

# On Instant Messaging Worms, Analysis and Countermeasures

Mohammad Mannan, Paul C. van Oorschot  
School of Computer Science, Carleton University  
Ottawa, Ontario, Canada  
{mmannan, paulv}@scs.carleton.ca

## ABSTRACT

We provide a collection of minor results on the area of Instant Messaging (IM) worms, which has received relatively little attention in the formal literature. We review selected IM worms and summarize their main characteristics, motivating a brief overview of the network formed by IM contact lists, and a discussion of theoretical consequences of worms in such networks. Existing methods to restrict an IM worm epidemic are analyzed in terms of usability and effectiveness, leading to the suggestion of two minor variations to limit IM worm propagation. We believe these variations are more user-friendly and effective than existing published methods. We also provide brief results of a three and a half year user study of IM text messaging and file transfer frequency in a moderate-size public IM network – the largest such study to date – which is of independent interest, but also supports in part the preceding claim regarding user-friendliness.

## Categories and Subject Descriptors

C.2.0 [Computer-Communication Networks]: Security and protection; D.4.6 [Operating Systems]: Security and Protection—*Invasive software*

## General Terms

Security

## Keywords

Instant Messaging worms, scale-free networks

## 1. INTRODUCTION AND BACKGROUND

Instant Messaging (IM) worms are worms that spread in IM networks, by exploiting features and vulnerabilities of IM clients and protocols.<sup>1</sup> We use a broad definition of worms following Kienzle and Elder [16]: “A worm is malicious code

<sup>1</sup>For background on Instant Messaging, see Appendix A.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WORM’05, November 11, 2005, Fairfax, Virginia, USA.  
Copyright 2005 ACM 1-59593-229-1/05/0011 ...\$5.00.

(standalone or file-infecting) that propagates over a network, with or without human assistance”.

It is well-known that IM worms are on the rise. The first IM worm made news in August 2001 (see e.g. [58]). Since then, IM networks have slowly become increasingly attractive breeding grounds for spreading worms. As of March 2005, IM worms are being reported so often (see e.g. [52]) that major public IM vendors (including Microsoft, AOL and Yahoo!) have formed an IM Threat Center [14] to track the latest IM and peer-to-peer (P2P) worms and vulnerabilities. A malware evolution analysis [10] for the first quarter of 2005 reports that email worms are on the decline due to improved anti-virus products; in contrast, IM worms are becoming more prevalent. Most IM worms propagate by generating automated file transfer requests or sending malicious URLs using valid IM user IDs (a compromised user’s IM user ID). Phishing scams using deceitful URLs are also increasing in IM networks (see e.g. [3, 28]). Currently available anti-virus products are inadequate to address most threats from IM worms (see e.g. [19]), mainly because anti-virus protections for IM currently guard against only known (or otherwise *detectable*) malware in file transfers, and cannot protect against malicious URL messages (i.e. URLs embedded in IM text messages). While monitoring email traffic is straightforward, the use of non-standard proprietary IM protocols makes IM traffic monitoring at the (corporate) gateway difficult. Hence, techniques developed to defend against email worms are less effective against IM worms [19].

Due to the prevalence of scanning- and email-worms, security researchers have employed large efforts to understand and restrict the propagation of those worms [43, 40, 36]. Although the war is far from being won, improved defensive techniques are making it increasingly difficult for worm writers. On the other hand, there appears to be very little published research detailing efforts to understand and contain the spread of IM worms. This largely motivates the present paper.

SCOPE. Most IM worms spread through popular public IM services such as MSN Messenger (Windows Messenger in Windows XP) [20], Yahoo! Messenger (YIM) [63], AOL Instant Messenger (AIM) [2], and ICQ [13]. Our discussion on IM worms here is mainly in the context of large public IM networks. Worms that spread via Internet Relay Chat (IRC) (see [32, 15]) are beyond the scope of this paper.

RELATED WORK. An overview of significant threats to IM, and a brief discussion on IM worms are available in our earlier work [19] (see also [17]). Symantec among others, regularly provides reports (e.g. [12, 11]) on IM protocols,

worms, threats and firewall issues. Available studies (e.g. [42, 62]) provide evidence that IM contacts form a *scale-free* network (see Section 4 for details). Smith [42] outlines defensive measures against IM worms. Williamson and Parry [62] introduced the *virus throttling* mechanism for IM to contain IM worms. For discussion of these proposed techniques see Section 5.1.

IM worms are quite different than regular scanning and email worms (e.g. see [62]). Scanning worms generally spread by generating *random* IP addresses and seeking to infect those machines. Many of these infection attempts fail as the targeted IP address may be invalid, or the host may not be vulnerable to the attack. In contrast, IM worms generally spread using online contact lists of the compromised hosts, and connect to target machines via the IM server. Thus the connection attempts are rarely unsuccessful or blocked by firewalls. Hence IM worms can bypass network-based scanning worm detection mechanisms which rely on the number of failed connection attempts (e.g. [59]). As another example, the DNS-based scanning worm detector proposed by Whyte et al. [60] would likely fail to detect an IM worm outbreak, because an IM worm's attack payload will most likely be forwarded through the IM server, using the target's user ID instead of a target IP address. Important differences between IM- and email-worms are discussed in Section 3.

OVERVIEW. In Section 2, we provide an overview of selected IM worms and vulnerabilities, and summarize major replication mechanisms for IM worms. Distinguishing features that make IM networks susceptible to fast-propagating worms are discussed in Section 3. We briefly discuss theoretical characteristics of IM networks to understand worms' behavior in such networks in Section 4. A critical review of existing mechanisms to address IM worms (e.g. virus-throttling for IM [62]) is provided in Section 5.1. We propose two new techniques to restrict the spread of IM worms in Section 5.2. In Section 5.3, we provide the results of a three and a half years user study of IM text messaging and file transfer frequency in a moderate-size public IM network – the largest such study to date – which is of independent interest, but also supports in part the usability claim of our proposed new techniques. Finally, in Section 6, we provide concluding remarks.

