



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Security Essentials: Network, Endpoint, and Cloud (Security 401)"
at <http://www.giac.org/registration/gsec>

Jeffrey Widom

SANS Security Essentials Practical Assignment Version 1.2f (amended August 13, 2001)

Title: Confidentiality: Can anyone read this message?

Introduction

Imagine the following conversation taking place between a father and son over the Internet.

Son: Hi dad!

Dad: Hi son, how are you?

Son: Fine, question for ya, can I please have some money to buy my books for next semester's classes?

Dad: Sure, use my credit card. Here's the number 2222 2222 2222 2222. Exp 3/03

Son: Thanks dad, you're the greatest!!

In today's society, with the ability to communicate in near real time, with anyone around the world, the thought of a parent and child having this type of discussion is not far fetched.

Recently, I had the opportunity to download and install Microsoft's MSN Messenger. A free Internet chat program which allows users to communicate via instant messages, establish voice sessions, and send files. During my first discussion via MSN Messenger I was surprised with the warning that was presented on the screen:

“Never give out your password or credit card number in an instant message conversation.”

My installation of MSN Messenger coincided with the recent release of Microsoft Chairman Bill Gate's memorandum entitled [Trustworthy Computing](#)^[1]. After reading Mr. Gate's memo, I wondered if MSN Messenger would be included in Mr. Gates's vision for a secure computing experience. In the future “.Net World”, will Microsoft provide applications which do not have built-in security features to protect the confidentiality of a user's information? That curiosity lead to this paper.

The purpose of this paper is to briefly demonstrate why Microsoft currently recommends that no passwords or credit card numbers are sent via their chat application.

Definition

The National Information Systems Security (INFOSEC) Glossary, [NSTISSI No. 4009](#)^[2], provides the following definition of confidentiality:

“Assurance that information is not disclosed to unauthorized persons, processes, or devices.”

For the purpose of this paper, I asked myself the following question:

“Is the information sent via MSN Messenger conducted in a manner which protects the confidentiality of the data and is not easily interpreted by an outside source?”

How does this stuff work?

Have you ever asked yourself, “When I am chatting over the Internet, how are my words making it half way around the world?” Before performing a specific test on MSN Messenger, let’s briefly describe how this process is accomplished.

Using the example provided in the Introduction, we’ll assume the son is sitting in his dorm room located in Los Angeles, California. His computer is connected to the university’s computer backbone and all users of the systems must first login to a centralized server prior to accessing the Internet.

The father in this scenario is located in Detroit, Michigan. His computer is located at his office, but because this is a personal matter, he has chosen to use his laptop’s modem to dial-in to his local Internet Service Provider (ISP) in order to open his chat application and talk to his son.

The diagram below (Figure 1) illustrates how a basic connection appears to an end user. For example, in our example, the father and son simply appear to have the ability to communicate with each other by opening their chat application. Each types their text into the box provided, and the words “magically” appear at the other end.

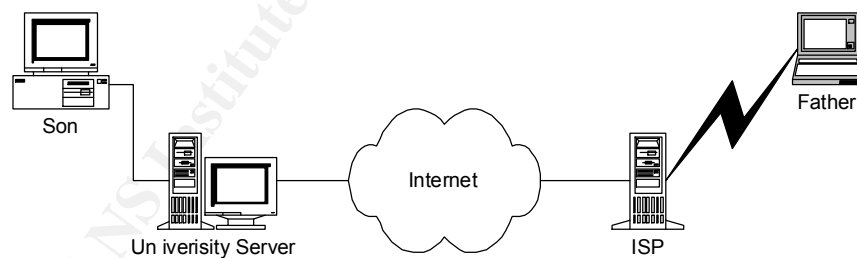


Figure 1

Unfortunately, although this may look simple to people chatting, this configuration does not accurately display the complexity of this session. In reality there are many more components to this discussion between father and son. (Figure 2)

