Security Analysis of Microsoft Notification Protocol

Oliver Zheng, Jason Poon

***Abstract* — This report studies the security of Microsoft Notification Protocol (MSNP), the underlying protocol used in Windows Live Messenger (WLM). Through the security analysis of the protocol, several security vulnerabilities were discovered and exploited by using our technique of MSNP-based TCP hijacking. MSNP-based TCP hijacking is a method of re-syncing the TCP stacks of the client and the server following an injection of a packet. By applying our developed technique, we managed to successfully spoof messages and the identities of others. This attack may be prevented if MSNP was encrypted or the protocol was changed to eliminate the use of certain commands.**

***Index Terms* — MSNP, security analysis, computer network security, instant messaging**

I. Introduction

First released in 1999, Windows Live Messenger has since grown to become one of the world's most popular instant messaging service. Windows Live Messenger, its predecessors MSN Messenger and Windows Messenger, and other third party instant messaging clients employ the Microsoft Notification Protocol (MSNP) to connect to the .NET Messenger Service. With over 300 million users worldwide, it is imperative to ensure that WLM and its underlying protocol, MSNP, are secure from attacks. Through the security analysis of MSNP, we will determine if a user's privacy and confidentiality are sufficiently protected while using instant messaging clients. The goal of our research is to identify any security flaws present in MSNP, exploit the weaknesses, and develop countermeasures for any possible vulnerability in the protocol.

In the next section, the architecture and underlying protocol of WLM will be described. Section III analyses the protocol to determine any security vulnerabilities present, while section IV exploits the vulnerabilities that were discovered. Section V discusses the methods of correcting the security flaws discovered.

II. Windows Live Messenger Overview

A. Architecture

Windows Live Messenger (WLM) is an instant messaging client that utilizes the .NET Messenger Service. The .NET network consists of a centralized cluster of servers; each individual server provides WLM clients with a specific service [1]. Two of the main servers within the cluster include the Notification servers (NS) and Mixers.

Upon a user's login, the WLM client opens a persistent transmission control protocol (TCP) connection to the NS; this connection must always be active, else the client will be disconnected from the service. Presence information (e.g. online status, user's display name) is relayed through the NS. The user broadcasts its status to other WLM users by updating the NS with new information. The NS then forwards this information to all subscribed clients (i.e. contacts who have this user on their list). Similarly, when other contacts update their status, the NS pushes updates to the user, all through the same TCP connection.

In the event that the user intends to initiate a conversation with a contact, the user requests for a session through the NS. The NS returns a session authentication ID and network destination internet protocol (IP) address and port that points to another server in the cluster called a Mixer. Mixers handle all forms of communication not meant as broadcasts, including instant messages and file transfers. The user connects to the designated Mixer and establishes another persistent TCP connection. Through this connection, the user invites others contacts to join the conversation. Utilizing a Mixer to interact between WLM clients abstracts the client's information from other WLM clients [2]. The invited contacts receive an invite through their NS connection and proceed to connect to this Mixer.

In essence, the connection with the NS controls all presence information and message invites, while connections with the Mixers are created upon requests and are used to relay chat messages. Figure 1 provides a visual representation of the WLM architecture.

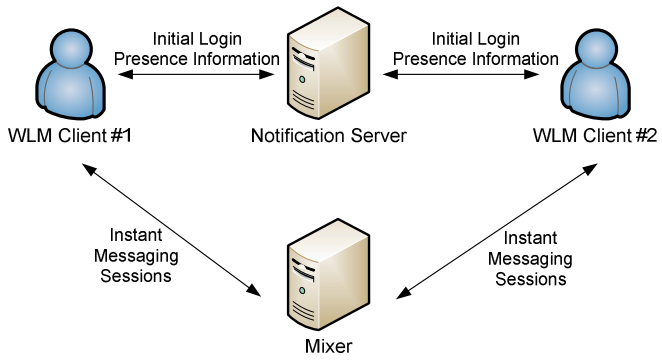


Figure 1. WLM clients are never in direct contact with other clients. Information is relayed through intermediary servers. Each server within the .NET messaging service serves a unique function.

