2008 IEEE Asia-Pacific Services Computing Conference

Zero-Configuration Personal Firewall for DLNA DMS

Shann-Chiuen Wu
Department of Accounting Information,
Tzu Chi College of Technology,
Hualien, Taiwan
E-mail: chiuen@tccn.edu.tw

Yung-Chung Ku
Department of Computer Science
and Information Engineering,
National Taiwan University,
Taipei, Taiwan.
E-mail: d93019@csie.ntu.edu.tw

Tzao-Lin Lee
Department of Computer Science
and Information Engineering,
National Taiwan University,
Taipei, Taiwan.
E-mail: tl_lee@csie.ntu.edu.tw

Abstract

When the DLNA products join the intranet of the home network, the attack from the internet will follow before one knows it. This paper will propose a design of a personal firewall for the Digital Media Server（DMS）providing the DLNA services.

Unlike traditional firewall, the access control list of this firewall can be easily configured and updated. The key concept of the design is to mimic the contact list used in the instant messaging system. If a Digital Media Controller (DMC) is added to the contact list of the DMS, this DMC can control and manage the media in the DMS. Also the DMS can reject the accessing of a DMC on the contact list by temporarily blocking.

1. Introduction

The Digital Living Network Alliance (DLNA) [1] is an international, cross-industry collaboration of consumer electronics, computing industry and mobile device companies. The main objective of DLNA is to establish an interoperable network architecture for personal computers (PC), consumer electronics (CE) and mobile devices in the home and to provide a seamless environment for sharing new digital media and content services.

There are more than 2000 products with DLNA (Digital Living Network Alliance) certification and UPnP (Universal Plug and Play) [2] capability by the first quarter of 2008. When those products join the intranet of the home network, the attack from the internet will follow before one knows it.

DLNA defines five classes of home network device category which are digital media server (DMS), digital media controller (DMC), digital media renderer (DMR), digital media player (DMP) and digital media printer (DMPr). Among these device classes, the DMS needs more security consideration since it store and share media content to other networked devices. PCs and network attached storage (NAS) devices are examples of DMS.

Though DLNA devices interoperate within a home network, the attack from the internet may penetrate the home router and access the private media content in the DMS. Another scenario is that one may want to restrict his/her private video/audio files from the access by the minors in the home.

This paper will propose a design of a personal firewall for the DMS. Unlike traditional firewall, the access control list of this firewall can be easily configured and updated. The key concept of the design is to deploy the contact list used in the instant messaging system. If a Digital Media Controller (DMC) is added to the contact list of the DMS, this DMC can control and manage the media in the DMS. However the DMS can still reject the access from a DMC on the contact list by temporarily blocking.

2. Related Works

The main concept of this paper is to combine personal firewall with instant messaging software. Instant messaging (IM) [3] is a technology that facilitate near real-time text based communication between two or more participants over a network. The IM architecture consists of contact list, presence information, message sending/receiving.

2-1 Contact List

A contact list is a collection of user names in the instant message system. Double-clicking on any name in the contact list will open an instant messaging session and allow one to talk with that person. The style of the contact list is different with the different IM programs, but all contact lists have similar capabilities. In this

978-0-7695-3473-2/08 $25.00 © 2008 IEEE
DOI 10.1109/APSCC.2008.244

paper, the contact list will be key concept to realize a zero-configuration personal firewall.

2-2 Presence Information

Presence information is a status indicator that conveys the ability and willingness of a potential communicating partner. A user's IM client provides presence information to a presence service, which is stored in what constitutes his personal availability record and can be made available for distribution to other users to convey his availability for communication. The proposed personal firewall in this paper utilizes this mechanism to trigger the protocol control module within the LSP layer which will be described in section 4.

2-3 Message Sending and Receiving

IM allows instant communication between users by transmitting information quickly. Due to this feature users can have a real-time conversation. Some IM systems allow users to use webcams and microphone which made them more popular than others. Due to this feature users can have a real-time conversation. In this paper, message sending and receiving can be used as an extra communication channel between DMS and DMC.

