Ethernet Wrapper: Extension of the TCP Wrapper

MoonSang Kwon, Jiman Hong, Yookun Cho
System Software Research Lab.
Dept. of Computer Science and Engineering
Seoul National University, 151–742, Seoul, Korea
151–742, Seoul, Korea
{kmscom, gman, cho}@ssrnet.snu.ac.kr

Abstract

One of the popular network security programs supporting host access control is the 'TCP Wrapper' [13]. TCP Wrapper is a software-only system and many computers connected to the Internet are using it. But, TCP Wrapper does 'IP address–based' access control. IP address is not such a reliable source when authenticating a host. In this paper, we point out two possible attacks against the TCP Wrapper, propose a new way to prevent them, and describe the prototype implementation, Ethernet Wrapper. By adding an 'ethernet address check', we augmented the TCP Wrapper. The test results showed that Ethernet Wrapper can prevent such attacks effectively.

1. Introduction

Network removed the time and distance constraints in using computer systems, and also changed the way to cope with the computer security problems. Just isolating the computer room and controlling the entrance is not sufficient any more. The facts that we should be able to manage the users who can use the valuable computer systems and only the qualified users should be able to use the computer systems are beyond discussion.

Access control mechanism, which allows only the qualified users to use the computer system, provides this functionality. One of the essential functionality of access control is "authentication". When someone requests a permission to use a computer system, we should know the identity of the requester to decide whether we should allow this or not. The authentication process can be divided into two parts, 'user authentication' and 'system authentication'. 'User authentication' is to identify the user himself. Normally 'user authentication' uses 'Something the user knows' (e.g. passwords, PINs)', 'Something the user has' (e.g. ID cards, SmartCards)', or 'Something the user is' (e.g. voiceprint identification'). Among these, 'Something the user knows' is the most general way. 'System authentication' or 'host authentication' is to identify the host from which the user tries to access the computer system. According to the host from which the user requests the access, we decide whether to allow or deny the access request. The combined use of the 'user authentication' and the 'system authentication' provides stronger access control. For example, if the system supports only the 'user authentication' \(^1\), the attacker may get an id and password information and can access the target system from any system. But, if the target system also supports the 'system authentication' and limits the hosts which can access it, the attacker should attack other systems that are allowed to access the target system.

The most popular program that supports the 'system authentication' is the 'TCP Wrapper' [13]. The administrator can specify which service requests from which hosts should be allowed or denied in advance. Only the service requests from the allowed hosts are accepted. Most of the Internet hosts managed by an administrator who is concerning the computer security problems are using the TCP Wrapper. But, the TCP Wrapper authenticates the source host using the IP address, which is not reliable enough.

In this paper, we show two possible attacks against the TCP Wrapper, propose a new way to prevent these attacks, and describe the prototype implementation. The last of this paper is composed as follows. Section 2, 'The limit of the TCP Wrapper' describes how the TCP Wrapper provides the system authentication, and shows two possible attacks. Section 3, 'Ethernet Wrapper for a local network' describes a new method we propose to prevent the attacks especially for a local network. We describe how to extend the 'Ethernet Wrapper' for an external network in section 4, 'Ethernet Wrapper for an external network'. Section 5, 'Usage Example' shows a usage example and finally we conclude in section 6, 'Conclusion'.

\(^1\)Normal UNIX system that does only user id and password check.
2. The limit of the TCP Wrapper

The 'TCP Wrapper' is a software-only firewall that provides host access control to a single host in which the TCP Wrapper is installed. It authenticates the host using the IP address of the source host. In this section, we show how the TCP Wrapper provides host access control, and two attacks against the TCP Wrapper.

2.1 Host access control using the TCP Wrapper

General way to provide the TCP/IP network services in UNIX is using a super server, inetd daemon. For example, telnet service is provided as follows.

- A user requests a telnet service to a UNIX server host using a telnet client program.
- Inetd daemon, which is running at the server host, takes the request. It knows that this request is for a 'telnet service' from the port number, on which it was listening, and executes a 'telnet service daemon'. Finally the inetd daemon turns over the new request to the telnet daemon.
- Telnet daemon executed by the inetd daemon continues to communicate with the telnet client using the telnet protocol [9].
- Inetd daemon turns back to the sleep state waiting for a new service request to come.