Manuscript received November 19, 2007.
O. Zheng (e-mail: ubc+eece412@oliverzheng.com).
J. Poon (e-mail: mr.j.poon@gmail.com).

B. Protocol

The Microsoft Notification Protocol (MSNP) is a text-based application protocol, the topmost layer of the Open Systems Interconnections (OSI) reference model [3]. With the exception of the initial login of the user to the .NET service, all communication between the WLM client and servers are expressed through MSNP.

Although the current version of MSNP is 15, the protocol has been proprietary and has not had a public specification since version 2 [4]. Due to the text-based nature of the protocol, attempts to reverse-engineer have proven to be relatively successful. Contributions to public documents [5][4][6] detailing the mechanisms of the protocol have propelled the growth of third-party implementations of.NET Messenger clients. These third party clients include Pidgin, Trillian, and Jabber.

MSNP is composed of a series of UTF-8 encoded commands which may or may not require confirmation replies from the received party. Both NS and Mixers can send commands to the WLM client and vice versa. These commands are used to indicate a change in the state of contacts or the arrival of new instant messages.

III. Design Flaws and Security Vulnerabilities

A. Violated Principles of Designing Secure Systems

MSNP achieves a secure login process through secure socket layer (SSL). Following the successful authentication of a user's identity, a dedicated TCP connection is opened between the NS and the user; this connection serves as a point of entry for all commands between the two endpoints. Traffic on this TCP connection is unencrypted and both endpoints assume that it is a secure channel. As a result, this design fails to obey several principles of designing secure systems: correct assumptions and least common mechanism.

Although MSNP has a built-in feature to ensure the validity of the connection, this feature can be exploited by an attacker. Two commands, pings (PNG) and challenges (CHL), are used by WLM clients and servers to keep the TCP connection alive and prevent Network Address Translators (NATs) from closing idle sockets [5]. However, it is unnecessary for both endpoints to have the capability to ping the other party. This protocol design violates the design principle of economy of mechanism. This design flaw is exploited by our technique as explained in section IV.C.

The third MSNP design flaw violates the principle of complete mediation. Once logged in, all commands sent between the client and the server(s) are not authenticated. No security checks (e.g. challenge-response) are generated for actions that should be authorized. For instance, when the user receives an invite to connect to a Mixer session, neither the Mixer nor the client authenticates the other party. The only checking that is performed is to ensure that the 3rd party that invited the user to the messaging session belongs to the "allowed" list and not the "blocked" list.

Another inherent vulnerability of MSNP deals with the fact that the WLM client trusts all messages from the NS, which again is related to the first flaw. Furthermore, the assumption that all messages from the NS are authenticate opens a more serious vulnerability. Users are capable of accessing their Windows Live Hotmail service through their WLM client. Although authentication through proper login is secure, the assumption the WLM client makes is that its credentials are being sent to the NS and not an attacker. This assumption opens a security hole in which an attacker can pose as the NS to obtain the login credentials of a user. Once obtained, the attacker will then have remote access to the victim's private email accounts.

All of these vulnerabilities affect the latest versions of all implementations of MSNP, including WLM 8.5, Pidgin 2.2.2, and Trillian 3.1.7.

B. Confidentiality, Integrity, and Availability

Computer security policies focus on three core goals: confidentiality, integrity, and availability of information [12]. However, the security vulnerabilities discovered in MSNP greatly increase the risk in all aspects of computer security.

The unencrypted nature of the protocol breaches the confidentiality of the user's privacy. All of the victim's activities including instant messaging conversations are visible to an attacker through a simple packet sniffer.

By applying the technique of application-based TCP hijacking which is further described in section IV.C, the origin of the data can no longer be assured. Through packet injection, it is possible to impersonate the NS and send a packet to the client.