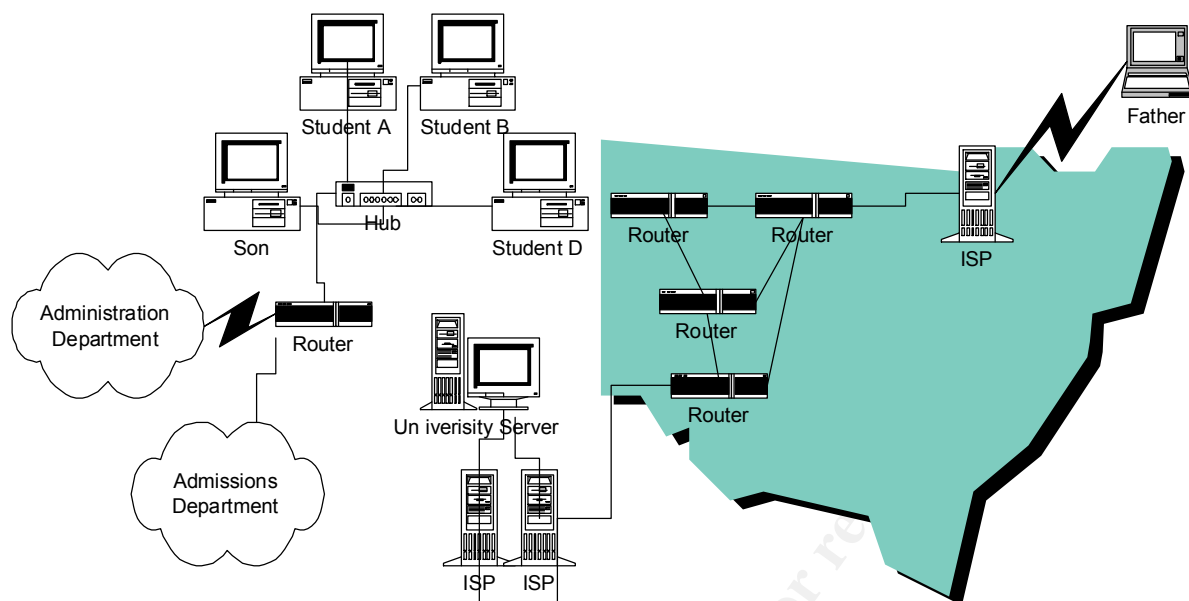


Figure 2

Although very primitive, the image displayed in Figure 2 more clearly depicts the complexity of chatting over the Internet. In our scenario, each word sent between father and son is relayed or passed via various components. In the son's environment, his Internet connection is actually being shared with multiple people: students in his dorm, the Administration department, and the Admissions department. Conversely, the father is simply creating a one-to-one connection with his ISP, but as you can see, his words still pass along multiple routers and paths.

Why is this concept being introduced?

When protecting the confidentiality of data, it is important to understand your system and how the data is passed between users. At some points in the transmission, you (the user) may have no control over how the data is protected. For example, once the son pressed the 'Send' key to initial the initial greeting to his father, how did the bits and bytes of his message arrive on his father's screen? In terms of MSN Messenger and the demonstration provided in the subsequent portion of this paper, it is imperative to have a clear understanding that although the conversation taking place may be only displayed as one-to-one, the process and complexity of the communications path is enormous.

Identify the threat

When using any type of application over the Internet, it is important to understand the different threats. Applied to this paper, there are two main types: Internal and External. Each type of threat is defined below:

Internal - Security personnel, system user, or malicious attacker located within the operational environment of the network being assessed. For example, any student located within the college would be considered an internal threat to the system.

External - Security personnel or malicious attacker who is not directly connected to the network being assessed, but is able to gain access to the network communications path via hacking, cracking, operating system vulnerabilities, or by any other means.

Confidentiality: Can anyone read this message?

The steps listed below explain the process utilized to determine the risks associated with using MSN Messenger. The goal of this demonstration is to determine if sensitive information can be easily captured while chatting via the Internet. The test was conducted via a dial-up connection on a personal computer running Windows 98 and MSN Messenger.

1. The first requirement for conducting an assessment on MSN Messenger was to locate an application which would be able to monitor (“sniff”) the data being sent to and from my local computer. I had to find a way to clearly identify if unencrypted data was being sent over the Internet while chatting with my friends. The requirements for the monitoring program were as follows:

- **Cost:** Any monitoring software downloaded and installed for this test would need to be free or a fully functional trial version.
- **Functionality:** The monitoring software would need to include built-in support for the MSN Messenger protocols
- **Ease of Use:** Because I am not familiar with command line applications, the software would need to provide a simple Graphical User Interface (GUI)
- **Windows Based:** The test being conducted would be run from a computer running a Windows based Operating System (OS)

Based on the above requirements, an extensive search on the Internet resulted in locating the “[Ethereal](#)” program. This application not only met the basic prerequisites, but also included built in support for the [MSN Messenger protocol](#)^[3].