The TCP Wrapper provides host access control by slightly modifying the above steps. In a system that using a TCP Wrapper, step 'b' is changed as follows. Instead of executing the 'telnet service daemon' directly, the inetd daemon executes the TCP Wrapper. When the TCP Wrapper is executed, the telnet server program is passed as an argument. The TCP Wrapper process gets the IP address of the host, which has requested this telnet service, from the network connection given by the inetd daemon, and based on the source IP address, decides whether to allow or deny this request. Rules about which service requests from which hosts should be allowed are specified in two different access control files, normally 'hosts.allow' and 'hosts.deny'. If the (service, host) pair is specified in the 'hosts.allow' file, TCP Wrapper executes the program passed as an argument. If the (service, host) pair is specified in the 'hosts.deny' file, TCP Wrapper executes the trap program if a trap program to execute is specified, and denies the network request. If the (service, host) pair is not specified in both of the files, the service request is accepted 2. Here are examples of the '/etc/hosts.allow' and '/etc/hosts.deny' files.

<table>
<thead>
<tr>
<th>in.telnetd: LOCAL, .snu.ac.kr</th>
</tr>
</thead>
</table>

Table 1. Example of the '/etc/hosts.allow' file

Both files are specifying the access rules for the 'in.telnetd' service. In both of the files, each line is specifying the rules. Each line consists of three fields delimited by a ':' character. The first field represents the service. The second field specifies the hosts to be allowed (if used in the '/etc/hosts.allow' file) or denied (if used in the '/etc/hosts.deny' file). We can specify the hosts with a DNS name or an IP address. The last field, which may be omitted, specifies the program to execute when the service request from the hosts specified at the second field is arrived. According to the access control rules specified above, when the 'in.telnetd' service is requested, TCP Wrapper first refers the '/etc/hosts.allow' file. It searches the source host at the second field. 'LOCAL' means the local system itself in which the TCP Wrapper is installed. '.snu.ac.kr' means the hosts whose domain names end with '.snu.ac.kr'. The third field is omitted in this example. If the source host is not specified here, the Wrapper refers the '/etc/hosts.deny' file. The keyword 'ALL' means all hosts. Thus, in this example, every request for the 'in.telnetd' service which is not processed at the first check, is processed according to the rule specified in '/etc/hosts.deny' file. The third field is specified to execute the 'finger' command for the source host and send the result to the user 'kmscom' using an e-mail.

2.2. The limit of the TCP Wrapper – two possible attacks against it

Host access control using the TCP Wrapper seems to be effective to a certain degree, and it is being used in most of the UNIX systems connected to the Internet. But, because it authenticates the source host based on an IP address or a DNS name, some attacks are possible.

2.2.1 DNS server attack

In this attack, an attacker first compromises the DNS server that the target system is using. After compromising the DNS server, the attacker modifies the DNS server's database so that it replies with false information. For example, let's assume that the '/etc/hosts.allow' file of the target host is specified as follows.

<table>
<thead>
<tr>
<th>in.telnetd: secure.snu.ac.kr</th>
</tr>
</thead>
</table>

If the attacker’s host is using an IP address of '149.49.129.1' and a DNS name of 'hack.snu.ac.kr', the attacker can modify the DNS server's database, so that it replies the DNS name for the IP address '149.49.129.1' is 'secure.snu.ac.kr'. The Wrapper may do double checks for the DNS name and
the IP address. But, both checks will be passed if the DNS server has been fully compromised. 'DNS server attack' is possible because the TCP Wrapper believes the reply of a DNS server which can be modified by the attacker. The extension of the DNS protocol that using a digital signature is being developed [1]. But most of the DNS servers are not supporting it. And, if the DNS server is fully compromised, the digital signing itself is not seemed to be great help.

2.2.2 IP address Changing attack

Compromising a DNS server is not impossible, but it is rather difficult. Much easier way to attack the TCP Wrapper–installed system exists. The attack is done as follows.

First, the attacker manages to know the hosts that are allowed to access the target system, and then finds one that is inactive. 'Inactive' means that the host does not show any network–related activities as if it is not connected to the network. Even when the attacker cannot find one, this does not discourage him. He may wait till one of the hosts is to be power–off or make one of the active hosts that are allowed to access the target system be inactive, using other kind of attacks such as SYN flooding [2]. Finally, the attacker changes the IP address of his host as that of the inactive host. From now on, the attacker can access the target system by spoofing his IP address.