Furthermore, the availability of the .NET messaging service is crucial as many users rely on it as a form of communication. However, it is possible to stage a denial of service attack on a WLM client by bombarding the client with either TCP FIN packets or the MSNP 'OUT' command. These two packets will close the TCP connection and sign the user out, respectively.

IV. Exploits

A. Setup and Tools

For our test environment setup, Windows XP with Service Pack 2 and the latest version of Windows Live Messenger, version 8.5, were installed within a virtual machine using Microsoft Virtual PC 2007. The virtual machine acts as the victim, and the hacker performs his attacks through the host machine.

The rationale for using a virtual machine was to allow the hacker a simplified mechanism to sniff the victim's packets and disallow any modification of network traffic going to and from the victim's machine. Through this setup, the hacker will not be able to access or modify any resources within the virtual machine (i.e. the victim). This assumption is fair since in a public network it is likely that the hacker has access to the sniffed packets of neighbouring machines yet is not able to act as the man-in-the-middle as modern switches and routers have been setup to circumvent this security concern.

Furthermore, our exploits make use of WinDump, a

command-line packet sniffer, and Bittwist, a command-line packet editor and generator. Both of these programs are open source tools and are publicly available on the Internet. We have also developed several Perl scripts and a server application written in C to assist in automating various tasks to exploit the security vulnerabilities within MSNP.

All of the exploits deal with network packet manipulation, specific to MSNP. The hacker does not require any modification of the WLM client running on the victim's machine.

B. Unencrypted Messages and Files

Although the initial login of the user is performed through SSL, and therefore not viewable in plaintext, all other messages are unencrypted. As a result, all commands and messages sent and received by the WLM client can be viewed using a packet sniffer such as Wireshark or TcpDump. The data exposed in the unencrypted network traffic include and is not limited to the email addresses and display names of a user's entire contact list, presence updates of the user and their contacts, and all instant messages and file transfers sent and received by the user. By using simple packet sniffers, hackers are capable of compromising the privacy and confidentiality of WLM users.

C. Application-Based TCP Hijacking

Traditional TCP hijacking is a form of the man-in-the-middle attack and involves address resolution protocol (ARP) cache table poisoning of either one or both sides of the TCP connection [7]. The attacker then acts as a relay between the two endpoints and will have the ability to inject packets to either side without disconnecting the connection.

While TCP hijacking is applicable to all applications that utilize TCP [10], it is often infeasible and a waste of resources to relay the entire conversation between the two endpoints. As a result, we have developed a new technique – application-based TCP hijacking. Application-based TCP hijacking is a variation of the traditional TCP hijacking method and very effective in compromising victims. By taking advantage of the ping and challenge commands used in MSNP, our technique accomplishes two goals: injection of a spoofed command to either the client or the server and maintaining the connection between the two endpoints. The latter goal of keeping the TCP connection alive is crucial in keeping the victim oblivious to the exploit.

In order for TCP packets to be accepted by the destination application, sequence and acknowledgement numbers are required to match the expected values of the destination TCP stack [8]. As shown in Figure 2, if an attacker were to inject a packet with the correct sequence and acknowledgement numbers (which can be obtained from sniffing network traffic), the recipient of the injected packet would respond with a TCP acknowledgment (ACK) to the other end of the TCP connection, which would ACK back to indicate that no such packets were sent. This back and forth of ACK-ing creates what is known as an ACK storm and will eventually lead to the disconnection of the TCP connection [9].

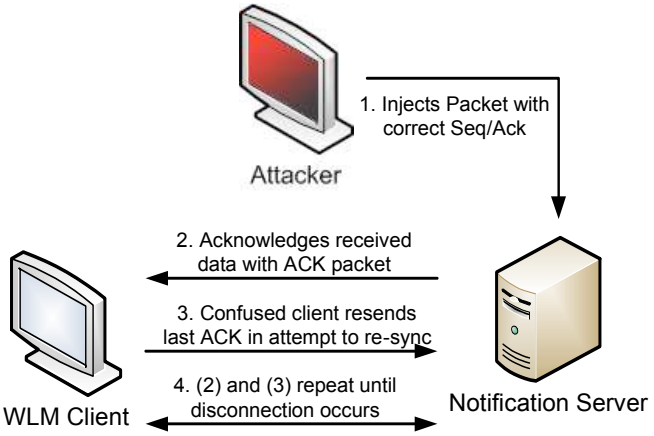


Figure 2. An injection of a packet will lead to an ACK storm which will eventually cause the disconnection of the WLM client.