2. Installing Ethereal was done with a standard installation wizard. The only additional software required for using Ethereal was the [WinPcap](#) Packet Capture Architecture for Windows. The packet filter is a device driver that adds to Windows 95, Windows 98, Windows ME, Windows NT and Windows 2000 the ability to capture and send raw data from a network card, with the possibility to filter and store in a buffer the captured packets.

3. Next, I opened Ethereal, and selected Capture from the menu bar, then Start. (Figure 3)

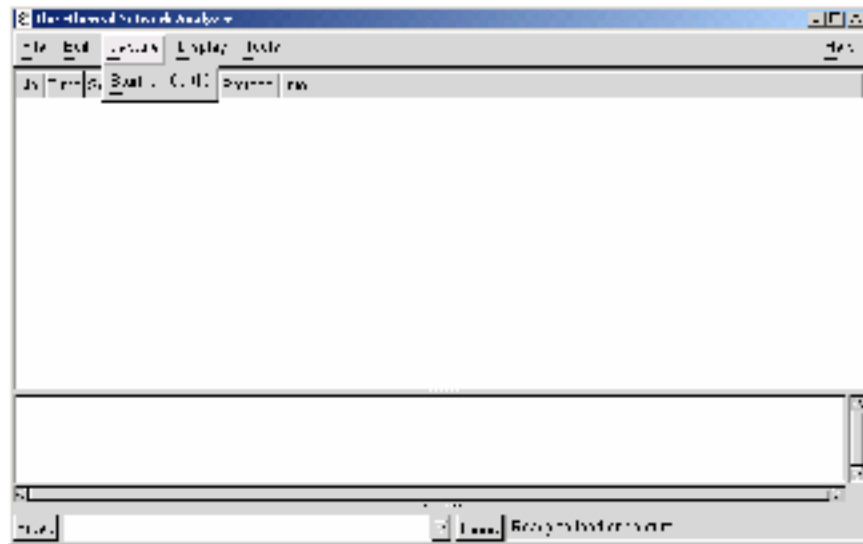


Figure 3

4. Selecting Start opened the Capture Preferences screen. (Figure 4) For the purpose of this scan, I decided not to modify the basic selections determined by Ethereal.

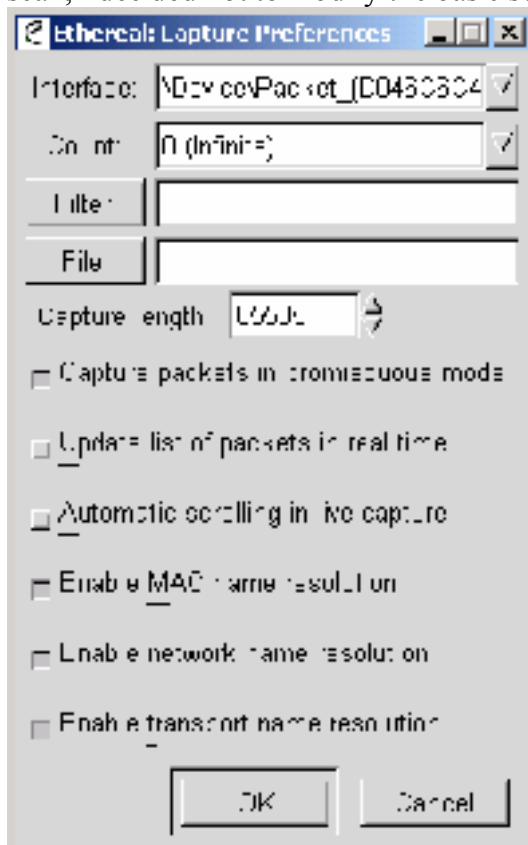


Figure 4

5. Next, I opened MSN Messenger, double clicked on one of my contacts, which displayed a basic chat window. For the purpose of this paper, the contact's identity has been concealed. (Figure 5)

