HoneyIM: Fast Detection and Suppression of Instant Messaging Malware in Enterprise-like Networks

Mengjun Xie Zhenyu Wu Haining Wang
The College of William and Mary
{*mjxie, adamwu, hnw*}*@cs.wm.edu*

Abstract

Instant messaging (IM) has been one of most frequently used malware attack vectors due to its popularity. Distinct from other malware, it is straightforward for IM malware to find and hit the next victim by exploiting the current victim's contact list and playing social engineering tricks. Thus, the spread of IM malware is much harder to detect and suppress through conventional approaches. The previous solutions are ineffective to defend against IM malware in an enterprise-like network environment, mainly because of high false positive rate and the requirement of the IM server being inside the protected network. In this paper, we propose a novel IM malware detection and suppression mechanism, HoneyIM, which guarantees almost zero false positive on detecting and blocking IM malware in an enterprise-like network. The detection of HoneyIM is based on the concept of honeypot. HoneyIM uses decoy accounts to trap IM malware by leveraging malware spreading characteristics. Fed with accurate detection results, the suppression of HoneyIM can conduct a network-wide blocking. In addition, HoneyIM delivers attack information to network administrators in real-time so that system quarantine and recovery can be quickly performed. The core design of HoneyIM is generic, and can be applied to the scenarios that either enterprise IM services or public IM services are used in the protected network. Based on open-source IM client `Pidgin` *and client honeypot* `Capture`*, we build a prototype of HoneyIM and validate its efficacy through both simulations and real experiments. Our results show that HoneyIM provides effective protection against IM malware in enterprise-like networks.*

1. Introduction

Instant Messaging (IM) has been stepping into the workplace as well as people's daily life at remarkable speed. It is estimated that enterprise IM users will grow to 78 million by the end of 2008 [9]. However, large user-base and communication immediacy also attract malware to land on IM, which is particularly ideal for malware propagation. By virtue of IM features and social engineering tricks, IM malware can spread quickly and stealthily, which poses a serious security threat not only to home IM users but also to organizations which allow the use of IM in workplace. The IM malware studied in the paper refers to any malicious code that spreads through Internet-based IM networks such as Windows Messenger series (MSN) and AOL Instant Messenger (AIM), which have dedicated servers for account management and message relay. Bropia [7] and Opanki [8] are typical examples of such malware. Although most of known IM malware spreads on popular public IM networks, enterprise IM systems such as [6] and [14] can also be penetrated as these corporate IM services usually provide connectivity and interoperability with public IM services. In 2005, the outbreak of a variant of Kelvir worm even forced Reuters to shut down its IM service [4].

File transfer and URL-embedded message are two major spreading vectors of IM malware. After compromising an IM client, the malware propagates itself by either making a malicious file transfer or sending a text message containing a malicious URL to the online users[1] in the victim's contact list. The contact list is also called buddy list. Once those invigilant contacts click the file or URL, malicious code will be triggered to execute or be downloaded from the URL and executed, and subsequently the malware propagation continues at an exponentially increasing speed.

Although the threat of IM malware, especially the outbreak of zero-day IM malware, is on the rise, network administrators still lack effective solutions to protect enterprise-like networks such as campus networks and corporate networks. Conventional protections using firewalls and anti-virus products are insufficient to defend against IM malware due to the unique propagation feature of IM malware. Most of popular IM protocols are able to circumvent firewalls if their default ports are blocked. Signature-based anti-virus products cannot detect zero-day IM mal-

[1] Offline contacts may also be attacked but this type of attack is rare.

1063-9527/07 $25.00 © 2007 IEEE
DOI 10.1109/ACSAC.2007.24

ware. Meanwhile, anomaly detection techniques, such as Norman Sandbox technology [18], may also be ineffective in catching evasive malware which behaves differently in the sandbox environment. Compared to malicious file transfers, malicious-URL-embedded IM messages are even harder to be identified by firewalls and anti-virus programs.

IM providers may take quick responses, e.g., releasing patches and mandating client upgrade, to newly discovered vulnerabilities in their products. They may even proactively block potentially malicious file transfers. However, these filtering mechanisms still could be bypassed [22, 23]. Moreover, it is extremely hard for IM providers to protect against malicious URLs that exploit the vulnerabilities of Web browsers or other related applications [20]. While some protection schemes, such as CAPTCHA and virus throttling for IM [13, 33], can enhance IM security, the incurred overhead and usability degradation could be significant, and thus prohibit IM providers from using them in near future.