Assuming the TCP connection was somehow kept alive, further messages sent between the client and server would lead to another ACK storm as the sequence numbers between the server's and client's TCP stack have become misaligned due to the injected packet.

Our technique of application-based TCP hijacking prevents the ACK storm and maintains the connectivity between the server and the client. For each injected packet that is to be sent, a carefully calculated number of pings and challenges are sent to the server and client. Through this process, the sequence numbers can be re-synced resulting in no ACK storm and the maintenance of the TCP connection.

A simplified example is shown in Figure 3 where the hacker's objective is to inject a packet of 30 bytes to the client (victim) posing as the NS. Server ACK is the sequence number that the server believes the client is at; client ACK is the sequence number the client believes the server is at. The hacker makes use of pings to the server (10 bytes) and client (5 bytes) and the responses from the server (20 bytes) and client (10 bytes) in order to create a gap in sequence numbers in which the spoofed packet fits. Note that it is the due to the different sizes of pings to and responses from the server and client that make this hack successful. After sending two pings and receiving two responses from the server, the hacker then sends two pings to the client and receives two responses. Now the client believes that the server's sequence number is at 10, while in reality the server has sent 40 bytes of data and will continue to send packets starting at sequence number 40. This creates an opportunity for the hacker (who designed for this to occur) to send 30 bytes of information to the client. The 30 bytes of information could contain any information and the client will accept it believing it originated from the server. After this last spoof message, the sequence numbers are lined up again, as client SEQ equals server ACK and server SEQ equals client ACK.

Another way of comprehending this method of attack is that for every byte of data the hacker wishes to send, the hacker must also force the spoofed source of the message to send an identical sized packet. For instance, if the hacker wishes to send 30 bytes of data to the client, the hacker has to somehow

force the server to actually send 30 bytes of data such that the sequence numbers of the client and server are still synchronized. The hacker accomplishes this by injecting packets to ping the server; however, during this process, the pings themselves require the client to actually send the same sized packets as the ping. The hacker achieves this by pinging the client. As a result, in order to successfully inject a packet, the hacker must also ping the server 'x' times and the client 'y' times for their sequence numbers to line up. The values of 'x' and 'y' can be solved using simple algebra.

In reality, the pings and responses have variable but predictable lengths. Our exploits utilize a Perl script that sniffs network traffic to automatically calculate the correct number of pings and challenges to send.

D. Command Spoofing

By applying application-based TCP hijacking techniques, it is possible to alter a user's WLM status. For example, the modification of a user's display name requires the injection of a single packet in which its payload is:

```
PRP 123 MFN hacked
```

The above packet consists of the command (PRP), transaction ID (123), my friendly name (MFN), and the new display name (hacked).

