Protection Against Internal Intruding using Host Identifier Authentication

Basil Y. Alkhayaat
College of Computers Sciences and mathematics
University of Mosul

Abdulsattar M. khidhir
Technical Institute
Commission of Technical Education/Nenivah

Omar A. Ibraheem
College of Computers Sciences and mathematics
University of Mosul

Received on: 24/11/2008 **Accepted on: 04/12/2008**

ABSTRACT

Network security has become one of the most important interesting areas for researches. Protecting the network can be done by many mechanisms. Among the most effective one is the network firewall. While the firewall protecting the network from the external intruding, it does upward nothing about the internal intruding. Internal intruding or Inside attacks can lead to a big loosing. One of these attacks is attaching an unauthorized host to the network to get benefits of using the network resources provided by the server (like Internet service) or to leak information into the outside.

To solve this problem, this paper suggested that two new programs have to be built under windows operating system in a client server model, one at the client and another at the server. The first supplies an authenticated unique host-ID to each packet destined to leave the network, while the latter verifies these information, allowing the authenticated packet to pass into its destined while dropping and documenting the unauthorized one.

This work is considered as a logical extension of the conventional network firewall and can be installed with any of these firewalls. While the conventional firewall is protecting against outside attacks, this work is protecting against one of these inside attacks.

Keywords: protection, intrusion, authentication, host identifier.

الحماية ضد التطفل الداخلي باستخدام توثيق مُعّرف الحاسبة المسؤولة

باسل الخياط
كلية علوم الحاسوب والرياضيات
جامعة الموصل

عبد الستار خضر
المعهد التقني
هيئة التعليم التقني/الموصل

عمر إبراهيم
كلية علوم الحاسوب والرياضيات
جامعة الموصل

تاريخ الاستلام: 2008/11/24 **تاريخ القبول: 2008/12/04**

الملخص

أصبحت أمنية الشبكات واحدة من أهم الميادين إثارة للباحثين. من الممكن حماية شبكة الحاسبات بعدة وسائل ويعد جدار النار الأهم بين هذه الوسائل. حيث يكون فعالا في حماية الشبكة من التطفل الخارجي، إلا أنه لا يقوم بأي حماية تقريبا من التطفل الداخلي. وهذه الهجمات قد تؤدي الى خسارة كبيرة. أحد انواع التطفل الداخلي هو ارتباط حاسبة غير مخولة الى الشبكة من قبل. وقد يحدث هذا بشكل غير ملحوظ مما يؤدي بدوره إلى إمكانية

الاتصال مع خارج الشبكة لتسريب المعلومات أو لاستخدام خدمات الشبكة (مثل خدمة الانترنت) بصورة غير مخولة.وهذه الأنواع من التطفل قد تكون خطرة جدا بالنسبة للمؤسسة.

يقدم البحث حلا لهذه المشكلة عبر تكوين برنامجين تحت بيئة نظام ويندوز أحدهما يوضع في كل حاسبة مستفيد والآخر في الحاسبة المُخَدِم. إذ يقوم البرنامج في حاسبة المستفيد بتجهيز كل حزمة معلومات خارجة منه إلى الشبكة الخارجية بمُعَرِف الحاسبة المسؤلة عن ارسال الحزمة الحالية. كما يقوم البرنامج بتوثيق وتشفير هذه المعرفات من أجل ضمان حقيقة وصحة المعلومات عند استلامها من قبل البرنامج الثاني في حاسبة المُخَدِم.

أما البرنامج في الحاسبة المُخَدِم فسوف يقوم باستلام حزم المعلومات من الحاسبة المستفيد والتأكد من وثوقية هذه الحزم ومن كونها مخولة. ثم يقوم بإعادة هذه الحزم إلى أصلها وتمريرها إلى الخارج وفي نفس الوقت يمنع الحزم الغير مخولة من المرور بالإضافة إلى تسجيل ذلك الخلل.

يعتبر هذا العمل كتوسيع منطقي لجدران النار الاعتيادية ومن الممكن تنصيبه مع اي منها. فبينما تقوم جدران النار الاعتيادية بالدفاع ضد الهجمات الخارجية، فأن العمل المقترح يقوم بالدفاع ضد احد الهجمات الداخلية.

الكلمات المفتاحية: الحماية، التطفل، التوثيق، مُعّرف الحاسبة المسؤولة.

Introduction

According to the annual CSI/FBI, Computer Crime and Security Survey, the number of successful attacks from the inside is roughly equal to numbers from the outside. So just protecting from outside attacks covers just half the threats [1, 2].

Attaching an unauthorized host to the network can be classified under internal intruding; the solution for this problem is that the network firewall at the server has to know the source of the packet in order to distinguish between trusted and untrusted hosts. This enables the firewall to let only the packets that came from the trusted hosts to pass the network.

