Title: Cross-validation based man-in-the-middle attack protection

Name Xiaofei Cui

CROSS-VALIDATION BASED MAN-IN-THE-MIDDLE ATTACK PROTECTION

Xiaofei Cui

MSc

2017

UNIVERSITY OF BEDFORDSHIRE

CROSS-VALIDATION BASED MAN-IN-THE-MIDDLE ATTACK PROTECTION

By

Xiaofei Cui

A thesis submitted to the University of Bedfordshire, in fulfilment of the requirements for the degree of Master of Science by research

March, 2017

ABSTRACT

In recent years, computer network has widely used in almost all areas of our social life. It has been profoundly changing the way of our living. However, various network attacks have become an increasingly problem at the same time. In local area networks, Man-in-the-Middle attack, as one kind of ARP attack, is the most common attack.

This research implemented a cross-validation based Man-in-the-Middle attack protection method (CVP). This approach enables a host to check whether another host that responds the initialising host with an ARP reply packet is genuine. It then allows the ARP cache table of the initialising hosts to be updated with the MAC address and IP address pairs of the genuine host and to place the MAC address of inauthentic hosts into a blacklist.

This research introduced ARP and ICMP firstly, including the structure of ARP and ICMP packets, and their workflows. Secondly, this research discussed the types of ARP attacks and the existing ARP attacks protection methods, including their principles, applicable environment, advantages and disadvantages. Then, this research proposed and implemented a cross-validation based Man-in-the-Middle attack protection method. Simulations and experiments were performed to examine the effect of CVP method. The results show the effectiveness of the proposed cross-validation based method in protecting network from Man-in-the-Middle attack. Compared with the existing Man-in-the-Middle attack protection methods, CVP requires no extra devices and administration, leading to more secure local area networks and low cost. It also has made a “tabu” to attackers. That is, it places the MAC address of attackers into a blacklist. So they will be identified immediately if they try to attack the network again.

DECLARATION

I declare that this thesis is my own unaided work. It is being submitted for the degree of Master of Science by Research at the University of Bedfordshire.

It has not been submitted before for any degree or examination in any other university.

Name of candidate: CUI Xiaofei

Signature: _____

Date: _____

ACKNOWLEDGEMENT

I wish to thank to my supervisor: Professor Dayou Li for his guidance and supervision throughout this research.

I also wish to express my thanks to my parents, Mr Xianshun Cui and Mrs Guangxia Dai, for their support and encouragement during the year of study.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

CHAPTER 1: INTRODUCTION

1. 1 Motivation

Address Resolution Protocol (ARP) in any local area network (LAN) allows the host in the network to acquire the physical address (Plummer, 1982), known as MAC address, of another host in the same network to establish communications between the two hosts. ARP operates based on the mutual trust between the hosts and does not have any extra authentication measures. This exposes the hosts to ARP attacks. Man-in-the-Middle is a typical ARP spoofing attack. An attacker cheats a pair of hosts that are establishing communications by associating the IP address of the two hosts with the attacker's own MAC address, so any host of the pair will regard the attacker as the host it wants to communicate with. The attacker can then capture the packets sent from one host to the other, change them and send the changed packets to the other host, without being recognised as an attacker.

LANs are increasingly used to connect smart devices including robots, in particular, in domestic environments, such as smart home, to allow human users to remotely control the devices. The devices and the entire smart homes can become vulnerable to ARP attacks due to the weak security measure of ARP itself. The ARP attacks can cause the devices to take unauthorised actions, leading to physical and psychological harms to the users.

LANs are also used as part of infrastructure of smart cities. ARP attacks could cause public services failures.

The protection of network hosts from Man-in-the-Middle attacks has been studied

since 1997 in “ARP and ICMP redirection games” (Volobuev, 1997). Reported techniques include S-ARP (Mohamed, et al, 2003) and attacks detection scheme (Tripathy R, et al, 2007), etc. Their drawbacks are given in following:

1. They rely on the third-party device to check each ARP packet in network. Any failure of the third-party device will result in an insecure network.
2. The use of the new protocols that encrypt packets to prevent Man-in-the-Middle attack to replace ARP can be costly, because of the expensive network devices needed.
3. The network administrator is required for binding MAC and IP addresses pairs in some approaches, leading to inflexibility and heavy workload.

This MSc research has focused on the development of a cross-validation based method to protect Man-in-the-Middle attacks. In this method, there are two tables used to validate whether the ARP packets are genuine, Whitelist and ARP table. The Whitelist stores the MAC and IP addresses pairs of all “live” hosts, the ARP table stores the MAC and IP addresses pairs of all “live” hosts which have communicated previously. When having received an ARP packet, the host run this method to check the MAC and IP addresses pair in the received ARP packet, and to enable the host to update its ARP cache table or adds the pair in Blacklist, which depends on whether the pair can be found in Whitelist and ARP table. The advantages of this method are given in following:

1. This method can fundamentally prevent Man-in-the-Middle attack. The function of this method is to ensure the ARP packets received in smart systems are genuine by cross-validation. Thus, no attacker can implement the Man-in-the-Middle attack by sending forged ARP packets.
2. This method provides an uncomplicated approach to prevent Man-in-the-Middle attack in smart systems. When the validated result is that the ARP packet is genuine, the smart system starts to communicate whether the smart system that uses this method communicates with others depends on

the cross-validation. It makes the process of protecting Man-in-the-Middle attack uncomplicated. The cost of using this method in smart systems is also lower, as that the smart systems do not have to be reformed before using this method and any smart system can use directly.

3. This method can prevent Man-in-the-Middle attack in smart systems by itself, which means there is no third-party device needed to apply in this method. The ARP packet sent in a normal manner, in other word, it need not be sent or forwarded by other devices. This makes the method more effective to prevent Man-in-the-Middle attack.

1. 2 Aim and Objectives

The aim of this research is to develop a cross-validation based Man-in-the-Middle attack protection (CVP) method to protect local area networks where ARP is used from a special type of ARP attack called Man-in-the-Middle attacks.

The objectives are:

1. To simulate Man-in-the-Middle attacks – This will confirm the mechanism of Man-in-the-Middle attack and its possible damages on smart systems.
2. To design and implement the cross-validation based Man-in-the-Middle attack protection algorithm – This will specify the functionality of the components of the method and will integrate the components together to achieve the expected attack protection.
3. To validate the CVP algorithm in the sense of its effectiveness – This will include simulation/experiment where Man-in-the-Middle attack is initialised in a network and the CVP algorithm is running to protect the network.

1. 3 Structure of Dissertation

This dissertation has six chapters.

Chapter 2 gives a literature review of the research area of ARP and ICMP. The three main areas of ARP include the structure of ARP packet, ARP attacks and ARP attacks protection. The structure of ICMP packet which can be used to simulate ARP attacks is also introduced in this chapter.

Chapter 3 describes the simulation of Man-in-the-Middle attack. There are three parts in this chapter. The first part is simulation environment which contains the software and hardware. The design of simulation of Man-in-the-Middle attack is described then. There are three scenarios in this part. Scenarios 1 and 2 show the processes of how an attacker becomes the Man-in-the-Middle between two hosts. Scenario 3 shows the process of making Man-in-the-Middle attacks. The third part is about how to implement attacks. After becoming Man-in-the-Middle, the attacker captured and modified packets from one host and sent a forged packet back to the host or other host. An attack example is also given in this part. The final part is summary which describes the main works of this simulation.

Chapter 4 contains three parts. The first part is background of cross-validation, which was used to design the cross-validation. The second part is about designing the cross-validation and establishing two tables, Whitelist and ARP table which can be used to search MAC and IP address pairs. To prevent the Man- in-Middle attack, the design of cross-validation is to ensure that ARP packet is genuine, which means the MAC and IP addresses pair in ARP packet can be found in Whitelist and ARP table. The details of establishing Whitelist and ARP table are introduced then. Whitelist was established to store the MAC and IP addresses pairs of all “live” hosts which were obtained from gateway. The MAC and IP addresses pairs of all “live” hosts which communicated previously in ARP table were obtained from host’s ARP cache table.

The third part presents details of cross-validation implementation. The cross-validation was used to check whether the ARP packet is genuine, which can protect smart systems from Man-in-the-Middle attack. The algorithms are also given in this part.

Chapter 5 describes the simulation of cross-validation based Man-in-the-Middle attack protection. The environment and process of this simulation are also described here. DHCP was used to issue IP addresses to hosts in this simulation. Then Whitelist obtained the MAC and IP addresses pairs of all “live” hosts from the lease file of DHCP. ARP table obtained the MAC and IP addresses pairs of all “live” hosts which communicated previously from host’s ARP cache table. The details of simulating cross-validation are given secondly. The cross-validation checked whether the MAC and IP addresses pair in ARP packet was genuine, which means it can find them in Whitelist and ARP table, and then allowed the ARP cache table to update or added the pair of the ARP packet in Blacklist. Simulation results are given in this chapter.

Chapter 6 gives conclusions and further work.

CHAPTER 2: BACKGROUND KNOWLEDGE AND LITERATURE REVIEW

2. 1 Background Knowledge

2.1.1 Address Resolution Protocol

Address Resolution Protocol (ARP) operates in the second tier, i.e. the data-link layer, of Open System Interconnection (OSI) seven-layer network model. It is a highly efficient data link layer protocol. Its main function is to acquire MAC address based on the IP address it knows. It is widely used in networking smart devices, including robots, smart living environments such as smart house, smart city, etc. If one host knows an IP address and wants to communicate with another host that holds the IP address in the same network, it will broadcast the IP address with a request packet and waits for reply packet from the network. The host that holds the IP address will respond to the request with a reply packet containing its MAC address. When the requesting host receives the response, it will then use the MAC address to communicate with the responding host. The IP address and the MAC address will then be paired and saved in the requesting host's ARP cache. Next time, if the host wants to communicate with the same host, it simply use the MAC address stored in its cache.

According to the functionality, ARP packets can be classified into two types:

- ARP request packet, used to obtain MAC address from receivers and containing an IP address and FF-FF-FF-FF-FF-FF(MAC address in ARP request packet) for the receiver to fill in with its MAC address.
- ARP reply packets, used to provide MAC address and contains the sender's IP

and MAC addresses.

The ARP request packet and the ARP reply packet are also called broadcast packet and non-broadcast packet, respectively. The structure of an ARP packet is shown in Figure 2-1.

OCTET/OFFSET	0	1
0	"Hardware type (HTYPE)"	
2	"Protocol type (PTYPE)"	
4	"Hardware length (HLEN)"	"Protocol length (PLEN)"
6	"Operation (OPER)"	
8	"Sender hardware address (SHA)"	
10		
12		
14	"Sender protocol address (SPA)"	
16		
18	"Target hardware address (THA)"	
20		
22		
24	"Target protocol address (TPA)"	
26		

Figure 2-1 Structure of ARP packet

The meanings of fields are listed below:

- “Hardware type (HTYPE)” -- stating the network protocol type in an exact and detailed way. For example, for Ethernet, it is 1.
- “Protocol type (PTYPE)” -- specifying internet work protocol, as ARP request is designed differently for different internet work protocols.

- “Hardware length (HLEN)” -- indicating the length of a MAC address (in octets), which can be a router, a host, or a switch. The length is 6 in Ethernet,
- “Protocol length (PLEN)” -- giving the length of addresses that is employed in the upper layer protocol (also in octets).
- “Operation (OPER)” -- specifying operations conducted by the sender, taking values of 1 if the operation is requesting and 2 if it is replying, respectively.
- “Sender hardware address (SHA)” -- containing the sender’s MAC address. This field contains the address of the host that sends the request if OPER = 1. Otherwise, it has the address of the host to which the request was sent. Switches take no account of this field, specifically in finding out MAC addresses.
- “Sender protocol address (SPA)” -- showing internet work address of the sender.
- “Target hardware address (THA)” -- containing the intended receiver’s MAC address. When OPER = 1, this field is ignored. When OPER = 2 this field is used to show the address of the host that generates the ARP request.
- “Target protocol address (TPA)” -- providing internet work address of the intended receiver.

2.1.2 Internet Control Message Protocol

Internet Control Message Protocol (ICMP) is regarded as a supporting protocol in an internet protocol suite. Network devices employ ICMP protocol to send error messages. ICMP is completely different from transport protocols such as TCP and UDP in the sense that it is not normally used for exchanging data between network hosts. Besides, it is not usually employed by end-user network applications. The structure of ICMP packet is shown in Figure 2-2.