## 2. GENERALIZATIONS BASED ON ANALYSIS OF SELECTED IM WORMS AND VULNERABILITIES

In this section, we briefly consider a few illustrative IM worms, and note their main characteristics. Further details on these worms are available on many security web sites (e.g. [10, 44, 55]), or via search engines (e.g. Google). Then we list a few IM clients' vulnerabilities (exploitable IM features or implementation bugs) which could be exploited to create more destructive and faster spreading IM worms. We then summarize the most common propagation mechanisms of IM worms. A chronological history of IM worms and vulnerabilities is beyond the scope of the present paper; see IM threat tracking web sites (e.g. [14, 9]).

### Examples of IM Worms.

1. The **Choke** (June 2001) worm [48] hooks to MSN Messenger, and when a user initiates an IM conversation with an infected host, the worm sends a text message along with an invitation to download a file (which turns out to be the worm-file) from the infected host.
2. The **SoFunny** (July 2001) worm [51] spreads as a file attachment using AIM. It steals AIM login information, and emails the user ID and password to a designated address. It runs as a *service process* in Windows systems to hide from the Windows Task Manager.
3. The **JS\_Menger** (Feb. 2002) worm [54] appears as a message in MSN Messenger with the URL of a web site hosting the worm code. It exploits a vulnerability in the Microsoft Internet Explorer (IE) browser.
4. Different variants of the **Bropia/Kelvir** (Jan. – Feb. 2005) worms attempt to spread to all online contacts as a file using MSN Messenger. **Bropia-M** [47] disables several system processes in a compromised host, including Windows System Configuration Utility, Registry Editor and Task Manager. **Bropia-F** [57] disables debugging tools in an infected host, and installs a variant of the **Agobot/Spybot** worm [56], opening a backdoor on the host. The Agobot worm propagates by exploiting many known vulnerabilities e.g., SQL server buffer overflow, IIS (Internet Information Services) or WebDAV (Web-based Distributed Authoring and Versioning) vulnerabilities, and can be used to log keystrokes and relay spam. In addition to propagating a copy of the Spybot worm, **Kelvir.HI** ([49], see also [27]) checks a compromised host's Windows registry setting for language preference, and forwards a text message in that language (although it is limited to 10 different languages) and a malicious URL to all online contacts.
5. Several **Serflog** variants (Mar. 2005, e.g. **Serflog.A** [50]) spread via MSN Messenger (as a URL) or P2P file-sharing networks (e.g. **eMule** [34]). It terminates anti-virus and other well-known system and security processes, and modifies the system's HOSTS file to block access to well-known security web sites.

Most IM worms install backdoor programs to further exploit infected hosts. However, to date such worms generally have not carried any damaging payload other than the replication engine. Social engineering techniques are typically used to convince users to accept a worm file transfer or visit a malicious URL link.

### Examples of Client Vulnerabilities.

Numerous vulnerabilities have been uncovered in popular IM clients. These are drawing the attention of IM service providers and security communities who recognize that worm writers could easily exploit these vulnerabilities to write more damaging IM worms.

As an example, buffer overflows were discovered (and subsequently fixed) in the past in AIM and YIM clients (see e.g. [11, 45, 29]). In a recent (May 2005) buffer overflow attack on YIM, attackers could delete files from an infected host [31]. The ICQ sound scheme file location vulnerability [41] could be exploited to place malicious content on a user's machine. This could allow attackers to run a malicious script in the compromised machine exploiting a known IE vulnerability. The **Bizex** worm [46] exploited this vulnerability to spread in the ICQ network.

While most known client vulnerabilities require user actions to exploit a host, the recent (Feb. 2005) MSN Messenger PNG (Portable Network Graphics) vulnerability (see e.g. [24]) could have been exploited to spread a worm without any additional user-interaction. MSN Messenger uses the PNG format to enable the “Display Picture” feature (which shows a small picture of a contact in a chat window). The picture data is sent to a user whenever the user wants to send text messages to a contact even though the recipient’s image may not be displayed in the sender’s chat window (depending on the sender’s IM client settings). Therefore, disabling this feature in an MSN Messenger client could not prevent the exploitation. Microsoft fixed this vulnerability and forced users to update the IM client; users who did not upgrade were unable to log in to the MSN Messenger service.

Similar to the PNG vulnerability, the MSN Messenger GIF image processing vulnerability (April 2005) could lead to remote code execution [25]. This vulnerability can be exploited by sending a maliciously crafted *emoticon* (i.e. emotional icons in text messages) or display picture to an online MSN Messenger client. Microsoft subsequently updated the IM client software to fix this bug.

### IM Worm Replication Mechanisms.

From the above discussion, the following are evidently among the most popular mechanisms for propagation of IM worms.

1. Malicious file transfer (requires user-action).
2. Malicious URL in a text message (requires user-action).
3. Exploitation of implementation bugs in IM clients (may or may not require user-action).
4. Exploitation of vulnerabilities in operating systems or commonly used software applications (may or may not require user-action).

It follows that mechanisms for limiting the spread of IM worms (see Section 5) should take into account these vectors.

## 3. DISTINGUISHING FEATURES OF IM NETWORKS

We now list four major reasons that IM networks are particularly vulnerable to a worm attack. Although most replication strategies of IM worms are apparently common in email worms, here we discuss important differences in the execution of those strategies.

### A) Popular and Connected.

Public IM networks connect millions of users. According to Microsoft, MSN Messenger had 130 million customers as of July 2004 [23]. As more users continue to move to high-speed Internet connections, more remain online for extended periods of time [6]. Therefore, the set of online IM users forms a larger and larger network as time passes with one consequence being that the number of potential victims possibly outnumbers any known worm-outbreaks to date. In contrast, although email is a very popular Internet application, email clients do not generally form a network of “connected” users to the same extent as IM clients.

### B) Instant Hit-List.

IM contact lists enable users to track the *presence* status of their contacts. To a worm, an online contact list provides an instant *hit-list* [43] – a list of users vulnerable (with a very high probability) to the worm from an infected host. Most IM users are connected through a homogeneous platform (i.e. the same operating system and IM client), which increases the success ratio of the hit-list. For example, online users in the contact list of an MSN Messenger client are mostly connected in the MSN IM service using the default MSN Messenger client on Windows. Users can access the MSN IM service from Mac (using the MSN Messenger for Mac [21]) or Linux (using e.g. *aMSN* [33]) machines. In present reality, however, the number of users accessing the MSN IM service using clients other than the MSN Messenger in Windows is negligible.