Of course, for this attack to be done successfully, the attacker’s host should be located at the same local network where the 'inactive' host is connected. But, in many cases, many different user domains are sharing the same local network, and each domain is using just part of the IP address subnet 4.

Figure 1 shows an example of the 'IP address changing attack'. Host A is the target system and its IP address is '149.49.129.10'. It allows access for the 'in.telnetd' service from the host B, whose IP address is '149.49.129.11', but does not allow access from the host C, whose IP address is '149.49.129.12'. All the three systems are connected to the same local network. First, the attacker at the host C makes the host B inactive, or waits till it becomes inactive. Then he changes host C’s IP address as that of the host B, in this case '149.49.129.11'. Finally, he accesses the target system, host A. Changing the IP address is much easier than compromising the DNS server. So, this kind of attack can be done very effectively. And, in this case, because the attacker’s host is at the same local network, the attacker can run a sniffer program that captures the user’s id and password with ease. This kind of attacks are possible because the TCP Wrapper authenticates the source host with the IP address, which is not reliable enough. To solve these problems, we need another information that is more reliable.

3 Ethernet Wrapper for a local network

As you have seen in section 2, identifying only the IP address in authenticating a host is not enough. We need something new, and 'ethernet address' is the one we choose. From here, we consider only the ethernet network [7]. Ethernet address is a unique 48–bit hardware address that all ethernet network cards have. ethernet address has the following characteristics against the IP address.

- Each ethernet network card has a factory–made unique ethernet address.
- The sender of the ethernet network packet should know the ethernet address of the destination system (at least in a local network).
- Every ethernet packet contains the ethernet addresses of the source and the destination.
- The IP address is a logical one, but the ethernet address is a physical one.
- Changing the ethernet address of the sender is more difficult than changing the IP address

You may wonder if it is really more difficult to change the ethernet address than to change the IP address. The IP address is a logical one, and you can change it just by changing the numerical values in your system configuration files. It can be done within one minute. And many IP spoofing attacks are popular in the Internet world. But, we have never heard about "ethernet spoofing". Ethernet address also can be modified, but to do so, the attacker should change the network module in the operating system and recompile it, which is at least more difficult or time consuming than modifying the IP address. Recently we have found

---

3 ‘Target system’ means the system which the attacker is trying to access and the Wrapper is installed on.
4 At least here we are working.
an Internet document about cloning the ethernet card by re-programming the EEPROM where the ethernet address is stored [4]. Re-programming the EEPROM is much more difficult than changing the IP address, we think ³. Based on the hypothesis that 'ethernet address spoofing is more difficult than IP spoofing', we modified the authentication process of the TCP Wrapper as follows. When authenticating the source host, we first do the IP address check as the original TCP Wrapper program does. If the access is allowed, then we perform additional ethernet address check. This check is done like this. First, we assume that we know the (IP address, ethernet address) pairs for all the hosts that want to access the target system, where the Ethernet Wrapper program is installed. At the ethernet address checking phase, we check if the ethernet address of the source host is same to that in the local (IP address, ethernet address) database. If the ethernet address is not same, we think, the source host is not the original host, and someone else is using the IP address at some other host. A result this connection request is denied. If the ethernet address of the source host is same to that saved in the local database, this service request is accepted. Now the two attacks explained above will fail. The service request will pass the IP address check, but not the ethernet address check.

If you don't agree, you don't have to read this paper any more.

³If you don't agree, you don't have to read this paper any more.