OFFSET	0	4
OCTET	0	32
0	Type	Rest of Header
1	Code	
2	Checksum	
3		

Figure 2-2 Structure of ICMP packets

- Header: ICMP packet has a header containing 8-byte. The first 4 bytes of the header are fixed. The last 4 bytes, also known as Rest of Header, contains the type, code and checksum information as given below:
    - Type: ICMP type.
    - Code: ICMP subtype.
    - Checksum: specifying error checking data.
- Data section the size of which varies and up to 32 bytes.

ICMP can report errors in the following three situations:

1) where a datagram cannot reach its destination,

2) where a gateway does not have sufficient buffering capacity to forward a datagram, and

3) where a gateway found a shorter route for a host to send messages/data.

ICMP only sends the error messages to the device that sent the original data. The reason is that the original data can only be corrected by that device. After that device receiving the error messages, it can correct and determine how to resend the original data according to the error messages.

ICMP is often used in the network, for example, the “Ping” command that checks whether the network is blocked.

2.1.3 ARP Attacks

ARP is a commonly-used communication protocol for acquiring link layer addresses from Internet layer addresses. When an IP datagram is sent from one host to another in a LAN, the IP address is always useful in finding out the destination host’s MAC address to perform data transmission via the data link layer. The sender usually knows the IP address of the destination host. When it needs the host’s MAC address, the sender acquires by broadcasting a request packet, known as an ARP request, to the LAN. As the destination host has the IP, it then replies to the sender with an ARP reply package including its own MAC address.

ARP is a stateless protocol. Network hosts will automatically catch all ARP packets in all situations. Hosts will overwrite ARP entries once receiving a new ARP reply packet even the old ones are not yet expired. Hosts do not authenticate other hosts from which the packet is originated in ARP. This results in the risks of attacks.

ARP spoofing attacks

In principle, any host in a network can be an attacker. The reason is that there is no way in APR to check whether ARP packets are genuine or to confirm the identity of the sending hosts.

APR attacks associate attackers’ host MAC address with the IP address of a target host. Consequently, messages that should direct to the target host are then sent to the attackers. The attackers can inspect the packets, can modify the packages and then send to the actual destination host, or can launch a denial-of-service attack by causing

some or all of the packets on the network to be dropped.

The most common attack of ARP Spoofing is Man-In-Middle attack. An attacker starts from inserting itself into a network and then intercepts and forwards modified data to other hosts. The attacker is known as “Man-In-Middle” because it stands and blocks the direct communications between two hosts. The host believes the attacker is the host it wants to communicate when the first host is spoofed by an attacker.. that’s how the second host is cheated by the attacker in the same way. The attacker captures, modifies and forwards the modified ARP packets between the two hosts to collect valuable information, and finally launch an attack to the host by sending forged ICMP packets. Man-In-Middle attack is hard to discover in network owing to the lack of authentication in ARP. It resembles a legitimate "transparent broker" between hosts’ communications.

IP address conflict attacks

To access the Internet, each host must have a unique IP address when using the TCP/IP protocol so that a host knows where packages come from and where they should go. IP address conflict means that the IP addresses of two or more hosts in the same LAN are the same. When IP address conflict takes place, the system will issue an error message to the host that use the same IP address. The hosts will not be able to use the network resources properly.

As there is no security mechanism in the Internet for the default hosts to write their own IP address into their source IP address field, attackers can generate IP address conflict attacks by making forged IP addresses, pretending to be others, or by hiding the source IP address field. IP address conflict attack can destroy the safety and availability of the Internet, that is, the attackers can send requests with forging the source IP address as the victim's IP address and attack destination end host to respond

and send data to the victim.

IP address conflict attack packet flow is derived from multiple locations and can be harmful to the data flow of the legitimate hosts, for example, the attacker can be a forged host, which means the legitimate host who holds the same IP address cannot access the Internet, so the defence mechanism can't use source IP address to filter attack stream. It can also make it possible to interfere with the communication between the two parties, which means a message's being injected through an encrypted security channel makes the TCP connection be hijacked and the DNS cache invalid.

MAC flooding attacks

MAC flooding attack attacks switches by using network deception to make the MAC address table overflow, causing all the data frames that reach the switches being broadcast to the entire network which eventually leading to network congestion and data monitoring.

In an MAC flooding attack, a switch can encounter a number of forged ARP packets, each of which contains a different MAC address. The effect of the attack is to consume the limited memory set aside in the switch through flooding the MAC address table with the different sources of MAC addresses.

The intention of the attack from the attackers' perspective is to push the legitimate MAC addresses out of the MAC address table by sending the forged ARP packages to the network, leading to the situation where significant quantities of incoming frames (packages) flood out on all ports of a network. The MAC flooding attack has its name from its flooding behaviour.

2. 2 ARP Attack Protection

According to the types of ARP attacks, the methods of attack protection can be classified into ARP spoofing attack protection, IP address conflict attack protection and MAC flooding attack protection.

2.2.1 ARP Spoofing Attack Protection

The ARP spoofing attack is the problem faced by the ARP which can be replaced by Secure ARP Protocol (S-ARP) (Mohamed, et al, 2003). The main strategy for solving this problem is using Asymmetric Cryptographic Algorithm (ACA) and Digital Signature Algorithm (DSA) to add verification information in ARP packet. The ASA is a cryptography technique. ASA divides a secret key into a public key and a private key which are produced by DSA. The public key is assigned to the host which uses S-ARP instead of ARP, in the meanwhile, the ARP packet can be signed with a private key, and then any hosts with the public key is able to verify that the ARP packet was sent by some hosts possessing the corresponding private key. With ACA and DSA, every ARP packet in a LAN becomes more reliable, this is because the host which uses S-ARP instead of ARP will check the verification information, only in the case where the verification information is correct, the host will update its ARP table and start to communicate. Replacing an existing ARP protocol with S-ARP can effectively prevent MITM attacks, but this protection requires modifications to the existing ARP protocol, resulting in a heavy workload and cost.

Based on Man-in-the-Middle attack, an enhanced ARP was proposed (Kim, et al, 2010). The aim of this approach is to protect the device from Man-in-the-Middle attack by using a long-term ARP cache table and election mechanism. In this approach, every host in a LAN has a long-term ARP cache table which can store

MAC and IP addresses pairs for a long time. Within this table, there are MAC and IP addresses pairs of other hosts or gateways, which cannot be updated by ARP packets. If an attacker wants to send forged ARP packets to the host which has this table, the MAC and IP pairs cannot be changed, so the host still can communicate with reliable hosts. If a host which MAC and IP address pair is not in the long-term ARP cache table wants to communicate with the host which uses this enhanced ARP, this approach used election mechanism to confirm whether this host is reliable. Every host has unique and comparable identities which can be compared, if the host which wants to communicate has the higher identity, the host will be recognized as a reliable host. According to existing shortcomings of ARP cache, Enhanced ARP uses the Long - term ARP cache table to replace the existing in the ARP cache table, as well as add unique and comparable identity to hosts in each network. Therefore, each host is authenticated so as to prevent the occurrence of Man-in-the-Middle attack. But this method needs to modify the host and the ARP protocol and leads to large workload and costs.

A ticket–based address resolution protocol (Lootah.W, et al, 2009) was proposed to replace the ARP. There are two main strategies in this approach. The first is to store the MAC and IP addresses pairs in a Local Tickets Agent. In addition, this agent can produce the Ticket Information which will be added in ARP packet. The second is to check the ticket information in ARP packet whenever the host receives it. For example, the ARP packet likes a “visitor”, if it wants to “visit” a host, it has to show its “ticket” to get the permission. The host will start to communicate with the host which the ARP packet has the “ticket”. This approach verifies ticket information through the ticket agent, so as to verify the reliability of the host , even to realize its protection. It requires the internet that employs this method to protect Man-in-the-Middle attack must install network ticket agent, which will cost much for the protection.

A secure ARP and secure DHCP protocol (Issac, 2009) was proposed to mitigate ARP spoofing attacks. In this approach, the ARP cache is stored in DHCP database instead of storing in the hosts. The process of communication between two hosts is, for example, if a host A wants to communicate with the host B, it has to send an ARP request to DHCP database which stores all MAC and IP addresses pairs of hosts in the LAN firstly, then the ARP request packet will be encrypted and forwarded to the host B. After the host B receiving the encrypted ARP request packet, it will authenticated the ARP request packet with the secret key which is set before. If the ARP request packet is reliable, the host B will start to communicate with the host A and send a confirm information to DHCP database which will update its ARP cache table. This approach uses a DHCP database to store ARP cache table in order to prevent ARP spoofing attacks happened in the hosts. This approach verifies the secret key of the host through the DHCP database to prevent Man-in-the-Middle attacks. It requires a functional upgrade of DHCP database and a transformation of DHCP protocol, which is impractical.

A Fuzzy-Based Stateful ARP cache (Trabelsi, EI-Hajj, 2007) is an approach to prevent ARP spoofing attack. The stateful ARP cache is that the host will update its ARP cache table only after sending ARP request packet, in another word, the ARP cache table cannot be updated if the host does not send any ARP request packets. The Fuzzy Logic is used to check reliability of ARP packet. In this approach, the designer carried out a model to check the reliability of ARP packet, and if the probability of the ARP packet is over 50%, then this ARP packet is reliable. This approach replaces the existing ARP cache with the Stateful ARP cache, and verifies the reliability of the host through Fuzzy logic, thus effectively preventing Man-in-the-Middle attack. This stateful ARP cache solves ARP cache's stateless problem, but the changes of ARP is necessary which makes the cost of the method is high.

A Man-in-the-Middle intrusion detection scheme (Tripathy R, et al, 2007) was

proposed to prevent ARP spoofing attack. In this scheme, there is a database which stores the entire MAC and IP addresses pairs in the LAN and two types of protocols, an invitation-accept protocol and a request-reply protocol. For example, if host A wants to communicate with host B, host A needs to send an invite message to the database with host B's IP address. If there is the MAC and IP addresses pair in the database, the database will send an accept message to host A. The database will send a reply message with the MAC and IP addresses pair of the host B to host A after receiving the request message from host A, then the host will update its ARP cache table and start to communicate with the host B. This approach prevents Man-in-the-Middle from attacking through DHCP database and two new protocols mentioned above. This method presents a new network frame. Nevertheless, in the practical work, this new network frame's every communication must go through two protocols, result in a long time to verify the reliability of the host and low working efficiency.

2.2.2 IP Address Conflict Protection

Wang and Huang (2010) proposed an approach to prevent IP address conflict attack. In this approach, there is a DHCP database which can check the entire ARP packet in the gateway. If the source MAC and IP addresses pair in the ARP packet is the same with in the DHCP database, the gateway will forwarded this ARP packet in the LAN. If it is not, the gateway will abandon this ARP packet. This method is simple and effective, but the transformation of DHCP database can lead to the decrease of network transmission speed.

2.2.3 MAC Flooding Protection

Based on Simple Network Manage Protocol (SNMP), an Active Defense Mechanism

(Wu, et al, 2007) was proposed to prevent MAC flooding attack. The main strategy of this approach is to use SNMP Trap to detect the MAC address of the attacker. Once the gateway receives many ARP request packets, the SNMP will read the MAC addresses in ARP request packets which will be shown in the gateway. If there are many same MAC addresses, it will be recognized as the attacker's MAC address which will be "kicked out" from the LAN. The use of SNMP can effectively prevent the occurrence of MAC flooding attack. But in the case of the attacker knowing its protection principle, such as using different virtual MAC address in every attack, the effect of protection will become limited.

CHAPTER 3: ARP ATTACK SIMULATION

This study aims at a new method of protecting LAN from Man-in-the-Middle attacks. To confirm the features of the attacks and evaluate the protection method at later stage, the attacks need to be simulated. This chapter gives details about how the simulated attacks were designed and implemented and what possible damages it can impose to smart systems.

3. 1 Simulation Environment

A LAN network often contains only a small number of host computers. This simulation designed a LAN with three hosts, including two normal communication hosts and one attack host that take the role of Man-in-the-Middle. These three hosts are connected to the same router. This design is sufficient for illustrating Man-in-the-Middle attacks in a LAN. The network topology is shown in Figure 3-1.

The simulation software tools include Nessus and packet builder. Nessus is a proprietary vulnerability scanner with the functions of discovering hosts and scanning vulnerabilities, etc. In this simulation, Nessus was used as a network scanner for the attacker to scan the IP and the MAC addresses of all hosts in the network. The packet builder was used to acker to construct reply packets with the attacker’s own MAC address to respond to the unique and comparable identity of packets broadcasted by the hosts in the network.