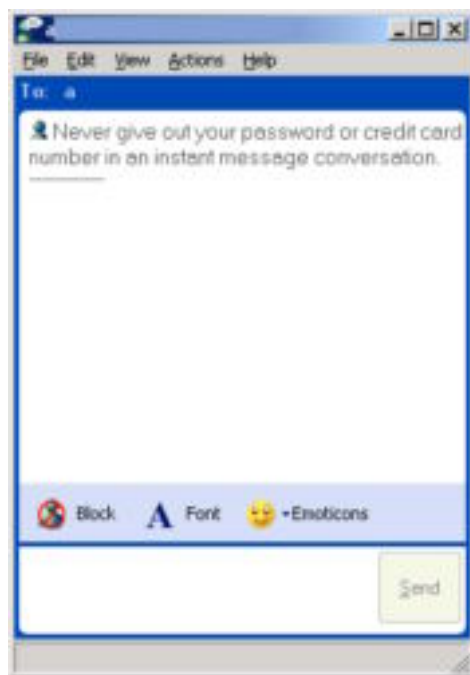
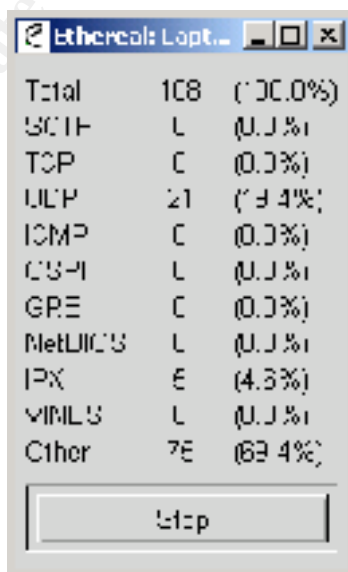


Figure 5

6. Before sending a message via MSN Messenger, I returned to Ethereal and clicked OK in the Capture Preferences screen. Once started, Ethereal provided a real-time monitor of the total number of packets captured. (Figure 6)



Total	108	(100.0%)
UDP	1	(0.9%)
TCP	0	(0.0%)
ICMP	21	(19.4%)
CSMA	1	(0.9%)
GRE	0	(0.0%)
NetBIOS	1	(0.9%)
IPX	6	(4.3%)
VMLB	1	(0.9%)
Other	76	(69.4%)

Help

Figure 6

7. Subsequently, with Ethereal monitoring my conversation, I sent a basic test message via MSN Messenger. (Figure 6)

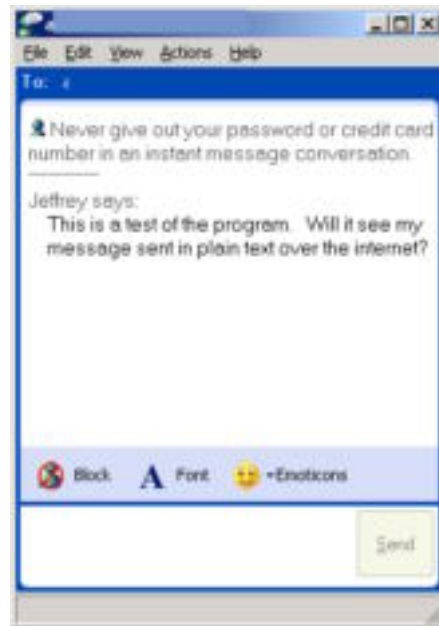


Figure 7

8. After sending the message via MSN Messenger, I returned to Ethereal and clicked the STOP button in the Capture window.

9. Upon clicking STOP, Ethereal proceeded to compile the information collected and present the data in sequential order. A sanitized display is shown below. (Figure 8)