Motivated by the shortage of effective defense against IM malware, we propose HoneyIM, a framework for automating the process of IM malware detection and suppression in an enterprise-like network. Based on the concept of honeypot, HoneyIM detects IM malware by leveraging its inherent spreading characteristics. Specifically, HoneyIM uses decoy accounts in normal users' contact lists as sensors to capture malicious content sent by IM malware, which achieves almost zero false positive. With accurate detection, HoneyIM suppresses malware by performing network-wide blocking. In addition, HoneyIM delivers attack information to network administrators for system quarantine and recovery. The core design of HoneyIM is generic and can be applied to a network that uses either private (enterprise) or public IM services. We implement a prototype of HoneyIM for public IM services, based on open-source IM client `Pidgin` [1] and client honeypot `Capture` [29]. We validate the efficacy of HoneyIM through both simulations and real experiments. The simulations show that even only a small portion, e.g., 5%, of IM users in the network have decoys in their contact lists, HoneyIM can detect the IM malware as early as after 0.4% (on average) of IM users are infected. The experimental results demonstrate that the prototype system succeeds in detection, suppression, and notification of IM malware within seconds.

The remainder of the paper is structured as follows. We first describe the major spreading mechanisms of IM malware and related work in Section 2. Then we detail the framework of HoneyIM in Section 3, followed by the implementation and evaluation of HoneyIM in Sections 4 and 5, respectively. We discuss possible evasion to HoneyIM and the countermeasures in Section 6. Finally, we conclude the paper in Section 7.

2. Background and Related Work

2.1. IM Malware

IM malware propagates mainly through two ways: malicious file transfer and malicious URL in text message. Usually the malware infection is triggered by the victim's action such as clicking the transferred file or the received URL. IM malware could also spread without victim's involvement, e.g., by exploiting the vulnerabilities in IM clients. However, this type of spreading vector is rare.

In the file transfer mechanism that has been used since early 2000s, IM malware propagates by initiating malicious file transfers to remote contacts. Malicious files are usually renamed to attract victims or to evade network filters. Once a victim clicks the file, the malware is invoked and will attempt to infect more victims in the contact list. To counter this type of malware spreading, some IMs such as MSN forbid IM clients to transfer certain types of files such as *.pif* files. While the actual file transfer is normally carried out directly between two IM clients, the messages for transfer establishment still go through IM server. Therefore, IM servers can easily detect the messages for establishing malicious file transfers and silently drop them to block malware propagation.

Nowadays malicious URL messages become much more popular than malicious file transfer for IM malware propagation. Instead of sending a file, IM malware sends a text message containing a malicious URL to remote contacts. Once a victim clicks the link, either a malware binary is downloaded and executed or some malicious web scripts run to exploit the vulnerabilities of the Web browser or other related applications. Compared to malicious file transfers, malicious URL messages have several advantages in propagation. First, malicious URL messages have more means to compromise a system. File downloading is just one of its attacking vectors. Second, malicious URLs can be used to collect victims' information by exploiting Web functionality. For instance, the URL sent by Kelvir.k [21] points to a php script and contains the contact's email address. The email address is harvested as soon as the URL is clicked. Last but not least, IM malware can play more social engineering tricks on URLs. For example, a malicious URL can be crafted to mimic the link on a reputable Web site [3]. The IM clients supporting HTML scripts also provide a playground for IM malware to fake URLs at their will. Those forged URLs appear normal but in fact point to malicious webpages.

After infection, IM malware may take different actions for propagation. Many types of malware start spreading immediately after they compromise IM clients, while others wait until they receive instructions to spread. The latter usually install certain bot programs on compromised machines,

through which the malware is controlled by the remote bot herder.

2.2. Related Work

The security threats posed by IM malware have been studied in [5, 12]. In [5], the spreading speed of IM malware is estimated, showing that 500,000 machines could be infected within a minute.

Previous defense schemes against IM malware are closely related to IM network modeling and traffic measurement. Based on individual measurement and analysis, [15, 24, 33] all verify that IM social networks formed by IM contacts are scale-free, that is, the IM network connectivities follow power-law distributions. However, a recent measurement study [34] suggests that Weibull distributions may be more appropriate for describing the connectivity of IM social networks. For scale-free networks, a small portion of nodes that are highly connected have significant effect on mitigating malware spread. Based on this observation, Smith [24] proposed to delay the propagation of IM malware by disabling the accounts of most connected IM users on the network. This scheme needs to be deployed on IM servers. It only reduces the spread speed and may have significant side-effects. Williamson *et al.* [33] applied their virus throttling mechanism to IM and demonstrated its effectiveness by simulation. The throttling to IM is also conducted at servers. The throttling becomes blind blocking if its threshold is very restrictive, which degrades the usability. Mannan and van Oorschot [13] proposed two defense methods, namely limited throttling and CAPTCHA-based challenge-response. They also provided a usage study on per-user frequency of IM text messages and file transfers to support the applicability of their second scheme. Liu *et al.* modeled the spread of IM malware using multicast tree [11] and analogous branching process with varied lifetime [10]. HoneyIM is orthogonal to all the schemes mentioned above, and can achieve accurate detection and blocking without degrading usability.