The conventional firewall can distinguish the source of the packets by reading both fields: source Media Access Control (MAC) address and source IP address. This is not always true, according to two reasons, first: Dynamic Host Configuration Protocol (DHCP) is a protocol, which enables a host to obtain an IP-address dynamically from a DHCP-server upon boot time. This means that the host can have a different IP-address whenever it is rebooted. Second, the internal adversary can change the values of these two fields (including the MAC address) of his packets or frames into the corresponding fields of an authorized host.

To solve this problem, we need to manage an enumeration of all protected hosts behind the firewall, and initialize each host with its unique ID.

Moller J. and Donbaek T. [3] used a normal string as the host ID which read from a configuration file in each client. But this method has disadvantages: first, the string can be repeated by the user or the adversary (whether he/she meant or not) on other hosts. Second, the string can be stolen or even the whole configuration file by the adversary.

Also Moller J. and Donbaek T. propose that every packet that destiny to leave the network, has to contain the host Identifier (ID) that send this packet. When this packet reached to the server, the extended firewall has to retrieve the host-ID to

determine the action (Drop/Pass) of this packet. The whole work is done in Linux (open source) operating system.

This paper proposed two points: first a new solution to overcome the weakness of choosing the Host-ID which by: first, the host ID can be a serial number of one of the hardware components e.g. manufacturer hard disk serial number (chosen in this case), manufacturer Compact Disk (CD) serial number, or CPU Identification number…etc. which are always unique. Second, this paper also proposed that the proposed application itself read the host ID at the beginning of its execution, not from file to prevent the adversary from copying or stealing the host ID to use it in his application. And even if the adversary has stolen the hooking application this will be useless. Second, the big challenge is that the whole proposed system is running under windows operating system, which is not open source code.

TCP/IP Communication Protocol

TCP/IP (Transmission Control Protocol/Internet Protocol) communication protocol is assumed, the TCP/IP protocol suite has now become a universal communication protocol. Since its development has paid little attention to traditional security aspects [4]; security means become essential to be added in a way that can protect the LAN which uses this protocol from one of the important security issues which is; the internal attacks.

This paper proposes a new method that implemented on a Client Server (C/S) model for a LAN. Clients and server are cooperate in such a way that each of the clients does authenticate each of its (hosts) to the server and the server will verify these information in turn.

This paper attempts to improve the security process in a communication network that employs TCP/IP. It focuses on defending against the internal attacks. It first identifies client server model, then suggests application scenario for each of the clients and the server in order to authenticate and verify the extra security information in order to enhance the network security for LAN model computer networks.

As mentioned earlier, the conventional firewall is unable to distinguish between legal and illegal network packets based solely on the information available in the TCP/IP-protocol. So, it needs more information. TCP/IP is still a very capable communication protocol suite, and it is not likely to be replaced, at least not within the next few years. But the information contained in an IP datagram is not enough therefore something additional is needed.

The proposed firewall must ensure that no information leaves the trusted network, except when the communication link originates from a trusted source (a trusted host). This means that the protected computers by the extended firewall must provide the extended firewall with these extra information in every single packet destined to going out.

The process of finding and adding these information to each packet and verifying of these information requires many changes to the operating systems of all hosts (clients and server), on the protected network, that need to communicate through the firewall.

Client-Server Model

The term Client/Server was first used in the 1980 in reference to Personal Computers (PCs) on a network. The actual Client/Server model started gaining an acceptance in the late 1980s [5]. The Client/Server model has become one of the central

