Security Analysis of Windows Live Messenger

December 6, 2010

Derick H., Christopher E., John K., and Oscar H..

Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, Canada

derick.hpy@gmail.com, pnutz0@gmail.com, jhyup.kang@gmail.com, oscarhou@gmail.com

Abstract--- **Windows Live Messenger (WLM) is a free online instant messaging application by Microsoft that allows users to chat with their friends around the world. This report examines the security of WLM's voice chat component, an alternative to the primarily text-based communication. Through our security analysis, we were able to find WLM's audio protocol for voice chat and using a Dynamic Link Library (DLL) called "rtmpltfm," it is possible to obtain the proprietary audio codec to snoop conversations by simply using an internet traffic analyzer such as Wireshark. To counter such attacks, voice encryption can be used from client to client or public encryption tools could be applied to prevent packet snooping of voice chats.**

I. INTRODUCTION

THIS document details an investigation into the security of Microsoft's Windows Live Messenger (WLM). WLM is an online instant messaging program which enables users to converse through text, voice or video. In June 2009, WLM had over 330 million users world-wide; it is likely that many of those users used, or still use the voice chat features offered by WLM [1]. The voice chat component is fast and is a cost effective way to save on long distance call fees that would apply in a telephone call. However, many users are either not concerned or unaware of security issues that exist while using WLM in a public network. This report will investigate the security in WLM voice chat and determine whether privacy is at risk for WLM users.

Similar analysis and research have been conducted on WLM prior to this report. A group in the 2007 cohort of EECE 412 performed a security analysis of Microsoft Notification Protocol (MSNP) where WLM and other instant messaging applications initiate MSNP on their clients then connects to the .NET Messenger Service [2]. There exists an open-source application called "aMSN" which is an instant messenger application that enables Linux users to connect to the WLM servers, allowing them to use their WLM accounts to start chatting.

We analyzed a full voice chat session while using WLM and were able to formulate the initiation procedures when commencing a voice chat program. We also determined the type of data sent during a voice chat which contained the primary audio protocol used to decode audio after transmission. The DLL component of WLM which is used to reference the audio protocol was then studied in order to reverse-engineer the audio codec. Required tools such as disassemblers and Application Program Interfaces (API) for media-based development were researched and familiarized to reproduce the codec and decoded voice chat sessions from WLM.

Despite our efforts in reverse engineering the DLL library, we were unable to produce a finished codec to play payload dumps produced from voice chat sessions. However, this is primarily due to the time-constraints involved in familiarizing the tools involved and understanding the level assembly required. Nonetheless, analysis of the DLL file reveals that reverse engineering of Microsoft's proprietary codecs is in fact possible. An implication of this finding is that WLM video and audio chat packets can be intercepted and replayed by a third party. This poses a security risk as many users are unaware of the lack of encryption involved in WLM conversations. Therefore, there will still be individuals using WLM video and audio chat features while assuming their session is secure.

Open Design principle of secure systems was found to be violated in our security analysis in addition to confidentiality. WLM's design information is not public to users hence it cannot be scrutinized by security communities. WLM does not provide confidentiality to its users as many of its activities can be observed using packet analyzer tools.

Currently, WLM add-on tools only add encryption to text-based chat messaging, not audio and video chat. The concept behind an add-on is similar to client-to-client encryption, although different where as the encrypted message produced by the add-on must be decrypted on the other side. Since the encryption method is secret, the same add-on must exist on the receiving end of the message in order to decrypt it. This is possible among small groups of contacts, but since millions of users use WLM that are unaware of the privacy faults of the program, it would not work in preventing the problem. We propose WLM incorporate a feature within their program to encrypt the data and decrypt it on the other end. Since it is undesirable to have a possible delay this encryption could produce from an otherwise real-time conversation, we suggest this feature be optional. This feature would only work if both users had the security function on during the conversation. WLM gives warnings to users in the chat box for reasons including the other user having a status of "busy" or "away." It would not be difficult to also include a message notifying a user that their contact has message security turned on; prompting them to do so as well if they wish.