To achieve a very rapid initial rate of infection (i.e. *getting off the ground*), a hit-list of vulnerable machines is required [43]. A worm writer could build a hit-list by scanning the whole Internet for vulnerable hosts before releasing a worm in the wild. By the time the worm hits the Internet, however, the hit-list may be stale, at least in part; some machines from the hit-list might be turned off, or some might be *patched* against the targeted vulnerability. In contrast, an online IM contact list offers a worm a reliable instant hit-list “for free” (i.e. without spending any time to build it). Note that most email clients provide an “address book” which does not reveal any online status of the users in that address book; an email address book is, at best, similar to an offline contact list of IM clients.

A simulation performed by Symantec [12] (with relatively conservative assumptions, e.g. considering only three unique contacts per user) shows that 500,000 machines can be infected with an IM worm in approximately 30 seconds. At least in theory, IM worms could out-perform email- and scanning-worms (in terms of total infection), which are typically limited by their ability to find vulnerable hosts across the entire Internet. IM worms would most likely be spread through server-mediated TCP connections (as IM connections are mostly TCP), and would thereby be limited by the network latency. Nevertheless, due to the inherent availability of a highly accurate hit-list, IM worms can spread among a large number of machines at a high rate.

### C) Reduced User-Interaction Time.

Despite numerous security measures, email remains one of today’s most effective worm propagation vectors [16]. This is despite the fact that an email worm, sent as an attachment, is not assured of reaching an intended destination address (e.g. the target email address may be invalid) or of immediate user-action (e.g. the target user may check email only occasionally). In contrast, as noted above, an IM worm gets a largely accurate instant hit-list from an infected user’s online contact list. A file transfer request or a URL sent to an online contact results, with high probability, in an instant user-action (despite some online IM users being away from their machine). Thus, reduced user-interaction time for propagation of worms is expected in IM networks.

### D) Increasing IM Integration in Popular Applications.

IM features are being integrated with many popular applications to meet the desire of IM users to share ideas and infor-

mation instantly. For example, the MSN Messenger contact list is accessible from the Microsoft Outlook Express email client; Microsoft’s Live Communication Server (LCS) [22] integrates MSN, AIM and YIM public IM services with Microsoft Office programs to enhance real-time collaboration in an enterprise. Greater integration increases usability, although the chance of IM-integrated applications being used by many users unknowingly or unwillingly increases as well. A security vulnerability in an IM protocol or client can exploit applications integrating IM features, and vice versa.

Although IM is just another Internet based application, the distinguishing features as listed above make IM networks particularly vulnerable to a fast-spreading worm.

#### 4. TOPOLOGY OF THE NETWORK FORMED BY IM CONTACTS

We now provide an overview of the network of IM contacts. Understanding the topology of such complex networks is necessary to obtain a better understanding of the rapidity of the spread of a worm in an IM network. This may, as well, help in devising novel ways to prevent or contain such an epidemic.

Paul Erdős and Alfréd Rényi introduced the ER (Erdős-Rényi) *Random Graph Model* in 1959 to analyze complex networks which are evident in communications and life sciences. ER networks (also known as *random networks*) are formed by nodes with randomly placed links, with most nodes having an approximately equal number of links. In the late 1990s, however, empirical network data (e.g. from the World Wide Web [1]) suggested that a common property of many complex real-world networks is that node linkages follow the power-law distribution<sup>2</sup> instead of the Poisson distribution of random networks. Barabási and Bonabeau [5] coined the term *scale free* (SF) to model the topology of such complex networks. The SF model originates (see [4]) from the following two mechanisms: networks grow continuously by the addition of new nodes, and new nodes attach *preferentially* to more highly connected nodes. This model accommodates the existence of *hubs* or *clusters* (i.e. a few nodes with a very high number of links, while most nodes have a few connections only) in real networks. SF architectures (e.g. WWW, Internet routers) are found to be highly resistant to accidental (i.e. random) failures, though very vulnerable to targeted attacks. Also, all viruses, even those that are weakly contagious, will spread and persist in a SF system. Consequently, once introduced, computer viruses are very difficult to completely remove from the Internet (see [35, 5]).

A general observation in human social structures is that some friends of a given person are typically friends themselves. Contacts in a user’s IM contact list also exhibit this feature; i.e. if user *A* has users *B*, *C*, *D* in her contact list, *B* will have *C* and/or *D* in his contact list with a high probability. Smith [42] analyzed the topology of the network formed by IM contacts, and found the network to follow the SF model. He studied a Jabber IM network of 50,158 nodes (young adult users) with nearly 500,000 links (contacts). The diameter of the network was only 4.35; i.e. any

<sup>2</sup>i.e. the probability that any node of the network is connected to *k* other nodes is proportional to  $k^{-\gamma}$ , where  $\gamma$  ranges between 2 and 3.

two users in the network studied were separated by only 4 to 5 users on average.

While IM networks may exhibit the SF model’s properties in reality, the following aspect of IM worms may complicate such a model. From any infected user, a worm’s spread may not be limited by the number of online contacts of that user; an IM worm may successfully guess IM contacts, at least for large public IM services. For example, in AIM a user ID consists of (case-insensitive) alphabetic characters and digits, and the maximum allowed length is 16; thus AIM theoretically supports approximately  $36^{16}$  different user IDs, so randomly guessing a user ID correctly would appear to have a very low probability. In reality, however, only a small (typically predictable) fraction of the user ID space is used. For example, any dictionary word or common name guessed as a user ID has a significantly higher than random chance of being an actual user ID. The large number of users in public IM networks makes such guessing of correct user IDs easier. Also, a large number of online user IDs can easily be obtained, for example, through the chat-room feature (e.g. YIM chat-rooms). In fact, each node can potentially become a hub, thereby dramatically increasing the number of hubs in the SF model. This may result in the failure of virus halting measures designed for SF networks (e.g. [7]) that aim to restore a *finite* epidemic threshold by patching most of the hubs in an infected network.