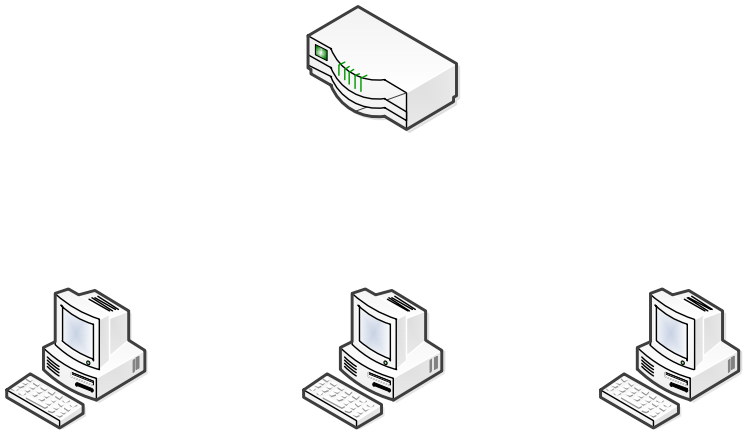

Figure 3-1 The network topology

3. 2 Attack design

The attacker C aims to let host A believe that C is host B and to let B believe that C is A through Man-in-the-Middle attacks. The following scenarios were designed:

Scenario 1:

Host A wants to communicate with a host whose IP address = 192.168.77.129. However, host A does not know the MAC address of the host that owns the IP address

1. Host A broadcast a request packet to the network, containing the IP address 192.168.77.129.
2. Attacker C responds immediately by sending a reply packet containing its own MAC address.
3. When host A receives the reply packet, it pairs the IP and the MAC within the reply packet and save it to its ARP cache, meaning that host A believes that attacker C is the host that holds the IP address of 192.168.77.129.

Scenario 2:

Host B wants to communicate with a host whose IP address = 192.168.77.128. However, B does not know the MAC address of the host which holds the IP address

1. Host B broadcast a request packet to the network, containing the IP address 192.168.77.129.
2. Attacker C responds immediately by sending a reply packet containing its own MAC address.
3. When host B receives the reply packet, it pairs the IP and the MAC, and save it to its ARP cache, meaning that host B believe that C is the host that holds the IP address of 192.168.77.128.

Now host C becomes the Man-in-the-Middle of hosts A and B. It can then send commands to A as it is B and to B as it is A.

Scenario 3:

Host A wants to send a command to host B. It first sends a “ping” command to confirm the establishment of a communication link between the two hosts.

1. Host A sends a “ping” command.
2. Attacker C captures the ICMP request packet and sends an ICMP reply packet. When host A receives the ICMP reply packet, it believes that it established the communication with host B.

Scenario 4:

Assume host B is a butler robot that has a database containing the personal information of a user. Host B is sending the username and password for the user to access the database to host A which is the mobile phone the user uses. Attacker C captures the packet after it becomes Man-in-the-Middle between hosts A and B.

3. 3 Attacks implementation

To simulate the four scenarios and to attack hosts A and B in a network, the attacker C must become Man-in-the-Middle first. It can then request information from hosts A and B. It can also capture and modify packets from one host and send a forged packet to the other.

3.3.1 Scanning IP address and MAC address

If an attacker wants to perform Man-in-the-Middle attacks to a host, first of all, it will have to get the IP address and MAC address of the host. The reason for this is that attacker C has to know the IP and MAC addresses of the host which it wants to attack.

Registering and logging in Nessus, by using the "Host Discovery", attacker C can scan the network and find IP address and MAC address of the hosts A and B, as shown in Figure 3-2. The scan result contains IP and MAC addresses of hosts A and B, and its own IP and MAC address. The blue strips represent the vulnerability of three hosts which can be used in other types of malicious attacks. If there is any vulnerability, the blue strip is not full of 100%. The third line is host A, the fifth line is host B and the sixth line is attacker C, respectively. Figures 3-3, 3-4 and 3-5 show their MAC addresses.

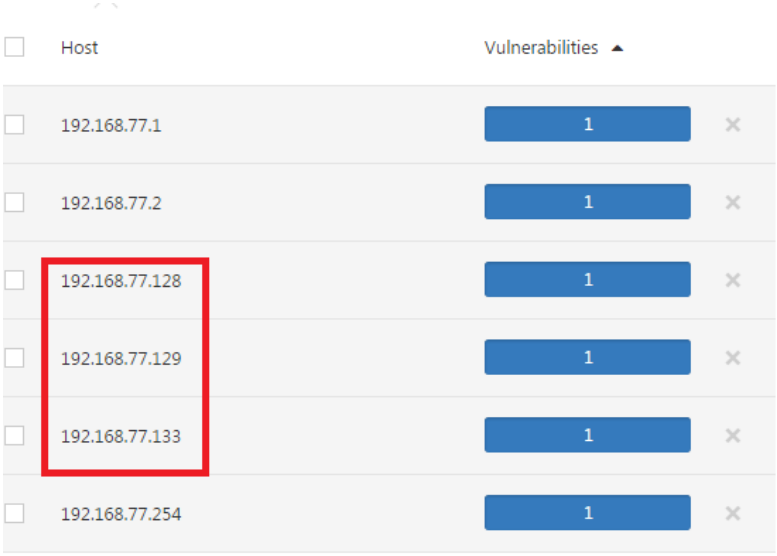

Figure 3-2 The scanning result of network

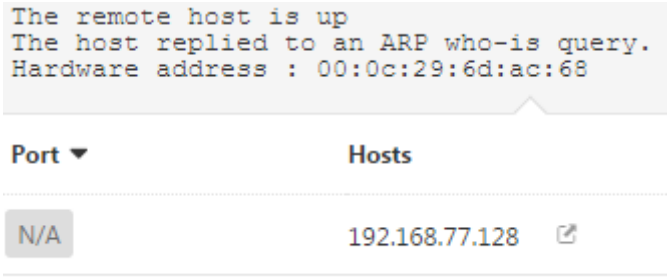

Figure 3-3 MAC address of host A in scan result

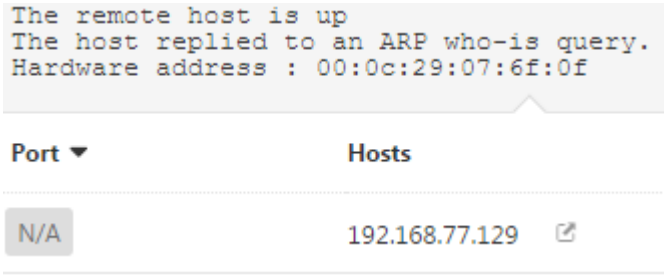

Figure 3-4 The MAC address of host B in scan result

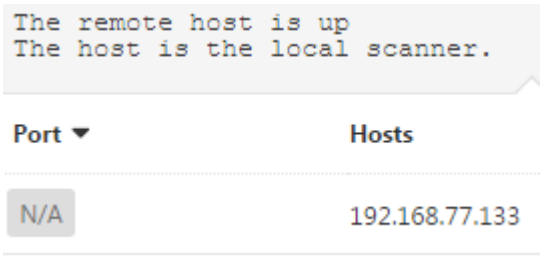

Figure 3-5 MAC address of attacker C in scan result

Table 3-1 summarises the IP address and MAC addresses.

Table 3-1 IP and MAC addresses

Host Name	IP Address	MAC Address
Host A	192.168.77. 128	00-0C-29-6D-AC-68
Host B	192.168.77.129	00-0C-29-07-6F-0F
Host C	192.168.77.133	00-0C-29-96-FF-82

3.3.2 Constructing and sending ARP packets

To place itself in between the two hosts A and B, C sent two ARP reply packets to host A and host B, respectively. Attacker C generated the ARP reply packet as shown in Figure 3-6 and sent to host A, and the ARP packet to host B in the similar way, shown in Figure 3-7.

OCTET/OFFSET	0	1
0	0001	
2	0800	
4	06	01
6	0002	
8	000C	
10	296D	
12	AC86	
14	C0A8	
16	4D80	
18	000C	
20	2996	
22	FF82	
24	C0A8	
26	4D81	

Figure 3-6 ARP reply packet sent to host A

OCTET/OFFSET	0	1
0	0001	
2	0800	
4	06	04
6	0002	
8	000C	
10	2996	
12	6F0F	
14	C0A8	
16	4D80	
18	000C	
20	2996	
22	FF82	
24	C0A8	
26	4D80	

Figure 3-7 ARP reply packet sent to host B

Hosts A and B updated their ARP cache table according to the ARP reply packets sent by C. In this way, the host A regards attacker C as host B and host B regards attacker C as host A. If there are any communications between host A and host B, the packets will be sent to host C. Figures 3-8 and 3-9 show ARP cache table of host B before and after receiving the ARP reply packet from C.

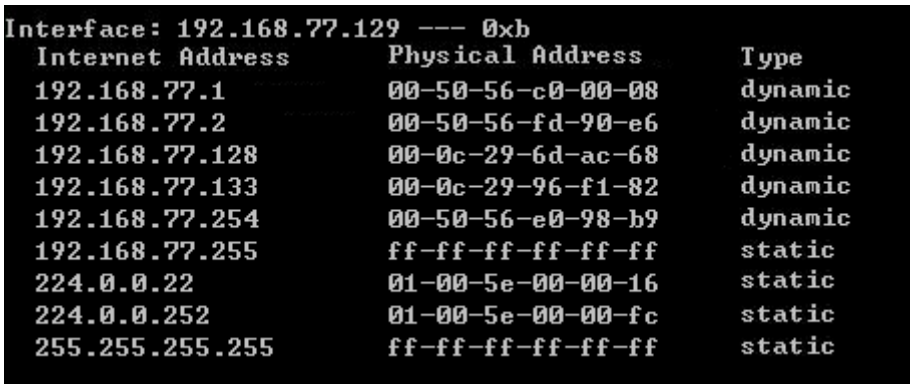

Figure 3-8 ARP cache table before Man-in-the-Middle attack

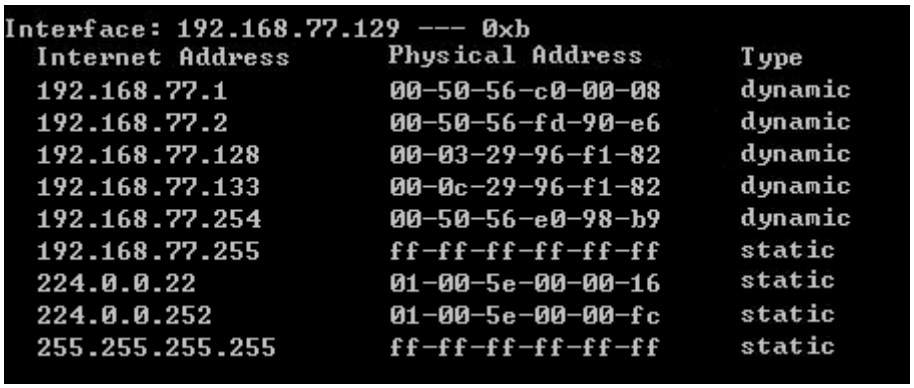

Figure 3-9 ARP cache table after Man-in-the-Middle attack

In Figure 3-8, there are IP and MAC addresses pairs of host A and attacker C. 192.168.77.128 is the IP address of host A, 00-0C-29-6D-AC-68 is the MAC address of host A. 192.168.77.133 is the he IP address of attacker C, 00-0C-29-96-F1-82 is the MAC address of attacker C. After Man-in-the-Middle attack, the MAC address of host B is changed to the MAC address of attacker C, as shown the line 5 of Figure 3-9.

In this simulation, the attacker becomes Man-in-the-Middle between hosts A and B. In the ARP packet which attacker C sent to host B, 00-0C-29-07-6F-0F is the MAC address of attacker C. 192.168.77.128 is the IP address of host A, 00-0C-29-07-6F-0F is the MAC address of host B, and 192.168.77.129 is the IP address of host B.

The MAC address of host A is changed to the same MAC address with attacker C,

when host A wants to communicate with the host B, attacker C carries out ARP spoofing by sending forged ARP packets to host A and host B continuously, which makes the communication between host A and host B intercepting by attacker C, more specifically, attacker C becomes the Man-in-the-Middle and all the communications between host A and host B will be forwarded by attacker C, the scenarios 1 and 2 were simulated successfully.

The ARP caches in hosts A and B are updated continuously, pairs IP and MAC addresses can only stay in a certain period of time. Therefore, attacker C needs to send forged ARP packets to hosts A and B periodically.

3.3.3 ICMP packets

Hosts in a LAN have to establish communication links with each other before starting the actual communications. Depending on who initiates communication, a host needs to send a “ping” command to the host that the first wants to communicate with. In this simulation, host A sent a “ping” command to host B. The command was captured by C.

Host A sent an ICMP request packet which is implemented by “ping” command to host B. Within the ICMP request packet, the MAC address of attacker C and IP address of host B were used, as shown in Figure 3-10, resulting at the situation where this ICMP packet was sent to attacker C. This is because that host A believes that attacker C is host B, so any packets which supposed to send to host B send to attacker C.

