



# Malicious Threats and Vulnerabilities in Instant Messaging

By Neal Hindocha and Eric Chien  
Symantec Security Response  
Symantec Corporation

Originally published in Virus Bulletin International Conference, September 2003.  
Copyright held by Virus Bulletin, Ltd., but is made available free of charge by permission of Virus Bulletin. For more information on Virus Bulletin, please visit <http://virusbtn.com/>

## INSIDE

- > Instant Messaging Architecture
- > Malicious Threats to Instant Messaging
- > Vulnerabilities and Blended Threats
- > Information Disclosure
- > Password Theft
- > Current and Future Solutions

# Contents

Abstract & Preface .....	3
Instant Messaging Architecture .....	4
User Verification .....	5
Data Encryption and Authentication .....	6
Enterprise Editions .....	6
Programmability .....	6
Malicious Threats to Instant Messaging .....	7
Backdoor Trojan Horses .....	10
Vulnerabilities and Blended Threats .....	10
Information Disclosure .....	14
Password Theft .....	16
Current and Future Solutions .....	18
Antivirus .....	20
Summary .....	21
References .....	22
About the Authors .....	23

## > **Abstract**

Instant messaging is an up and coming threat as a carrier for malware. More and more people are using instant messaging, both for personal and business reasons. Instant messaging networks provide the ability to not only transfer text messages, but also transfer files. Consequently, instant messengers can transfer worms and other malware.

Furthermore, multiple vulnerabilities have been discovered and have yet to be discovered in instant messaging clients. Such vulnerabilities not only give hackers remote access, but also provide access to fast spreading blended threats. Current blended threats are limited by their ability to find vulnerable hosts, but with instant messaging buddy lists, finding vulnerable hosts becomes significantly easier resulting in a blended threat that may propagate faster than CodeRed and Slammer.

This paper discusses current and future threats to the various instant-messaging networks, including how the major instant messaging networks operate. We will also demonstrate a variety of live attacks against instant messaging, including hijacking and monitoring entire instant messaging sessions, unauthorized remote access and control, and blended threat and classic worm propagation.

Finally, current and future solutions will be discussed for the various threats including how a company can secure instant messaging communication.

## > **Preface**

This paper primarily discusses Windows-based instant messaging clients. However, the analysis could equally be applied to other operating systems as well. Furthermore, MSN Messenger, AOL Instant Messenger (AIM), ICQ, Yahoo! Instant Messenger (YIM), and related third-party clients that operate on these networks are the primary focus of this paper. This paper does not discuss IRC or its clients.

While many of the explanations and examples site specific clients or instant messaging protocols, this discussion does not exclude other clients or instant messaging protocols. The examples have been primarily chosen due to their ease of explanation.

## > Instant Messaging Architecture

Instant messaging is a method for real-time communication over the Internet. The instant messaging system generally follows the client-server model. Communication between clients occurs either via a server, or a server brokers it. Furthermore, instant messaging servers provide clients with a channel for communication to other services such as email and stock quotes. This section describes the general architecture of instant messaging networks and clients.

### TYPICAL FEATURES

The primary function of an instant messaging client is to allow text or HTML messages to be sent to other users in real time. Files can also be transferred between users, and a range of other services have been implemented. This includes the ability to receive email notifications, stock quotes, multi-player online game brokering, video conferencing, voice over IP, and SMS sending.

Other features continue to be added to instant messaging clients including security features for enterprise editions.

### DATA EXCHANGE

There are two main ways messages and files are transferred between clients – server proxy and server broker.

#### *1) Server Proxy*

In the server proxy architecture, all instant messaging communication is passed through the server. For example, if the two users, Alice and Bob, want to exchange messages via instant messaging, they would not send the messages directly to each other. Instead, the messages would first be sent to the server. The server would then forward the message to the intended recipient. In this case, the server acts as a proxy between the users.

The benefit of this method is that both clients initiate outgoing connections to the server and either one does not require the ability to accept incoming connections on a port that a corporate firewall may block. However, sending messages to the server may first incur a time-delay and represent a privacy risk. In general, this is the default method that all major instant messaging networks use today.

#### *2) Server Broker*

In the server broker architecture, the only packets that are sent to the server are packets requesting the server to initiate communication between two clients. The server essentially facilitates the connection between the two clients. The server provides the clients with the connection information; the clients then directly connect to one another.

For example, if Alice wants to send a message to Bob, Alice will send a request to the server to initiate the session. The server will notify Bob that Alice wishes to chat with him. If Bob agrees, he replies to the server with his contact information (typically an IP address and port number) and this information is forwarded to Alice. Then, Alice can directly connect to Bob and messages between the two do not pass via the server.

This method reduces the load on the server and reduces the privacy risk, as potentially confidential messages are no longer sent to the central server. However, this method is often blocked by firewalls, as they are usually configured to not allow incoming connections. Yahoo! Instant Messenger (YIM) is an example of a client that uses server brokering. YIM will send the first message via the server and then attempt a direct connection. If the direct connection fails, YIM will continue to send messages via the server.

#### FILE TRANSFERS

In addition to simple text messages, all popular instant messaging networks support file transfers between clients. Most file transfers are done via server brokering.

Files transferred between clients do not pass through the server, but instead, the server acts as a broker and informs the sender on how to find the intended recipient. The file will then be transferred directly from one client to the other.