Trivedi *et al.* studied the network and content characteristics of spim, the spam messages on IM networks, by using a proxy server as honeypot [31]. Their work is different from HoneyIM, since [31] is a measurement study and it targets spim but not IM malware. The honeypot used in [31] refers to a SOCKS proxy, which is exploited by spimmers to conceal their identities.

3. HoneyIM Framework

HoneyIM aims to assist network administrators in IM malware defense by automating the process of malware detection and suppression in an enterprise-like network. Utilizing the innate spreading characteristics of IM malware and applying the concept of honeypot, HoneyIM can detect and block unknown IM malware at its early stage of spreading, which greatly facilitates network filtration and system quarantine and recovery. In this section, we first give an overview of HoneyIM, how and why it can detect IM malware early. Then, we discuss several issues that need to be considered when using HoneyIM in practice. After that, we present the design of HoneyIM and the functionalities of its components. Finally, we describe the deployment of HoneyIM in an enterprise-like network.

3.1. Overview

HoneyIM is based on the concept of honeypot. As an effective intrusion detection technology, honeypot has been used widely. According to [30], *a honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource*. Not only can a honeypot be a physical machine or a specialized program, which is the common case, but it can also be an e-mail address, or even an IM decoy user. Since IM malware always attempts to infect other users on the victim's contact list, HoneyIM exploits decoy users to detect IM malware. Under normal circumstances, a client user will not initiate a conversation with a decoy user. Therefore, if the decoy user receives a file transfer request or a URL-embedded text message originated from a client user, it is highly probable that malware is spreading and the request/message sender is compromised. Thanks to decoy users, HoneyIM can achieve almost zero false positive in detection. This strong guarantee, which is rarely offered by other schemes, relieves network administrators from worrying about possible interruption to normal IM users caused by the protection technique. In addition, HoneyIM can block malicious content that has been detected and inform network administrators of the attack information, e.g., the IP address of the compromised machine, in real-time.

Figure 1 illustrates the working mechanism of HoneyIM. The IM user with an icon of honeypot is the one whose contact list contains a decoy user. The events happen in the following sequence. (1) Some IM malware compromises an IM client and (2) propagates. However, (3) when it tries to spread again, it hits a decoy user and (4) is detected by HoneyIM. (5) HoneyIM blocks the malicious content in IM traffic (either at the edge gateway or at the IM server if the IM service is provided within the network) and non-IM traffic[2] instantly, and notifies the attack information to the network administrator.

HoneyIM is designed to be independent, with no restriction on the type and location of IM servers. Therefore, the framework of HoneyIM can be flexibly realized under the

[2]Doing this is to block accesses to malicious contents, e.g., malicious URLs.

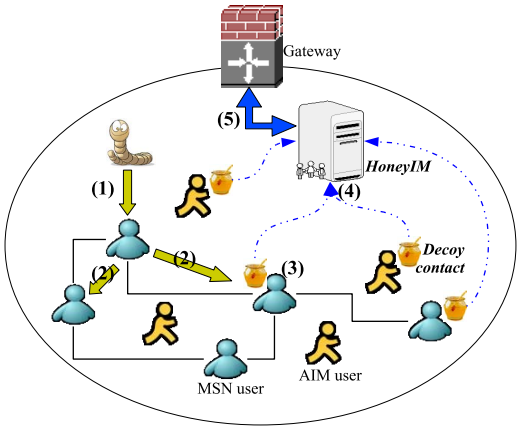

Figure 1. Working mechanism of HoneyIM

context of either public IM services or private (enterprise) IM services being used in the protected network. The core of HoneyIM is the same for either server-enhanced (with private servers) or serverless (with public servers) realization. The difference lies in the implementation and deployment, which will be discussed in Section 3.4. The framework of HoneyIM consists of several modules and these modules can be deployed in a single machine or at different places.

3.2. Design Issues

The success of HoneyIM largely depends on the use of decoy users. In the following, we discuss three issues of HoneyIM that are much related to decoy user, including initialization, sensitivity, and compatibility.

The initialization of HoneyIM mainly refers to the creation and addition of decoy user accounts. Strictly speaking, it is a deployment issue. If public IM services are used in the protected network, the network administrators need to create decoy accounts and solicit some volunteer IM users to add those decoy users into their contact lists. In contrast, if an enterprise IM service is employed, the creation and addition of decoy users can be done automatically by the IM server. However, the system must notify volunteer users the purpose and usage of decoy accounts, and provide a disable (or opt-out) option. This HoneyIM initialization is fulfilled at one time, and the update of decoy accounts could be performed if necessary. In addition to the volunteer policy for IM user cooperation, the network administrators might require the IM users who have high connectivity degrees (i.e., the super-nodes in IM networks) to include decoy accounts in their contact lists.