OFFSET	0	4
OCTET	0	32
0	08	61626364656 66768696a6b 6c6d6e6f707 17273747576 77616263646 566676869
1	00	
2	3A20	
3		

Figure 3-10 ICMP request packet from host A

Attacker C received the ICMP packet but host B did not, which is confirmed by showing the status of both C and B, as given in Figures 3-11 and 3-12, respectively. Compared with Figures 3-11 and 3-12, host B which IP address is 192.168.77.129 did not receive any ICMP packets, but attacker C which IP address is 192.168.77.133 received the ICMP packets from host A.

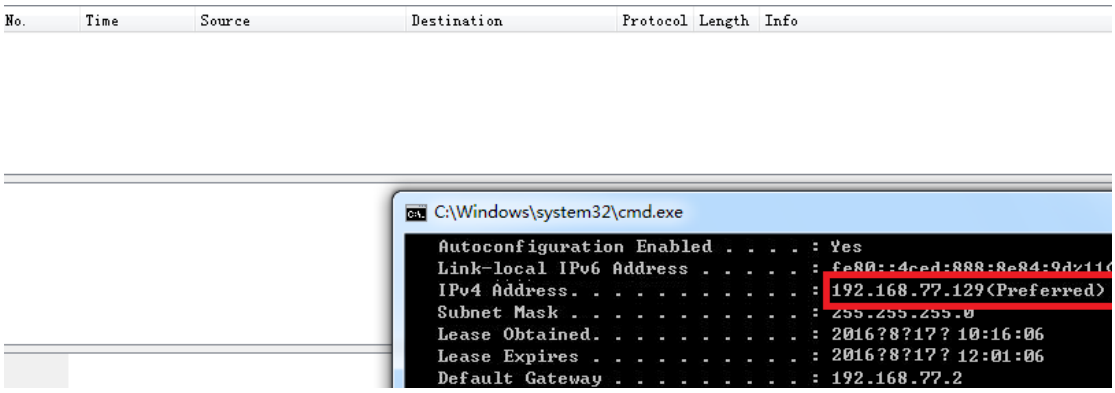

Figure 3-11 Status of host B

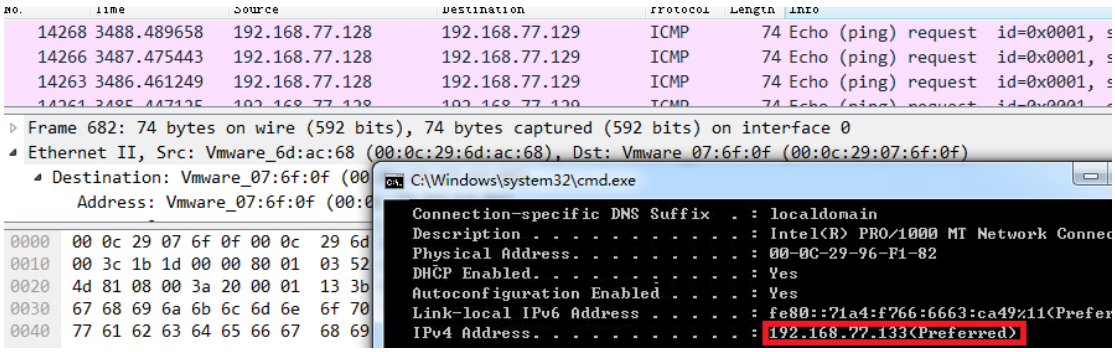

Figure 3-12 Status of attacker C

After attacker C receiving ICMP request packets from host A, it generated an ICMP reply packets, given in Figure 3-13, and sent to host A.

OFFSET	0	4
OCTET	0	32
0	00	61626364656 66768696a6b 6c6d6e6f707 17273747576 77616263646 566676869
1	00	
2	3A33	
3		

Figure 3-13 ICMP reply packet generated by C

The difference between this packet and the ICMP request packet from A is that this ICMP message is “00 00”, meaning an ICMP reply packet. When host A received the ICMP reply packet from attacker C, it believed that a communication link between itself and host B is established. The ICMP reply packets received in host A is shown in Figure 3-14.

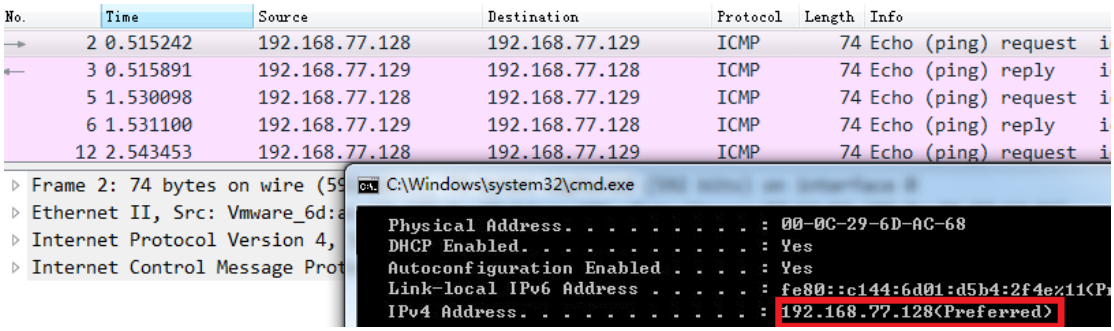

Figure 3-14 ICMP packets received by host A

In Figure 3-14, the ICMP reply packets sent by host C were regarded sent by host B in host A, which means host A believed the communication link between host B was established, the scenarios 3 was simulated successfully.

3.3.4 Attacking

After successfully becoming Man-in-the-Middle in a LAN, attacker C can capture communication packets that host A supposed to send to host B and vice versa. It can modify and forward the forged commands to the hosts in the network. It can also acquire personal information from the hosts.

As host B is a butler robot of a user, it has an FTP database which stores the important information of the user who uses his mobile phone, which is host A, to access the database. To let host A have the information, the FTP database set up a password and a username for the user and sent to host A with FTP packets. As having become Man-in-the-Middle between hosts A and B, attacker C captured the FTP packets and, hence, obtained the password and username.

The FTP packets which are captured in attacker C are shown in Figure 3-15. The second line in this diagram shows the username “host A”, and the fourth line shows the password “1234”.

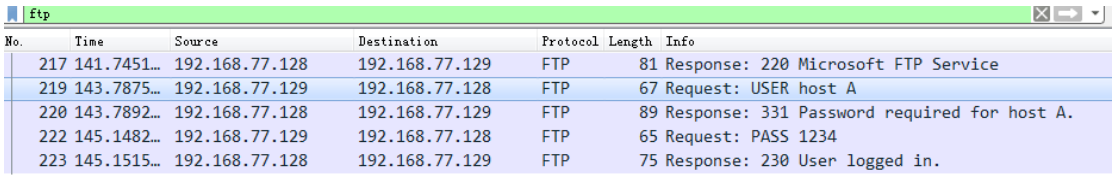

Figure 3-15 FTP packets captured by attacker C

With the username and password, attacker C can pretend it is host A to log in the FTP database. It can not only access all personal information of the user stored in the database, but also change the information in the database, the scenarios 4 was simulated successfully.

3. 4 Summary

This chapter gives details about the implementation of simulated Man-in-the-Middle attacks. To demonstrate the dangers of Man-in-the-Middle attack, the simulated attacker used ARP reply packet to become Man-in-the-Middle between two hosts A and B, used ICMP reply packet to let host A believed the communication link between hosts A and B was established, and obtained personal information from A through the communication link.

CHAPTER 4: CROSS-VALIDATION

4. 1 Background

Cross-validation is a method which uses at least two different sets to validate whether the user is genuine. This study performs cross-validation among three sets of IP and MAC addresses pair: Whitelist obtained from DHCP database, ARP table obtained from the existing ARP cache table of hosts and the pairs in ARP reply packets.

Let X be the pairs in ARP reply packet, Y be ARP table and Z be Whitelist. The principle of cross-validation is $X \cap (Y \cup Z)$.

If the IP and MAC addresses pair in ARP reply packet can be found in either Y or Z, the computer/smart device that replies ARP request with the ARP reply packet is considered genuine. Otherwise, it is regarded as an attacker.

4. 2 Cross-Validation to Protect Smart Systems from Man-in-the-Middle Attack

4.2.1 Cross-validation based Man-in-the-Middle attack algorithm

Communications between hosts/computers in a LAN is based on the MAC address of the hosts. That is, if a host A wants to communicate with another host B, A must obtain host B's MAC address first. An ARP table is designed for A to obtain B's MAC address, in the case where host B is a registered host in the network, the table contains the IP address and MAC address of all registered hosts in the network. A can search for host B's MAC address via B's IP address, refer to Figure 4-1 where both IP

address and MAC address can be seen in the table.

In the case where host B is not registered in the network, host A will broadcast IP address to all the hosts in the network using a request packet. Within the packet, there are the IP address and FF-FF-FF-FF-FF-FF for the host that has the same IP address to fill in with its MAC address. If a host in the network has the IP address as the same as the one in packet, it will replace FF-FF-FF-FF-FF-FF with its MAC address and send to host A. After having received the responds, host A will update its ARP table with the MAC address given by the responding host. Host A will then start to communicate with that host. The workflow can be seen in Figure 4-2.

It can be seen from the above description that there is little the ARP spoofing attack protection to secure communications between hosts in a LAN. An attacker, host C, can send a fake ARP reply package to the host A that initiates the communication with its own MAC address when having received this reply package, host A with update its ARP table by pairing the IP with C's MAC address suppose the IP address is the IP address of the host in the network. Host A will believe that host C is host B. Host C can also cheat host B in the same manner. As the result, host A and host B thought they communicated with each other, but actually they communicate with host C. The latter can give unauthorized commands to the hosts A and B.

```
[root@localhost studentadmin]# arp -a
gateway (172.16.114.1) at <incomplete> on eno16777736
? (192.168.1.198) at 00:0c:29:07:6f:0f [ether] on eno16777736
? (192.168.1.2) at 00:50:56:fd:ee:29 [ether] on eno16777736
```

Figure 4-1 ARP cache table

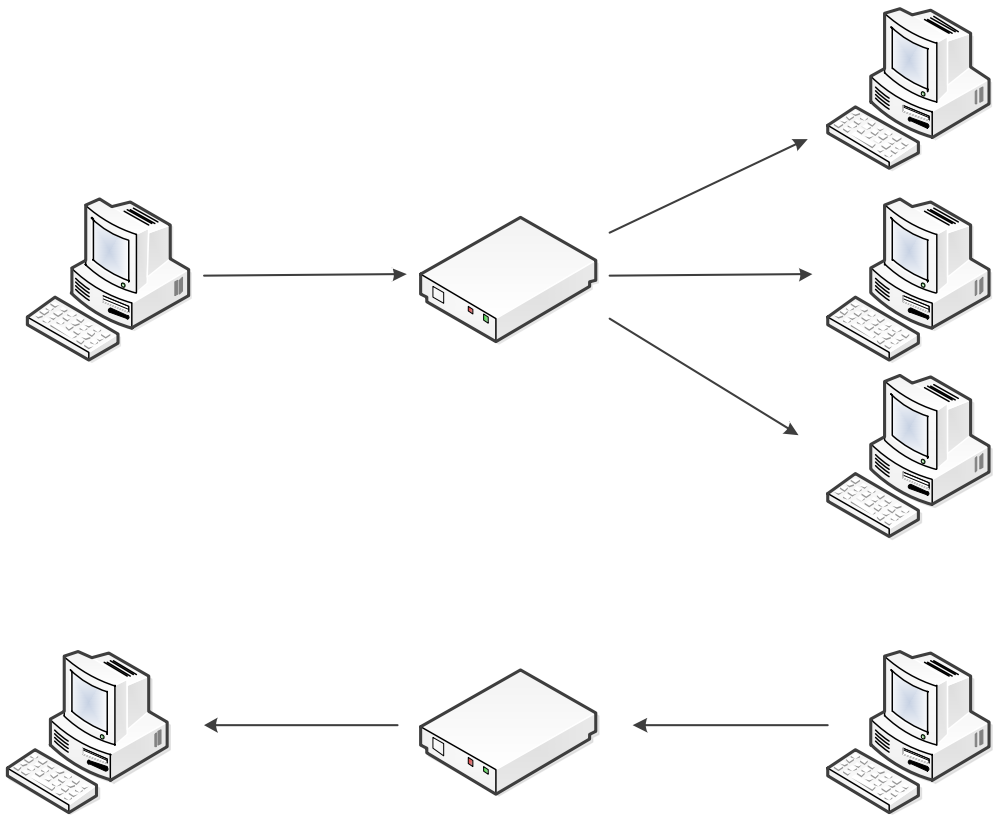

Figure 4-2 The workflow of ARP

To reduce the vulnerability, this study developed a cross-validation based Man-in-the-Middle attack protection (CVP) algorithm and introduced it into the communication process within LAN.