### > **User Verification**

A user must first log in to the instant messaging server before being allowed to chat to other users. One must have a pre-registered username and password. A challenge-response mechanism generally verifies the password. For example:

1. Alice starts her instant messaging client and types in her username and password.
2. The client sends the username to the server and receives a packet with a key (the challenge).
3. The client combines the key, the password, and some additional information, and then calculates a hash. The most common method for calculating the hash is to use the MD5 or SHA1 algorithm.
4. The client sends the resulting hash back to the server. If the hash is correct, Alice will be authenticated to the server. At this point, some networks will also send Alice a session cookie to be used with secondary services.

The challenge-response method is a fairly secure method for sending a password over an insecure network connection. However, the method is only as secure as the password itself, as the algorithm for calculating the hash remains static.

After a successful login, some networks will send the client a cookie. The client can use the cookie to log in to secondary servers to access other services, such as email and games. The cookie will be invalidated when the user disconnects, or after a predefined time period has elapsed since last receiving a packet from the client.

## > **Data Encryption and Authentication**

Today, the most popular instant messaging networks are designed for scalability and performance rather than security. Few of the popular instant messaging networks encrypt data by default. Therefore, all communication via instant messaging should be considered insecure and should never be used for sensitive communication.

Furthermore, data is not authenticated either. The instant messaging architecture does not provide a means for verifying that a message really originated from the sender. For example, a hacker can not only inject messages into an ongoing chat session, but can also hijack an entire session by impersonating one of the users.

Many third-party clients that interoperate with the popular instant messaging networks have added their own encryption mechanisms. For example, Trillian has the ability to encrypt data that is being transmitted over the AOL Instant Messaging network and PGP has a plug-in for ICQ.

## > **Enterprise Editions**

Recently, all the three major instant messaging network providers (Microsoft, AOL, and Yahoo) have announced corporate versions of their products. The corporate versions encrypt data transmitted over the network, as well as incorporate additional functionality, which corporations request, such as central logging, user access controls, and corporate screen names.

In addition, some versions will provide a gateway to the public instant messaging servers. Messages bound for people within the company will therefore stay behind the corporate firewall, whereas messages bound for people outside the company will be proxied to the public instant messaging servers.

## > **Programmability**

Two popular clients, ICQ and MSN Messenger, allow users to write their own applications that communicate with the instant messaging network and/or the client using documented APIs that can be found at the vendors' Web sites.

The APIs that exist for both MSN Messenger and ICQ allow developers to create applications that can send and receive messages, send and receive files, and receive notifications when certain instant messaging actions occur.

AOL does not provide any developer APIs for their client. Instead, they have documented a protocol called TOC that is interoperable with the AOL Instant Messaging network. The TOC protocol is not as feature-rich as the main AIM protocol known as OSCAR; however, the protocol provides the ability to transfer text messages and files to AOL IM clients. Thus, TOC allows developers to create custom third-party clients that can communicate with AOL IM clients.

Yahoo! Instant Messenger is the only popular instant messaging client that does not provide some level of interoperability.

While documented APIs allow developers to write legitimate applications that interoperate with the instant messaging client, virus writers can do the same and create worms and other malware that spreads via the instant messaging client. As a result, more than 20 worms spread using MSN Messenger, while none use Yahoo! Instant Messenger.

## > Malicious Threats to Instant Messaging

### CLASSIC INSTANT MESSAGING WORMS

Any Internet-enabled application is a potential carrier for worms and other malware. Instant messaging is no exception. Currently, more than 30 worms spread via instant messaging networks and their clients.

Generally, instant messaging worms use four methods to spread:

1. Utilizing exposed APIs by the vendor
2. Enumerating windows via the Windows OS APIs to interactively send a file
3. Sending a URL link instead of a file
4. Patching client DLLs to send itself along with the original message

#### *Instant Messaging APIs*

Using a documented set of APIs, developers can write applications that interface with the MSN Messenger and ICQ clients. Unfortunately, these APIs expose instant messaging functionality that is robust enough to create instant messaging worms. For example, the following MSN Messenger and ICQ APIs can be utilized to create an instant messaging worm:

`onTextReceived` (MSN) – When text is received, a structure will be sent to the calling process with the text of the message, the user who sent the message, and session information.

`onListAddResult` (MSN) – A notification will be sent to the calling process whenever a contact is added to the contact list.

`sendFile` (MSN) – Sends a file to another user. The sending user does not require a confirmation.

`sendFile` (ICQ) – Sends a file to another user. Distinctly different from MSN, as ICQ requires confirmation from the sending user before the file is sent.

`MessengerContacts.Item` (MSN) – This method allows one to enumerate each buddy in the buddy list.

`GetOnlineListDetails` (ICQ) – This API provides information regarding all users on the contact list.

With these APIs a virus writer can easily create a worm that spreads using ICQ or MSN Messenger. For example, a worm can be alerted when a message is received or when a user is added to the buddy list, and then send a file to that particular user. Furthermore, a worm could easily send itself to all contacts by enumerating the contact list.

With ICQ, a call to `sendFile` requires user confirmation before the file is sent. This requires the additional step of enumerating windows and sending Windows messages in order to mimic the confirmation by the user. Such techniques are discussed in the next section.

Unfortunately, utilizing exposed APIs to create an instant messaging worm is not simply theoretical – many worms already exist today that utilize these APIs. W32.Choke.Worm is a worm that replicates using documented MSN Messenger APIs and simply sends itself in reply to any incoming text message.

For example, Alice's system is infected with W32.Choke.Worm. Bob, a contact of Alice, sends Alice a message. Bob then receives the following message from Alice (which the worm actually sends) and the invitation to download a file.

`President bush shooter is game that allows you to shoot Bush balzz hahaha`

Of course, the file is the worm itself.