3. System Analysis

Establishing a personal firewall within DMS to enhance security is the main objective. However, configuring filter rules of firewall is not easy for every member in the home. Therefore zero-configuration issue is an important objective as well.

3.1 Zero-Configuration

Instead of setting rules of addresses and ports, right-clicking a certain user name in the contact list will be able to directly configure its network connection rules. Figure 3-1 shows a scenario while Alice is denied of the access to the DMS but Bob is permitted.

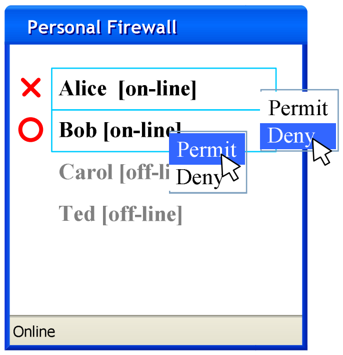

Figure 3-1

Usually an IM client would not perceive the IP address of users in the contact list hence IP lookup mechanism should be launched to convert a user name to the IP address which he or she uses. The IP lookup process will be implemented as a protocol control module and will be described in section 4.2.

3.2 Authentication

Since the proposed personal firewall is combined with IM architecture, it will be able to utilize the IM authentication mechanism and provide dual protections -- address oriented and user oriented protection. The address oriented protection is provided by the traditional firewall function which filters network traffics according to pre-configured rules while user-oriented protection is provided by IM user authentication mechanism.

The following figures 3-2 and 3-3 show the process that a user uses a DMC and controls the DMS transmitting a media file to the DMR for rendering. In this case, the user must use the IM client within the DMC to sign-in to the IM server first. Only when the user’s id or account is in the allow list of the DMS IM user, he/she can use the DMC to access the DMS. Otherwise, connections from DMC will be denied and blocked.

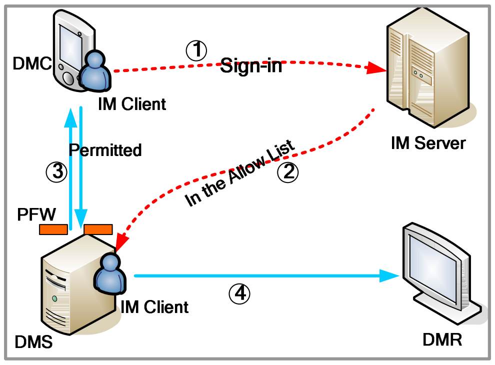

Figure 3-2 DMC in the Allow List of DMS

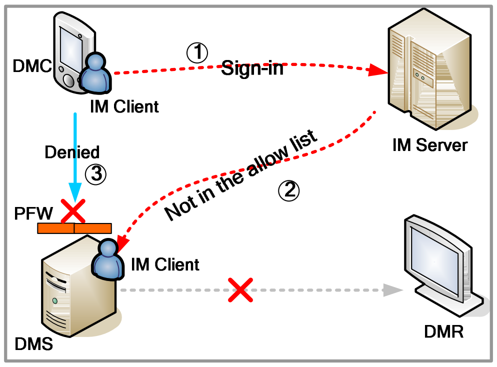

Figure 3-2 DMC not in the Allow List of DMS

4. Design and Implementation

In order to verify the feasibility of this architecture, a prototype system with MSN messenger bundled on Windows XP is implemented. Figure 4-1 below shows the system stacks with IM interface, protocol control and firewall as three major add-on modules.