The algorithm can be described as in Figure 4-3. In this algorithm, whitelist stores the IP and MAC addresses pairs of all “live” hosts of a network. When a computer joins the network, the Network with assign an IP to the computer and, hence, an IP and MAC addresses pair forms. The Network with then update whitelist with the new pair. ARP is a local ARP table associated with a host. This table stores IP and MAC addresses pairs of all “live” hosts that communicated with the host. In addition, the ARP keeps an IP and MAC addresses pair of a host for a short time then the host left

the network.

```
Struct whitelist [ ] {
        String IP;
        String MAC; };
Struct ARP [ ][ ] {
       String IP;
       String MAC; };
String X ← IP + MAC; // IP + MAC sent by a new user
Whitelist + [ ] ← NewWorkARP[ ];
ARP[ ] [n] ← Host_n_ARP[ ]; //hostn's ARP
Function Cross − validation( )
{
Booled a, b;
If(string X = = whitelist [i])
 a ← TRUE;
else{
 a ← FALSE;
   { If(string x == ARP[j][N])
     b ← TRUE;
else
b ← FALSE;
}}
If ((a = 0)&&(b = 0))
BlackList ← x;
Else
ARP[ ][N] update;
}
```

Figure 4-3 Cross-validation based Man-in-the-Middle protection algorithm

The algorithm performs cross-validation among 3 sets of IP and MAC addresses pair. That is, if the IP and MAC addresses pair in ARP reply packet can be found in either Y or Z, the newly joined computer will be considered genuine. Otherwise, it will be regarded as an attacker. The corresponding MAC address will be put into a blacklist, which means that the network will not grant this MAC address an IP address so it cannot be join the network anymore in the future.

When a new computer, for example, host C with physical address MAC_C obtain an IP address, such as IP_C. The pair IP_C and MAC_C will be stored in X and whitelist Z. If host C is not cheating, it will use this pair to respond to request packet from any existing host, for example, host A. When host A runs the cross-validation, as the same pair is in Z, equation (4.1) will return TRUE. Hence, host A can update its ARP with the pair and start communication with host C.

If host C is cheating, it will capture the request package and reply with IP address that in the request package e.g. IP_B and with its own MAC address replacing FF-FF-FF-FF-FF-FF. The pair cannot be in whitelist Z. It cannot be ARP, Y, either this is because that this is the first time host A tries to communicate with IP_B, otherwise, host A would not broadcast IP_B. Equation (4.1) will return FALSE.

The reason for including ARP, Y, in the cross-validation is to avoid hasty in putting host C into blacklist. In the case where host C joined the network, left and re-joins, all in a short period of time, the network would possibly assign two different IP addresses to host C, for example, host C had IP_B in the first time joining and IP_C for the second time. Host C may accidentally respond to the request package from where IP_B is given. Because IP_B and MAC_C stays in host A's ARP for a short period, though the pair is not in whitelist, Equation (4.1) will return TRUE. So host C will not be placed in blacklist.

This will not jeopardize the security. This is because if host C cheated before, its MAC_C will be in blacklist. Assigning an IP address to MAC_C is forbidden by the network.

4.2.2 Establish Whitelist

The whitelist contains the IP and MAC addresses of all “live” hosts in a network. In a LAN, after a host accessing the network, the gateway will distribute an IP address to this host, then store the host’s IP and MAC address in a table. The IP and MAC addresses of all “live” hosts in the whitelist are gotten from the table stored in the gateway.

The IP and MAC addresses pair in the whitelist is an element of cross-validation. When a user wants to communicate with other host which running this system in a LAN, if this user’s IP and MAC addresses pair can be found in the whitelist, this system will check whether it can be found in ARP table. Its algorithm flow chart is shown in Figure 4-4.

In this flowchart, this system will read the information of all connected devices after accessing the gateway. Within the information, there are device name, MAC address and IP address which can be cut by connective. So for obtaining to IP and MAC addresses pair, this system will find the two connectives then cut and stores the information between the two connectives.

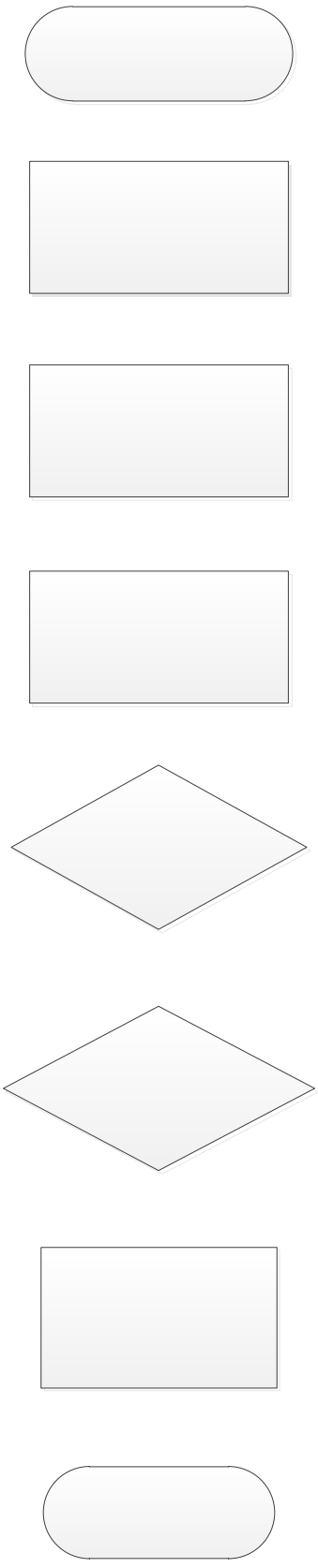

Figure 4-4 Whitelist algorithm

4.2.3 Establish ARP table

ARP cache table is used for storing IP and MAC addresses pair of existing hosts, essentially, it is a corresponding IP address and MAC address table which records the other hosts in the network. Each host has its own ARP table. When the host received a ARP request packet, it will search this MAC address according to the IP address in its ARP cache table firstly, if there is, the host will return the corresponding MAC address; if there is not, the host will only send ARP requests enquiries to other host.

ARP cache contains dynamic and static items. Dynamic items are automatically added and removed over time. A potential lifetime of each dynamic ARP cache entries is 10 minutes. New items can be removed from the cache table with a short time, which means if a ARP cache entry is not reused within 2 minutes, it will be removed from the ARP cache; If a ARP cache is already in use, that is, there are two hosts communicate by using this ARP entry in a LAN, it will receive the life cycle of 2 minutes; If a project is always in use, it will also receive the life cycle of 10 minutes. For the static entries, it will remain in the cache until the computer restarted.

When the system is running, the first thing is calling the ARP cache by entering the command “arp -a”, then read and stores all of the information in the cache table. In this information, every IP and MAC addresses pair is linked with some connectives, such as “at” or “on”. So the most important thing to get the IP and MAC addresses pair in the ARP table is to find the connectives, that is, the IP and MAC addresses pair can be gotten by cutting the information between the connectives. Its algorithm flow chart is shown in Figure 4-5.

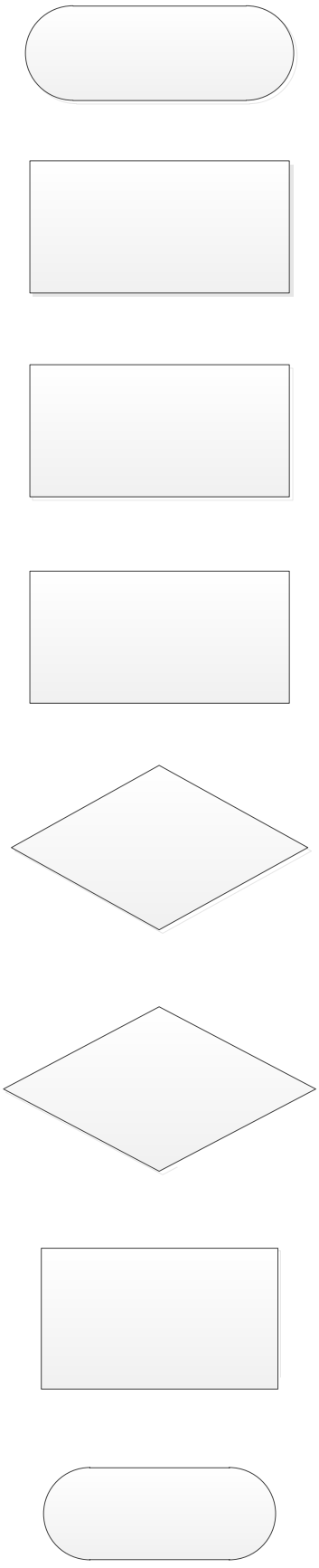

Figure 4-5 ARP table algorithm

After entering the command “arp –a” to query the ARP cache table, this system will read all the information which contains IP and MAC addresses pair in the table. Then, this system will search first connective in the information, if the first connective can be found, this system will search the second connective, if the two connectives all can be found, the information between the two connectives is the IP and MAC addresses pair which can be cut and stored.

4. 3 Implementation

The detailed flowchart of the cross-validation algorithm is given in Figure 4-6.

Figure 4-6 Cross-validation algorithm

“Receive an ARP reply packet” has the function of checking if the host receives an ARP reply packet, this was implemented as given in Figure 4-7. This function uses Wireshark to check if the host receives an ARP reply packet. Tshark is one function of Wireshark which can be used in computer terminal. In this algorithm, system will listen and store all ARP packets in computer.

```
void syetemnetworkmannager::serchstoreARPpacket()
{
System("tshark  -R "ARP">"D: \ARPPacket.txt"");
return 0;
}
```

Figure 4-7 Receiving an ARP reply packet function

“Obtain the MAC and IP addresses in the packet” has the function of obtaining MAC and IP address pair in ARP packet, this was implemented as given in Figure 4-8. In this function, this system opens the file which stores ARP reply packet firstly. Then in order to obtain the MAC and IP addresses pair in this packet, this system will cut the MAC and IP addresses pair according to the structure of ARP packet, which can be found from 22th byte to 32th byte in ARP packet.

```
using namespace std;
struct newUseripmac
{
char IPMAC[10];
};
void systemnetworkManager::getMACIPofnewUser(){
fstream infile("ARPPacket.txt",ios::in|ios::nocreate);
newUser data[10];
cout<<sizeof(data[0])<<endl;
for(int i=0;i<10;i++)
{
infile.seekg(i*32,ios::beg);
infile.seekg(22,ios::cur);
infile.read((char *)&data[i],sizeof(data[0]));
cout<<data[i].IPMAC[10]<< <<endl;
fin.close();
}
}
```

Figure 4-8 Obtaining the MAC and IP addresses in the packet function

“Establish a database of users with Whitelist and ARP” has the function of establishing a database of users with whitelist and ARP table, the whitelist was implemented as given in Figure 4-9. In this function, this system will open a file which stored the MAC and IP addresses pair of all “live” hosts firstly. Within the file, there are MAC and IP addresses of all “live” hosts and some connectives, which link MAC and IP addresses. This system will open the file which saved before, then cut the information between the two connectives, which can be added in whitelist.

```
void SystemNetworkManager::addMACIPinwhitelist(){
fstream infile("dhcpd.lease",ios::in|ios:: nocreate)
char *x;
    x = strtok(result,split);

    while (x!=NULL) {
        string stringTemp = x;
        size_t leasePos = stringTemp.find("lease ");
        size_t startsPos = stringTemp.find("starts ");
        size_t ethernetPos = stringTemp.find("ethernet ");
        size_t clientPos = stringTemp.find("client ");
        unsigned long ipclipCount = lease Pos - startsPos - 3;
        unsigned long macclipCount = ethernet Pos - clientPos - 3;
        string ip = stringTemp.substr(leasePos+3,ipclipCount);
        string mac = stringTemp.substr(ethernetPos+3,macclipCount);
        strcat(string ip,string mac);
        string macIPStart = macIP.substr(0,1);
        if (macIPStart.compare("(") != 0) {
            char *macIPChar = (char *)macIP.c_str();
            cout << macIPChar << "\n";
UserInfoDatabse::sharedDatabase()->addMACIPInWhitelist(macIPChar);
        }
        x = strtok(NULL,split);
}}
```