#### ENUMERATING WINDOWS AND WINDOWS MESSAGES

Certain Windows APIs allow one to enumerate all application windows and send Windows messages to any window without regard to any user or process security rights. Effectively, this means that a program can be created that would “click” any of the necessary buttons to enumerate a buddy list or send a file via an instant messaging client window.

For example, to send a file using Yahoo! Instant Messenger, a user would perform the following sequence of events:

1. Right-click the contact you wish to transfer the file to.
2. Choose “Send a File” in the list that appears.
3. Fill in the file name in the correct input box.
4. Click the “Send” button.

An application can also perform this sequence of events. The application would do the following:

1. Get a handle to the correct window by using **FindWindow**.
2. Use a Windows message to automatically right-click the contact name.
3. Get a handle to the opened sub-menu and use **SendMessage** to click the “Send File” menu option.
4. Fill in the filename and click the “Send” button via **SendMessage**.

While the above description is simplified, Microsoft has made available and documented all the required APIs.

An example of a worm that enumerates windows via this method is W32.Goner.A@mm. The worm uses the aforementioned ICQ APIs to enumerate and send itself to all the contacts on the contact list. However, because the ICQ **SendFile** API requires user confirmation (the “Send” button in a dialog box needs to be clicked), the worm also enumerates windows for the confirmation dialog box and automatically “clicks” the appropriate button to send the file.

While the worm utilized the ICQ APIs to enumerate the contact list and invoked sending itself, using window enumeration and Windows messages could have been easy to use for the entire process.



#### SENDING URLs

To avoid complex cases of windows enumeration and cases where exposed APIs lack the ability to send files, worm authors have sent text messages with a URL link rather than the file itself. The link then points to the worm file residing on a remote server or an infected machine.

W32.Aplore.A@mm uses this method to spread itself via AOL Instant Messenger. When the worm is executed, it will first start an HTTP server on port 8180 to serve itself to incoming connections.

The worm then sends a message to all contacts on the buddy list. The message is picked from a list of 24 possible messages and includes a URL link to the infected system.

When the receiving buddy visits the link, a page with the following text appears.

**Browser Plugin Required:**

You may need to restart your browser for changes to take affect.

Security Certificate by Verisign 2002.

MD5: 9DD756AC-80E057FC-E00703A2-F801F2E3

Click [HERE](#) and choose "Run" to install.

Of course, installing the file actually downloads and executes W32.Aplore.A@mm.

#### FILE PATCHING

The behavior of instant messaging clients can be modified by patching the various files associated with them. For example, a DLL file could be patched so that whenever a user sends a message, a malicious file will be sent along with the message. Alternatively, if a file is sent, a malicious file could replace the file. This DLL could be an instant messaging-specific DLL or an operating system DLL that deals with network connectivity, such as Winsock.

W32.AimVen.Worm is an example of a worm that uses this method to spread itself. The worm locates and modifies the file C:\Program Files\AIM95\lcbmft.ocm, which is simply a renamed DLL. lcbmft.ocm contains code relating to sending instant messages and files. Once the worm modifies the file, any time a .EXE file happens to be sent using the modified AIM client, the W32.AimVen.Worm replaces the file.

## > **Backdoor Trojan Horses**

Backdoor Trojan Horses use the same techniques as those utilized by instant messaging worms, but instead of sending themselves (replicating), backdoor trojans export sensitive information or wait for specific messages to arrive, instructing them to perform a malicious action.

For example, Backdoor.Kryost sends a message to the author of the virus informing him that the backdoor trojan has been executed on the victim system. Then, the hacker can send messages, which the installed backdoor trojan monitors, to the infected buddy. These messages contain commands to be executed on the infected system, effectively giving the hacker unauthorized remote access.

Furthermore, this unauthorized remote access is not blocked by a firewall as instant messaging communication has already been permitted.

In addition, a backdoor trojan can modify configuration settings of the instant messaging client allowing unauthorized remote access. For example, some instant messaging clients offer file-serving capabilities. This service is off by default and when enabled generally will only expose files in a particular directory. A backdoor trojan could modify such settings to enable file-serving on the entire drive, effectively giving the hacker remote read and write access to sensitive materials.

## > **Vulnerabilities and Blended Threats**

The discovery of vulnerabilities in network-enabled applications occurs everyday. Instant messaging clients are no exception. These vulnerabilities are common coding mistakes made by programmers. At best, these vulnerabilities can cause a Denial of Service (DoS), and at worst, can allow hackers unauthorized remote access. Furthermore, remotely injected code could contain classic worm replication functionality, forming an instant messaging blended threat. This type of threat would have the potential to spread significantly faster than even CodeRed and W32/Slammer.

While a variety of vulnerabilities have been publicly recorded, an AOL IM buffer overflow (CVE-2002-0005) is described below as an example of the type of vulnerabilities that exist in instant messaging clients. This particular vulnerability has been chosen solely because the vulnerability was publicly reported, has since been patched, and is a classic programming error that is easily exploitable.