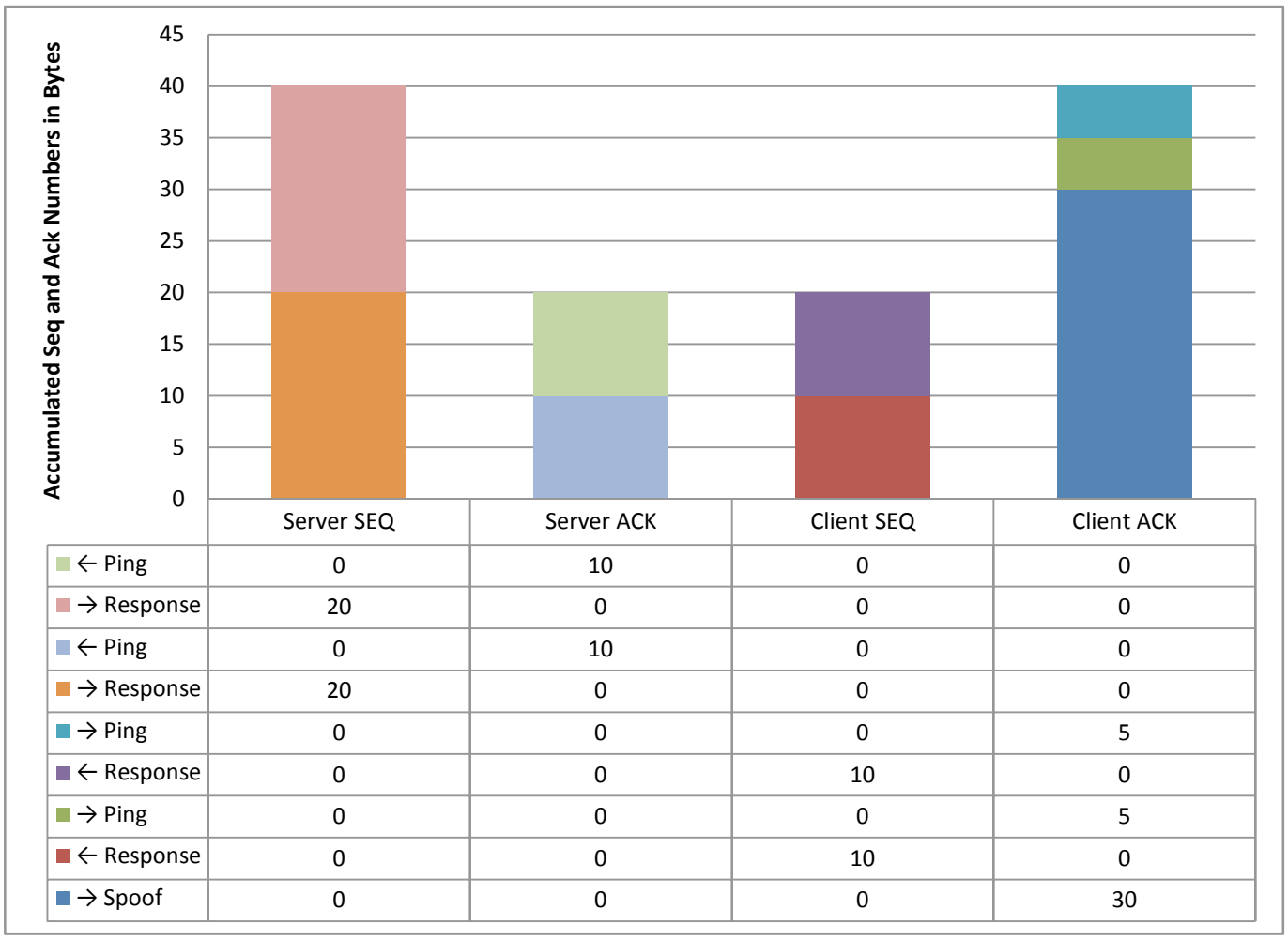
In another scenario, an attacker wishes to alter a user's personal message to display "*oliver+jason!*". This can be done by sending the following command:

```
UUX 1234 131
<Data><PSM>oliver+jason!</PSM>
<CurrentMedia></CurrentMedia><MachineGuid>
{01A9EEE1-3A31-4C46-9EDB-2ACEB026B6D4}
</MachineGuid></Data>
```

The UUX command sets a user's personal message or "currently playing" song and is followed by the transaction ID (1234) and payload size (131). The contents of the payload include the personal message (oliver+jason!), the current media, and the machine globally unique identifier.

Following the injection of these packets, it is necessary to utilize application-based TCP hijacking to re-synchronize the sequence and acknowledgement numbers of the two endpoints to prevent disconnection of the victim from the service.