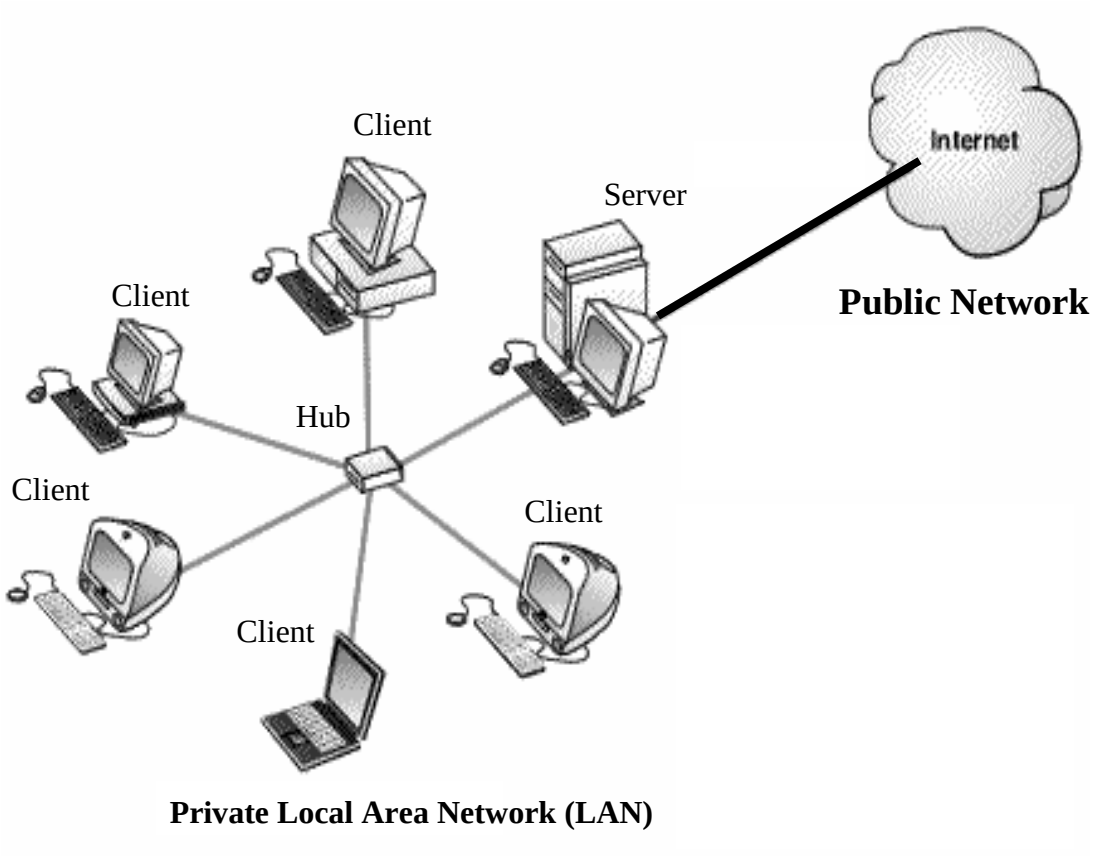

Figure (1): Client/Server model in a LAN topology connected to the Internet

ideas of networking. Most networking applications are being written today using the Client/Server model. The same can be said about Internet [6].

Client/Server is a network application architecture environment where the control of data is established at a server node and is available for access from the clients. Client/Server describes the relationship between two computer programs in which one program, the client, makes a service requested from another program, the server, which fulfills the request [7].

These days, most computers are multi-tasking (can run many programs simultaneously), so a single computer can run the server and client programs at the same time [8]. Often, in such a case this computer is a server of the internal network and is a client of the outer network. This work proposes a Local Area Network (LAN) topology based on a Client/Server model in which the server of the LAN is a client of a public network (Internet) and each LAN's client can connect to the public network through only the LAN's server. Figure (1) illustrates the relationship between the clients and server in a LAN topology connecting to a public network (Internet).

Framework

Protecting a Local Area Network (LAN) from an internal intruding needs a cooperation between two parts: client and server. This paper proposes that each client must send an extended information with each packet destined to the outside in order to authenticate the host which sent the packet. On the other hand, the server must verify these information and passes the authorized packet, while drops (and save in log file) the unauthorized one.

This process require a full control of the incoming/outgoing packets. One aspect of this paper is the implementation of the extended firewall (in both parts) on the windows operating system. Since the source code of the TCP/IP, which is proprietary to windows operating system, is not accessible, and since it is not possible to manipulate the packets in any TCP/IP protocol suit from level above the TCP/IP driver layer, these make the problem much more complicated. To overcome this problem, a hooking technique is developed during this work in order to control the packet at the point that links between the protocol driver and the NIC card(s), which is represented by the NDIS.

The work in this paper is, therefore, composed of two parts:

1-NDIS Hooking.

2- Authentication and verification of the extended information.

NDIS Hooking

The term hooking represents a fundamental technique that controls a particular piece of code execution. It provides a straight forward mechanism that can easily alter the operating system's behavior as well as third party products, without having their source code available [9].

The proposed software must be run in an user mode. This makes a comfortable and familiar interface to the user for handling routine data and gets the required information.

The user mode program can control a device driver by I/O commands, since the NDIS driver is not supporting I/O commands that provide a tool to deal with the user mode program, a new driver must be developed to hook the NDIS driver. This new driver will support the I/O commands in order to enable them to deal with the user mode program.

When the windows user-mode program can hook the NDIS, this program can monitor, control, add, and modify the NDIS incoming/outgoing packets.

Therefore, and in order to implement NDIS hooking it must be composed of two parts:

1- NDIS hooking driver.

2- NDIS hooking application.