### ANALYSIS OF CVE-2002-0005

On January 2, 2002, Matthew Conover reported a vulnerability allowing unauthorized remote execution of code in the AOL Instant Messaging client. The AOL IM client DLL, rvapps.ocm, contained the following code snippet that formats a user, supplied string into a clickable URL hyperlink:

```

push  [ebp+szUserSuppliedString]
push  offset szHTMLMarkup
push  eax          ; result_buffer[1024]
call  ds:sprintf

```

where szHTMLMarkup is the string:

```
<table width=100%><td align=center><a href="%s">%s</a></td>
```

where szHTMLMarkup is the string:

```

char result_buffer[1024];

sprintf(result_buffer,
        "<table width=100%><td align=center><a href=\"%s\">%s</a></td>",
        szUserSuppliedString,
        szUserSuppliedString2);

```

Unfortunately, szUserSuppliedString is never bounds-checked. One could easily supply a large string for szUserSuppliedString and overflow the allocated 1024-byte destination buffer on the stack, overwriting the return instruction pointer and redirecting the application to execute malicious code. This is illustrated by the following stack layout when the `sprintf` occurs.

char result_buffer[1024]
char buf1[260]
char buf2[260]
DWORD dw1
DWORD dw2
DWORD dw3
DWORD dw4
DWORD dw5
DWORD dw6
Old EBP
Return EIP

This code is reached when the AOL client parses a particular type of request. The original discovery pinpointed the vulnerability in processing AOL IM game requests, a feature that allows one to send a message to a buddy prompting them to join you in a particular multi-player online game. However, other types of requests could trigger the overflow as well.

After public notification of the vulnerability, AOL partially filtered any exploit requests at the AOL IM server, but clients remained affected for many months afterward. For example, on May 6, 2002, a modified exploit was discovered that bypassed the server filter. Fortunately, no publicly reported exploitation occurred. However, such a simple vulnerability could have easily been combined with replicating code to create a blended threat.

## BLENDED THREAT IMPACT

Instant messaging blended threats have the potential to replicate much more rapidly than the blended threats that have been seen to date, mainly because blended threats are currently limited by their ability to find vulnerable hosts.

For example, CodeRed II selected IP addresses closest to the infected system in attempt to make it more effective in rapidly scanning the possible  $2^{64}$  IP addresses. CodeRed II ended up taking 14 hours to iterate through this address space, eventually infecting over 350,000 machines.

In January 2003, W32/Slammer demonstrated that a blended threat could iterate through the entire IP address space in an even quicker amount of time. W32/Slammer was similar to CodeRed II in generating random IP addresses, but instead of TCP-utilized UDP datagrams, allowing it to send itself to remote hosts without waiting for a connection reply (TCP requires an acknowledgement from the remote host). Thus, W32/Slammer was not limited by network latency, but simply by network bandwidth. This allowed W32/Slammer to scan the entire IP address space in less than 15 minutes, infecting approximately 80,000 machines.

While an instant messaging blended threat will likely need to use TCP and be limited by network latency similar to CodeRed, an instant messaging blended threat will not need to iterate through the entire IP address space in search of vulnerable machines. Inherent to instant messaging is a buddy list, and therefore a pre-populated target list of potentially vulnerable machines. Furthermore, the number of instant messaging users far exceeds the number of Web servers or SQL servers on the Internet.

By eliminating the need to scan for vulnerable machines, one can infect hundreds of thousands of machines in seconds rather than minutes. The following approximation can be used to estimate the impact:

$$\begin{aligned} \text{Machines\_Infected} = & \\ & \text{Avg\_Unique\_Buddies}^{((1 * \text{Time\_To\_Send}) / \text{Time\_To\_Send})} + \\ & \text{Avg\_Unique\_Buddies}^{((2 * \text{Time\_To\_Send}) / \text{Time\_To\_Send})} + \\ & \dots + \\ & \text{Avg\_Unique\_Buddies}^{(\text{Time\_Elapsed} / \text{Time\_To\_Send})} \end{aligned}$$

where:

**Avg\_Unique\_Buddies:** The average number of buddies on the victim host that are different than the infecting host.

**Time\_Elapsed:** The time since the first infection.

**Time\_To\_Send:** The average time needed to send the malicious packet from one host to another.

**Machines\_Infected:** The number of machines that will be infected, assuming all machines on one's buddy list are vulnerable.

The following table details the approximate number of seconds required to infect 500,000 machines for a few realistic values:

## Number of Seconds to Infect 500,000 Machines

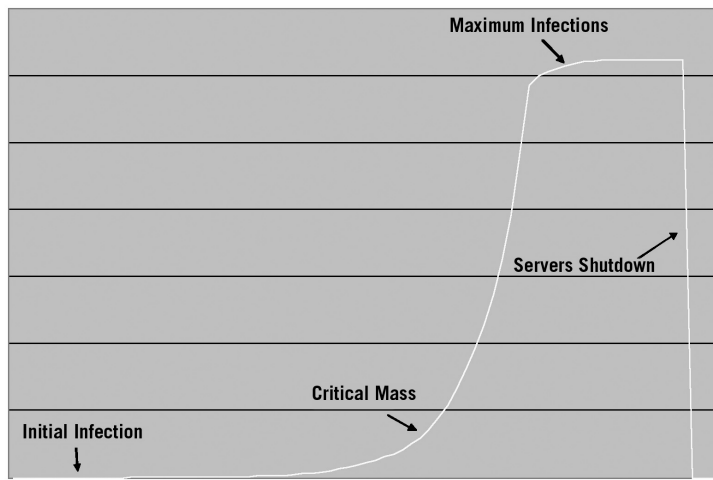
Unique Buddies	Time to Send (secs)					
	0.5	0.75	1.0	1.5	2.0	2.5
1.2	31	47	63	94	126	157
1.5	15	22	30	45	60	75
2.0	9	13	18	27	36	45
2.5	7	10	14	21	28	35
3.0	6	9	12	18	24	30

There are a variety of factors that make this an ideal calculation, but nevertheless, orders of magnitude exist between this infection rate and that of W32/CodeRed or W32/Slammer. Furthermore, after sending itself to the buddies on the existing buddy list, an instant messaging blended threat could also generate random buddy list names, thereby increasing its infection rate even further. Thus, an instant messaging blended threat could infect all vulnerable machines in seconds rather than minutes or hours.