Figure 4-9 Establish a database of users with Whitelist and ARP function

The ARP table was implemented as show in Figure 4-10. In this function, this system will query the ARP cache table by entering “arp –a” command in computer terminal firstly. Because every time the ARP cache table is queried, it will be saved as a file automatically. Within the file, there are MAC and IP addresses of all communicated hosts and some connectives, such as “at”, “on”. This system will open the file which saved before, then cut the information depends on the connectives, which can be

added in ARP.

```
void SystemNetworkManager::addMACIPinARP() {
    char cmd[10] = "arp -a";
    char result[1024];
    char buf_ps[1024];
    FILE *ptr;
    if ((ptr=_popen(cmd,"r")) != NULL) {
        while (fgets(buf_ps,1024,ptr) != NULL) {
            strcat_s(result,buf_ps);
            if (strlen(result)>1024) {
                break;
            }
        }
        _pclose(ptr);
        ptr = NULL;
    } else {
        cout << "popen" << cmd << "error\n";
    }

    const char *split = "\n";
    char *p;
    p = strtok(result,split);
while (p!=NULL) {
        string stringTemp = p;
        size_t atPos = stringTemp.find("at ");
        size_t onPos = stringTemp.find("on ");
        unsigned long macclipCount = onPos - atPos - 9;
        string mac= stringTemp.substr(atPos+1,clipCount);
        string ip= stringTemp.substr(3,atPos-1);
        strcat(string ip,string mac);
        string macIPStart = macIP.substr(0,1);
        if (macIPStart.compare("(") != 0) {
            char *macIPChar = (char *)macIP.c_str();
            cout << macIPChar << "\n";
        UserInfoDatabase::sharedDatabase()->addMacAddresInARP(macIPChar);
        }
        p = strtok(NULL,split);
    }}
```

Figure 4-10 ARP table implementation

“Match with the MAC and IP addresses in whitelist and ARP” has the function of checking the reliability of new user, “Add this MAC address Blacklist” and “Update the ARP cache table” are the result of running this block. This was implemented as shown in Figure 4-11. In this function, the MAC and IP addresses pair of new user is compared with in the whitelist and ARP table. If the result is FALSE, which means this new user is an attacker, and then adds this new user into blacklist. If the result is TRUE, which means it can be found one same pair in whitelist or ARP table, this

system will update the ARP table with the MAC and IP addresses pair of the new user.

```
bool UserInfoDatabse::haveIPMacAddress(char *ipmacAddress) {
string ipmacAddressString = ipmacAddress;
UserInfoDatabase::sharedDatabase()->addMACIPInBlacklist(newUseripmac );
 cout <<"add  into blacklist with     " << newUseripmac   << "\n";
bool result = FALSE;
    for (int i = 0; i < arrayIndex; i++) {
        string newUseripmac  = ipmacAddresArray[i];
        if (ipmacAddressString.compare(newUseripmac) == 0) {
            result = TRUE;
system("arp  -s newUseripmac");
 cout <<"update ARP cache table with    "<<newUseripmac  << "\n";
        }
    }
    return result;
}
```

Figure 4-11 Matching with the MAC and IP addresses in whitelist and ARP function

4. 4 Summary

This chapter gives details about cross-validation to protect smart systems from ARP Spoofing attack method. In order to implement this method, this study established a whitelist which contains the MAC and IP addresses pair of all “live” hosts and an ARP table which contains the MAC and IP addresses pair of all communicated hosts as standards of cross-validation. The main idea to protect smart systems from ARP spoofing attack is to check every MAC and IP addresses pair in ARP packet, that is, whether it can be found in whitelist and ARP table. The implementation of this method written in the C++ programming language, the test of this method is given in next chapter.

CHAPTER 5: SIMULATION AND EVALUATION

5. 1 Aim and Environment

After implementing the cross-validation based method to protect smart systems from Man-in-the-Middle attacks as mentioned in Chapter 4, this chapter gives the details of a simulation in order to evaluate the proposed method in terms of protecting the network from Man-in-the-Middle attacks. During this simulation, the three functions were evaluated: Whitelist, ARP table and Cross-validation.

The aim of this simulation is to prove the effectiveness of the proposed cross-validation based method in protecting LANs from Man-in-the-Middle attacks. The expected results include:

1. The MAC and IP addresses pairs in the Whitelist of all “live” hosts are the same as the ones in a DHCP database, meaning Whitelist established correctly.
2. The MAC and IP addresses pair newly added to ARP table is the same as in the current pair in ARP cache table, meaning ARP table established correctly.
3. The ARP cache table does not update after Man-in-the-Middle attack, meaning the attack failed.

The achievement of the expected result one indicates attack simulation scenarios 1 and 2 will be prevented. The achievement of the expected results 2 and 3 means the protection of LANs from being affected as simulated in scenario 3.

The environment of this simulation was set up as follows:

The system: Windows 7, Linux.

The hardware: Three hosts.

The network topology is shown in Figure 5-1.

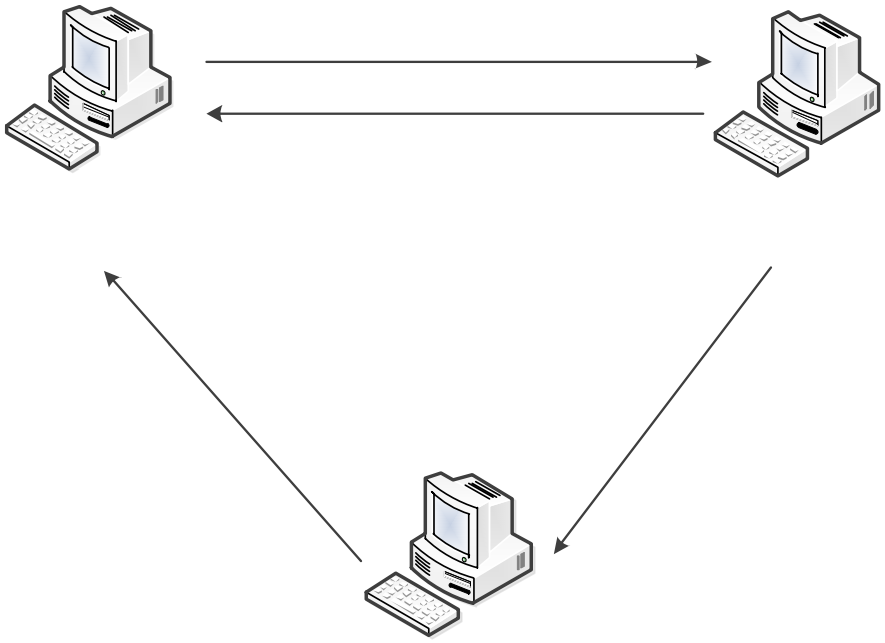

Figure 5-1 The network topology

Host A and host B are two normal users in the network, host C is an attacker that wants to launch Man-in-the-Middle attacks. The MAC addresses of the three hosts are shown in Table 5-1.

Table 5-1 Mac addresses of the hosts

Host Name	MAC address
Host A	00-0C-29-30-DE-BF
Host B	00-0C-29-07-6F-0F
Host C	00-0C-29-96-FF-82

The proposed cross-validation based method to protect smart systems from Man-in-the-Middle attacks were used in both hosts A and B.

The reasons for establishing this environmental setting are：

1. This environment is similar to the real LAN. The aim of cross-validation to

protect smart systems from Man-in-the-Middle attack method is that protect smart systems in a LAN, so the environment is closer to the LAN, the results of simulation are more realistic.

2. This environment meets the minimum requirements of simulation. This simulation requires that there are at least two hosts as normal users and one host as attacker.
3. The software and hardware are universal.

5. 2 Establishment of Whitelist

Dynamic Host Configure Protocol（DHCP）is a LAN protocol, which issues IP addresses to hosts in a LAN. When a host wants to access the network, the network picks up IP address from DHCP database and assigns it to the host. There are three ways to assign IP address:

1. Automatic allocation. When a host has gotten an IP address from DHCP database, it will always use this IP address to access the network.
2. Dynamic allocation. An IP address will be assigned to a host when it access the network and re-assigned to other hosts after the host left the network.
3. Manual allocation. The IP addresses will be assigned by network administrator.

DHCP was used to issue dynamic IP addresses to hosts, that is, an IP address will be assigned to a host and re-assigned to the other host after the host leaving the network. The keys are the range of IP addresses and max lease time. The range of IP addresses in this simulated network is from 192.168.1.128 to 192.168.1.255, which means every host’s IP address must be in this range. The max lease time is about six days, which means the IP addresses of connecting host will be changed after six days.

If a host accesses the network and is assigned with an IP address from DHCP database, a lease file will be created which contains the information of connected hosts in DHCP database. This lease file contains information, such as the connected host's IP address which was assigned from DHCP database and the host's MAC address. Figure 5-2 gives an example showing a host with MAC address 00-0C-29-30-DE-BF (Line 1) is assigned with IP address 00:0c:29:30:de:bf (Line 6) and is connected to the network.

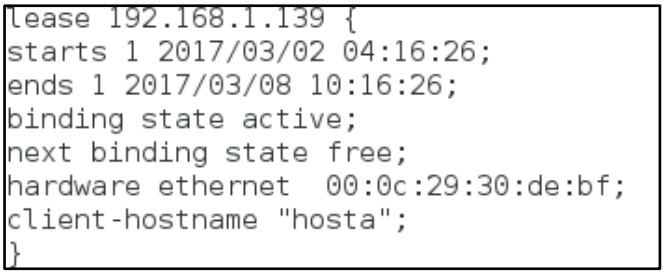

Figure 5-2 Information in lease file

The Whitelist used in the proposed cross-validation based method to protect smart systems from Man-in-the-Middle attacks contains the MAC and IP addresses pairs of all "live" hosts and is able to update itself whenever a host becomes "alive". To establish such as Whitelist, dynamic allocation was used. The configuration file is given in Figure 5-3.

One of the expected results is to have the MAC and IP addresses pairs in a Whitelist are the same as in DHCP database. The lease file which stores the information of the three hosts in DHCP is shown in Figure 5-4. In this diagram, there are three hosts have connected to this network with their MAC and IP addresses.

The Whitelist of host A generated in this simulation is shown in Figure 5-5. The obtained Whitelist contains the MAC and IP addresses pairs of three "live" hosts, which means the three hosts are now in this network. Comparing Figures 5-4 and 5-5,

it can be found that IP and MAC addresses pairs in DHCP are the same as those that are in the Whitelists.

```
ddns-update-style none;
ignore client-updates;
subnet 192.168.1.0 netmask 255.255.255.0
{
option routers 192.168.1.254;
option subnet-mask 255.255.255.0;
 option broadcast-address 192.168.1.255;
option domain-name-servers 192.168.1.3;
option domain-name "www.1234.com";
option domain-name-servers 192.168.1.3;
option time-offset -18000;
 range dynamic-bootp 192.168.1.128 192.168.1.255;
 default-lease-time 259200;
 max-lease-time 518400;
 host ns
{
hardware ethernet 52:54:AB:34:5B:09;
fixed-address 192.168.1.9;
}
}
```

Figure 5-3 Whitelist configuration

Figure 5-4 Connected hosts information

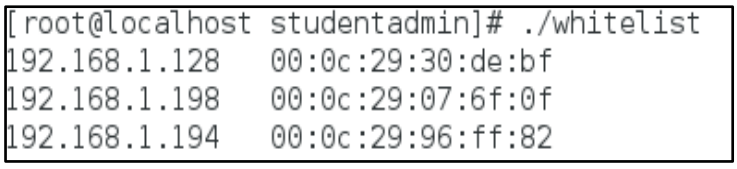

Figure 5-5 The result of simulating whitelist

5. 3 ARP Table

The ARP table of a host stores the MAC and IP addresses pairs of all “live” hosts which have previously communicated with the host. The ARP table is used to validate whether the host is genuine.

The MAC and IP address pairs in an ARP table are obtained from the ARP cache table.

ARP cache table of host A which runs the cross-validation based method to protect smart systems from Man-in-the-Middle attack is shown in Figure 5-6. There are two “live” hosts in the ARP cache table. The two hosts have previously communicated with host A. In this diagram. 192.168.1.198 and 00-0C-29-07-6F-0F in the second line are the MAC and IP addresses of host B. The other host is the gateway with its MAC and IP addresses pair as given in the bottom line.

```
[root@localhost studentadmin]# arp -a
? (192.168.1.198) at 00:0c:29:07:6f:0f [ether] on eno16777736
? (192.168.1.2) at 00:50:56:fd:ee:29 [ether] on eno16777736
```

Figure 5-6 ARP cache table of host A

To form IP and MAC pairs and add them into ARP table, ARP table algorithm as mentioned in the previous chapter first looked for connectives, “at” and “on” in the ARP cache table. These connectives lead to MAC and IP addresses of a host, respectively. The algorithm of ARP table searched for “at” from the ARP cache table and took the IP address before “at”. It searched for “on” and took MAC address before “on”. The algorithm then used the MAC and IP addresses to form the pairs and stored them in ARP table which is shown in Figure 5-7.

```
[root@localhost studentadmin]# ./arp
192.168.1.198   00:0c:29:07:6f:0f
192.168.1.2   00:50::56:fd:ee:29
```

Figure 5-7 ARP table obtained from the ARP cache table of host A

5. 4 Cross-validation

The cross-validation based method checks whether the MAC and IP addresses pairs in

ARP packet are genuine, that is, it can find them in Whitelist and ARP table.