#### 5. MEASURES FOR LIMITING IM WORM EPIDEMICS

In this section, we review proposals from the existing literature to limit IM worm outbreaks. We also propose two new techniques to effectively restrict worms in IM networks. The findings from a new large-scale user study on file transfer and text messaging usage on IM networks are also summarized.

##### 5.1 Existing Techniques and Remarks Thereon

An overview of existing techniques, as well as comment on their shortcomings, is given below.

###### A) Temporary Server Shutdown.

All IM connections go through or are initiated by an IM server. A typical (but effective) solution that takes advantage of this centralized structure to stop an IM worm-outbreak is the following [12]: shutdown the IM server, manually analyze the worm and build a client patch; then force update when users attempt to login as the IM server comes alive again. For example, in April 2005, Reuters Messaging was shut down for several hours to stop an IM worm spread (e.g. [30]). Obviously, this is not a user-friendly solution. Moreover, as it has already been observed, IM worms may enable the propagation of other worms exploiting known (and generally ubiquitous) vulnerabilities in infected systems, and spread via traditional mechanisms such as email attachment. Therefore, shutting down the IM server may not effectively contain an IM worm epidemic.

As all IM connections are server-mediated, IM servers also appear to open up interesting detection and filtering approaches (assuming that the server has access to the plaintext of messages). For example, a server can track frequently transferred messages with identical content, such as identical URL messages. However, attackers could easily defeat such an approach by randomly padding identical messages.

Again, file-data are transferred P2P in many cases (see Appendix A), leaving no clue for the server to act on. On the other hand, server filtering may be considered to be more attack-resistant than client filtering, because a typical user may not have sound software protection mechanisms at the client-end. Therefore, it is apparent that an IM worm detection and filtering technique should consider both the server and clients.

## B) Temporarily Disabling the Most-Connected Users.

Exploiting the scale-free nature of IM contacts, Smith [42] proposed the following measure to contain a worm outbreak: disconnect the most-connected users (determined by the size of contact lists), which may effectively increase the network’s diameter, and thereby slow down the spread and allow time to build and apply a patch. It has been shown by Smith, however, that disabling the top 10% connected users in the IM network still would leave 90% of the remaining network connected although the network’s diameter has increased almost twofold. Note that Smith’s study derives users’ connectivity from the total number of contacts, not from the number of online contacts (which may vary significantly); of course, an IM server can easily also track the most-connected online users.

## C) Virus Throttling for IM.

To limit the propagation of fast worms, Williamson [61] introduced a general throttling mechanism based on the observation that worm-spreading network traffic is significantly different than normal traffic on many Internet protocols (e.g. TCP/IP, email). Generally, IM users interact with a slowly varying subset of contacts while IM worms send messages to all the online contacts in a user’s contact list. The idea behind virus throttling for IM, as proposed by Williamson and Parry [62], is to limit the rate at which users can interact with their contacts in a way that impedes a worm-spread (even for unknown, *zero-day* worms) without noticeable impacts on IM users.

**THE THROTTLING MECHANISM.** In essence, the throttling mechanism for IM is as follows. A short recent history list or *working set* of contacts for each user is maintained. This is the list of contacts that a user has *recently* interacted with. The mechanism attempts to ensure no more than one message is sent to a new contact (a user from the contact list who is not in the working set) per time period; see Figure 1. Every time a user attempts to send a message to a contact, the recipient’s user ID (‘h’ in Figure 1) is compared with the working set. If it is in the set, the message is sent immediately; otherwise, the message is placed on a delay queue for sending later. Alternatively, in a preferred implementation (as the authors advocated), the message will be sent instantly while recording a message entry in the delay queue as well (hence the delay queue does not actually delay messages here); the later message entries are removed at regular intervals, and the working set is updated accordingly. If a client’s delay queue length exceeds a predefined threshold, an attempted worm-spread is assumed, and all messages (to new contacts) from that client are blocked, and the user is asked to confirm the legitimacy of the queued messages. It is suggested that the mechanism be implemented at the IM server, because the IM server processes or initiates all user-messages, and users would not be able to bypass the server.

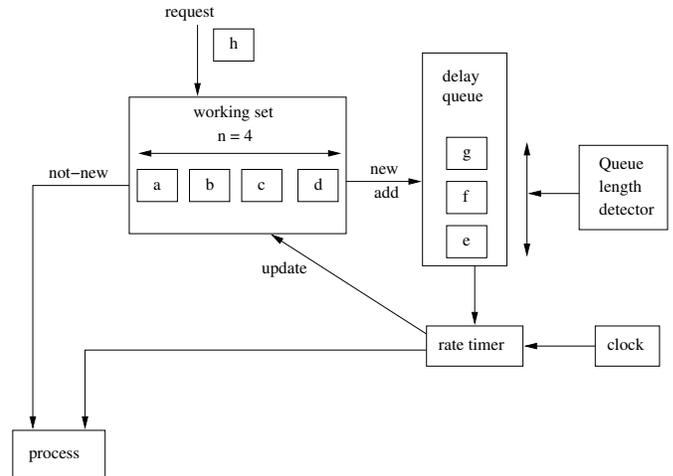


Figure 1: Throttling algorithm for IM [62]

**DATA ANALYSIS.** Williamson and Parry collected IM usage data from a Jabber server deployed at Hewlett-Packard. Contact lists of 710 corporate users and 39,740 messages from 223 senders in a 72 day period were analyzed. Setting a working set size to five resulted in less than 10% of messages being placed in the delay queue for most users. Although setting the allowed rate at around one new contact per minute appears more user-friendly, to restrict a worm-spread effectively, Williamson and Parry suggested the allowed rate to be one new contact per day. In either case, some messages will escape before the throttle mechanism is applied. Also from the collected data, the network formed by IM contacts was shown to be scale-free. The *hubs* of this network (i.e. users with large contact lists) appear to release the fewest messages before being blocked when the throttle is applied. This property apparently provides evidence that the throttling mechanism may effectively restrict IM worms.

**SHORTCOMINGS.** Throttling for IM may help stop IM worms to some extent, and appears to be one of the few technical mechanisms proposed to date. Thus it provides an important benchmark for comparison. However, we note the following limitations.