While an instant messaging blended threat could spread extremely fast, such a threat could also be eliminated relatively quickly. As this threat relies on the instant messaging server, as soon as the instant messaging servers are shut down, the blended threat will no longer replicate, unless the threat carried some additional propagation component not reliant on instant messaging.

In any case, the instant messaging servers would likely suffer a DoS due to the network traffic generated by the blended threat itself, preventing further replication. Thus, if the instant messaging servers are restarted and force the user to upgrade to a patched client or have a filter to prevent the forwarding of malicious traffic, the threat is then effectively neutered.

The following timeline gives a sample prediction of how an instant messaging blended threat would spread over time. Notice that once the instant messaging servers are shut down, the number of infections drops immediately to zero.



## > Information Disclosure

Using instant messaging can put one at risk of having confidential information disclosed. A hacker can obtain passwords, system configuration information, and sensitive files via instant messaging. This data can be stolen without a breach of the actual system and without the knowledge of the instant messaging user. More importantly, the resultant damage due to information disclosure can outweigh the direct damage due to a malicious threat.

This section will cover hijacking instant messaging sessions, impersonating other users, maliciously proxying data, sniffing network traffic, password theft, and exporting data via instant messaging.

### HIJACKING AND IMPERSONATION

One can hijack an instant messaging connection or impersonate other users in a variety of ways. Some methods are based on the inherently insecure nature of TCP/IP and related protocols, while others are specific to the design of the instant messaging protocol. These techniques necessitate that secure instant messaging clients utilize methods of data authentication to ensure that the data truly originates from the supposed source.

#### *Session Cookie Attacks*

Many instant messaging protocols use different servers for user authentication and the actual transmission of chat messages. Such designs allow an instant messenger client to authenticate to a particular server, retrieve a session cookie, and then utilize this session cookie to log in to secondary servers, such as the messaging server.

However, if this session cookie is captured via network sniffing, it can be used to impersonate the authenticated user.

For example, Alice logs in to the login server and receives a session cookie from the login server. When the cookie is received, Alice's client disconnects from the login server and attempts to connect to the messaging server. In the meantime, Eve has been sniffing network traffic and captures Alice's session cookie. Eve then uses the session cookie to log in to the messaging server herself, impersonating Alice. At the same time, Eve also conducts a DoS attack on Alice to prevent her from logging in to the messaging server. Eve is now successfully able to use instant messaging and appear as Alice.

*Man-In-The-Middle Attacks*

All data between the instant messaging server and client is unauthenticated. Instant messaging traffic is therefore vulnerable to classic TCP/IP man-in-the-middle attacks.

For example, Eve, a hacker, can poison the DNS cache for the user Alice, so that Alice's computer believes that Eve's computer is the instant messaging server.

When Alice attempts to log in to the instant messaging server, she is redirected to Eve's system. Eve's system simply proxies all the data to the real instant messaging server, but is able to both spy on all traffic passing through it and also to inject her own messages into the message stream. Eve effectively has full control over the session and can impersonate Alice to other users and vice versa.

*Data Export*

Due to the fact instant messaging network traffic is not encrypted, all data sent across the wire can be captured using a network sniffer.

For example, the following two screen shots show a network dump of two users chatting online:

```

0050 08 65 63 76 62 32 30 30 33 00 00 00 03 00 01 00 .ecvb200 3.....
0060 02 00 11 00 0f 00 04 00 00 00 2a 00 03 00 04 .....*.....
0070 00 02 00 7e 05 01 00 03 01 01 02 01 01 .....~.....
0080 00 73 00 00 00 00 3c 48 54 4d 4c 3e 3c 42 4f 44 .s....<H TML><BOD
0090 59 20 42 47 43 4f 4c 4f 52 3d 22 23 66 66 66 66 Y BGCOLOR="#ffff
00a0 66 66 22 3e 3c 66 6f 6e 74 20 66 61 63 65 3d 22 ff"><font face="
00b0 41 72 69 61 6c 22 3e 48 69 20 44 61 64 21 20 49 Arial">H i Dad! I
00c0 27 6d 20 6c 6f 77 20 6f 6e 20 63 61 73 68 2e 20 'm low o n cash.
00d0 20 43 61 6e 20 79 6f 75 20 73 65 6e 64 20 79 6f Can you send yo
00e0 75 72 20 43 43 23 3f 3c 2f 42 4f 44 59 3e 3c 2f ur CC#?< /BODY></
00f0 48 54 4d 4c 3e HTML>

0050 08 65 63 76 62 32 30 30 33 00 02 00 75 05 01 00 .ecvb200 3...u...
0060 03 01 01 02 01 01 00 6a 00 00 00 00 3c 48 54 4d .....j ....<HTM
0070 4c 3e 3c 42 4f 44 59 20 42 47 43 4f 4c 4f 52 3d L><BODY BGCOLOR=
0080 22 23 66 66 66 66 66 66 22 3e 53 75 72 65 20 73 "#ffffff ">Sure s
0090 6f 6e 2e 20 20 49 74 20 69 73 20 31 32 33 34 20 on. It is 1234
00a0 31 32 33 34 20 31 32 33 34 2e 20 1234 123 4 1234.
00b0 20 53 70 65 6e 64 20 61 6c 6c 20 79 6f 75 20 77 spend a ll you w
00c0 61 6e 74 21 3c 2f 42 4f 44 59 3e 3c 2f 48 54 4d ant!</BO DY></HTM
00d0 4c 3e L>