The above two hacks both occur without the victim's knowledge. Altering either the user's display name and/or personal message will cause presence updates to be pushed to the user's contacts, but the user will not be aware of any alterations to their information. These hacks are possible due to the assumption by both endpoints that all messages received are authentic and therefore do not require authentication.



	Server SEQ	Server ACK	Client SEQ	Client ACK
← Ping	0	10	0	0
→ Response	20	0	0	0
← Ping	0	10	0	0
→ Response	20	0	0	0
→ Ping	0	0	0	5
← Response	0	0	10	0
→ Ping	0	0	0	5
← Response	0	0	10	0
→ Spoof	0	0	0	30

Figure 3. Application-based TCP hijacking takes advantage of the MSNP asynchronous commands PNG and CHL to create a disparity between the sequence numbers of the TCP stacks of the notification server and WLM client to successfully inject a TCP packet.

E. *Identity Spoofing*

Another application of the MSNP-based TCP hijacking is the spoofing of an instant messaging session. When a contact sends a message to the user, the NS notifies the user of a Mixer request. The user then connects to the Mixer and the Mixer will notify the user of the contact's identity and relay messages. This attack spoofs the notification from the NS to the user and forces the user to establish a connection to a spoofed Mixer, which will then spoof the identity of the contact who is attempting to talk to the user.

The victim will not be able to see anything amiss and will assume that it is an authentic instant messaging session. However, the hacker is capable of impersonating any contact.

The specific message for inviting the victim (eece412_bob@hotmail.com) to the Mixer session is:

```
RNG 1312697181 67.207.145.17:1863 CKI
17021696.318151 eece412_alice@hotmail.com
Alice U messenger.msn.com 1
```

After sending a pre-calculated number of pings to the server and challenges to the client, the TCP sequence numbers leave a gap of exactly the size of this message. After sending this message, the TCP connections of both the server and client are re-synced, and the client will attempt to connect to the Mixer whose IP address and port is specified in the RNG message. In this case, the client will connect to IP 67.207.145.17 and port 1863.

As part of the exploit, we wrote a multi-threaded server application, hosted on 67.207.145.17 port 1863, which is capable of handling Mixer connections. As soon as the victim connects to this address, the Mixer will send the identity that the hacker is attempting to impersonate, in this case, Alice (eece412_alice@hotmail.com).

```
IRO 1 1 1 eece412_alice@hotmail.com Alice
1985855532
```

The victim will then recognize the email address as a contact on their contact list and assume that the instant messaging session just established is with Alice (eece412_alice@hotmail.com) where in actuality, it is with a hacker. All messages are relayed through this fake Mixer server, and the hacker, typing in front of this Mixer application, can pose as the contact Alice to the full extent of a regular contact.

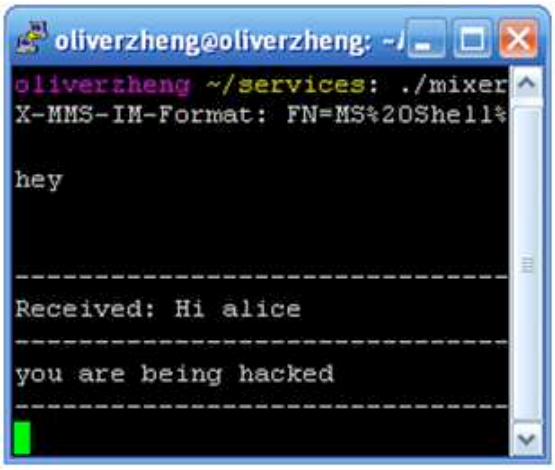


Figure 5. Attacker's perspective of the spoofed Mixer session. The user is able to spoof the identity of any contact.