1. Allowing only one new contact/day may be *too restrictive* for IM users (home or enterprise). If users need to confirm the legitimacy of messages *often*, throttling may find little acceptance in IM communities.
2. When the delay queue threshold is crossed for a given user, new messages from that user will be blocked until the user confirms the legitimacy of those messages. This will cause delay (albeit temporarily) which is not what IM users expect in an IM environment; the whole point of using IM is being *instant*.
3. A user must confirm the legitimacy of messages when that user’s delay queue threshold is exceeded. A worm may be able to do that on behalf of the user, and thereby compromise the throttling mechanism.
4. The size of IM usage data analyzed by Williamson and Parry is quite small, and appears to be unrepresentative of active IM usage. For example, their study’s

average of less than 2.5 messages per user per day appears to largely fall short of the number of messages in real IM practice (our study in Section 5.3 shows that in a mid-size public IM network, on average a user sends 334 text-messages per day). Also, the network of IM contacts formed by only 710 users does not likely represent the characteristics of a public IM network with millions of users.

5. Throttling does not deal with group-chat or chat-room situations, where a user sends a message to multiple recipients concurrently.
6. Implementing the throttling mechanism, as suggested, requires per-online-user state memory at the server (e.g. working set, delay queue).
7. As the working set is stored in the server, a worm apparently cannot access the working set from a client machine. However, as an advanced attack, a worm might wait (e.g. a few minutes or hours) in a client machine to monitor (plaintext) user communications (e.g. messages sent to contacts), and build the working set for that user. Theoretically, the worm could spread to all contacts in the working set, as well as to other contacts before the delay queue threshold is crossed, without being throttled. The suggested parameters for the working set (e.g. five) and the delay queue (e.g. two), albeit small, may still be sufficiently large to allow a worm epidemic (recall the Symantec simulation in Section 3 – a worm could spread quite fast using only three unique contacts per user).

We have discussed several limitations of available techniques designed to limit IM worm-outbreaks. We briefly propose containment strategies in the next subsection to avoid most of these shortcomings.

## 5.2 New Proposals

In this section, we briefly suggest two mechanisms to further reduce the propagation of IM worms, and to do so in a more user friendly manner. While these mechanisms may appear to be straight-forward, we record them nonetheless, as an early contribution to the scant literature in this research area.

Our discussion in Section 2 suggests that file transfer and URLs in text messages are two major vectors for malware propagation in IM networks. It is apparent that, if we can restrict automatically generated file transfer requests and text messages with malicious URLs, the propagation of IM worms can be significantly delayed. In essence, our new proposals aim to control these two vectors. We assume that all IM text messages and file transfer requests (not necessarily the file data) are relayed through the IM server – which is generally true in current public IM connections (see Appendix A).

In Section 5.1, we listed limitations of the virus-throttling mechanism for IM. We believe that such a mechanism can do little while working on top of an insecure IM protocol. Similarly, a *secure IM protocol*<sup>3</sup> alone is unlikely to eliminate

<sup>3</sup>By “secure” we mean a protocol intending to provide confidentiality, authentication and integrity to client-server and client-client communications; the IM client itself may still have vulnerabilities.

the possibility of IM worms because IM worms may take advantage of implementation bugs of a “secure” IM client. A two-step method – a throttling mechanism as well as a secure IM protocol, e.g., Off-The-Record messaging (OTR) [6] – appears to us to be more effective against IM worms. (Note that, if a secure IM protocol needs to offer APIs to the application programmer, e.g., for better integration and custom application design, those APIs must be exposed in a secure manner.) Here we propose two techniques derived from the throttling mechanism (similar to it in spirit), and which we recommend be deployed with OTR or a similarly-featured protocol so that IM connections (to the server or a contact) are not easily compromised or spoofed.

1. Only use a throttling mechanism to limit file transfers and messages containing URLs. We believe this will be as effective as the proposal of Williamson and Parry [62] in most cases with less impact on users, due to the general observation that IM users send file transfer requests (e.g. see Section 5.3) or text messages with URLs only occasionally. In other words, we propose that the throttling mechanism be applied only to file transfer requests and URL messages, instead of on all new connection requests. Now normal text messages (the most frequently used feature of IM) will not be delayed, and there is no threshold at which users must confirm the legitimacy of text messages. Instead, URL messages and file transfer requests may be delayed, and in the case of file transfer the usability impact is further minimized due to the fact that a file transfer is generally not *instant* (even for a small size file). When sending a file transfer request or URL message to multiple users at a time (e.g. in a group-chat), the sender needs to confirm the authenticity of the request to the server or a selected recipient.
2. Challenge senders of a file transfer request or URL message with a (e.g. visual) CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) [53] generated by the IM server or the recipient’s client. This will not affect usability much, if as noted above, these features are used infrequently. CAPTCHA tests are computer generated, easy for humans to solve but difficult for machines, and the probability of guessing the answer correctly is low. Answering a CAPTCHA correctly increases the chance of the sender being a *human* user, rather than an IM worm trying to automatically propagate using the sender’s user ID. IM users are no stranger to these tests as while registering for an IM account, all major public IM vendors challenge users with CAPTCHA tests to stop automated account acquisition by software bots (e.g. by spammers).

To expand the latter technique, we note the following. Some users may send more files than others. Challenging such users for each file transfer with a CAPTCHA is less user-friendly than desired, and may be avoided by using *secure cookies* in a manner similar to Pinkas and Sander [37]. Assume that user *A* wants to send a file to user *B*, and both are logged in to the same IM server *S*. When *A* sends her file transfer request to *B* through *S*, *S* challenges *A* using a CAPTCHA. If *A* successfully responds to the challenge, then *S* forwards *A*’s request to send the file to *B*; *S* also sends

a secure cookie (i.e. data encrypted symmetrically under a key known only to the server) to  $A$ , and  $A$  stores that cookie to the local system. The cookie will expire after a certain interval, e.g., one hour. If  $A$  sends more files to  $B$ , or any other user before the expiry of the cookie,  $S$  will not challenge  $B$ . If  $A$  wants to send a file to multiple recipients at a time (e.g. while in a group chat),  $A$  will be challenged by  $S$  only once.

To reduce the load of generating a large number of CAPTCHAs by the IM server, the recipient’s IM client can be used instead to generate the challenge CAPTCHA sent to the originator, as user machines (e.g. personal computers) are generally under-utilized.