Simulations are given as follows:

Host A broadcasted an ARP request packet which is shown in Figure 5-8 to request the MAC address of the host that holds that IP address. Within the packet, there are an IP address of the host to which host A wants to communicate and FF-FF-FF-FF-FF-FF for the host o fill in with its MAC address. In this simulation, host A wants to communicate with host B but it did not know host B's MAC address.

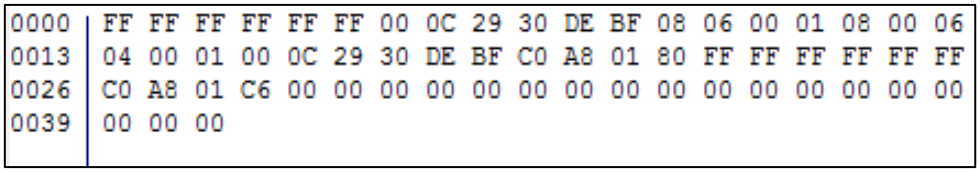

Figure 5-8 ARP request packet host A broadcasted

The ARP cache table of host A before broadcasting sending the ARP request packet is shown in Figure 5-9. In this diagram, there is only one pair of 192.168.1.2 (IP address) and 00-50-56-FD-EE-29 (MAC address), meaning host A has previously communicated with that host. The host is the gateway of the network.

```
[root@localhost studentadmin]# arp -a
gateway (172.16.114.1) at <incomplete> on eno16777736
? (192.168.1.2) at 00:50:56:fd:ee:29 [ether] on eno16777736
```

Figure 5-9 The ARP cache table of host A before broadcasting the ARP request packet

Every host in this LAN can receive this ARP request packet. When host B received the ARP request packet from host A, it sent an ARP reply packet with its MAC and IP addresses to host A.

Host A checked whether the MAC and IP addresses pair in the ARP reply packet from

host B is genuine by running cross-validation. The cross-validation algorithm as given in Section 4.3 compared the MAC and IP addresses pair of host B with the ones in its Whitelist and ARP table. Refer to the Figure 5-5, the MAC and IP addresses pair of host B was found in Whitelist, which means host B is a genuine host. Cross-validation then updated host A's ARP cache table with host B's MAC and IP addresses as given in Figure 5-10. 192.168.1.198 is the IP address of host B, 00-0C-29-07-6F-0F is the MAC address of host B.

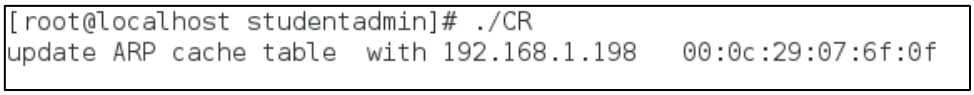

Figure 5-10 ARP cache table of host A updating

The ARP cache table of host A after receiving ARP reply packet from host B became the one as shown in Figure 5-11. Compared with the ARP cache table before receiving ARP reply packet, there is one more MAC and IP addresses pair added. This is host B's MAC and IP addresses pair which is in the fourth line of Figure 5-10, which means host A can start to communicate with host B.

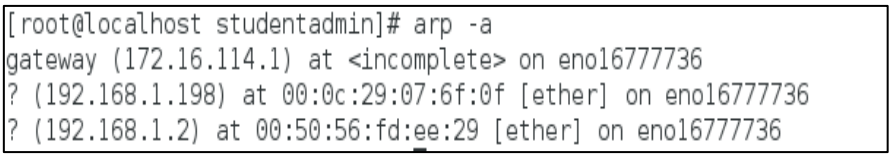

Figure 5-11 ARP cache table of host A after receiving ARP reply packet from host B

Host C also received the ARP request packet from host A. As an attacker, host C initiated a Man-in-the-Middle attack to host A by sending an ARP reply packet which is shown in Figure 5-12. Within this ARP reply packet, the MAC address is host C's and the IP address is host B's. In this diagram, 00-0C-29-96-FF-82 is the MAC address of host C in the first line, the IP address of host B is shown in hexadecimal which is C0-A8-01-80 in the third line.

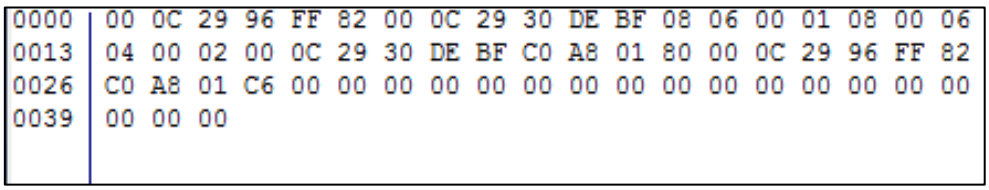

Figure 5-12 ARP reply packet from attacker (host C)

When host A received the ARP reply packet sending from host C, the MAC and IP addresses pair in the packet was checked in the same way as the ARP reply packet from host B. But the pair could not be found in Whitelist and in ARP table, refer to Figures 5-5 and 5-7, the IP address 192.168.1.198 (host B's IP address) has assigned to the host which MAC address is 00-0C-29-07-6F-0F (host B's MAC address), it cannot belong to the host which MAC address is 00-0C-29-96-FF-82 (host C's MAC address). This is means host C is recognized as an attacker, t cross-validation added host C's MAC address in blacklist as shown in Figure 5-13.

```
[root@localhost studentadmin]# ./CR
add into blacklist  with 192.168.1.194   00:0c:29:96:ff:82
```

Figure 5-13 The result of receiving a forged ARP reply packet from host C

The ARP cache table of host A which is shown in Figure 5-14 was not updated, in another word, it was the same as the one before receiving the ARP reply packet from host C. This can be seen by comparing with Figure 5-11, the ARP cache table of host A after receiving ARP reply packet from host B.

```
[root@localhost studentadmin]# arp -a
gateway (172.16.114.1) at <incomplete> on eno16777736
? (192.168.1.198) at 00:0c:29:07:6f:0f [ether] on eno16777736
? (192.168.1.2) at 00:50:56:fd:ee:29 [ether] on eno16777736
```

Figure 5-14 The ARP cache table of host A after receiving the ARP reply packet from attacker C

5. 5 Summary

This chapter implemented the simulation of the cross-validation based method to protect smart systems from Man-in-the-Middle attacks. The simulation shows the effectiveness of the proposed method in the sense of protecting from Man-in-the-Middle attacks.

1. The ARP cache table is updated only when the cross-validation proves that the host from which an ARP reply packet is received is genuine. The host to which this method is applied only updates its cache in the case where the cross-validation can find the MAC and IP addresses pair in ARP reply packets in Whitelist or ARP table.
2. All ARP reply packets are checked in hosts. When a host receives an ARP reply packet, it runs cross-validation to check the MAC and IP addresses pair in the packet. This avoids the acceptance of IP and MAC pairs from ARP attackers.
3. ARP attackers' MAC addresses are placed in a blacklist to prevent further attacks from these attackers.

CHAPTER 6: CONCLUSION AND FURTHER WORK

6. 1 Conclusion

This research aims at the development of a cross-validation based Man-in-the-Middle attack protection (CVP) method. The method enables the hosts that initialise communications to check whether responding hosts are genuine by looking at the ARP reply packets sending back from those hosts. It allows the ARP cache table of the initialising hosts to be updated with the MAC address and IP address pairs of the genuine hosts and to place the MAC address of inauthentic hosts into a blacklist. The following work has been done:

1. Simulating Man-in-the-Middle attack to reflect the damages it can impose to smart systems. The simulated attacker used ARP packets to become “Man-in-the-Middle”, and then captured and modified the packets to obtain the personal information from the simulated hosts.
2. Establishing a Whitelist and an ARP table to store the MAC and IP addresses pairs. Whitelist was established to store the MAC and IP addresses pairs of all “live” hosts which were obtained from lease file of DHCP. The MAC and IP addresses pairs of all “live” hosts which communicated previously in ARP table were obtained from host’s ARP cache table.
3. Implementing the cross-validation to check whether the ARP packet is genuine. To ensure that the ARP packet is genuine, the cross-validation checked whether the MAC and IP address pairs in ARP packets can be found in Whitelist and ARP table, and then allowed the ARP cache table to update or added the pair of the ARP packet in Blacklist based on whether the ARP packet is genuine.
4. Setting up a simulation environment to prove the effectiveness of the proposed

cross-validation based method in protecting network from ARP attacks. The ARP cache table of host which used CVP method did not update after Man-in-the-Middle attack, which means this method can prevent Man-in-the-Middle attack.

The simulation results presented in this dissertation show:

1. Man-in-the-Middle attack can be implemented by sending ARP packet. After sending ARP packet, the attacker becomes “Man-in-the-Middle”, and then can capture and modify packets to attack.
2. The cross-validation proves that the host from which an ARP packet is received is genuine, and then allows the ARP cache table of the host to update.
3. The cross-validation also proves that the attacker from which an ARP forged packet is received is inauthentic, which means the MAC and IP addresses pair in the packet cannot be found in Whitelist and ARP table, and then adds the MAC and IP addresses pair of the forged ARP packet to Blacklist.

Compared with the existing Man-in-the-Middle attack protection methods, the proposed CVP method has the following advantages:

1. No extra devices is needed and no changes is necessary in the existing LANs,
2. Low cost resultant from 1,
3. More secure due to the use of blacklist which contains the MAC address of attackers,
4. No administrator is required because of the automated process of adding new hosts to networks of the proposed cross-validation.

6. 2 Further Work

This research aims at developing a cross-validation based Man-in-the-Middle attack

protection which can be used in smart systems. Apart from Man-in-the-Middle attacks, there are other ARP attacks such as MAC flooding attacks and IP conflict attacks. CVP method is not validated for protection networks from those ARP attacks. There is a need for different approaches to be designed to prevent those ARP attacks.

DHCP is used to issue IP addresses to the hosts in network and create a lease file which contains the same MAC and IP addresses pairs with in the Whitelist. Every host, include the attacker, can obtain an IP address from the DHCP. In further work, permission information such as a key word can be used in CVP method. In the case where the host has the permission information, the DHCP issues the IP address to it, which can further reduce the cost of CVP method and make a more secure network.

In some situations, the attacker can send a large number of ARP reply packets to influence the rate of network communication. Though the attacker is not genuine, after being checked by CVP method in hosts, the attacker aims at implementing network congestion instead of becoming “Man-in-the-Middle” or launching other ARP attacks. This method is used in hosts now, to improving the speed of network transmission in LANs, this method can be used in the router in the future to check whether the host from which receives the ARP reply packet is genuine, and then the router only forwards the ARP reply packets which are from genuine hosts.

REFERENCES:

Yuri, Volobuev (1997). ARP and ICMP redirection games[EB/OL]. Available at: http://insecure.org/sploits/arp.games.html [Accessed 15 Mar. 2017].

RFC 826, An Ethernet Address Resolution Protocol -- or -- Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware[S].

Charles, M, Kozierok. The TCP/IP Guide[M]. 2005. 277-305

RFC 792, INTERNET CONTROL MESSAGE PROTOCOL[S].

Kotenko, I. and Skormin, V. (n.d.). Computer Network Security. 1st ed.

Pandey, A. and R. Saini, J. (2016). ARP cache rectification for defending spoofing and poisoning attacks. International Conference on Computing for Sustainable Global Development, pp.3487-3492.

Xing, W., Zhao, Y. and Li, T. (2010). Research on the defense against ARP Spoofing Attacks based on Winpcap. 2010 Second International Workshop on Education Technology and Computer Science, pp.762-765.

Yang, R. (2014). Study on ARP Protocol and Network Security in Digital Manufacturing. Applied Mechanics and Materials, 484-485, pp.191-195.

Xu, Y. and Sun, S. Yang, R. (2010). Study on ARP Protocol and Network Security in Digital Manufacturing. Applied Mechanics and Materials, 484-485, pp.191-195. 2010 2nd International Conference on Future Computer and Communication, 3,

pp.465-467.

Salim, H., Li, Z., Tu, H. and Guo, Z. (2012). Preventing ARP Spoofing Attacks through Gratuitous Decision Packet. 2012 11th International Symposium on Distributed Computing and Applications to Business, Engineering & Science, pp.295-299.