```

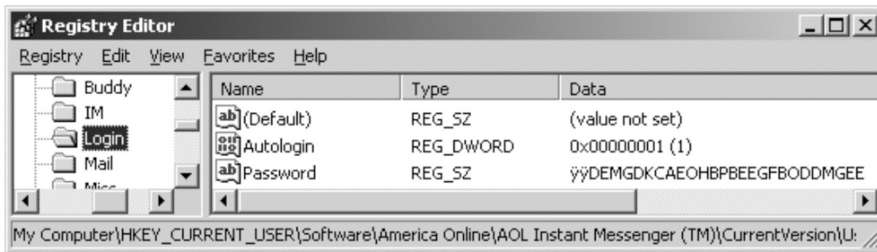
Network sniffing is not the only method of retrieving sensitive data via instant messaging. Instant messaging could also be used as a communication channel to export data found on the system. This could be most easily achieved by installing a simple backdoor trojan as described previously.

## > Password Theft

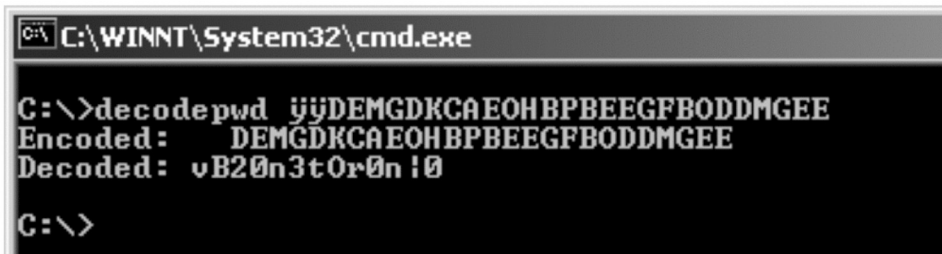
### CACHED PASSWORDS

Instant messaging clients offer the ability to cache previously used passwords. These passwords are generally stored in an obfuscated manner in the registry or a file on the system. Recovering the plain text password is trivial in these cases. For example, AOL IM stores the obfuscated version of the password in the registry value **Password** under the key:

```
HKCU\Software\America Online\AOL Instant Messenger (TM)\ CurrentVersion\
Users\<username>\Login
```



The password can be instantly recovered as it is a mere obfuscation as seen below:



In this case, AOL obfuscates passwords simply by taking the high and low nibbles of the character byte separately and translating the value based on its value and position within the string.

The other popular instant messaging clients employ similar obfuscation algorithms.

### PASSWORDS IN NETWORK TRAFFIC

Information to recover the password can also be obtained without access to the actual system by sniffing network traffic. The challenge-response mechanism utilized by most instant messaging clients requires a brute force attack. However, passwords with a length up to and including six characters could be easily broken over a weekend using a standard PIII 2 GHz CPU.



The following two screenshots show an example of AOL Instant Messaging network traffic. The first highlights the challenge key and the second highlights the response from the client:

```
0030 40 00 a5 21 00 00 2a 02 c0 6f 00 16 00 17 00 07 @...!...*. .0.....
0040 00 00 00 00 00 00 0a 31 39 36 32 38 32 32 38 ..... 19628228
0050 39 37 97
```

```
0050 6e 00 25 00 10 85 46 aa 1b 65 f5 45 44 b4 98 9b n.%...F. .e.ED...
0060 80 9e 0d b2 fe 00 03 00 32 41 4f 4c 20 49 6e 73 ..... 2AOL Ins
0070 74 61 6e 74 20 4d 65 73 73 65 6e 67 65 72 20 28 tant Mes senger (
0080 52 4d 70 7c 70 76 65 73 73 60 6f 60 70 74 70 77 SM) ver sion 4.7
```

Neither of these byte-sequences is useful alone. Together though, there is enough information available to perform a brute force attack. For example, we were able to recover a password 5 characters long in just over 2 minutes on a P4 Mobile 2GHz machine:

```
[c:\lmd5brute.exe 1962822897 8546aa1b65f54544b4989b809e0db2fe
z = bd62fd95bda60661b79d10e5044e2680
75 tries in 0 secs
zz = 3ccc7fe45f7b43d86e7934c13076b06b
5700 tries in 0 secs
zzz = 6099830f19d0af9993b99f492b6ff911
427575 tries in 0 secs
zzzz = e0812c7c42c7c7c26c0805e427a3952f
32068200 tries in 39 secs

Found!
2k3Ub = 8546aa1b65f54544b4989b809e0db2fe
120259851 tries in 137 secs
```

At this rate, the five-character key space can be completed in 45 minutes; a six-character key space in 56 hours. The time needed would increase if special characters are allowed and would decrease if additional computing systems were used.

Thus, while a challenge-response mechanism is useful in preventing the recovery of a plain text password, the password must be of sufficient length or it can be easily extracted using brute force.

## > Current and Future Solutions

While striking the balance between the legitimate need for instant messaging and the dangers inherent in its use is difficult, one can minimize their risk with a basic set of security policies.

The single most effective method of minimizing risk is to have a corporate policy restricting the use of instant messaging. In addition, a variety of products exist to assist in minimizing the risk of infection from malicious threats or information disclosure.

The solutions section will cover corporate policy, firewall rules, domain blocking, using a proxy, intrusion detection systems, antivirus software, and potential future solutions.

### FIREWALLS

A firewall can be very effective at blocking incoming connections and rogue outgoing connections. However, in the case of instant messaging, instant messaging traffic can frequently travel to commonly allowed destination ports, such as HTTP (port 80).

