volume 1: The Protocols. Addison–Wesley Professional Computing Series, January 1994.

[3] D. Plummer. An Ethernet address resolution protocol, Nov. 1982. RFC 826.

[4] Mohamed Al-Hemairy, Saad Amin, and Zouheir Trabelsi, “Towards More Sophisticated ARP Spoofing Detection/ Prevention Systems in LAN Networks," *2009 International Conference on the Current Trends in Information Technology (CTIT),* pp.1-6, December 2009.

[5] Hu Xiangdong, Gao Zhan, and Li Wei "Research on the Switched LAN Monitor Mechanism and its Implementation Method based on ARP spoofing," *International Conference on Management and Service Science.(MASS '09)*, pp. 1-4, Sept. 2009.

[6] Marco Antônio Carnut and João J. C. Gondim, "ARP spoofing detection on switched ethernet networks: a feasibility study," *5th Symposium on Security in Informatics held at Brazilian Air Force Technology Institute*, November 2003.

[7] Cristina L. Abad and Rafael I. Bonilla, "An Analysis on the Schemes for Detecting and Preventing ARP Cache Poisoning Attacks," *27th International Conference on Distributed Computing Systems Workshops, 2007. (ICDCSW '07)*, page(s): 60, June 2007.

[8] Somnuk Puangpronpitag and Narongrit Masusai, “An Efficient and Feasible Solution to ARP Spoof Problem," *6th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2009. (ECTI-CON 2009)*, 3(1), pp. 910—913, May 2009.

[9] S. Whalen, "An introduction to ARP spoofing," 2600: The Hacker Quarterly, 18(3), 2001, (accessed 13-9-2012). [Online].:http://servv89pn0aj.sn.sourcedns.com/gbpprorg/2600/arp spoofing intro.pdf

[10] http://technet.microsoft.com/en-us/library/cc958841.aspx. ARP Cache, (accessed May 8, 2013).

[11] Zouheir Trabelsi and Wassim El-Hajj, "Preventing ARP Attacks using a Fuzzy-Based Stateful ARP Cache," IEEE International Conference on Communications.(ICC '07), pp. 1355 -1360, June 2007.

[12] Dr. S. G. Bhirud and Vijay Katkar, "Light Weight Approach for IP-ARP Spoofing Detection and Prevention," *2011 Second Asian Himalayas International Conference on Internet (AH-ICI)*, page(s):1-5, November 2011.

[13] Xiangning HOU, Zhiping JIANG, and Xinli TIAN, "The detection and prevention for ARP Spoofing based on Snort," *2010 I*

[14] Andre P. Ortega, Xavier E. Marcos, Luis D. Chiang and Cristina L. Abad, " Preventing ARP cache poisoning attacks: A proof of concept using OpenWrt," Latin American Network Operations and Management Symposium. (LANOMS), pp. 1-9, Oct. 2009.

[15] Ai-zeng Qian, "The Automatic Prevention and Control Research of ARP Deception and Implementation," *2009 WRI World Congress on Computer Science and Information Engineering*, , 2(1), pp. 555-558, April 2009.

[16] Boughrara, A.; Mammar, S., "Implementation of a SNORT's output Plug-In in reaction to ARP Spoofing's attack," 2012 6th International Conference on Sciences of Electronics Technologies of Information and Telecommunications (SETIT), pp.643,647, 21-24 March 2012

[17] R. K. Jain, “The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling,” Prince Hall, April 1991.