NDIS-Hooking driver

The NDIS-Hooking driver is logically similar to the NDIS Intermediate (IM) driver, but it is implemented differently. It inserts itself between TCP/IP and all of the adapters that bind with it as shown in figure (2).

When TCP/IP sends a packet, the packet comes to the NDIS-Hooking driver first. Likewise, packets that are to be indicated (received) on TCP/IP will go to the NDIS-Hooking driver first.

The NDIS hooking driver intercepts services that are exported by the windows NDIS wrapper at a point in the load sequence before NDIS protocols begin their binding process. These services are implemented in NDIS library. They deal with the TCP/IP stack and the NIC driver by sending/receiving a single or array of packets and other system events.

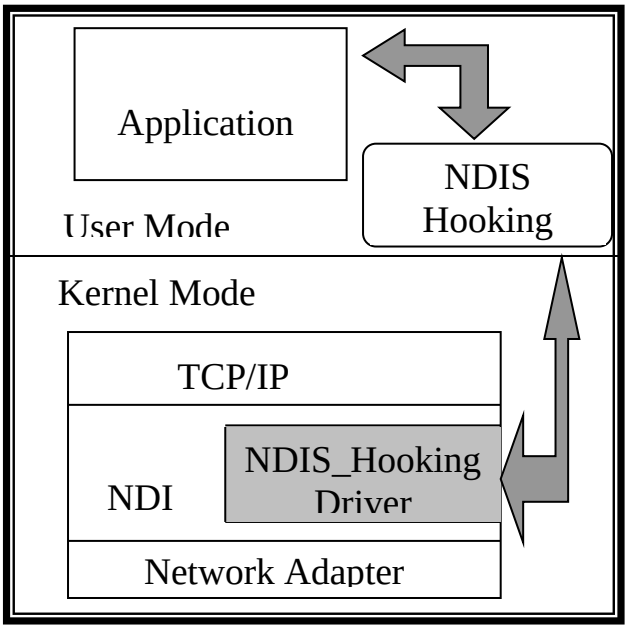

Figure (2): NDIS-hooking driver with relation to user

NDIS Hooking Application

The second part of NDIS hooking is a user mode application, it controls the hooking driver, which means controlling the packet traveling. The hooking application must have the following functions:

1- Loading/Restoring the hooking driver for the specified adapter.
2- Controlling the original NDIS driver (by hooking driver), so that the ability for moving the NDIS packet from the kernel mode to user mode and vice versa.
3- Continuing for the sending/receiving operation.
   The functions in the user mode which do these jobs are programmed in a Dynamic Link Library (DLL) file using visual C++.net language.

The packet treatment process must continue until the application end (in case of resetting to the old driver). The execution of this process (function) containing this loop will prevent this function form terminating and the inability to control the rest of the program. To solve this problem, threading must be applied.

Threading is a technique that enables the programmers to execute one or more functions (in the same program) concurrently with the remaining parts of the program.

Getting the Host Identifier

In order to get the host identifier at application running time, Windows Management Instrumentation (WMI) must be used.

5.6.1 Windows Management Instrumentation (WMI)

The Windows Management Instrumentation (WMI) job is to provide status or performance data about computers (possibly in networks) to applications. It retrieves these data from various sources like the registry or other custom defined sources. In

particular, the actual location of data is hidden in favor of a more abstract view of the entirety of status or performance data [10].

WMI introduces the notion of a managed object. A managed object is some logical or physical component on a computer that is managed by WMI. For example, a partition on a hard disk drive is a managed object. Another example would be a physical component like the processor.

Web-Based Enterprise Management (WBEM)

WMI is a Microsoft's implementation of Web-Based Enterprise Management (WBEM) which is an industry initiative and has evolved into Distributed Management Task Force (DMTF) to develop a standardized, non-proprietary means for accessing and sharing management information in an enterprise network [10]. WBEM will result in technology that enables customers to collect, associate, and aggregate management data from diverse sources. The WBEM initiative is intended to solve the problem of collecting end-to-end management and diagnostic data in enterprise networks that may include hardware from multiple vendors, numerous protocols and operating systems [11].

The goal of WBEM was first, to harness the power of the web for management interoperation, which is mean, that management data and the actual managed elements should be accessible from the Internet; second, build a Common Information Model (CIM) for management which is mean that all managed entities are accessed through a single unified standard protocol [11, 12].

CIM is based on an object oriented model, and it is used by different management applications to have a common understanding of the managed elements regardless of how that information is stored or transported [13].

WMI Architecture

WMI architecture consists of [13, 14]:

1- Consumers: the application that obtains information about managed objects from WMI.