The sensitivity of HoneyIM is measured by the ratio between the number of infected users and that of all IM users in the protected network when the spreading of IM malware is first detected. The key factor affecting the sensitivity of HoneyIM is the coverage of HoneyIM—the portion of the IM users equipped with decoy user accounts among all IM users within the network. It is obvious that HoneyIM cannot detect malware for those users who do not include decoy accounts in their contact lists. Moreover, IM malware may intentionally or inadvertently bypass HoneyIM by not hitting decoy users in the infected users' contact lists. The word "intentionally" does not mean that the IM malware knows the decoys in advance, but reflects its capability of distinguishing decoys from other contacts. Here we assume that the threat comes from the outside of the protected network and the inside IM users do not collude with the outside attackers. Given the coverage of HoneyIM, which is usually determined by the network administration policy, we will consider how to counter evasive IM malware to improve HoneyIM sensitivity in Section 6.

Compatibility is not an issue if HoneyIM is deployed on an enterprise IM server, since the server can maintain the compatibility with supported IM clients. However, the compatibility has to be taken into account if public IM services are used in the protected network. Under this circumstance, various types of public IM systems may coexist. This is especially true on the networks with less strict IM usage policies such as campus networks. Thus, HoneyIM should be able to talk with different types of IM clients.

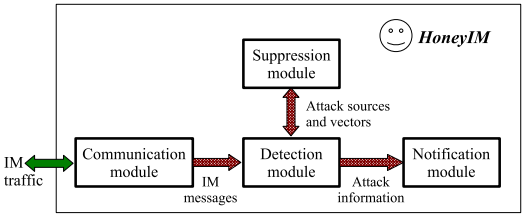

Figure 2. Framework of HoneyIM

3.3. System Components

Figure 2 shows the general framework of HoneyIM, which comprises four modules each performing a specific functionality. Note that these modules could be deployed either on the same machine or on different hosts (or network devices). As displayed, the communication module is responsible for handling IM traffic. It parses the IM traffic to decoy users and delivers it to the detection module. The detection module extracts attack vectors and related information from IM messages, and then feeds them into the suppression and notification modules. The suppression module sifts through network traffic and filters out malicious traffic containing attack vectors. Meanwhile, the notification module informs network administrators of the detected malware spreading.

3.3.1 Communication Module

The communication module is the base of HoneyIM. Decoy accounts use it to join IM networks and communicate with normal IM clients. This module realizes all necessary functions of a normal IM client, such as signing on/off, setting presence status, receiving messages and files, etc. These functions are automatically executed by default and can also be manually operated by a network administrator. The module only accepts the messages from the users on the contact list for blocking "spim", the spam on IM networks. The communication module should support all IM protocols that are used by the protected IM services, and allow multiple accounts to log into different IM networks simultaneously if necessary.

3.3.2 Detection Module

The detection module serves three purposes: (1) detecting compromised IM clients, (2) identifying attack vectors, and (3) validating attack vectors. It accomplishes the first two tasks by consulting the communication and suppression modules and scrutinizing IM messages delivered by the communication module, and attains the last task by conducting deep-inspection.

The detection module classifies a sending IM client as compromised, when a decoy account receives a file transfer request or a text message with URL from the IM client. The reason is that it is very rare for a normal user to issue such a request or message to the decoy account[3]. The detection is not affected by client-to-client or client-to-server traffic encryption because the IM messages received by a decoy (as a client) must be in plain-text. If IM malware spreads through file transfer, the attack source, i.e., the IP address of the compromised machine, is immediately known as a file transfer is usually done between two IM clients directly. However, if IM malware spreads through URL message, we cannot identify the sender directly because the message is usually relayed through server. Under this circumstance, the attack source is inferred with the help of the suppression module, which will be described shortly. The detection module can easily generate the attack vector information such as malicious file names and malicious URLs from the received IM messages.

Furthermore, the detection module performs deep-inspection to verify the virulence of the received file or URL. There are many techniques available to achieve this purpose. For example, we can use dynamic taint analysis based techniques such as TaintCheck [17] and Argos [19] to examine if a received binary can compromise system and to generate the corresponding signature if a compromise occurs. We also can adopt the technique used by HoneyMonkey [32] to check received URLs. HoneyMonkey detects Web exploits by browsing URLs inside a virtual machine and monitoring the change of system states. In general, any effective and efficient host-based anomaly detection techniques can be used for deep-inspection. HoneyIM does not contain any specific technique for analyzing IM malware, but rather provides a platform to apply existing techniques for malware dissection and leave the choice of what technique to use to network administrators. The adopted techniques are implemented as plug-ins of the detection module, and the deep-inspection is conducted in a contained environment such as a virtual machine to prevent HoneyIM itself from being compromised.