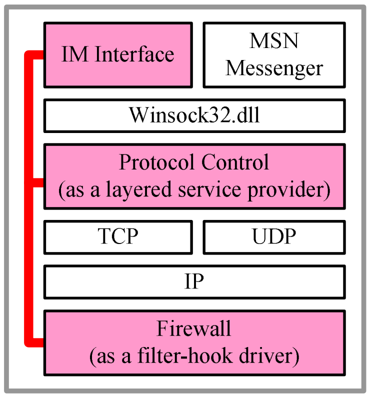

Figure 4-1 System Stacks

4.1 The IM Interface

The IM interface module which would provide a friendly user interface and control the major process flow of the whole system is implemented using MFC. The major part of this module is to fetch the MSN contact list which will be used for subsequent firewall configurations. Fortunately Microsoft released some MSN APIs [4] which can be utilized to program the auto sign-in, contact list fetching, etc. The following are MSN Messenger APIs which were used in the IM interface codes.

- Signs the local client in automatically using the last sign-in name and saved password information.

```
HRESULT AutoSignin(VOID);
```

- Creates a MessengerContact object that can be used to call methods of the Windows Messenger interfaces, such as IMessenger, IMessengerContact, and IMessengerContacts.

```
HRESULT GetContact( BSTR bstrSigninName, BSTR
                    bstrServiceId, IDispatch **ppMContact );
```

- Retrieves the status of the local user in a service.

```
HRESULT IMessengerService::get_MyStatus(
                                  MISTATUS *pmiStatus);
```

- Sets the status of the local user in a service.

```
HRESULT IMessenger::put_MyStatus(MISTATUS pmiStatus);
```

Except for the contact list and status operations, there are no suitable message exchanging APIs to be invoked by a program resided in DMS and DMC. The released InstantMessage method defined in IMessenger interface could only launch a conversation window which would be annoying. Therefore a piggyback approach was adopted in the protocol control module.

4.2 Protocol Control

The main function of the protocol control module is IP inquiring. The module resided in DMC side should respond with its IP to the inquiring command issued from the corresponding module on the DMS side. Therefore the protocol control module must be installed on both the DMS and the DMC.

This module is implemented as a Layered Service Provider (LSP) [5] which is a DLL that uses Winsock APIs to insert itself into the TCP/IP stack. Once in the stack, an LSP can intercept and modify inbound and outbound Internet traffic. Therefore it can be used to process all the TCP/IP traffics taking place between the MSN clients and piggyback additional predefined command for address inquiring and responding.

However, the LSP is triggered by the upper layer application hence the MSN client is made to transmit something that would result in nothing important. Changing the MSN status from online to busy and back to online is a trick that can make the MSN client sending a status altering message to each one in the contact list and trigger the LSP to attach extra commands. Figure 4-2 in the following shows the process flow of IP inquiring process.

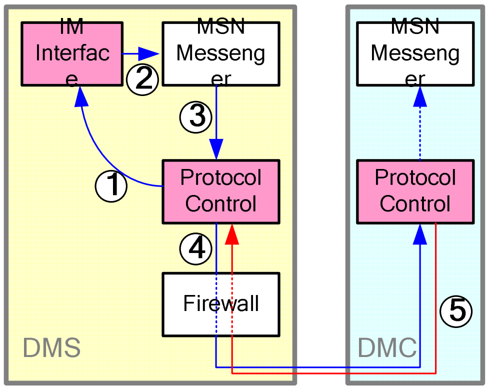

Figure 4-2 IP Inquiring Process

(1) Protocol control module is ready to inquire IP address in the contact list hence notify the IM interface.

(2) IM Interface uses the **IMessengerService::put_MyStatus** method to alter the MSN messenger status.

(3) MSN messenger sends out a message with status altering command.
(4) Protocol control module attaches additional message with predefined IP inquiring command.
(5) Protocol control module in DMC receives the additional message and responds its own IP address.

4.3The Firewall

The firewall module is implemented using filter-hook driver approach [6]. A filter-hook driver is a kernel-mode driver that implements a callback function called a filter hook and registers that callback function with the system-supplied IP filter driver. The IP filter driver then uses the filter hook to determine how to process incoming and outgoing packets.

