An empirical analysis of the RVP-based IM (MSN Messenger Service v3.6)

November 2001

Dimitrios Petropoulos
Encode Security Labs
3, Romanou Melodou Street
Athens - 15125 - Greece
d.petropoulos@encode-sec.com

Summary

An empirical analysis of the Microsoft Instant Messaging implementation based on Exchange 2000 and using the MSN Messenger Service v3.6 client, was performed by the Encode Security Labs with the intention to assess its security. The protocol analysis was based on the protocol reconstruction via packet capture due to the lack of any other kind of informative documentation.

The most important findings about the IM service are:

- it does not offer any confidentiality whatsoever
- it is vulnerable to man-in-the-middle attacks
- its authentication methods are weak and only employ unilateral authentication (the client never gets to authenticate the server)
- it does not offer any form of data origin authentication
- the IM service is not easy to firewall since the server uses arbitrary port numbers to deliver messages to clients

Background

On Friday 6 July 2001 a message was posted to BUGTRAQ, analysing the Messenger/ Hotmail password authentication mechanism and concluding that it contains serious vulnerabilities (see ref. 3).

Based on the above, Encode Security Labs decided to take a look at the underlying IM protocols and assess the security of the implementation.

IM Protocol Analysis

The basics

Communication between the IM client and the IM server are performed over TCP/IP. The TCP protocol is used with the client using various ports to generate the requests and the server using port 80 to receive the requests. For outbound notifications, the server is using ports > 1024.

The transport protocol is HTTP and the payload is a mixture of HTML and XML data.

User logon

The user logon process involves identifying the user to the IM client and setting and retrieving fundamental information. The client subsequently notifies the IM server in order for the user to be shown as 'online'. These are performed through a series of HTTP requests containing various special HTTP methods. The key HTTP request methods seen were:

- SUBSCRIBE
- PROPPATCH
- SUBSCRIPTIONS
- ACL

Each one of the above HTTP requests is refused by the server which returns, for each one, a HTTP reply with status code 401 (Unauthenticated) and requests digest access authentication (see ref. 6 for detailed discussion of digest access authentication). The client then repeats each of the four HTTP requests, this time including the 'Authorization' field containing the digest authentication string (see reference 5). Below is a schematic representation of the exchange between client (C) and server (S), shown for only one of the four requests.

```
C → S :   SUBSCRIBE                                    (1)
C ← S :   ACCESS_DENIED, WWW-Authenticate, Nonce       (2)
C → S :   SUBSCRIBE, MD5(userID, pwd, Nonce, URI, …)   (3)
```

Message exchange between users

When a user wants to send a message to another user, they double click on the recipient's name on their MSN Messenger Service GUI window or use the 'Send' button from the menu. At that time, an HTTP message with the NOTIFY method is sent to the IM server, indicating the name of the recipient. The server replies with a SESSION_ID, which the client includes in the message sent to the recipient. The session ID is derived by the server and is not modified by the communicating clients in any way. A schematic representation of this exchange is presented below:

```
C → S :   NOTIFY(recipient name)                        (1)
C ← S :   SESSION_ID                                    (2)
C → S :   NOTIFY(recipient name), SESSION_ID, MESSAGE (3)
```

If the recipient of the message wants to reply in the same pop-up window they received the message in then the interaction will have the same session ID. If they choose to open a different window, the reply will have a different session ID.

From the recipient's point of view the messages seen are:

```
C ← S :   NOTIFY(sender name), SESSION_ID             (1)
C ← S :   NOTIFY(sender name), SESSION_ID, MESSAGE    (2)
```

Security implications of the above

General observations

The exchange of any kind of message in this protocol is performed in the clear and no cryptographic protection is available. Consequently, if intercepted, all messages are intelligible (in fact it was through interception that this protocol analysis has been made possible).

User Logon

On user logon, the server requests Digest Access Authentication (computed using a password) as proof of the client's identity. Digest Access Authentication has a number of limitations, an extensive discussion on which can be found on the relevant RFC (see ref. 6)

A bigger problem than the one above however is that authentication is unilateral with the server authenticating the client. Since the client never gets to authenticate the server, the protocol is vulnerable to a man-in-the-middle attack, where an attacker might pose as a legitimate server to the user and relay the server's challenge to the client and the client's authentication request to the server, or schematically (where 'C' is the client, 'S' is the server and 'A' is the attacker):


```
C → A :   SUBSCRIBE                                    (1)
A → S :   SUBSCRIBE                                    (2)
A ← S :   ACCESS_DENIED, WWW-Authenticate, Nonce       (3)
C ← A :   ACCESS_DENIED, WWW-Authenticate, Nonce       (4)
C → A :   SUBSCRIBE, MD5(userID, pwd, Nonce, URI, …)   (5)
A → S :   SUBSCRIBE, MD5(userID, pwd, Nonce, URI, …)   (6)
```

Since none of the subsequent interactions between client and server rely on any common (or at least mutually computed) secret, the attacker manages to achieve total control over the stream (i.e. can arbitrarily modify all messages to and from client and server).

Message exchange

The message exchange between two clients does not carry any data origin authentication information (e.g. message authentication code, digital signature, etc.). After the initial user authentication, the server does not require any further proof nor any data origin authentication information in order to forward messages from one user to another. Consequently, spoofing a message is trivial and an attacker could therefore send a message pretending to be any legitimate user of the system.

The session ID created by the server does not have any security functionality and does not improve security in any way (since the server does not perform any authentication prior to issuing one, therefore a session ID can be issued to an impostor).

Because of the lack of server authentication on behalf of the client, man-in-the-middle attacks are also still possible.

References

1. http://www.microsoft.com/exchange/techinfo/administration/2000/IM_FAQ.asp

2. http://www.microsoft.com/exchange/downloads/2000/IMclient.asp

3. http://www.securityfocus.com/archive/1/195656

4. Berners-Lee, T., et al. *Hypertext Transfer Protocol -- HTTP/1.0*, RFC 1945, 1996

5. Berners-Lee, T., et al. *Hypertext Transfer Protocol -- HTTP/1.1*, RFC 2068, 1997

6. Franks, J. et al. *HTTP Authentication: Basic and Digest Access Authentication*, RFC 2617, 1999

- END -