### 5.3 User Study on Per-User Frequency of IM Text Messaging and File Transfer

Here we summarize our findings for text messaging and file transfer statistics in an IM service, from a new IM user study which we believe should be of independent interest. Usability issues related to our proposals (of Section 5.2) are also examined. Although it is generally believed that the number of file transfer (FT) requests in an IM network is far less than the number of text-messages (TM), we found no real data supporting this belief in the literature. Therefore we carried out an independent study to quantify the file transfer usage in a reasonable-size IM network.

We used (with permission) data collected from the Eyeball Chat [8] service for a period of three and a half years (Sept. 2001 to April 2005). The data used contained only *collective* statistics that do not violate the privacy of Eyeball’s users. While the primary focus of this service is to enable *live* video conversation over the Internet, it nevertheless supports text messaging and file transfer. In total, 17.84 million FT requests and 3.2 billion TMs were tracked. The average number of FT, TM and online users over 15-minute periods is given in Table 1. The number of online users in the table is the average of the maximum and minimum online users in the 15-minute intervals. Table 2 shows that the ratio of FT to IM is quite low (i.e. one FT per 181 TM, on average). Also, per 24-hour period, an online user in this dataset sent 334 text-messages, and 1.84 file transfers on average. In a specific time interval, it would be interesting to present additional statistics, e.g., how many users sent one or more files, the maximum number of files sent by a user, etc. However, due to the lack of server statistics available to us from Eyeball on those items, we are unable to do so at present.

<i>Feature</i>	<i>Avg. Number</i>
File Transfer (FT)	143
Text Message (TM)	25953
Online Users	7459

**Table 1: Average file transfer, text messages, and online users over 15-minute intervals**

<i>Ratio</i>	<i>Value</i>
FT/TM	0.0055
FT/user/day	1.84
TM/user/day	334.03

**Table 2: Comparison of file transfer (FT) and text message (TM) usage**

From this it is apparent that IM users (at least in this dataset) infrequently transfer files, although it is a useful feature of IM. Hence throttling file transfers or challenging the initiator (sender) of a file transfer with a CAPTCHA would appear to be practical.

Our proposed mechanisms may be applied to limit automatic URL propagation as well. However, the usability impacts for CAPTCHA-protected URL messages are not known, because of the unavailability of such usage data.

## 6. CONCLUDING REMARKS

Although IM worms are still in their infancy (approximately 5 years old, as of June 2005), they are increasingly common and potent. We have discussed characteristics of a few IM worms and IM networks, and the implications with respect to IM worm epidemics. As one of our proposals involves the use of a CAPTCHA for improved defense against IM worms, the security of such tests must be taken into account, as another ongoing arms race is that between the designers, and attackers, of CAPTCHAs (e.g. [26]).

Our proposed techniques provide protection against what we consider to be the two most significant IM worm propagation vectors – malicious file transfers and URL messages. In contrast, throttling for IM [62] tries to minimize the number of IM worm connections which implies the following: a worm can establish a certain number (albeit small) of connections unchecked. Also, we believe our techniques are more user-friendly than the throttling mechanism [62]. We require a user to be challenged only when sending a URL message or a file transfer request, which are evidently less frequent events. In contrast, the throttling mechanism requires a user to verify each connection attempt after a certain threshold is crossed. However, throttling connections is more effective than our techniques when connection establishment implicitly transfers user-configurable file data (e.g. MSN “Display Picture” file) automatically. This is because we aim to check worm-generated automated file transfer requests rather than connection attempts. However, in general, we recommend that such implicit file transfer functionality should be disallowed as an IM feature to avoid easy exploitations.

We emphasize that when considering security enhancements to IM, usability issues should be a paramount concern, because most IM users are casual, and expect *instant* communication when using IM. Our proposed techniques as presented are preliminary, and we expect may require enhancements and critical review. A primary motivation of our work is to further academic research into defenses against IM worms, with the intention of stimulating security improvements in IM systems in the longer term.

## Acknowledgements

We thank anonymous reviewers, as well as Matthew Williamson, for their constructive comments which helped us improve the quality of this paper, and all members of Carleton’s Digital Security Group for their enthusiastic discussions on this topic, especially Dave Whyte. We also thank Eyeball Networks for access to their server statistics. The second author is Canada Research Chair in Network and Software Security, and is supported in part by an NSERC Discovery Grant, the Canada Research Chairs Program, and MITACS.