2- Provider: A Component Object Model (COM) that maintains information from a managed object and relays it to the consumer. Providers implement specific interfaces defined by CIM and are registered with WMI. By implementing these interfaces, developers can write their own providers specific to their needs. WMI allows applications and systems to be built from components supplied by different software vendors. WMI comes with a number of predefined providers, which are called standard providers. Figure (3) illustrates the relationship between providers and consumers and the interaction with WMI.

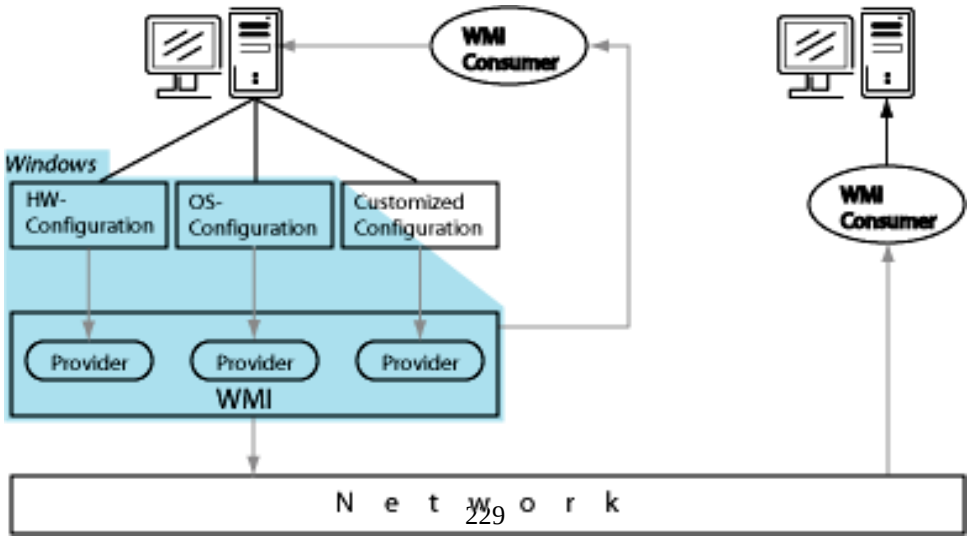

Figure (3): WMI architecture

Interaction with providers is blurred by WMI, to the client both standard and custom providers are hidden; data structures exposed by WMI do seem to have merged with the operating system.

Information about managed objects is exposed by WMI as instances of classes created by WMI providers. In general, consumers do not access these instances directly; instead, consumers query WMI, which then relays the query to the appropriate provider. From the requested subset of the managed object's properties, the provider builds a result set which is returned to the consumer, again via WMI (Figure 4). Queries can by specified by Windows Management Instrumentation Query Language (WQL).

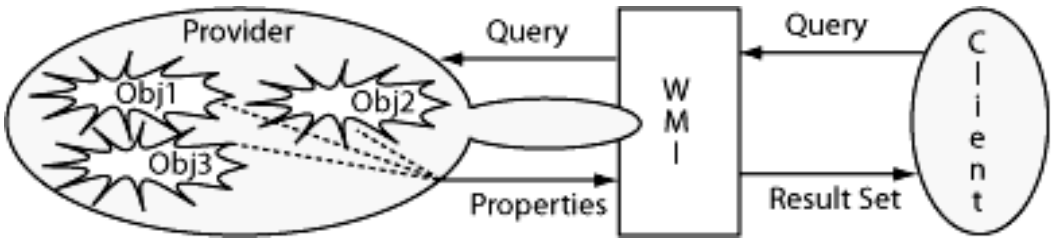

Figure (4): WMI Queries

WMI Query Language (WQL)

WQL is a subset of ANSI Structure Query Langauge (SQL) with a few extensions to fit WMI's needs. The most important difference between SQL and WQL is that WQL supports read-only queries [13].

Object Manager queries are used in order to access the management Information (managed by the CIM). The syntax of the query is defined by the WQL. Unlike SQL, WQL can only be used to retrieve data; it cannot be used in operations involving modification, insertion, or deletion of data., examples of WQL queries are:

"SELECT FreeSpace,Size FROM Win32_LogicalDisk WHERE DeviceID='C:'"

This query will only retrieve the amount of free space and the size of the logical disk named 'C:'.

Connecting to the CIM Object Manager

To be able to access the functionality of WMI, a connection to the functionality CIM object manager within the service *WinMgmt.exe* has to be established. This is done by the following steps:

- First, setup the COM in the WMI consumer by calling the *CoInitialize* function.
- Second, set the system security context into "*process-wide*" by calling the *CoInitializeSecurity* function in the consumer with an appropriate parameters.

After COM and COM security have been set up, class *IwbemLocator* is used to access the WMI service. On the other hand, *CocreateInstance* is called in order to get the interface handle to the *IwbemLocator*. This function returns the interface handle at the last argument as a *IwbemServices* object.