Nam, S., Kim, D. and Kim, J. (2010). Enhanced ARP: preventing ARP poisoning-based man-in-the-middle attacks. IEEE Communications Letters, 14(2), pp.187-189.

Trabelsi, Z. and El-Hajj, W. (2007). Preventing ARP Attacks using a Fuzzy-Based Stateful ARP Cache. IEEE Communications Society, pp.1355-1360.

G. Gouda, M. and Huang, C. (2003). A secure address resolution protocol. comuter networks, 41(1), pp.57-71.

Bruschi, D., Ornaghi, A. and Rosti, E. (2003). S-ARP：a secure address resolution protocol. in 1 9th Annual Computer Security Applications Conference.

Lootah, W. Enck, W. and Mcdaniel, P. (2007). TARP：ticket-based address resolution protocol. Computer Network, 51(15), pp.4322-4377.

Raval, N. and Chaudary, P. (2015). Detection and Prevention of ARP Cache Poisoning. International Journal of Engineering Trends and Technology, 30(3), pp.159-160.

Goyal, V. and Tripathy, R. (2005). An efficient solution to the ARP cache poisoning problem. Lecture Notes in Computer Science, 3573, pp.40-51.

Younes, O. (2016). A Secure DHCP Protocol to Mitigate LAN Attacks. Journal of Computer and Communications, 04(01), pp.39-50.

Issac, B. (2009). A Secure DHCP Protocol to Mitigate LAN Attacks. Journal of Computer and Communications. International Journal of Network Security, 8(2), pp.39-50.

Trabelsi, Z. and Shuaib, K. (2008). A NOVEL MAN-IN-THE-MIDDLE INTRUSION DETECTION SCHEME FOR SWITCHED LANs. International Journal of Computers and Applications, 30(3).

The Propagation Method by Means of an Attack of The 'Man-in-The-Middle' Type in The DHCP Protocol. (2014). Systems and Means of Informatics.

Hao, G. and Tao, G. (2009). Principle of and Protection of Man-in-the-middle Attack Based on ARP Spoofing. Journal of Information Processing Systems, 5(3), pp.131-134.

M., A., S.Elkilani, W. and M.Amin, K. (2014). An Automated approach for Preventing ARP Spoofing Attack using Static ARP Entries. International Journal of Advanced Computer Science and Applications, 5(1).

Ranjith Kanna Kanna, R., Sunkari, V. and Chander, P. (2011). Dos and ARP Spoofing Attacks Analysis through Agent Software. Indian Journal of Applied Research, 3(5), pp.253-255.

Forouzan, Behrouz A. (2007). Data Communications And Networking (Fourth ed.). Boston: McGraw-Hill. pp. 621–630. ISBN 0-07-296775-7.

GGller, D. (n.d.). Contract, Renegotiation, and Holdup: When Should Messages Be Sent?. SSRN Electronic Journal.

LIU, J. (2008). ICMP-based method for logical network topology discovery and analysis. Journal of Computer Applications, 28(6), pp.1498-1500.

Ko, B., Chung, S. and Cho, G. (2013). A Design of Network Management System for Efficiently Isolating Devices Infected with ARP Spoofing Virus. The Journal of the Korean Institute of Information and Communication Engineering, 17(3), pp.641-648.

Hassan, M. (2010). Smart pattern recognition of structural systems. Smart Structures and Systems, 6(1), pp.39-56.

Forouzan, B.A. (2000). TCP/IP: Protocol Suite (1st ed.). New Delhi, India: Tata McGraw-Hill Publishing Company Limited.

Seo, C. and Lee, K. (2016). ARP Spoofing attack scenarios and countermeasures using CoAP in IoT environment. Journal of the Korea Convergence Society, 7(4), pp.39-44.

Kang, J., Lee, Y., Kim, J. and Kim, E. (2014). ARP Modification for Prevention of IP Spoofing. Journal of information and communication convergence engineering, 12(3), pp.154-160.

Ko, B., Chung, S. and Cho, G. (2013). A Design of Network Management System for Efficiently Isolating Devices Infected with ARP Spoofing Virus. The Journal of the Korean Institute of Information and Communication Engineering, 17(3), pp.641-648.

Srinath, D., S.Panimalar, S., Simla, A. and J.Deepa, J. (2015). Detection and

Prevention of ARP spoofing using Centralized Server. International Journal of Computer Applications, 113(19), pp.26-30.

Kang, J., Lee, Y., Kim, J. and Kim, E. (2014). ARP Modification for Prevention of IP Spoofing. Journal of information and communication convergence engineering, 12(3), pp.154-160.

Song, M., Lee, J., Jeong, Y., Jeong, H. and Park, J. (2014). DS-ARP: A New Detection Scheme for ARP Spoofing Attacks Based on Routing Trace for Ubiquitous Environments. The Scientific World Journal, 2014, pp.1-7.

Hong, S., Oh, M. and Lee, S. (2013). Design and implementation of an efficient defense mechanism against ARP spoofing attacks using AES and RSA. Mathematical and Computer Modelling, 58(1-2), pp.254-260.

Lin, J., Koo, M. and Wang, C. (2013). A Proposal for a Schema for ARP Spoofing Protection. Applied Mechanics and Materials, 284-287, pp.3275-3279.

APPENDIX:

Main.cpp

```
#include < iostream >
#include "PermissionManager. hpp"

int main(int argc, const char * argv[]) {
    PermissionManager(). run();
    return 0;
}
```

PeimissionMangager.hpp

```
#ifndef PermissionManager_hpp
#define PermissionManager_hpp

#include < stdio. h >
#include "UserInfoDtabase. hpp"
#include "SystemNetworkManager. hpp"
#include < string >

class PermissionManager {
public:
    void run();
    PermissionManager();
    bool havePermissionForIPMacAddress(char *);
  }
private:
    int choice;
```

```
    SystemNetworkManager * systemNetworkManager;

    void searchstoreARPpacket();
    void displayvalidatedresultt(); };
```

PeimissionMangager.cpp

```
#include "PermissionManager.hpp"
#include < iostream >

void PermissionManager::run() {
    while (1) {
        cout <
            < "\nwelcome to network manager, please select a number
            \n";
        cout << "1.serchstoreARPpacket\n";
        cout << "2.displayvalidatedresult\n";
        cin >> choice;
 (choice) {
            case 1:
                this-> searchstoreARPpacket();
                break;
                case 2:
                this-> displayvalidatedresult();
                break;

            default:
                break;
        }
        choice = 0;
```

```
    }
}

void PermissionManager::searchstoreARPpacket() {
    this->systemNetworkManager->searchstoreARPpacket();
}
void PermissionManager::displayvalidatedresult() {
UserInfoDatabse::sharedDatabase()->displayvalidatedresult();
}

bool PermissionManager
            ::havePermissionForIPMacAddress(char *ipmacAddress)
{
    bool result = UserInfoDatabse::sharedDatabase()-
            >haveMacIPAddress(ipmacAddress);
    return result;
}
```

SystemNetworkManager.hpp

```
#ifndef SystemNetworkManager_hpp
#define SystemNetworkManager_hpp

#include <stdio.h>

class SystemNetworkManager {
public:
    void addMACIPinARP();
    void addMACIPinwhitelist();
    void getMACIPofnewUser();
```

```
    void serchstoreARPpacket();
};
```

SystemNetworkManager.cpp

```
#include "SystemNetworkManage. hpp"
#include < iostream >
#include < string >
#include "UserInfoDtabase. hpp"

using namespace std;
void SystemNetworkManager: : addMACIPinARP() {
    char cmd[10] = "arp -a";
    char result[1024];
    char buf_ps[1024];
    FILE * ptr;
    if ((ptr = _popen(cmd, "r")) != NULL) {
        while (fgets(buf_ps, 1024, ptr) != NULL) {
            strcat_s(result, buf_ps);
            if (strlen(result) > 1024) {
                break;
            }
        }
        _pclose(ptr);
        ptr = NULL;
    } else {
        cout << "popen" << cmd << "error\n";
    }

    const char * split = "\n";
```

```
    char * p;
    p = strtok(result, split);
while (p! = NULL) {
        string stringTemp = p;
        size_t atPos = stringTemp.find("at ");
        size_t onPos = stringTemp.find("on ");
        unsigned long macclipCount = onPos − atPos − 9;
        string mac = stringTemp.substr(atPos + 1, clipCount);
        string ip = stringTemp.substr(3, atPos − 1);
        strcat(string ip, string mac);
        string macIPStart = macIP.substr(0,1);
        if (macIPStart.compare("(") ! = 0) {
            char * macIPChar = (char *)macIP.c_str();
            cout << macIPChar << "\n";
        UserInfoDatabase::sharedDatabase()−
            > addIPMacAddresInARP(macIPChar);
        }
        p = strtok(NULL, split);
    }
}

void SystemNetworkManager::addMACIPinwhitelist(){
fstream infile("dhcpd.lease", ios::in|ios:: nocreate)
char * x;
    x = strtok(result, split);

    while (x! = NULL) {
        string stringTemp = x;
        size_t leasePos = stringTemp.find("lease ");
```

```
            size_t startsPos = stringTemp.find("starts ");
            size_t ethernetPos = stringTemp.find("ethernet ");
            size_t clientPos = stringTemp.find("client ");
            unsigned long ipclipCount = lease Pos − startsPos − 3;
            unsigned long macclipCount = ethernet Pos − clientPos − 3;
            string ip = stringTemp.substr(leasePos + 3, ipclipCount);
            string mac = stringTemp.substr(ethernetPos + 3, macclipCount);
            strcat(string IP, string mac);
            string macIPStart = macIP.substr(0,1);
            if (macIPStart.compare("(") != 0) {
                char ∗macIPChar = (char ∗)macIP.c_str();
                cout << macIPChar << "\n";
            UserInfoDatabse::sharedDatabase()−
                > addMACIPInWhitelist(macIPChar);
            }
            x = strtok(NULL, split);
        }
}

void systemnetworkManager::getMACIPofnewUser(){
struct newUseripmac
{
char IPMAC[10];
};
fstream infile("ARPPacket.txt", ios::in|ios::nocreate);
newUseripmac data[10];
cout << sizeof(data[0]) << endl;
for(int i = 0; i < 10; i + +)
{
```

```
infile. seekg(i * 32, ios: : beg);
infile. seekg(22, ios: : cur);
infile. read((char *)&data[i], sizeof(data[0]));
cout << data[i]. IPMAC[10] <<  << endl;
fin. close();
}
}
```

UserInfoDtabase.hpp

```
#ifndef  UserInfoDtabase_hpp
#define  UserInfoDtabase_hpp

#include  < stdio. h >

class  UserInfoDatabse  {
public:
      static  UserInfoDatabse  * sharedDatabase();
      bool  haveIPMacAddress(char  *);
      void  displayvalidatedresult();

protected:
      char *  ipmacAddresArray[100];

private:
      UserInfoDatabse();
      UserInfoDatabse(const  UserInfoDatabse&);
      static  UserInfoDatabse  * m_databse;
};
```

UserInfoDtabase.cpp

```
#include "UserInfoDtabase. hpp"
#include < iostream >
using namespace std;
#define ARRAY_LENGTH 20

static int arrayIndex = 0;

UserInfoDatabse: : UserInfoDatabse(const UserInfoDatabse&) {
}
UserInfoDatabse * UserInfoDatabse: : m_databse = new UserInfoDatabse();
UserInfoDatabse * UserInfoDatabse: : sharedDatabase() {
    return m_databse;
}

void UserInfoDatabse: : displayvalidatedresult(char * ipmacAddress) {
string ipmacAddressString = ipmacAddress;
        cout << "add into blacklist with " << newUseripmac <
            < \n;
}
  for (int i = 0; i < arrayIndex; i + +) {
        string newUseripmac = ipmacAddresArray[i];
        if (ipmacAddressString. compare(newUseripmac) == 0) {
         cout << "update ARP cache table with " << newUseripmac <
            < \n;
}
 return 0;
}
```

```
bool UserInfoDatabse: : haveIPMacAddress(char ∗ ipmacAddress) {
string ipmacAddressString = ipmacAddress;
UserInfoDatabase :: sharedDatabase() → addMACIPInBlacklist(newUseripmac);
 cout << "add into blacklist with " << newUseripmac << "\n";
bool result = FALSE;
    for (int i = 0; i < arrayIndex; i + +) {
        string newUseripmac = ipmacAddresArray[i];
        if (ipmacAddressString. compare(newUseripmac) == 0) {
            result = TRUE;
 system("arp –s newUseripmac ");
 cout << "update ARP cache table with " << newUseripmac << "\n";
        }
    }
    return result;
}
```