The problem with using a disassembler on a shared library or executable is valid for all programs, not just WLM. Since we chose to analyze WLM and not the protection of these files, we did not look into a means of protection for these files. The obfuscation of these shared libraries could help to prevent hackers from accessing the addresses and overall assembly of the DLL. Some gaming companies ensure their files are not tampered with a privacy agreement when installing the game. This privacy agreement would allow the company to hack into the user's system and snoop on what programs they are using [3]. If users run programs that modify the registry of the company's files, they act out a form of punishment towards the user. Even though this method would work, we would not suggest this for WLM because of the many privacy concerns it would bring up upon release.

II. ANALYZED SYSTEM

WLM is a program that is used by millions of users on a daily basis. Primarily as a text-based instant messaging client, the addition of voice and video chat features have been welcomed as two new means to pass along information. The system used in order to allow these new features to bring about a free flow of information is built upon the Microsoft servers. As all network communication revolves around the sending and receiving of packets, WLM works in the same way. Each user must log in to their account using their e-mail and password on the WLM client and initiate either a voice or video conversation with somebody on their contact list. The reason this conversation can exist is because both contacts have stored the e-mail address of the other. In reality, what is happening is a sequence of packets is being sent to the corresponding Microsoft server for voice chat: http://relay.voice.messenger.msn.com. The server will respond to the client with the IP address of the other user as well as a port to send packets to, which can be used to receive a steady stream of data.

A session initiation protocol (SIP) hand-shaking conversation will occur, where one user INVITEs another to a voice chat. After the other user accepts, information is passed between the two users in a packet-based standard SIP conversation in order to properly setup a real-time voice chat. The SIP portion of each packet is encrypted with Base64. SIP uses TCP to complete this, while immediately after the proper initiation finishes, a UDP conversation begins the actual transfer of data. These packets are sent using the real-time transfer protocol (RTP). Although this data is not encrypted in any way, it is encoded with Microsoft's proprietary codec, x-msrta. WLM uses a variety of codecs, but the most commonly used one is x-msrta [4]. This codec was found in all of the packets that we worked with for audio chat.

III. RELATED WORK

A group in the 2007 cohort of EECE 412 performed a security analysis of MSNP in WLM. MSNP is the protocol on the client-side WLM applications that communicate with the .NET Messenger Servers regarding notifications and initiation of chat conversations. For analysis, they set up a virtual machine using Windows XP operating system as the victim and used tools such as WinDump and Bittwist to packet-sniff and edit or generate packets. They revised TCP hijacking and created their own attack called application-based TCP hijacking. MSNP did not authenticate commands sent between the client and the .NET Message server hence they were able to spoof commands to the client and server and were able to keep the end-to-end connection alive.

aMSN software is an open-source application used for Linux-based operating systems that allow users to connect to their WLM accounts. Although this software is not a direct result of an analysis project, much analyzing and design was performed to create it. aMSN first began with a simple text-based functionality but has grown into have full audio, video features and many elements that exist in WLM. Much reverse-engineering was performed by its developers as many of the WLM design is not open for the public. For voice chat however, they avoided reverse engineering the audio codec and instead chose to initiate the chat using another codec supported by WLM which has an open implementation [5].

IV. ANALYSIS METHODOLGY

IDA is a freely available software that can be found online to disassemble executable or DLL files. Using this, we were able to disassemble WLM file to see the assembly code of how the program functioned. The next step we took was to find related fields or strings where the messenger client interacted with codecs, encoders, decoders and the x-msrta codec. Another function in the IDA program that we used was the "Translate to pseudo code" function. It helped us translate the assembly code into a more legible language. IDA also makes finding address locations easy by making cross-references such as sub-routine calls accessible with a simple click.