In fact, filter-hook driver isn't a network driver; it is a kernel mode driver. In this filter-hook driver, a callback function should be implemented and registered. The main steps are shown below:

(1) Creates a filter-hook driver as a kernel mode driver. This step includes creating a device, creating a symbolic link and creating dispatch points.

```
RtlInitUnicodeString(&deviceNameUnicodeString,
                     NT_DEVICE_NAME);
ntStatus = IoCreateDevice(DriverObject,0,
             &deviceNameUnicodeString,
             FILE_DEVICE_DRVFLTIP,0,FALSE,
             &deviceObject);
……
RtlInitUnicodeString(&deviceLinkUnicodeString,
                     DOS_DEVICE_NAME);
ntStatus = IoCreateSymbolicLink(&deviceLinkUnicodeString,
                     &deviceNameUnicodeString);
……
DriverObject->MajorFunction[IRP_MJ_CREATE] =
DriverObject->MajorFunction[IRP_MJ_CLOSE]=
DriverObject->MajorFunction[IRP_MJ_DEVICE_CONTROL] =
                     DrvDispatch;
DriverObject->DriverUnload= DrvUnload;
```

(2) Gets a pointer to IP filter driver device

```
RtlInitUnicodeString(&filterName, DD_IPFLTRDRVR_DEVICE_
ntStatus = IoGetDeviceObjectPointer(&filterName,
                     STANDARD_RIGHTS_ALL,
                     &ipFileObject, &ipDeviceObject);
……
```

(3) Installs the filter function. It can be done by sending a specific IRP.

```
filterData.ExtensionPointer = filterFunction;
KeInitializeEvent(&event, NotificationEvent, FALSE);
irp = IoBuildDeviceIoControlRequest(
                IOCTL_PF_SET_EXTENSION_POINTER,
               ipDeviceObject, (PVOID) &filterData,
                sizeof(PF_SET_EXTENSION_HOOK_INFO),
                NULL, 0, FALSE, &event, &ioStatus);
if(irp != NULL)  {
   ntStatus = IoCallDriver(ipDeviceObject, irp);
   ……
}
```

(4) Processes packets filtering within the filter function. This function is always called when the host receives or sends a packet. Depending on the return value of this function, PF_FORWARD, PF_DROP and PF_PASS, the system will process the packet accordingly. The prototype of this function must be:

```
typedef  PF_FORWARD_ACTION (*PacketFilterExtensionPtr)(
  IN unsigned char *PacketHeader,
  IN unsigned char *Packet,
  IN unsigned int PacketLength,
  IN unsigned int RecvInterfaceIndex,
  IN unsigned int SendInterfaceIndex,
  IN IPAddr RecvLinkNextHop,
  IN IPAddr SendLinkNextHop
  );
```

The filter-hook driver is not the best method to develop firewalls for Windows since it cannot filter lower layer headers. Fortunately it is sufficient for this proposed architecture since the UPnP network is based upon the TCP/IP stack.

5. Conclusions

Instead of traditional address oriented firewall, this paper proposed a user oriented personal firewall which filters network traffics according to user id of IM. This new firewall architecture provides the zero-configuration features and suits the DLNA architecture.

References

[1] Digital Living Network Alliance, “DLNA Overview and Vision Whitepaper 2007”, retrieved from http://www.dlna.org
[2] UPnP^{TM}, “UPnP™ Device Architecture”, retrieved from http://upnp.org/standardizeddcps/default.asp
[3] RFC2778, “A Model for Presence and Instant Messaging”, http://www.ietf.org/rfc/rfc2778.txt
[4] Microsoft, “Windows Messenger Client Reference-Interface”, MSDN Library
[5] Wei Hua, Jim Ohlund, Barry Butterklee, “Unraveling the Mysteries of Writing a Winsock 2 Layered Service Provider”, Microsoft Systems Journal, 1999 http://www.microsoft.com/msj/0599/LayeredService/LayeredService.aspx
[6] Microsoft, “Windows Driver Kit: Network Devices and Protocols-filter-hook drivers”, MSDN Library