## 7. REFERENCES

- [1] R. Albert, H. Jeong, and A.-L. Barabási. Diameter of the World Wide Web. *Nature*, 401:130–131, 1999.
- [2] America Online, Inc. AOL Instant Messenger. <http://www.aim.com/> [Accessed: Mar. 22, 2005].
- [3] Anti-Phishing Working Group. Phishing activity trends report, Feb. 2005. [http://www.antiphishing.org/APWG\\_Phishing\\_Activity\\_Report\\_Feb05.pdf](http://www.antiphishing.org/APWG_Phishing_Activity_Report_Feb05.pdf) [Accessed: Mar. 31, 2005].
- [4] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.
- [5] A.-L. Barabási and E. Bonabeau. Scale-free networks. *Scientific American*, 288:60–69, 2003.
- [6] N. Borisov, I. Goldberg, and E. Brewer. Off-the-record communication, or, why not to use PGP. In *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society, WPES '04*, pages 77–84. ACM Press, 2004.
- [7] Z. Dezső and A.-L. Barabási. Halting viruses in scale-free networks. *Physical Review E*, 65(5), May 2002.
- [8] Eyeball Networks Inc. Eyeball Chat – Free video Instant Messenger. <http://www.eyeballchat.com> [Accessed: June 15, 2005].
- [9] FaceTime. IMPact Center – IM & P2P Threats. <http://www.facetime.com/impactcenter/default.aspx> [Accessed: Aug. 17, 2005].
- [10] A. Gostev. Malware evolution: January - March 2005. Kaspersky Lab. <http://www.viruslist.com/en/analysis> [Accessed: Apr. 27, 2005].
- [11] N. Hindocha. Threats to Instant Messaging, 2003. Symantec Security Response. <http://securityresponse.symantec.com/avcenter/reference/threats.to.instant.messaging.pdf> [Accessed: Dec. 7, 2003].
- [12] N. Hindocha and E. Chien. Malicious threats and vulnerabilities in Instant Messaging. In *Virus Bulletin Conference, vb2003*, Sept. 2003.
- [13] ICQ Inc. ICQ Pro 2003b. <http://www.icq.com/> [Accessed: Mar. 22, 2005].
- [14] IMlogic. IMlogic Threat Center. [http://www.imlogic.com/im\\_threat\\_center/index.asp](http://www.imlogic.com/im_threat_center/index.asp) [Accessed: Mar. 17, 2005].
- [15] C. Kalt. Internet Relay Chat: Architecture, Apr. 2000. RFC 2810, Status: Informational. <http://www.faqs.org/rfcs/rfc2810.html> [Accessed: June 22, 2004].
- [16] D. M. Kienzle and M. C. Elder. Recent worms: A survey and trends. In *Proceedings of the 2003 ACM Workshop on Rapid Malcode*, pages 1–10, Washington, DC, USA, Oct. 2003. ACM Press.
- [17] N. Leavitt. Instant Messaging: A new target for hackers. *Computer*, 38(7):20–23, July 2005.
- [18] M. Mannan. Secure Public Instant Messaging, Aug. 2005. Masters Thesis (Computer Science). <http://www.scs.carleton.ca/~mmannan/publications/msthesis.pdf> [Accessed: Aug. 21, 2005].
- [19] M. Mannan and P. C. van Oorschot. Secure public Instant Messaging: A survey. In *Proceedings of the 2nd Annual Conference on Privacy, Security and Trust (PST'04)*, pages 69–77, Fredericton, NB, Canada, Oct. 2004. <http://www.scs.carleton.ca/~mmannan/publications/pst04.pdf> [Accessed: June 23, 2005].
- [20] Microsoft. MSN Messenger. <http://messenger.msn.com/> [Accessed: Mar. 22, 2005].
- [21] Microsoft. MSN Messenger for Mac. <http://www.microsoft.com/mac/default.aspx?pid=msnmessenger> [Accessed: May 3, 2005].
- [22] Microsoft. Office live communications server. <http://www.microsoft.com/office/livecomm/prodinfo/default.mspix> [Accessed: Mar. 18, 2005].
- [23] Microsoft PressPass. Flirting's moved online! New research from MSN reveals millions swap IM addresses with potential dates, July 2004. <http://www.microsoft.com/presspass/press/2004/jul04/07-08FlirtingPR.asp> [Accessed: Mar. 17, 2005].
- [24] Microsoft TechNet. Microsoft security bulletin MS05-009: Vulnerability in PNG processing could allow remote code execution (890261). <http://www.microsoft.com/technet/security/bulletin/MS05-009.mspix> [Accessed: Mar. 18, 2005].
- [25] Microsoft TechNet. Microsoft security bulletin MS05-022: Vulnerability in MSN Messenger could lead to remote code execution (896597). <http://www.microsoft.com/technet/security/Bulletin/MS05-022.mspix> [Accessed: June 20, 2005].
- [26] G. Mori and J. Malik. Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 134–141, 2003. UC Berkeley Computer Vision Group. <http://www.cs.berkeley.edu/~mori/gimpy/gimpy.html> [Accessed: June 14, 2005].
- [27] News.com Staff. IM worm speaks your language, Aug. 2005. News article, <http://news.com.com/> [Accessed: Aug. 24, 2005].
- [28] News.com Staff. Phishers target Yahoo Messenger, Mar. 2005. News article, <http://news.com.com/> [Accessed: Mar. 27, 2005].
- [29] News.com Staff. Trillian IM flaw exposed, Mar. 2005. News article, <http://news.com.com/> [Accessed: Mar. 25, 2005].
- [30] News.com Staff. Worm attack forces Reuters IM offline, Apr. 2005. News article, <http://news.com.com/> [Accessed: Apr. 15, 2005].
- [31] News.com Staff. Yahoo fills in Messenger hole, May 2005. News article, <http://news.com.com/> [Accessed: June 17, 2005].
- [32] J. Oikarinen and D. Reed. Internet Relay Chat protocol, May 1993. RFC 1459, Status: Experimental. <http://www.faqs.org/rfcs/rfc1459.html> [Accessed: June 10, 2004].
- [33] Open Source. aMSN : Alvaro's Messenger. <http://amsn.sourceforge.net/index.php> [Accessed: May 3, 2005].
- [34] Open Source. eMule Project. <http://www.emule-project.net/> [Accessed: Aug. 21, 2005].
- [35] R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Physical Review Letters*, 86(14):3200–3203, 2001.