A. Dynamic-Link Library

Dynamic-Link Library (DLL) files are Microsoft's implementation of a shared library. A DLL file contains functions which can be loaded by programs at run time. The functions which were critical in reverse engineering the Microsoft proprietary codecs (x-msrta, x-rtvc1) resided in the rtmpltfm.dll (Real-Time Communication Platform)[6]. From this knowledge, it was decided that the best course of action was to analyze the DLL to obtain the function names or addresses of the relevant functions. If the decoding and encoding functions were made transparent, the functions could be used to replay the audio data sent between WLM clients.

After performing an initial investigation of the DLL file, it was discovered that many functions were privately exported. Because of this, the function names were not available for linking in the normal manner. However, by searching for the defined string "x-msrta", a slot in memory was found. This memory slot was referenced by two words worth of memory. No cross-references by functions were called to this point in memory, but by analyzing the defined memory surrounding the x-msrta section, we noticed the labels SIREN, g723, wma9, wma8, PCMU, and many more. These labels all happened to be WLM voice chat codec names. At the top of the section was an offset section of memory that was cross-referenced by another function. This proved that these codecs made up an array of strings. After further analysis of the cross-references we were able to determine that the function in the shared library would determine which payload was used to encode the audio and then fetch the payload codec information and sample rate.

B. Replay Attack

A replay attack was tested on WLM. We planned to send the data sniffed in packet form from one of our computers to another, using the WLM connection setup methods. We decided to try this method since it seemed more feasible than manually extracting the codec. The client receiving the data would decode it for us and convert it to audio so we could listen. Essentially, we had to analyze packets sent and received to generate our own packets, using the correct data necessary to initiate a SIP session. Most of the data could be found in advance, such as port number, IP address, and e-mail address. There was one parameter within the SIP authentication which was unique with every session. This was the "epid" value, which is a unique session ID in a SIP conversation. It is based off of the e-mail address of the sender, but the methodology in how the "epid" is developed is proprietary to Microsoft. This parameter is an additional security measure optional to SIP sessions that is active in all Microsoft voice communication systems. Without the value, an error: "SIP/2.0 400 Mismatched epid parameter value would appear. We decided this method would not work, since we had no means of breaking this proprietary protocol with no knowledge of in which DLL it was created in.

C. Video Chat

Using the same idea, we captured packets of video streams sent between two users having a video conversation. We again decoded and analyzed the SIP message at the initiation of the video chat. Two types of information were retained from this analysis: the packet type and the video codec. The video conversation for WLM uses the RTP type-121 protocol. As mentioned earlier, RTP is a real-time transfer protocol used for many media streaming and VoIP services. After further analysis of these RTP type-121 packets, we were able to determine that the codec used for the video conversation was the x-rtvc1 codec (RTVideo, Microft Proprietary Codec) [6]. Now, to analyze the packets captured, we had Wireshark decode the UDP packets as RTP. Next, we saved the payload and converted to a VC-1 media file. The reason we converted to a VC-1 file was suggested from the codec name itself that x-rtvc1 could be an extension of the widely available VC-1 codec. After further research, we were able to confirm that x-rtvc1 was indeed an extension of the VC-1 codec.

Now that we had the media file, we wanted to see the content. We tried many commercial media players including Windows Media Player, VLC Player, and SMPlayer.

V. Results

A. Video Chat

In analyzing the RTP streams of a video conversation, we were able to translate the data sent back and forth into a playable video file. Upon using ffplay, a Linux-based media player, we were able to see some frames of the video. After further analysis, we noticed that editing the payload data before each packet would result in a more complete video stream.