The incorporation of deep-inspection is justified by the following considerations. First, deep-inspection can further reduce false positives. It is possible that innocent URLs or files could be sent with malicious content by IM malware to disguise their malice. Second, deep-inspection helps discover additional or real attack vectors used by IM malware. For example, file deep-inspection can generate the signature of malware binary, based on which the filtering is much more robust against evasion than based on file name. IM malware can also use different URLs in its spreading, which in fact are doorway webpages redirecting traffic to the same website that hosts real exploits. With URL deep-inspection, the protection can be further enhanced because not only doorway URLs but also real exploit URLs can be discovered. Last but not least, deep-inspection uncovers the IM malware activities, such as the infection mechanism and the infected files, for network administrators.

After attack vector extraction and validation, the detection module supplies the validated attack vectors and sources to the suppression module for immediate network traffic filtration. In the meantime, the detection module feeds all collected attack information into the the notification module, which informs network administrators of the occurrence of an attack in real-time for prompt system quarantine and recovery.

3.3.3 Suppression Module

The suppression module in essence is a network filter. It takes the attack source and vector information from the detection module as input. Then, it blocks any traffic from attack sources and filters out network traffic that contains attack vectors. Different from other modules that have no requirement for deployment location, the suppression module should be installed at a network vantage point, where it can monitor all traffic passing through the protected network. The location of the suppression module will be further discussed in Section 3.4.

The suppression module consists of two components: non-IM traffic filter and IM traffic filter. These two compo-

[3]Even if a normal user accidentally sends a message to the decoy account, the message is usually a pure text message.

nents are logically independent for flexible implementation and deployment. The non-IM traffic filter fulfills two tasks: blocking attack sources and filtering non-IM network traffic. For the former, the filter simply drops any packet from the attack sources to terminate malware propagation. For the latter, the filter examines contents of inbound and outbound packets to identify if an internal user is attempting to access a malicious webpage or transfer a virulent file. Any packet containing a matched attack vector will be discarded.

The IM traffic filter also provides two functionalities. The first is traffic filtration, which weeds out the IM messages that either come from (or go to) the compromised clients or contain identified malicious file names or URLs. Although a file is usually transferred between two clients, the IM messages for establishing transfer connections are relayed through servers in plain-text for mainstream IM products. Therefore, blocking malicious file transfer by dropping connection establishment messages is not affected by client-to-client encryption. The second functionality of the IM traffic filter is to help identify malicious URL sending hosts within the protected network. Because messages are relayed through server, the detection module cannot identify the sources of malicious URL messages. To track the IP address of the compromised host, the IM traffic filter records the URLs and the corresponding IP addresses of their senders. With this information, the detection module can easily pinpoint the malicious URL senders.

3.3.4 Notification Module

The notification module plays the role of messenger. Its job is to inform network administrators of the occurrence of IM malware spread upon the detection of an attack. Given the fast spread of IM malware, the notification to network administrators should be made in real-time or near real-time by means of SMS (Short Messaging Service) or IM. The notification module can also notify the victim about the fact that his machine has been infected with IM malware via IM or email.

3.4. Deployment

As mentioned in the overview section, HoneyIM can be deployed with a private IM server inside the protected network (server-enhanced deployment) or with public IM services outside the network (serverless deployment). The major differences between the two deployments lie in the function location and system initialization of HoneyIM. In serverless deployment, the non-IM and IM traffic filters of the suppression module have to be placed on the network edge device. However, in server-enhanced deployment, while the non-IM traffic filter still needs to be on the network edge device, the best place for the IM traffic filter is the private IM server, where the filter can see all IM traffic. Moreover, in practice many IM servers already include the message filtering functionality, making IM traffic filtering much easier there.

The deployment of HoneyIM also involves system initialization, i.e., the creation and addition of decoy accounts. In serverless deployment, network administrators need to register accounts for decoy users on public IM services before running HoneyIM. Due to the maximum size of contact list (e.g., 600 for MSN) and the protection consideration, the administrators can create multiple decoy accounts and use them for different groups of IM users. Then, the decoy accounts are added into the volunteer IM users' contact lists with their cooperation. By contrast, the server-enhanced deployment saves the efforts of network administrators and IM users by automating the creation and addition of decoy accounts, just like the use of AIM Bots for shopping and movie guide. This can be achieved by adding a decoy account management module to the private IM server. The module can also be used to (1) provide IM users with the information of decoy accounts and the option to enable/disable them, and (2) update decoy accounts periodically against potential evasion.