By default the popular instant messaging clients use the following destination ports:

- AOL IM: port 5190
- ICQ: port 5190
- MSN Messenger: port 1863
- Yahoo! IM: port 5050

However, if these ports are blocked, these instant messaging clients will attempt connections on common destination ports such as telnet (23), FTP (20/21), SMTP (25), POP (110), and NNTP (119). Often, this is enough to bypass a corporate firewall.

However, a protocol analysis firewall can block such traffic by default, as the instant messaging traffic does not match the protocol typically utilized on that particular port. For example, instant messaging traffic does not resemble SMTP email traffic.

If a client is unable to access the instant messaging server via a common destination port due to protocol analysis, instant messaging traffic can instead be tunneled via HTTP (80). Typically, instant messaging packets are embedded into an HTTP POST request and thus can bypass both types of firewalls, as the instant messaging traffic will appear to them to be a standard POST request.

In order to block such traffic, one must block by domain name or IP address. By default the popular instant messaging clients use the following domains for login:

- AOL IM: login.oscar.aol.com, aimexpress.oscar.aol.com
- ICQ: login.icq.com
- MSN Messenger: messenger.hotmail.com, gateway.messenger.hotmail.com, loginnet.passport.com
- Yahoo! IM: scs.msg.yahoo.com, scsb.msg.yahoo.com, scsc.msg.yahoo.com, scs.yahoo.com, shttp.msg.yahoo.com

Unfortunately, these domains are liable to be changed at any time and consist of a range of IP addresses. In addition, some of these domains translate into the same set of IP addresses.

Finally, a centrally managed desktop firewall can effectively block instant messaging via a “white list.” Desktop firewall applications match network traffic to the calling application. A “white list” would allow certain applications, such as Internet Explorer, but would deny all other applications network access, including instant messaging clients.

#### PROXIES

Even with the above domains blocked, some instant messaging clients can still access the instant messaging servers via an unblocked proxy. Third-party proxies exist to forward traffic to the official instant messaging server. These proxies’ domain names and IP addresses change regularly, and keeping up with them will require vigilance in analyzing firewall logs and blocking, accordingly.

One can instead set up a company proxy and, by blocking all other outgoing Web traffic, force employees to use it. The company proxy will prevent access to other proxies and should be configured to block the ability to connect to the public instant messaging servers. This, combined with blocking at the firewall, can effectively prevent instant messaging.

#### NETWORK INTRUSION DETECTION SYSTEMS

Network intrusion detection systems can also be used to identify unauthorized instant messaging traffic. However, more importantly, intrusion detection signatures can detect worms, blended threats, and hacking attempts on instant messaging clients.

Network intrusion detection systems can detect malformed or suspicious packets as they travel through the network. As instant messaging protocols are well-understood, extraordinarily long packets or byte-sequences often used in buffer overflows can easily be distinguished from legitimate traffic.

Unfortunately, intrusion detection systems are still relatively immature and well-crafted exploits are less likely to be detected generically, so the detection of new exploits will often require a specific signature to be created. With time, network intrusion detection systems will likely mature to support anomaly detection specific to instant messaging traffic.

## > Antivirus

Currently, some antivirus programs hook into popular instant messaging clients as a plug-in. The instant messaging client will pass files to the antivirus plug-in for scanning prior to notifying the user that the file has arrived. The antivirus program does not scan all instant messaging traffic, but rather just the files that are transferred via instant messaging. Thus, some types of blended threats or hack attempts will not be detected by such an antivirus plug-in.

Antivirus products that scan specifically for instant messaging threats at the gateway currently do not exist, but are in development. Solutions will plug into existing proxies that recognize instant messaging traffic, or the antivirus solution itself will proxy instant messaging traffic.

Unfortunately, files and other instant messaging traffic between users in a single corporation will often occur as a direct connection (not passing through a gateway). Thus, antivirus at the client level will always be required.

Enterprise instant messaging servers are also now available. These servers will proxy instant messaging traffic (file transfers may still occur directly between users) and are an ideal tier for antivirus scanning to take place. Thus, in the future, antivirus solutions will also plug into enterprise instant messaging servers, scanning for potentially malicious content.

While instant messaging antivirus solutions for corporations are still in their infancy, development in this area is ongoing, and several solutions are already on the horizon.

### CORPORATE POLICY

Corporate policies should be the first rather than the last line of defense. They are often the most effective and the easiest to implement. First, one should simply determine whether instant messaging is a business necessity, weighing the business necessity against the potential risk of information disclosure or infection from malicious threats.

If instant messaging is required for the organization, one should standardize on a particular client. If the client does not support enterprise features such as central logging and encryption, then strict rules should be enforced regarding the usage of the client. In particular, the use of instant messaging to discuss any business matters should be prohibited. Additionally, nicknames should not reflect their association with the corporation. Obviously, these requirements may make instant messaging almost useless for pure business communication.

Furthermore, file sharing should be blocked, and even potential incoming file transfers should be regulated. If incoming file transfers are required, users should be briefed on the policies of accepting and executing unknown files that arrive via instant messaging.

Also, a password policy should exist for complex passwords of at least eight characters in length (the longer the better) and passwords should not be cached.

Finally, a locked-down system is preferable, whereby an administrator can install and configure the instant messaging client according to prescribed policy.

Establishing a corporate policy can go a long way toward minimizing the risk of instant messaging at minimal cost.