F. *Additional Exploits*

A few other interesting exploits include the modification of a victim's contact list. Commands are sent to the NS from the client to block, unblock, add, or delete contacts. Although an attacker can easily inject a packet to instruct such commands, the latest version of MSNP associates the contact list with an authenticated SOAP service. Therefore, for each command to be successfully completed, in addition to the MSNP command sent to the NS, a coupled command has to be sent to a central SOAP service using SSL.

Another critical exploit we investigated was WLM's integration with Windows Live Hotmail. Users are capable of checking their email by clicking a button inside WLM, launching a browser with an automatically authenticated session into the Hotmail email service. According to the (outdated) documents detailing the reverse-engineered protocol [5][4][6], WLM accomplishes this by generating a temporary HTML page that submits hashed login credentials to an HTTPS server. As these credentials are authenticated using SSL, an attacker will not be able to view the information. However, the hacker is able to spoof the MSNP command that sends the uniform resource locator (URL) of the email inbox, where WLM redirects the temporary HTML page. Multiple attempts were made to inject a packet with a spoofed URL to the victim where the credentials could be submitted to the attacker. That spoofed URL could then redirect the user to the authentic Hotmail service. Theoretically, the exploit should work, and the victim would not realize that the hacker has obtained credentials to login to Hotmail as the victim. Unfortunately, the exploit did not achieve the desired result. The victim was always redirected to

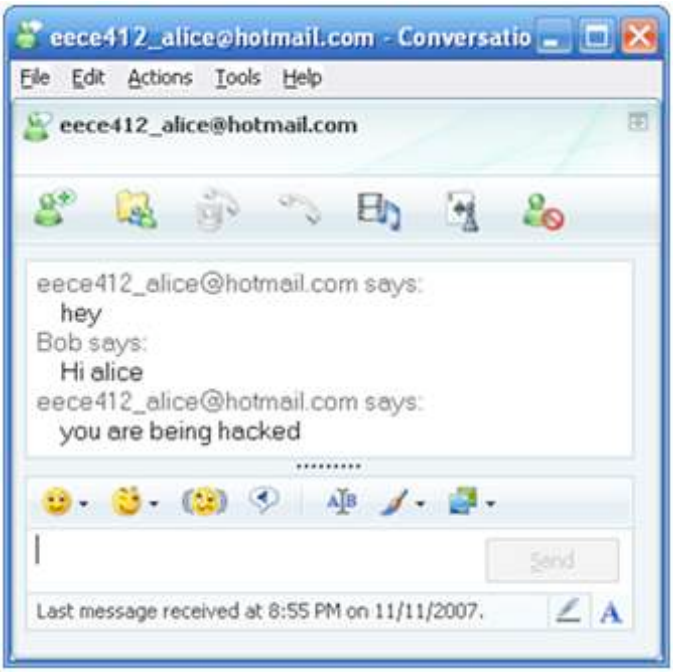


Figure 4. Victim's view of the spoofed Mixer session. Note that the instant messaging session with the attacker looks authentic.

the HTTPS server. An explanation of this failure was that the URL is actually contained as part of the WLM binary and cannot be modified by the NS.

G. Feasibility

All of these hacks require two abilities – the hacker requires the capability to view the victim's network traffic, and the hacker has the ability to send packets with the media access control (MAC) and IP addresses of the victim and the NS server. In our setup of a virtualized victim, both of these requirements are satisfied. In a publicly switched network, these two assumptions can also prove true. The second requirement depends on the hacker's gateway switch or router. If a packet could contain a range of IP addresses, it is likely that the victim's IP address will be one of them.

V. Countermeasures

A. Encryption of Messages

Encrypting the messages between the WLM client and the server will greatly increase the user's privacy. SimpLite-MSN achieves this by encrypting all messages between supporting WLM clients using RSA encryption [11]. With SimpLite-MSN, the user is sets up a pair of keys the public key is automatically distributed to contacts that support SimpLite-MSN. Should the contacts choose to accept the public key, the two WLM clients can establish a session key to be used for encrypting the instant messaging conversation.