4. Prototype

To demonstrate the efficacy of HoneyIM, we have built a prototype of the serverless HoneyIM, which can be easily transformed to the server-enhanced HoneyIM prototype with minor changes in function location and system initialization. We implement the HoneyIM modules using different techniques. We use a full-fledged open-source IM client `Pidgin` (formerly known as `Gaim`) [1] to build the communication module. The detection module employs `Capture` [29], a high interaction client honeypot on Windows systems, for URL deep-inspection. The detection module extracts URLs from the communication module and feeds them into `Capture`, which decides whether a URL is malicious by comparing the system states such as registry and running processes before and after the URL is accessed. For any file transfer request HoneyIM does not perform deep-inspection but immediately fires an alert instead, given that the file transfer method is relatively unpopular in IM malware spreading and most IM users and programs are vigilant to this type of threat. HoneyIM receives the delivered file and sends it to network administrators via email. In the construction of the suppression module, we use Perl IPQueue module for iptables [16] to perform URL logging and pattern-matching. We implement the notification module with two communication means: email and SMS. The suppression module communicates with the detection module via network socket, and thus can be deployed on a separate machine.

Because `Pidgin` supports multi-protocol and multi-account, HoneyIM can log into multiple accounts on multiple IM networks simultaneously. Therefore, it can provide protection for multiple public IM networks. Note that the choice of `Pidgin` and `Capture` is mainly due to the availability of their source code. Upon the accessibility of source code, any IM clients or anomaly detection systems can be used to construct HoneyIM.

5. Evaluation

In this section, we first evaluate the detection sensitivity of HoneyIM under different coverages via simulation. Then, we validate the applicability of HoneyIM through real experiments.

5.1. Simulation

When adding decoy accounts is voluntary for IM users on the protected network, it is very possible that HoneyIM does not cover all IM users. Under this circumstance, how effective would HoneyIM be? Because we cannot carry out a large-scale experiment in practice, we turn to simulation for answering this question. We adopt the simulation model from [35] due to the similarity in propagation between IM malware and Email worms [35]. The major metric we use is the percentage of IM users being infected by the time the IM malware is firstly detected by HoneyIM (the percentage of infected IM users for short), and we investigate its variation under different HoneyIM coverages.

5.1.1 Simulation Model

The simulation model of IM malware propagation is described as follows. First, when an IM user receives an IM message, she may or may not read the message immediately. The reading delay for user i, denoted by T_i, is a stochastic variable. When the user receives a message with a malicious URL [4], she clicks the URL with a clicking probability denoted as C_i. We assume that C_i is a constant for user i. If the malicious URL is clicked, the malicious code is downloaded and executed immediately. It infects the current IM client and sends malicious URLs to all the victim's contacts with no delay. The malware will not spread again unless the user receives the same URL and clicks it again.

Before we start the simulation, we need to determine the IM network topology and the values of each C_i and T_i. Here the IM network refers to the virtual network composed by the contact lists of the IM users on the protected network. According to [24] that studies an IM network containing 50,158 users, over 80% of the user contacts are bidirectional, indicating that most of users are also in the contact lists of their buddies. Thus, we model the IM network topology by an undirected graph $G =< V, E >$. For $\forall v \in V$, v denotes a node (IM user), and for $\forall e = (u, v) \in E, u, v \in V$, e represents an edge that connects two users, u and v, who are in each other's contact list. $|V|$ is the total number of nodes, and $D(i)$ is the degree of node i, i.e., the number of edges connected to node i. The size distribution of contact lists has been identified as scale-free by [15, 24, 33], except that [34] claims that Weibull distribution has a better fit. However, [34] does not give the parameters of Weibull distribution and the number of their monitored IM users is small compared to [15, 24, 33]. Therefore, we model the IM network topology as power law and set the power law exponent α to 1.7, based on the measurement results from [15] and [24]. The network is generated by using GLP power law generator [2] with the given α, the number of nodes $|V|$, and the average node degree $E[D]$. We generate three IM networks with the number of nodes $|V| = 1000/6000/6000$ and the average node degree $E[D] = 8/8/16$, respectively. The maximum node degrees of the generated networks are all below 600, the maximum size of a contact list for MSN.

Similar to [35], we assume that IM users have independent behaviors. Due to the large number of users $|V|$ and independent behaviors, the mean values of user reading delay T_i and clicking probability C_i, denoted by $E[T_i]$ and $E[C_i]$ ($i = 1, 2, \cdots, |V|$), can be assumed to follow Gaussian distribution. That is, $E[T_i] \sim N(\mu_T, \sigma_T^2)$ and $E[C_i] \sim N(\mu_C, \sigma_C^2)$. We also assume that T_i follows exponential distribution and C_i is a constant for user i, and the generation of T_i and C_i is constrained by $T_i \geq 0$ and $C_i \in [0, 1]$. In simulation, we use $N(20, 10^2)$ and $N(0.5, 0.3^2)$ to generate $E[T_i]$ and $E[C_i]$, respectively.

5.1.2 Simulation Results

Given the network topology, we *randomly* deploy decoys in the network with different coverage $\mathfrak{R}$ and run simulation experiments. Each simulation run stops once IM malware hits a decoy user (blocking is in effect immediately) or timeout occurs. The number of infected users and detection time are the simulation output. For each coverage $\mathfrak{R}$, we vary the decoy deployment 10 times and run simulation 100 times for each deployment, and have the mean and median values derived from these $1,000$ simulation experiments.