#### ENTERPRISE INSTANT MESSAGING

Any deployment of secure instant messaging must utilize features such as data authentication and encryption, central logging, client management, and user access controls. Currently, these features do not exist in most popular free instant messaging clients. One must implement a third party client or one of the major vendor's enterprise versions of their instant messaging infrastructure. With such features, instant messaging can be utilized securely to discuss sensitive business matters and be more easily protected from external malicious threats.

However, even with the deployment of an enterprise instant messaging client one must remember another usage of instant messaging by employees is to communicate with family and friends outside of the enterprise. Unapproved instant messaging clients and communication must still be blocked accordingly.

#### > **Summary**

Instant messaging has clearly taken off as a means of communication. The ability to communicate in real-time makes it an ideal medium for both business and personal communication. Unfortunately, threats that affect instant messaging already exist today, including worms and vulnerabilities that can give hackers remote access to vulnerable computers. These threats, combined with the potential for unintentional disclosure of business information make instant messaging relatively insecure for company use.

The impact from potential instant messaging blended threats has yet to be seen. A threat that can replicate in seconds can affect more than just instant messaging—the Internet infrastructure as a whole. End users and corporations should employ basic security practices and products such as intrusion detection and antivirus to mitigate the risk. Most importantly, corporations at the outset should assess whether instant messaging is even a business necessity. If so, enterprise versions of the instant messaging products should be utilized and administrators should be on the lookout for future enterprise security solutions that specifically address instant messaging threats.

> **References**

- 1 Messenger APIs - <http://msdn.microsoft.com/downloads/list/messengerapi.asp>
- 2 ICQ APIs - <http://www.icq.com/api/>
- 3 AIM for the Enterprise - <http://enterprise.netscape.com/products/aimsvcs/index.html>
- 4 Yahoo! Messenger Enterprise Edition - <http://enterprise.yahoo.com/products/msg/>
- 5 W32.Goner.A@mm - <http://securityresponse.symantec.com/avcenter/venc/data/w32.goner.a@mm.html>
- 6 W32.Aplore@mm - <http://securityresponse.symantec.com/avcenter/venc/data/w32.aplore@mm.html>
- 7 W32.AimVen@mm - <http://securityresponse.symantec.com/avcenter/venc/data/w32.aimven.worm.html>
- 8 W32.Choke.Worm - <http://securityresponse.symantec.com/avcenter/venc/data/w32.choke.worm.html>
- 9 CodeRed:A Case Study – <http://www.caida.org/outreach/papers/2002/codered/codered.pdf>
- 10 The Spread of Slammer – <http://www.caida.org/outreach/papers/2003/sapphire/sapphire.html>
- 11 CVE-2002-0005 - <http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2002-0005>
- 12 AOL Remote Buffer Overflow - <http://www.securityfocus.com/bid/3769>
- 13 AOL IM AddExternalApp Overflow - <http://www.securityfocus.com/bid/4677>

> **About the Authors**

**Neal Hindocha**

After finishing his studies in Sweden, Neal Hindocha joined Symantec Corporation as a virus researcher for the Security Response division in Leiden, Netherlands. Previously, Neal had worked in the IT industry in Sweden and the United Kingdom.

Neal currently works at the Symantec Security Response office in Dublin, Ireland, where he focuses primarily on security threats.

Neal has written several articles, presented whitepapers, and has spoken at various conferences regarding computer security.

**Eric Chien**

Eric Chien joined Symantec Corporation at the Symantec Security Response headquarters in Santa Monica, California in 1997. Chien graduated from the University of California, Los Angeles with a Bachelor of Science degree in Electrical Engineering and Molecular Genetics.

Currently, Chien heads research in the Europe, Middle East, and Africa (EMEA) regions, analyzing current virus threats and researching new threats in the world of viruses and malicious software. He has been a key developer in projects such as the Digital Immune System (DIS), Symantec's automated system of virus analysis, and the Seeker project, which proactively finds viruses on the Internet.

Chien has spoken at various conferences and published a variety of papers addressing threats to computer security via malicious software.

**SYMANTEC, THE WORLD LEADER IN INTERNET SECURITY TECHNOLOGY, PROVIDES A BROAD RANGE OF CONTENT AND NETWORK SECURITY SOFTWARE AND APPLIANCE SOLUTIONS TO INDIVIDUALS, ENTERPRISES AND SERVICE PROVIDERS. THE COMPANY IS A LEADING PROVIDER OF VIRUS PROTECTION, FIREWALL AND VIRTUAL PRIVATE NETWORK, VULNERABILITY ASSESSMENT, INTRUSION PREVENTION, INTERNET CONTENT AND EMAIL FILTERING, AND REMOTE MANAGEMENT TECHNOLOGIES AND SECURITY SERVICES TO ENTERPRISES AND SERVICE PROVIDERS AROUND THE WORLD. SYMANTEC'S NORTON BRAND OF CONSUMER SECURITY PRODUCTS IS A LEADER IN WORLDWIDE RETAIL SALES AND INDUSTRY AWARDS. HEADQUARTERED IN CUPERTINO, CALIF., SYMANTEC HAS WORLDWIDE OPERATIONS IN 36 COUNTRIES.**

**FOR MORE INFORMATION, PLEASE VISIT [WWW.SYMANTEC.COM](http://WWW.SYMANTEC.COM)**

**WORLD HEADQUARTERS**

**20330 Stevens Creek Blvd.  
Cupertino, CA 95014 U.S.A.  
408.517.8000  
800.721.3934**

**[www.symantec.com](http://www.symantec.com)**

**For Product information  
In the U.S. call toll-free  
800.745.6054**

**Symantec has worldwide  
operations in 36 countries.  
For specific country  
offices and contact numbers  
please visit our Web site.**