3.1. Using the ethernet address in host access control

In an ethernet network, the data of high–level protocols such as IP, TCP, or UDP, is transferred from the source to the destination within an ethernet packet frame. Figure 2 shows the paths that network packets of high–level protocols pass when they are arrived at the ethernet network card. The packet arrived at the network cable passes each protocol layers in sequence. For example, the TCP packet first arrives at the network card then passes the ethernet layer, where the ethernet address of the sender is available. Then it passes the IP layer, in which the source and destination IP addresses are available. And, finally it arrives at the TCP layer. The TCP Wrapper operates at the application layer. When the TCP Wrapper checks the IP address, the ethernet address is also available, and we can use it in authenticating the source host. But, unlike the IP address, there is no system call that we can use to get the ethernet address for a network connection. We should make some other mechanism to get the ethernet address. Another difficulty in using the ethernet address is that the ethernet address has meaning only in a local network where the ethernet packet frames are broadcasted. The source address of an ethernet packet frame is changed to that of the gateway when it passes a gateway. So, ethernet address can be used only in a local network. For now, we consider only the hosts that are at the same local network with the target host. This is why we named the title of this section "Ethernet Wrapper for a local network" instead of "Ethernet Wrapper". In this section, we only consider the local hosts, but we will also provide solution for the hosts that are not in a local network later.

3.2 Design of the Ethernet Wrapper

Checking the ethernet address for the TCP/IP service request is not seemed to be easy at a glance. The ethernet address, not like the IP address, is not available instantly for the TCP/IP connection, or socket file descriptor. We want to enhance the host access control functionality of the TCP Wrapper by adding an ‘ethernet address checking’. An ethernet address has meaning only in local network. So, we are sure that the additional check can be helpful to enhance the access control at least for the hosts at the local network. In this section, we describe the prototype design of the "Ethernet Wrapper (from here referred as EW)". We consider only the hosts at the local network for now. The EW is composed of two programs, "Ethernet Address Monitor (from here referred as EAM)" and "Ethernet address Checker Module (from here referred as ECM)". The two programs run independently. The EAM monitors the ethernet address, and the ECM refers the monitoring results in checking the ethernet address of the source host. One of the two difficulties in using the ethernet address discussed above, was 'how to get the ethernet address of the source host at the application layer'. If we want to get the ethernet address as we get the IP address, we should add a new system call that takes the IP address as input and returns the ethernet address for that IP address. This is not such a difficult work, but it will introduce another problem. It is not desirable to modify the operating system to support an application, which hurts the portability. We should not make an application that forces the operating system to be modified. To overcome this difficulty, we decided to use the "Ethernet Address Monitor". The EAM monitors the changes of the ethernet addresses of all hosts that are connected to the local network. We can
monitor the ethernet addresses by inspecting all the ARP and IP packets. Of course, \{IP address, ethernet address\} information of all the local hosts should be ready in advance to detect the changes. Running the EAM in learning mode can do this, or the administrator can record it in person. When it finds a host that uses a different ethernet address, the EAM records this change to the local monitoring file. The ECM is part of the original TCP Wrapper. The EW first does an IP-based access check for the host that requests the service. If the IP-based access check is passed, the ECM module inside the EW does the ethernet address check. The check is very simple. The ECM reads the record from the monitoring file in which the EAM has recorded the monitoring results. If the record for the IP address of the host that requested the service says that the ethernet address has been modified, the ECM denies the request.

The ECM is part of the original TCP Wrapper. The EW first does an IP-based access check for the host that requests the service. If the IP-based access check is passed, the ECM module inside the EW does the ethernet address check. The check is very simple. The ECM reads the record from the monitoring file in which the EAM has recorded the monitoring results. If the record for the IP address of the host that requested the service says that the ethernet address has been modified, the ECM denies the request.

change has been recorded, the access request is denied and the trap program is executed if it is specified. If the ethernet address change has not been recorded, the service request is accepted and the telnet server program is executed.

d. If the telnet service is accepted, the telnet server process continues to communicate with the telnet client using the telnet protocol [9].

e. 'in.tel daemon' turns back to the sleep state waiting for a new connection request to come.

In step b' and c, if the ethernet address checking of the EW occurs before the packet capturing and recording of the EAM, the attacker may pass the check of the EW, which we do not want. This is not a general case, but to minimize the chance, we delay the EW check one or two seconds (the duration may be specified by the administrator). When the EW does the 'ethernet address check', it calls sleep() function to sleep for a short time (more than 1 second) to give the EAM enough time to capture the request packets.

3.3 Prototype implementation

In this section, we describe how we implemented the two programs, EAM and EW.