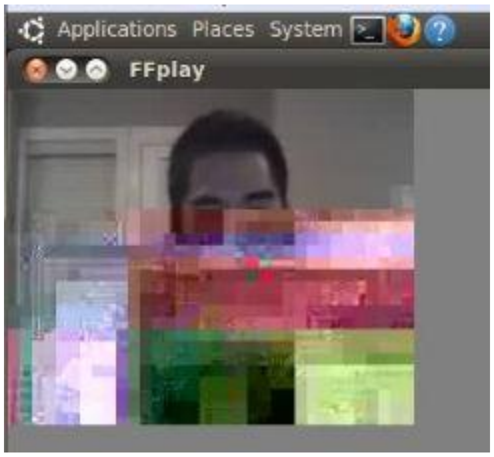

Figure 1 Decoded Video File of a Video Conversation

B. Voice Chat

As for audio, we were able to discover the function within the shared library that used the specific codec x-msrta we were looking for. However, due to inexperience with program disassembly, we were unable to find the correct function parameters and work with the function in a programmed C code of our own. Theoretically, the Windows API supports loading shared libraries into C codes and calling the DLL function from a code. The functions were made for use with streaming audio, where the client would read a multitude of packets and the shared library function would determine the rate the packets were coming in. With this information, the function could calculate the frame rate of the audio and sampling rate.

VI. Discussion

The analysis of WLM confirms that video and audio conversations are in fact replay-able. Due to this, the privacy of video conversations between WLM users are at risk from attackers. If users are unaware of the lack of security present in WLM, then they may share private information which could be intercepted by a third party. Even if the users are aware of the lack of security, and information discussed throughout the conversation may not be critically sensitive, many individuals would prefer to have their privacy.

WLM has violated the open-design principle in secure system design. By keeping the codecs and the system proprietary, other users have no option of adding on to the system to make it more secure. There is little or almost no extension made for WLM as a result of its proprietary system. If outside developers were allowed to work on the software, there could be a solution for its insecure chat features. As we noted earlier, WLM lacks confidentiality for its users by having unencrypted packets of text-based, audio and video conversations between its end users. This empowers attackers to read sensitive data and the users are susceptible to identity theft, which could lead to many more social engineering problems. A potential solution to increase security could be to encrypt data sent between WLM clients so that attackers would not be able to replay the video or audio even if they had access to the codec. This would deter attackers from attempting to intercept and replay conversations, as there is an additional level of difficulty.

Throughout the course of the analysis, it was seen that progress was greatly hindered due to a lack of experience with the tools that were used (IDA, Assembly, gstreamer). These tools each possess their own learning curve which slowed progress as time was required to familiarize with the tool. As such, attempts at analysis and reverse engineering were less detailed and thorough than they could have been.

VII. Conclusion

The main purpose of our analysis of WLM was to prove that voice and video chat are unsafe features on an insecure network. We were able to prove this by finding the source of the audio encoding and displaying a portion of the video conversation. Insecure network packet sniffing is an incredibly easy-to-do task with a program such as Wireshark. This supports our final analysis defining WLM as insecure. The problem with the security of WLM is that since it is intercept-able, it produces the concerns that come with telephones and wire-tapping. Even though WLM is a free service, it is a widespread communication tool so users should be informed of the possibility of privacy breaches. Microsoft makes no effort to inform the program's users about this problem. Many users are naive in thinking the service is perfect and are unaware that whatever they are saying to their contacts is heard. No system is perfect, but WLM could potentially make it harder for sniffers to perceive what information is passed through the service. Encryption to the packet data would definitely make breaking the system a much more tedious and difficult task.

References

[1] The Windows Live Messenger Team. (June 15, 2009). Share your favorite personal Windows Live Messenger story with the world! Available: http://messengersays.spaces.live.com/Blog/cns!5B410F7FD930829E!73591.entry

[2] O. Zheng and J. Poon, "Security Analysis of Microsoft Notification Protocol", unpublished.

[3] M. Ward (2005, October 31) Warcraft game maker in spying row. *BBC News.* Available: http://news.bbc.co.uk/2/hi/technology/4385050.stm

[4] MDSN Library. Representing New Payload Types. Available: http://msdn.microsoft.com/en-us/library/dd949621(office.12).aspx

[5] Y. Alaoui, aMSN Project Manager, online communication, November 2010

[6] MSDN Library. (June 2002). Integrating Rich Client Communications with the Microsoft Real-Time Communications API. http://msdn.microsoft.com/en-us/library/ms997607.aspx