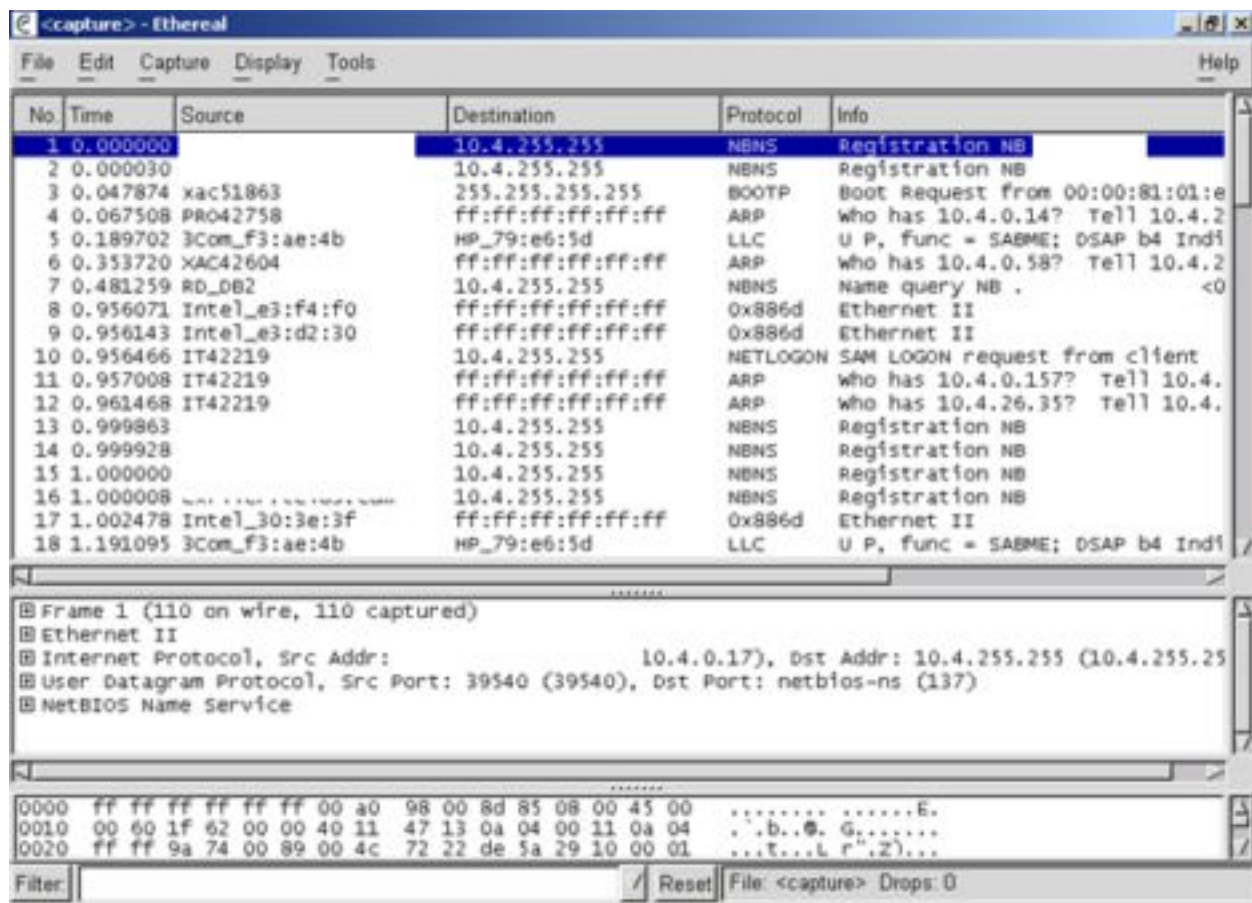


Figure 8

10. Scrolling down to packet number fifty-three (53), I was able to clearly identify the conversation I was performing with my contact. The (sanitized) information captured is listed below:

Frame 53 (284 on wire, 284 captured)
 Ethernet II
 Internet Protocol, Src Addr: xx5555 (xx.x.252.13), Dst Addr: msgr-sb44.xxx.xxxx.com (xx.x.xx.193)
 Version:
 Header length: 20 bytes
 Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
 Total Length: 270
 Identification: 0xc183
 Flags: 0x04
 Fragment offset: 0
 Time to live: 128
 Protocol: TCP (0x06)
 Header checksum: 0xe58f (correct)
 Source: xx5555 (10.4.252.13)
 Destination: msgr-sb44.xxx.xxxx.com (xx.x.xx.193)