3.3.1 Ethernet Address Monitor

The role of the EAM (Ethernet Address Monitor) is detecting the changes of the ethernet addresses. A well-known program that does the similar work was modified to implement the EAM. As you may know, it is 'ARPWATCH' [5]. This program monitors the local network and mails or logs the ethernet address changes. It is using the Berkeley Packet Filter library [6, 11] and operates in a promiscuous mode to capture all the ARP request and reply packets. We modified it to monitor the IP packets also and record the changes into an additional monitoring file.

ARP (Address Resolution Protocol) is used to get the ethernet address (physical address) for an IP address [8]. In an ethernet based network, a host must know the ethernet address of the other host with which to communicate (we are considering only the local network here). Each host is saving the ethernet addresses of other hosts that are connected to the same local network in its own local cache. If the ethernet address is not in the local cache, it broadcasts an ARP request packet that contains the IP address of other host. The host whose IP address is contained in the broadcasted ARP response packet captures the packet, and replies an ARP reply packet that contains the ethernet address for that IP address. The host that has broadcasted the ARP request packet receives this ARP reply packet and saves the \{IP address, ethernet address\} information at the local cache. After that it uses the saved ethernet address to communicate with that host.

Figure 3 shows the overall Ethernet Wrapper system architecture. Now, the steps that a telnet service request for the host that using the EW is processed are as follows.

a. A user requests a telnet service to a UNIX server host using a telnet client program.

b. 'in.tel daemon', which is running at the server host, takes that request. It knows that this request is for a 'telnet service' from the port number which it was listening on, and executes the EW (Ethernet Wrapper). Finally the inet daemon turns over the new request to the EW process.

b'. The EAM captures TCP/IP packets for this telnet service request [6]. It checks if the ethernet address is not changed. If the ethernet address has been changed, the EAM records it in the monitoring file, or ignores the packets [7].

c. The EW process executed by the inet daemon does the IP address check as the original TCP Wrapper does. If the check succeeds, the EW reads the monitoring record for the source IP from the EAM monitoring file. If the ethernet address

\footnote{Because the telnet service is using TCP/IP protocol and the EAM is monitoring all the TCP/IP and ARP packets.}

\footnote{Of course, the EAM might have detected the change and recorded it long before.}
Figure 4. ARP request and reply packet structure.

Table: ARP packet structure

<table>
<thead>
<tr>
<th>field</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ether_dhost</td>
<td>Ethernet address of the sender</td>
</tr>
<tr>
<td>ether_shost</td>
<td>Ethernet address of the receiver</td>
</tr>
<tr>
<td>ether_type</td>
<td>Specifies the ethernet packet type</td>
</tr>
<tr>
<td>ar_hln</td>
<td>Ethernet address of the sender</td>
</tr>
<tr>
<td>ar_pln</td>
<td>Ethernet address of the receiver</td>
</tr>
<tr>
<td>ar_op</td>
<td>Specifies the address type</td>
</tr>
<tr>
<td>arp_sha</td>
<td>Ethernet address of the sender</td>
</tr>
<tr>
<td>arp_tha</td>
<td>Ethernet address of the receiver</td>
</tr>
<tr>
<td>arp_spa</td>
<td>IP address of the sender</td>
</tr>
<tr>
<td>arp_tpa</td>
<td>IP address of the receiver</td>
</tr>
</tbody>
</table>

Figure 4 shows the ARP request and reply packet structure. Ether_dhost, ether_shost are the ethernet addresses of the sender and receiver of the packet. To send a packet, ethernet address of the receiver should be available. Ether_type is specifying the ethernet packet type. For example, IP packet has the value of 0x0800, ARP packet has the value of 0x0806, and RARP packet has the value of 0x8035 [8]. Ar_op specifies if this packet is for ARP request or ARP reply. Arp_sha, arp_spa are the ethernet and IP address of the sender, and arp_tha, arp_tpa are the ethernet and IP address of the receiver of this packet. Possible checks in the ARP packets are as follows.

a. For the arp_spa host, check if the ethernet_shost and the arp_sha are same
b. For the arp_dpa host, check if the ethernet_dhost and arp_tha are same (for the ARP reply packet only)
c. For the arp_spa host, check if the original ethernet address and the ether_shost or arp_sha are same
d. For the arp_tpa host, check if the original ethernet address and the ether_dhost or arp_tha are same (for the ARP reply packet only)