Finally, the connection to *WinMgmt.exe* is then done with a call to the command *IwbemLocator::ConnectServer*. The first parameter of this function specifies the CIM namespace that the consumer is connect to. The namespace for most of the pre-defined WMI structures including the structure that contains the information about the physical hard disk reside in the root\CIMV2 namespace. However, the second and third parameters are user and password. For the local machine (as in this case), these parameters must be set to NULL. The last parameter will be the *IwbemServices* object.

When the connection to *WinMgmt.exe* is done then the communication with the managed elements through the CIM object manager can be also established.

Querying the Required WMI Properties

The first step to query the required WMI properties, is to determine the class that contains instances of the required information, e.g. class *Win32_Processor* contains instances about the CPU like CPU identification number, CPU clock frequency.

This paper proposes that the serial number of the physical hard disk is the host ID. The instance of this information is contained in *Win32_PhysicalMedia* class. As for the second step, it is executing the *IwbemServices::ExecQuery* function with the "*Select*" command as one of its parameter, e.g. *"Select * from Win32_PhysicalMedia"*. The last parameter of the *ExecQuery* function is a pointer to the corresponding first (if more than one) enumeration object.

Getting the next enumeration object (if needed) can be done by executing the *IwbemClassObject::Next* function repetitively. At this point the instance of the required *Win32_PhysicalMedia* is obtained. This instance acts as wrappers implementing access to the properties requested only those by query, see figure (5).

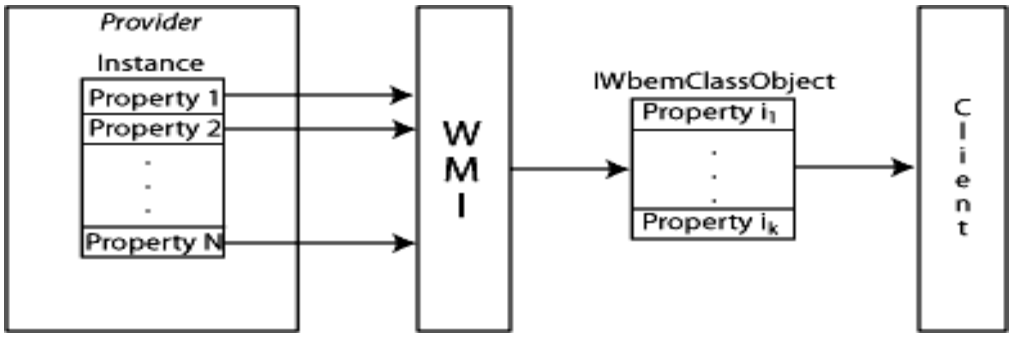

Figure (5): Accessing the required properties requested by query

Finally, to access the required WMI properties of the *Win32_PhysicalMedia*, a *IwbemService::Get* function is used with the name of the required properity as a parameter. After getting the serial number of the hard disk, this Host-ID will append to each single outgoing IP packet.

The Getting Host-ID algorithm is as follow:

Step-1- Setup the Component Object Model in the WMI consumer
Step-2- Set the system security context into process-wide in the consumer
Step-3- Define a Iwbemlocator object used to access the WMI service
Step-4- Get the interface handle to the Iwbemlocator

Step-5- Connect to WinMgmt.exe
Step-6- Determine the class that contains instances of the required information
Step-7- Query the required type information
Step-8- Get the enumeration objects with their instances
Step-9- Get the required information as a HostID

Authentication Importance

The additional information that passed from the protected hosts to the extended firewall must be authentic. So, any fabrication or even simple modification to the packet will be detected. Otherwise, the adversary is able to modify the packet's content to his own benefit, due to his physical access to the protected network [15].

Message authentication is a procedure to verify that a received message comes from the alleged source and have not been altered [16, 17].The authentication scheme must be strong enough to overcome the adversary's limited computational resources. It must be applied to every network packet; it must be small enough to fit into every packet without loss of too much bandwidth. Furthermore, it has to be very fast to calculate and validate, since the firewall has to validate network packets from a large number of hosts which, yielding hundreds or thousands of packets each second [3]. UMAC method is very fast and effective method that authenticate a whole message and the result will be a value in a form of 4 bytes[18]. This value will append also the to the end of the IP packet. Using normal method of authentication does not prevent the intruder from the ability of understanding what the packet contain. So, encryption technique is important.

Encryption Importance

As the authentication is protecting the packets from fabricating or modification, an encryption technique will prevent the intruder from the ability to understand them. This factor will be very useful in order to prevent the intruder from knowing what extra identifiers are chosen to append to each packet for making the packet as authorized one. So, encryption is a necessity [19, 20].