- [36] V. Paxson, editor. *Proceedings of the 2004 ACM Workshop on Rapid Malcode (WORM'04)*, Washington, DC, USA, Oct. 2004. ACM Press.
- [37] B. Pinkas and T. Sander. Securing passwords against dictionary attacks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 161–170, Washington, DC, USA, 2002. ACM Press.
- [38] P. Saint-Andre. Extensible messaging and presence protocol (XMPP): Core, Oct. 2004. RFC 3920, Status: Standards Track. <http://www.ietf.org/rfc/rfc3920.txt> [Accessed: Mar. 2, 2005].
- [39] P. Saint-Andre. Extensible messaging and presence protocol (XMPP): Instant Messaging and presence, Oct. 2004. RFC 3921, Status: Standards Track. <http://www.ietf.org/rfc/rfc3921.txt> [Accessed: Mar. 3, 2005].
- [40] S. Savage, editor. *Proceedings of the 2003 ACM Workshop on Rapid Malcode (WORM'03)*, Washington, DC, USA, Oct. 2003. ACM Press.
- [41] Secunia. ICQ predictable file location weakness, Feb. 2004. Secunia Advisory SA10970, <http://secunia.com/advisories/10970/> [Accessed: June 17, 2005].
- [42] R. D. Smith. Instant Messaging as a scale-free network, June 2005. cond-mat/0206378, <http://arxiv.org/abs/cond-mat/0206378> [Accessed: June 20, 2005].
- [43] S. Staniford, V. Paxson, and N. Weaver. How to Own the Internet in your spare time. In *Proceedings of the 11th USENIX Security Symposium*, pages 149–167. USENIX Association, 2002.
- [44] Symantec. Search and latest virus threats page. <http://www.symantec.com/avcenter/vinfodb.html> [Accessed: Mar. 18, 2005].
- [45] Symantec. Securing Instant Messaging. <http://securityresponse.symantec.com/avcenter/reference/secure.instant.messaging.pdf> [Accessed: June 27, 2004].
- [46] Symantec. W32.Bizex.Worm. <http://securityresponse.symantec.com/avcenter/venc/data/w32.bizex.worm.html> [Accessed: June 21, 2005].
- [47] Symantec. W32.Bropia.M. <http://securityresponse.symantec.com/avcenter/venc/data/w32.bropia.m.html> [Accessed: June 21, 2005].
- [48] Symantec. W32.Choke.Worm. <http://securityresponse.symantec.com/avcenter/venc/data/w32.choke.worm.html> [Accessed: June 21, 2005].
- [49] Symantec. W32.Kelvir.HI. <http://securityresponse.symantec.com/avcenter/venc/data/w32.kelvir.hi.html> [Accessed: Aug. 25, 2005].
- [50] Symantec. W32.Serflog.A. <http://securityresponse.symantec.com/avcenter/venc/data/w32.serflog.a.html> [Accessed: June 21, 2005].
- [51] Symantec. W95.SoFunny.Worm@m. <http://securityresponse.symantec.com/avcenter/venc/data/w95.sofunny.worm@m.html> [Accessed: June 21, 2005].
- [52] TechWeb.com Staff. IM threats growing 50% per month, Mar. 2005. TechWeb.com news article, <http://informationweek.com> [Accessed: Mar. 25, 2005].
- [53] The CAPTCHA Project. Telling humans and computer apart (automatically). <http://www.captcha.net/> [Accessed: Mar. 19, 2005].
- [54] Trend Micro. JS\_MENGER.GEN. [http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=JS\\_MENGER.GEN](http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=JS_MENGER.GEN) [Accessed: June 21, 2005].
- [55] Trend Micro. Virus encyclopedia. <http://www.trendmicro.com/vinfo/virusencyclo/> [Accessed: Mar. 18, 2005].
- [56] Trend Micro. WORM\_LAGOBOT.AJC. [http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM\\_LAGOBOT.AJC](http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM_LAGOBOT.AJC) [Accessed: June 21, 2005].
- [57] Trend Micro. WORM\_BROPIA.F. [http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM\\_BROPIA.F](http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM_BROPIA.F) [Accessed: June 21, 2005].
- [58] USA Today Staff. IM viruses opening a new can of worms, Aug. 2001. News article, <http://www.usatoday.com> [Accessed: Mar. 17, 2005].
- [59] N. Weaver, S. Staniford, and V. Paxson. Very fast containment of scanning worms. In *Proceedings of the 13th USENIX Security Symposium*, pages 29–44, Aug. 2004.
- [60] D. Whyte, E. Kranakis, and P. C. van Oorschot. DNS-based detection of scanning worms in an enterprise network. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, Feb. 2005.
- [61] M. Williamson. Throttling viruses: Restricting propagation to defeat malicious mobile code. In *Proceedings of the 18th Annual Computer Security Applications Conference (ACSAC)*, pages 61–68, Dec. 2002.
- [62] M. Williamson and A. Parry. Virus throttling for Instant Messaging. In *Virus Bulletin Conference, vb2004*, Sept. 2004.
- [63] Yahoo! Inc. Yahoo! Messenger. <http://messenger.yahoo.com/> [Accessed: Mar. 22, 2005].

## APPENDIX

### A. BACKGROUND ON INSTANT MESSAGING (IM)

For convenience, we provide here introductory background on IM, largely extracted from [18].

IM is an Internet based application which facilitates communications, typically between two end-user clients, with the aid of a server. Most IM communications are client-server based. Messages among users are also typically relayed through the server (mainly to avoid firewall issues). However, server-mediated P2P communications also occur in some situations (e.g. audio/video chat, file transfer); i.e., connection setup messages are relayed through the server, but payload is transferred directly, e.g., a file transfer request and response messages are relayed through the server but the file data is sent P2P.

While an IM server appears to be a single entity to a client, it may be a group of servers controlled by a single IM service provider (e.g. AOL), or a collection of servers from

independent IM service providers (e.g. Jabber [38, 39]). If user *A* wants to communicate with user *B*, both must log into the same IM service. Messages from *A* to *B* will be delivered by the server depending on *B*'s privacy settings. For direct communications between *A* and *B*, the server provides the necessary information (e.g. network address) to each party.

Table 3 summarizes common terminology used in IM.

<i>online user</i>	A user successfully logged in to an IM server.
<i>presence</i>	Presence information reveals whether or not a user is logged in to an IM server.
<i>availability/user mode</i>	Availability information reveals a user's willingness (e.g. "busy", "do not disturb") to send/receive messages, or status (e.g. "away", "on the phone").
<i>contact/buddy list</i>	The list of user IDs whose presence and availability a user has currently subscribed to.
<i>block list</i>	The list of user IDs explicitly barred from getting the current user's presence and availability information; listed users also cannot send any messages to the current user.
<i>allow list</i>	The list of user IDs allowed to send messages to the current user and which can track the user's presence and availability information.
<i>one-to-one chat</i>	A user sends or receives messages from another user, generally through the IM server.
<i>group chat</i>	More than two users exchange messages at a time. Users form a virtual "group", generally which is short-lived. Users in a group chat are usually closely related.
<i>chat room</i>	A virtual room, generally consisting of many users who exchange instant messages on some closely related topics.
<i>IM user</i>	A human user of an IM service.
<i>IM client</i>	A (software) program that enables a human user to use an IM service.
<i>IM server</i>	A (software) program that enables IM clients to access IM features in an IM service.
<i>IRC</i>	A client-server based multi-user messaging system of large networks. Users meet on "channels" (rooms/virtual places, usually with a certain topic of conversation) for group or personal messaging.
<i>URL message</i>	An IM text message containing one or more URLs.

**Table 3: IM-related definitions**