Transmission Control Protocol, Src Port: 1091 (1091), Dst Port: 1863 (1863), Seq:
3643520148, Ack: 2279486975
Data (230 bytes)

```
0000 4d 53 47 20 31 34 35 20 4e 20 32 31 35 0d 0a 4d MSG 145 N 215..M
0010 49 4d 45 2d 56 65 72 73 69 6f 6e 3a 20 31 2e 30 IME-Version: 1.0
0020 0d 0a 43 6f 6e 74 65 6e 74 2d 54 79 70 65 3a 20 ..Content-Type:
0030 74 65 78 74 2f 70 6c 61 69 6e 3b 20 63 68 61 72 text/plain; char
0040 73 65 74 3d 55 54 46 2d 38 0d 0a 58 2d 4d 4d 53 set=UTF-8..X-MMS
0050 2d 49 4d 2d 46 6f 72 6d 61 74 3a 20 46 4e 3d 4d -IM-Format: FN=M
0060 53 25 32 30 53 68 65 6c 6c 25 32 30 44 6c 67 3b S%20Shell%20Dlg;
0070 20 45 46 3d 3b 20 43 4f 3d 30 3b 20 43 53 3d 30 EF=; CO=0; CS=0
0080 3b 20 50 46 3d 30 0d 0a 0d 0a 54 68 69 73 20 69 ; PF=0....This i
0090 73 20 61 20 74 65 73 74 20 6f 66 20 74 68 65 20 s a test of the
00a0 70 72 6f 67 72 61 6d 2e 20 20 57 69 6c 6c 20 69 program. Will i
00b0 74 20 73 65 65 20 6d 79 20 6d 65 73 73 61 67 65 t see my message
00c0 20 73 65 6e 74 20 69 6e 20 70 6c 61 69 6e 20 74 sent in plain t
00d0 65 78 74 20 6f 76 65 72 20 74 68 65 20 69 6e 74 ext over the int
00e0 65 72 6e 65 74 2e ernet.
```

11. Once finished with the scan of the MSN Messenger conversation, Ethereal provided an excellent assortment of file formats to store the data including libcap, Red Hat Linux 6.1 libcap, and Nokia libcap. (Figure 9)

© SANS Institute 2000 - 2002

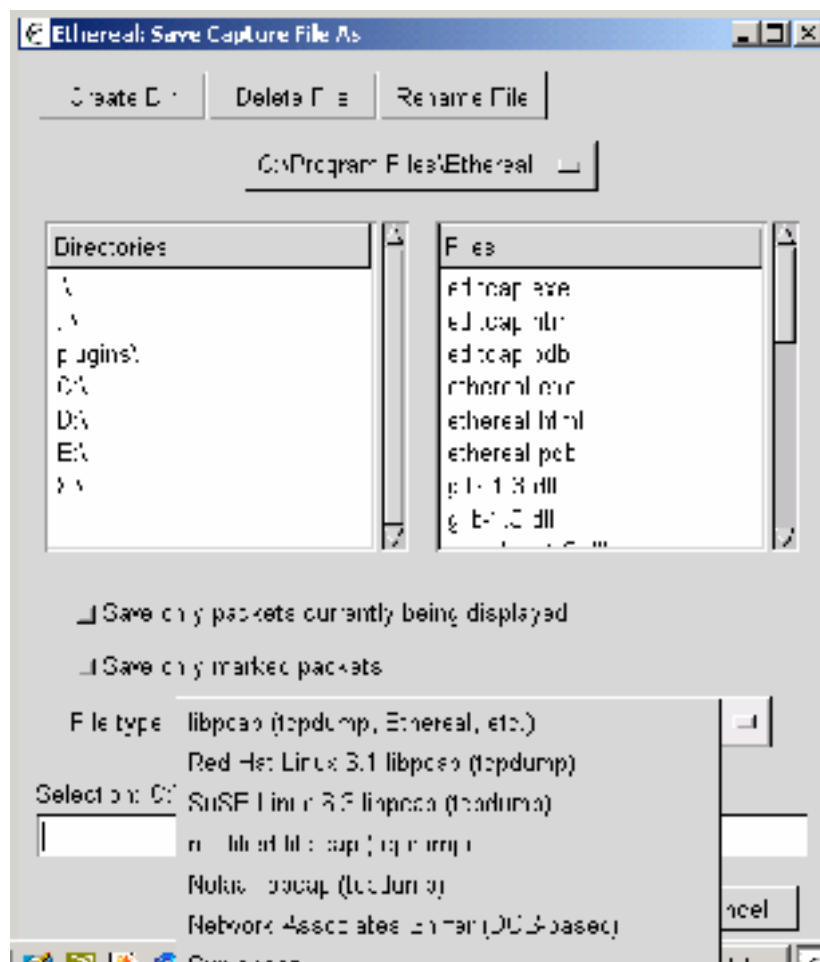


Figure 9

Additionally, by selecting File from the menu bar, and then Print, Ethereal allowed me to “Print” the information collected to either a plaintext or Postscript file format. (Figure 10)