Figure 6. SimpLite-MSN provides usable security with recognizable dialog boxes displaying the security status of the instant messaging session. Green notifies the user of a secure session, while red notifies the user of an unsecure session.

This method of encryption enables the confidentiality of messages. It also provides, to some degree, integrity of messages. If the user knows that a contact has the capability of encrypting messages, then the identity spoofing hack described earlier might not carry as much weight, given that the user knows it may not be who the contact claims to be.

B. Redesign the Protocol

The protocol design of allowing bidirectional pings opens up the opportunity for attackers to successfully spoof messages. This inherent flaw, while meant as a feature, is the main reason why these attacks work. If this feature was removed from the protocol and only a one-sided ping was allowed, hackers would not be able to spoof a message and maintain the integrity of the connection.

While this solution would require modification to the server applications that operate the NS, it would not break backward compatibility since the lack of a ping does not result in alteration of connection states. The original problem this flaw was intended to address – NAT connections – could be regulated by setting timeout periods for the connections that do not send pings to the server. Instead of actively querying a connection status, the server could wait a certain time for the ping from the user before disconnecting potentially closed connections.

VI. Discussion and Conclusion

The implications of an unsecure popular instant messaging service are rather severe. To say the least, personal privacy is compromised. Additionally, as businesses begin to adopt technologies such as WLM, damages to the business can originate from exposed confidential business decisions or impersonated messages. As a result, we advise all users to reconsider their use of IM clients using the .NET messaging service for communication requiring confidentiality and integrity.

The design of Microsoft Notification Protocol did not follow several principles of designing secure systems. As a result, the protocol contains a plethora of security vulnerabilities that can easily be exploited by an attacker. By applying our technique of MSNP-based TCP hijacking, it is possible to decrease the overall security of MSNP in regards to confidentiality, integrity, and availability. Since our exploits attack the fundamental protocol itself, the only countermeasure to our attacks is to redesign the protocol.

References

[1] Charles. (2006, July 13). *Windows Live Messenger – What. How. Why.* [Online] Available: http://channel9.msdn.com/Showpost.aspx?postid=215459

[2] N. Williams and J. Ly, "Securing Public Instant Messaging (IM) At Work" Centre for Advanced Internet Architectures, Swinburne Univ. of Technology, Melbourne, Australia. Rep. 040726A , July 2004.

[3] C. Sanders, "Packet Analysis and Network Basics," in *Practical Packet Analysis*. San Francisco: No Starch Press, 2007, pp. 4-7

[4] P. Piccard, B. Baskin, C. Edward, G. Spillman, and M. Sachs. *Securing IM and P2P Applications for Enterprise.* Rockland, MA: Syngress Publishing, 2006, pp.99

[5] M. Mintz and A. Sayers. (2003, Dec 19). *MSN Messenger Protocol.* [Online]. Available: http://www.hypothetic.org/docs/msn/index.php

[6] 2007, June 4. MSNPiki: Unofficial MSN Protocol Documentation [Online]. Available: http://msnpiki.msnfanatic.com/index.php/

[7] M. Stamp. "Network Security Basics" in *Information Security: Principles and Practice*. Hoboken, NJ: John Wiley and Sons, 2006, pp. 349- 350

[8] D. Comer and D. Stevens. *Internetworking with TCP/IP*. Upper Saddle River, NJ: Prentice-Hall, Inc., 1999, pp.200-202

[9] M. Gregg. *Hack the Stack: Using Snort and Ethereal to Master the 8 Layers of an Insecure Network*. Rockland, MA: Syngress Publishing, 2006, pp.209-211

[10] J. Scambray, S. McClure, and G. Kurtz. *Hacking Exposed: Network Security Secrets & Solutions*. NY: McGraw-Hill Publishing, 2001, pp.530-533

[11] W. Stallings. *Cryptography and Network Security Principles and Practices*. Upper Saddle River, NJ: Prentice-Hall, Inc., 2006, pp.257-259

[12] Beznosov, Konstantin, "Introduction to Computer Security", unpublished.