When a 'Proxy ARP' is used, ethernet_shost and arp_sha may be different [12]. In this case, the check (a) is omitted. The information that we can get from the IP packet is the ethernet and IP address pairs of the sender and the receiver. Possible checks in the IP packets are as follows.

e. For the IP source host, check if the ethernet address is same to the address known before
f. For the IP destination host, check if the ethernet address is same to the address known before

e. The checking results are recorded in the 'NPD.mon' file. 'NPD' is the abbreviation of 'Network Pseudo Device'. For example, the network pseudo device of Solaris is '/dev/le'. In this case, the checking results are stored in the '/etc/security/le.mon' file. If two or more network interfaces exist, the monitoring file is made per each interface. The information saved in the 'NPD.mon' file for each IP address is as follows.

typedef struct {
    u_int ip;
    char ether[6];
    time_t t_registered;
    char ether_new[6];
    time_t t_detected;
    time_t t_logged;
    int is_allowed;
    int is_gateway;
} T_ether_monitor_record;

ip : 32-bit IP address of the host
ether : NIC's ethernet address of the IP host
t_registered : time that the {ip sol ether} information was registered or updated
ether_new : new ethernet address is saved here when it is detected
t_detected : time that ether_new was saved
t_logged : time that the log saying that 'ethernet address spoofing is detected' was written. This field is needed to prevent the DoS(Denial of Service) attack. If we write the log whenever we detect the 'ethernet address spoofing', the file system will be full of the log soon. We decided to give the logging interval. The default interval is 10 seconds.
is_allowed : specify whether to allow or deny the service request from the ip host. '0' means 'deny' and '1' means 'allow'. When an 'ethernet address spoofing' is detected, this flag is set to 0.
is_gateway : specify whether the ip host is a gateway or not. '1' means gateway. If specified as a gateway, this host may announce other host's ethernet address. Generally, it is desirable to disallow service requests from a gateway host.

When 'ethernet address spoofing' is detected, the new ethernet address detected is compared with the ether_new. If they are different, the new one is saved at the ether_new and t_logged is updated. If they are same and the time interval between the current time and t_logged is greater than a threshold value (the default value is 10 seconds), this event is logged and t_logged is updated to the current time.

3.3.2 Ethernet Wrapper

The EW was implemented by adding additional checking function to the original TCP Wrapper. First, it does IP address check, then for the request that passed the first check, it does the 'ethernet address spoofing' check. It reads the monitoring record from the 'NPD.mon' file which is made and managed by the EAM and checks the is_allowed field. If this field has the value of '1', the request is accepted, otherwise rejected and trap function (if specified) is executed. The second check has meaning only when the EAM is running. If the EAM is not running, the EW simply rejects the request.
4 Ethernet Wrapper for an external network

The ethernet address of a host which is not connected to the local network has no meaning. The ethernet address of a packet from an external network is set to that of the gateway, and the EW explained above cannot be used directly for the hosts that are not connected to the local network. To support the external hosts, we designed 'Local Network Connection Monitor (from here referred as LNCM)'.

4.1 Design of the "Local Network Connection Monitor"

The LNCM (Local Network Connection Monitor) is a network–monitoring program that monitors all the TCP/IP network connections whose source hosts are the local hosts. It manages the following information for each TCP/IP network connection.

\[
\text{typedef struct } \\
\{ \\
\text{u_int s_ip; } \\
\text{u_int s_port; } \\
\text{u_int d_ip; } \\
\text{u_int d_port; } \\
\text{time_t t; } \\
\} \text{T_conn_pair; }
\]

\(s_ip, s_port, d_ip, d_port\) represents a unique identifier of a TCP/IP connection. \(t\) is the time that this TCP/IP connection is detected. When a new TCP/IP network connection is detected, the LNCM saves it to the local list in memory. This information is kept valid in the list for short time (several seconds). Now the process of access control of the EW is changed as follows.