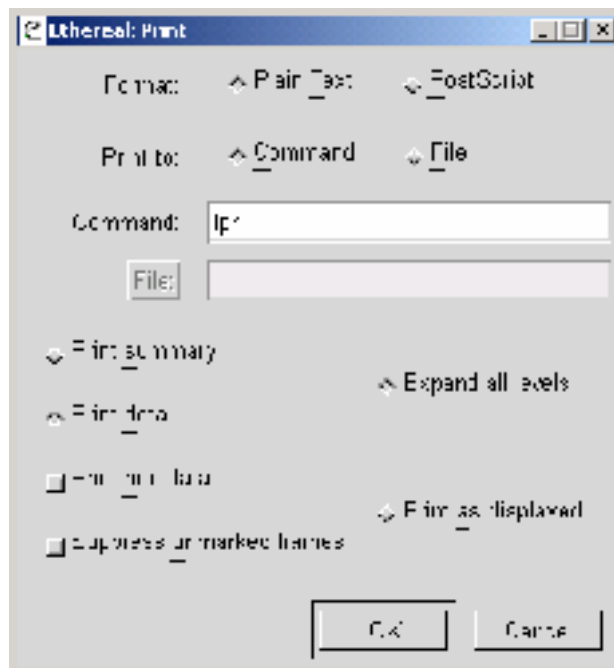


Figure 10

Assessment

Although I had never conducted any type of monitoring in the past and after analyzing the data, I was able to clearly recognize Ethereal's ability to capture the conversation and present the information without any requirement to decrypt the data. If this information had been classified or proprietary in nature, any malicious attacker with access to my network (internally or externally) would have the ability to easily see all conversations.

As currently designed, MSN Messenger provides no means to secure the information provided when chatting. Moreover, the help features included with MSN Messenger provide no explanation of the initial warning banner received when initializing a chat session.

Finally, a review of Microsoft's [.NET Messenger Service Statement of Privacy](#)^[4] states:

“ Please keep in mind that if you directly disclose personally identifiable information or other sensitive data to other users through the .NET Messenger Service, this information may be collected and used by the recipient(s). Note: Microsoft does not read any of your private online communications, your .NET Messenger Service messages are completely confidential.”

This statement, although technically correct, may lead a new user of MSN Messenger to assume that this “confidentiality” applies to all communications made via MSN Messenger, rather than just the portion controlled by Microsoft and its affiliates. The statement assumes that personally identifiable information is being disclosed willingly to another user via Microsoft's .Net Messenger Service. No where in Microsoft's document do they address responsibility for the unintentional loss of confidentiality due to information being sent in plain text.

Conclusion

The process used in the demonstration above may be applied against many types of Internet chat applications. From a major overflow vulnerability in [AOL's Instant Messenger](#)^[4] to a security flaw in [X-Chat](#)^[5], hackers will certainly continue to look for new means to intercept and exploit chat applications. Will this architecture change in the future? Only time will tell.

The concept of a secure chat environment is not new. In their [paper](#), the Internet Engineering Tasks Force (IETF) discusses the development of a standard Internet message format which can be signed or encrypted using MIME security multipart in conjunction with an appropriate security scheme. Does the IETF have the ability to enforce these standards? Many questions still remain unanswered.

In closure, the features and tools offered by today's communications applications are phenomenal. Just remember, as demonstrated above, it doesn't take a lot of time, money or effort to gather the tools necessary to invade your privacy when using these programs. Until they are explicitly protected by encryption, it is important not to disclose personal or confidential information via these types of applications.

References:

- [1] Gates, Bill. [Trustworthy Computing](#), Microsoft Memorandum, URL: <http://zdnet.com.com/2100-1104-817343.html?legacy=zdnm> (January 15, 2002)
- [2] National Information Systems Security (INFOSEC) Glossary, NSTISSI No. 4009, URL: <http://www.nstissc.gov/Assets/pdf/4009.pdf> (September 2000)
- [3] Ethereum, Version 0.2.80, Product Features, URL: <http://www.ethereum.com/introduction.html#features>
- [4] Microsoft, .NET Messenger Service Statement of Privacy, URL: <http://messenger.microsoft.com/support/privacypolicy.asp>
- [5] Conover, Matt. [AOL Instant Messenger overflow](#), w00w00!, URL: <http://www.w00w00.org/advisories/aim.html>
- [6] BUGTRAQ, ID # 3830, X-Chat CTCP Ping Arbitrary Remote IRC Command Execution Vulnerability, URL: <http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=info&id=3830>, (January 2002)