The DES method has been chosen because of two reasons: first, it is symmetric encryption so it is fast; second, it is still effective method.

Maximum Transfer Unit (MTU)

Two information are added to the end of each IP packet. The length of each information is as follow:

HostID: 15 bytes
UMAC value: 4 bytes

This means that each outgoing IP packet length will be increased by 19 bytes, which approximately 0.0125 of the maximum Ethernet packet (The Ethernet maximum packet length is 1514 bytes). Since, no packet must exceed the Maximum Transfer Unit (MTU) of the network [21, 22, 23], in this case (Ethernet), therefore, the MTU must be decreased by the total length of the extended information which is 19 bytes.

Necessity of Check Sum Recalculation

The error detection method used by most TCP/IP protocols is called the check sum. The check sum protects against the corruption that may occur during the transmission of a packet. It is redundant information added to the packet.

The check sum is calculated at the sender and the value obtained is sent with the packet. The receiver repeats the same calculation on the whole packet including the

check sum. If the result is satisfactory (match) then the packet is accepted; otherwise, it is rejected.

Since, the lengths of the IP and UDP/TCP packets are increased by adding the extra information, and since the contents of these packets are changed, a recalculation of the check sum for both the IP and the UDP/TCP packets are necessary.

Client's program algorithm

Step-1- Specify the network card
Step-2- Load the hooking driver
Step-3- Perform the host ID function to get the host ID
Step-4- Wait until outgoing packet is reached to the driver
Step-5- Transfer the packet from the NDIS buffer (kernel mode) to the hooking application buffer (user mode)
Step-6- Add host ID to the packet
Step-7- Perform authentication process to the packet payload using UMAC method
Step-8- Perform ciphering process to the packet payload using DES method
Step-9- Recalculate checksum of the packet
Step-10- Transfer the packet to the kernel mode
Step-11- Transmit the packet
Step-12- go to step -4-

Verification Process

When the extended firewall (verifier) receives an extended IP datagram from a protected host, it has to check whether it should accept or deny. So, the processing for each incoming message begins with loading the hooking driver for the specified adapter, and then transfers the incoming packet from kernel mode to user mode by the hooking application. This operation is also important for retransmitting the authorized packet. When the packet is in user mode, the verifying process will be easy.

The next step is deciphering the message. After that, the authenticity of the message must be checked. This is done by verifying the MAC value received from the client with the message appended to it. If this value is correct, the extended firewall will verify the authenticity for the hostID.

The verification process of the hosted is accomplished by searching in a data base file which contains the authorized information.

If MAC value was incorrect or the authenticity of the hostID was incorrect, this packet will be dropped and logging in a log file. If the packet was authorized, the original packet must be extracted from the received one by removing the extended information and recalculating the checksum. The last step is resending the packet. The problem with the server is that it may receive a large number of packets in a short time, according to this fact; the server has to process and verify the incoming packets as fast as possible. The server deciphers each packet by using a symmetric ciphering method, which is very fast comparing to asymmetric ciphering methods. Also, the server uses the fastest method of authentication process.

The matching process of the host-ID (search in each data base file) is slow. In order to increase the speed of this process, cache information is presented.

The server main procedures algorithm is as follow:

Step-1- Specify the network card
Step-2- Load the hooking driver
Step-3- Open authorized user ID file

Step-4- Wait until incoming packet is reached to the specified driver
Step-5- Transfer the packet from kernel mode to user mode
Step-6- Deciphering the packet
Step-7- Recalculate the packet authentication value using the key
Step-8- if Match with the sent one go to step -14-
Step-9- if Match the host ID with authorized one go to step -14-
Step-10- Rebuild the IP packet without the extended information
Step-11- Recalculate checksum
Step-12- Send the packet
Step-13- go to step -4-
Step-14- Send into a log file
Step-15- Drop the packet
Step-16- go to step -4-

Caching Authorized Information

In order to increase the speed of verifying the incoming extended information, caches for the authorized information are presented. This paper proposes the use of cache of Host-ID at the server:

Caching Framework

When the incoming packet is deciphered and authorized by the UMAC method, the HostID is read from the packet and search it in the HostID cache. There are two situations:

First, if it is not found, then search it in the authorized HostID file. If not found in the file, then this HostID is unauthorized and this packet will be dropped and logged. Otherwise, a new HostID entry cache is added.

Discussion

The practical side of this paper is implemented on a LAN network topology consisting of five computers; one server and three authorized clients in addition to one unauthorized client.

In order to test the proposed implementation, the execution of the client portion is beginning with reading the HostID and appending to each packet going to the server. Table (1) illustrates the clients' computer names with their HostID (retrieved by the proposed solution at the client) beside the authorization state (depending on a database at the server) and the server's action for this packet.