With the increase of HoneyIM coverage, the corresponding percentages of infected IM users on three different IM networks are shown in Figure 3, in which the solid curves are for mean values and the dashed curves are for median values. The mean curves are above the median curves for very small coverage values, and both types of curves drop sharply and converge to zero with the increase of coverage. This clearly demonstrates the effectiveness of HoneyIM. Figure 4 further zooms in on y-axis and compares the

[4] The situation for malicious file transfer is similar.

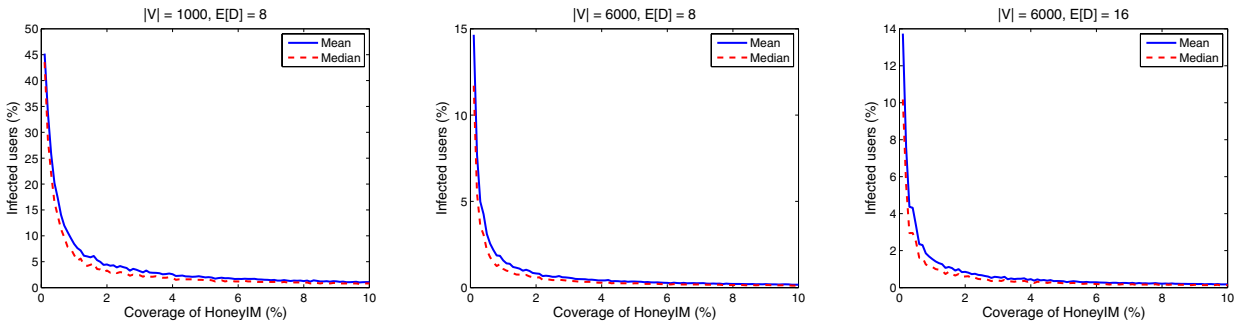

Figure 3. Relations between HoneyIM coverages and infected user percentages

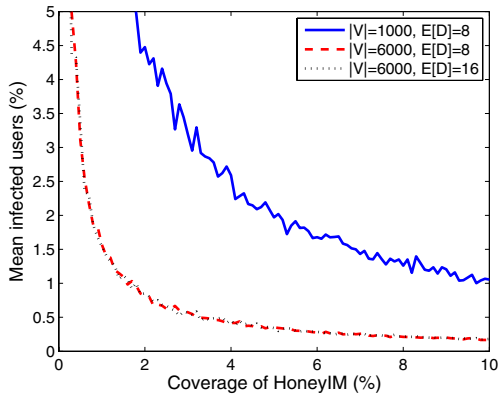

Figure 4. Comparisons among mean curves

mean curves of the three IM networks. Even with the 5% coverage, HoneyIM can detect the spread of IM malware only after 2% (or 0.4%) of all IM users are infected for the network with $|V| = 1,000$ (or $|V| = 6,000$). Compared to the number of nodes $|V|$, the average node degree $E[D]$ has much less effect on the performance of HoneyIM. Two mean curves, the dashed one for $|V| = 6,000, E[D] = 8$ and the dotted one for $|V| = 6,000, E[D] = 16$, are almost identical.

We also compare the performance of HoneyIM with that of IM throttling [33]. The throttling of IM malware is usually conducted on an IM server. We use the "no-delay" mode of IM throttling and configure the working set size and threshold to 5 and 2, respectively, as suggested. Since it is difficult to simulate the working set for each user at run time, we simplify the propagation model by (1) randomly determining a node's working set between 0 and 5 right before the node is propagating and (2) blocking the node after its propagation (no matter whether the delay queue is overflowed or not). Therefore, the maximum number of the nodes that a compromised node could infect is its working set size plus 2 (the threshold). Note that this model is conservative compared to the original scheme, as we block an infected node permanently once it starts spreading.

Figure 5 shows the performance comparisons between HoneyIM (coverage $\Re = 3\%$) and throttling on the three IM networks. The solid curves represent HoneyIM and the dotted curves represent throttling. The dashed curves show the spreading of IM malware with no mitigation. Note that the y-axis is logarithmic, and all the results for throttling and no mitigation are the mean values for 100 runs. Compared with throttling, HoneyIM can achieve similar performance in terms of the number of infected users on a small network ($|V| = 1,000$), and perform much better when the network becomes bigger ($|V| = 6,000$) and has more edges ($E[D] = 16$). More importantly, HoneyIM can accurately detect the malware and block its spread right after detection, while throttling cannot differentiate malicious traffic from normal traffic, let alone block them in an effective manner.