- The EW does the IP address-based checks using the '/etc/hosts.allow' and '/etc/hosts.deny' files to see whether the service request from the source host should be allowed or not.
- If the request is allowed, the EW checks if this request is from a local network. If it is from the local host, it does 'IP address spoofing check' from the monitoring file that is managed by the EAM as described above.
- If the request is from an external network, the EW makes a network connection to the LNCM, which is running at the external network to which the host, which is the source of the request, is also connected. Then it sends the source IP, source port, destination IP, destination port information to the LNCM to probe the validity of this connection.
- The LNCM checks if the connection source IP, source port, destination IP, destination port exists in the connection list. If the connection does not exist, or the time interval between the connection detection time and the current time is greater than a certain pre-specified value (default is 5 seconds), it sends a NACK reply.
- If the connection exists and the time interval is within the threshold value, the LNCM does 'IP address spoofing check' from the monitoring file that is managed by the EAM which is running at the same host.
- If no 'ethernet address spoofing' is found, the LNCM sends an ACK reply. In other cases, it sends a NACK reply.
- According to the reply of the LNCM, the EW accepts or denies the request.

Because the EW depends on the reply of the LNCM, it should be able to authenticate the LNCM and its reply. Public key cryptography such as RSA can solve this problem [10].

4.2 Implementation of the "Local Network Connection Monitor"

The LNCM should do the following works concurrently.

- Monitor all the TCP/IP network connections and add it to the connection list
- Accept the probe requests from the EW, check the connection list, and send ACK or NACK reply
- Discard the connection records which are not valid any more

The LNCM was implemented as three threads using the POSIX thread library in the Solaris. The first thread(from here referred as thread 1) captures all the TCP/IP connections and saves the \{source IP, source port, destination IP, destination port, current time\} information to the connection list which is managed using a hash table. The second thread(from here referred as thread 2) is a garbage collecting thread. It is scheduled periodically, and removes all the connection records which are not valid any more. All the records become invalid after at most 5 seconds (this value can be adjusted by the administrator). So, it should be removed soon. Every new record is added at the head of the list. So, when a connection record, which is not valid, is found, the following records can be discarded safely. The final thread(from here referred as thread 3) opens a TCP/IP server port and processes the probing requests from the EW, which are running at the external network. The hash table is protected using a mutex variable. Each thread first should gain a mutex lock for the hash table entry to modify the list.

![Figure 5. Test environment.](image-url)
5 Usage Example

To test the Ethernet Wrapper completely, we need at least two separated networks. Let's assume two networks, net1 and net2. Figure 5 shows the test environment. The EW is installed at Host A, which is the target host. The EAM is running at the Host B. Both the Ethernet Wrapper and the TCP Wrapper are on the Solaris 2.x operating system. The Host P and the Host Q are PCs and both are using a Windows 95 operating system. The network is a 10M/bps ethernet. Two tests were done. The first was for the access control of the local host, Host P. The second was for the access control for the external host, Host Q. Because the TCP Wrapper program executes the EW, we cannot get the screen output instantly. So, we show here only the output of the LNCM when an attacker tries to access the telnet service of the target host A at the Host Q. Figure 6 shows the test results. The IP address of the Host A is '149.49.129.92'. The IP address of the Host B is '149.49.130.87'. The IP address of the Host Q is '149.49.130.55'. A telnet server daemon is configured to run at the '149.49.129.92' on the port 24. Messages in italic characters are the logs we concern. We inserted several debugging messages to clear the results. When a thread code is executed, 'in thread' and 'out thread' messages are printed. When a user tries to connect to the target host at the local host, '149.49.130.55', the Ethernet Wrapper running at the target host, 149.49.129.92, sends a probe request for the {149.49.130.55:4860 -> 149.49.129.92:24} connection to the Host B. The probe request arrived before the connection information is saved by the LNCM running at the Host B. Thread 3 sleeps 1 second, then checks if this connection has no problem. In the first case, there was no problem and the LNCM sent an ACK reply. But, in the second case, as the IP address spoofing was detected, the LNCM sent a NACK reply and the second connection request was rejected successfully.

6. Conclusion

As the Internet is becoming more and more popular, security problems are also becoming more and more important. In this paper, we proposed the "Ethernet Wrapper", which augmented one of the most popular hosts access control software, TCP Wrapper. By adding the ethernet address checking, the Ethernet Wrapper can prevent attacks such as "DNS attack" or "IP address changing attack", which can not be prevented by the TCP Wrapper. The ethernet Wrapper is not perfect. Reconfiguring the network module inside the operating system or cloning the ethernet NICs may defeat the Ethernet Wrapper. But, it is certain that the Ethernet Wrapper can make the attacker do more work to successfully attack the target system.