Table (1): Clients' computer names, HostID, their authorization state and server reaction

Computer Name	Computer HostID	Authorization State	Server's Reaction
Computer1	WD-WMA8E8787590	Authorized	Pass
Computer2	WD-WMA8E8787862	Authorized	Pass
Computer3	WD-WMA8E8787931	Authorized	Pass
ComputerHack	WD-WMA8E8787996	Unauthorized	Drop

The above table shows that each packet coming from the authorized client (Computer1, Computer2, and Computer3) was succeeded and passes this step. In the opposite site, ComputerHack had strange HostID (unauthorized), this state was recognized by the server and drop this packet as a reaction.

Any violation to the token authorization or incorrect authentication from any client will be detected and reported in a log file.

Conclusions

From the previous, we can conclude that: TCP/IP protocol has many security holes especially that can be used for internal intruding attacks, this can be done by modifying or fabricating any IP packet contents easily and this will enable the adversary to communicate with the external network. So, the HostID can be more reliable than the IP and MAC addresses. Also, the Host-ID can be used by the programmer to use it when developing a software in order to guarantee that this copy of software is the only copy that cannot execute on another host.

Also, the work is conclude that there is no unique security solution for a network; instead it is necessary to make different and multistage defense lines.

REFERENCES

[1] Gordon A., Loeb P., Lucyshyn W., and Richardson R., "CSI/FBI Computer Crime and Securty Survey", Annual Report, Computer Security Institute, 2004. http: //i.cmpnet.com/gocsi/db_area/pdfs/fbi/FBI2004.pdf.

[2] Strand L., "Adaptive Distributed Firewall Using Intrusion Detection", University of Oslo, 2004.

[3] Møller J. and Donbæk T., "Internal Network Security", Department of Computer Science at the University of Aarhus, 2001.

[4] Ahsan k., "Covert Channel Analysis and Data Hiding in TCP/IP", Department of Electrical and Computer Engineering, University Of Toronto, 2002.

[5] Hammo A., "Designing a Client/Server Model for Real Time Data On the Internet", College of Computers and Mathematical Sciences, University of Mosul, 2005.

[6] Hammoshi M., "Problem Solving by Using High Performance Computing Techniques", Mosul University, 2005.

[7] Carr S., "Networking Concepts", University of Watreloo, 1998.

[8] Bagwill R., Barkly J., Carnahan L., "Security in Open Systems", Published by Computer Systems Technology, U.S., Department of Commerce, 1994.

[9] Ivanov I., "API hooking revealed", at web site http://www.codeproject.com/webservices/articlerss.aspx?cat=1.

[10] "Web-based Enterprise Management", at web site http://www.dmtf.org/download/spec/wbem.ppt

[11] Saikat B., "An Application of the CIM Model to Diagnostics: AMIDiag the CDM way", American Megatrends, Inc., 2002.

[12] Westerinen A., Strassner J., "Common Information Model (CIM) Core Model, version 2.4.", DMTF White Paper, 2000.

[13] Jens A., "Supervision of Computer Equipment in ABB OperateIT Using WMI", Department of Automatic Control, Lund Institute of Technology, 2000.

[14] Jeffrey C., "Windows Management Instrumentation: Administering Windows and Applications Across Your Enterprise", MSDN Magazine, 2000.

[15] Tour I., "A Secure Authentication Infrastructure for Mobile Communication Services over the Internet", University of Ottawa Ontario, Canada, 2001.

[16] Bishop M., "Computer Security: Art and Science", Addison Wesley, 2002.

[17] Seberry J., Pieprzyk J., “Cryptography- An Introduction to Computer Security”, Published by Prentice Hall of Australia Ltd., 1989.

[18] Mironov I., "Hash Functions: Theory, Attacks, and Applications", Microsoft Research, Silicon Valley Campus, October 2005.

[19] Menezes A., Oorschot P. and Vanstone S., "Handbook of Applied Cryptography", CRC Press, 1996.

[20] Jiang S., Smith S. and Minami K., "Securing Web Servers Against Insider Attack", Department of Computer Science, Institute for Security Technology Studies, Dartmouth College, 2001.

[21] Sanchez M., "Iterations in TCP/IP – Ethernet Network Optimization", Faculty of California, Polytechnic State University, 1999.

[22] Hasegawa T., Ogishi T., and Toru H., "A Framework on Gigabit Rate Packet Header Collection for Low-cost Internet Monitoring System", KDDI R&D Laboratories Inc., Japan, 2002.

[23] Jelger S. and Jaafar M., "Performance of a Slotted MAC Protocol for WDM Metropolitan Access Ring Networks Under Self-Similar Traffic", University of Wales Swansea, U.K., 2002.