5.2. Real Experiment

We set up a small testbed comprising three machines. We use one machine as the IM client and the other two as HoneyIM and the network gateway. The suppression module of HoneyIM is deployed on the network gateway. Both the IM client and HoneyIM run inside virtual machines for security and ease of experimentation. We first use real IM malware binaries we have collected to test HoneyIM by running malware on the IM client machine. We test Jitux-A [26], Kelvir-F [27], Kelvir-M [25], and Kelvir-Q [28], respectively, all of which spread through malicious URL messages on MSN platforms. The URLs for Jitux-A and Kelvir-F lead to .exe and .scr file downloading, while the URLs for Kelvir-M and Kelvir-Q point to .php scripts which also harvest victim's email addresses. Unfortunately, due to the legal reaction taken by the IM providers and security community, the webpages pointed by these known malicious URLs are either invalid or have been removed by the hosting websites[5]. The URL message sent by Kelvir-F is not even received by HoneyIM, because of the filtering in MSN servers. No detailed information about IM malware is

[5]This situation also applies to other known IM malware.

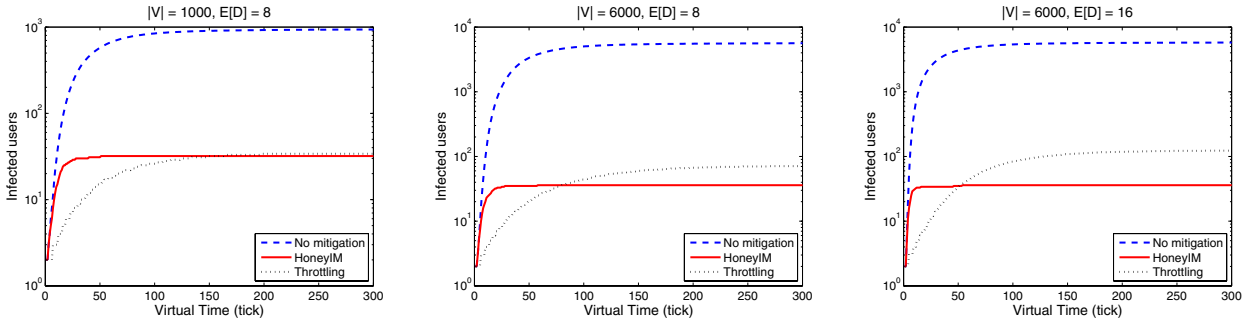

Figure 5. Effect comparisons between HoneyIM and IM throttling

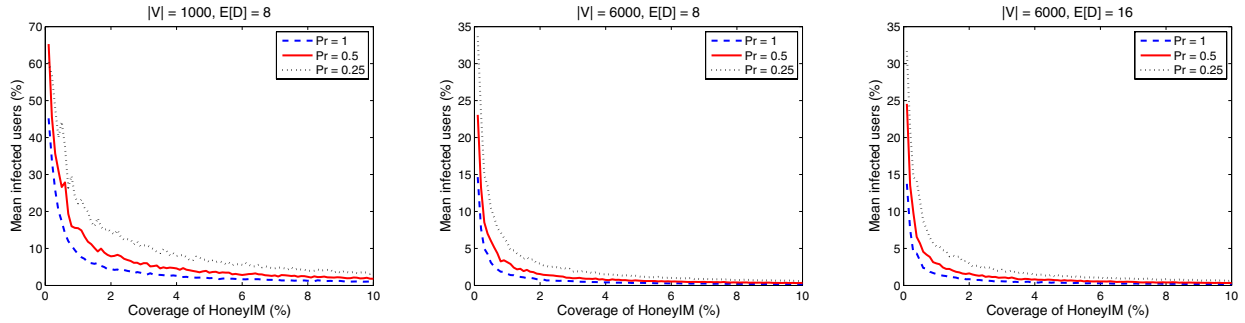

Figure 6. Effects of randomly selecting infection targets on HoneyIM

given by deep-inspection. Thus, we reconfigure the detection module to skip the deep-inspection step and rerun the tests. The suppression and notification modules work well as expected.

We also test the prototype using a generic approach which overcomes the difficulty caused by the invalidity of the known malicious URLs. We mimic IM malware by sending malicious URLs collected by ourselves to decoy accounts. The malicious URLs we used, in principle, have no difference from those carried by known IM malware in terms of Web exploits. Thus, they should have the same effect on normal IM clients and HoneyIM. The URL process time of HoneyIM is mainly determined by deep-inspection, which is usually finished within 30 seconds. Overall, HoneyIM successfully detects all malicious URLs, updates the URL blacklist, and sends the attack information to the designated recipient via SMS and email. For emulated malicious file transfers, HoneyIM automatically receives files, reveals file names to the suppression module, and sends file payloads to the designated recipient via email. The whole process takes seconds to complete, since no deep-inspection is performed for file transfer.

6. Discussion

In previous sections, we assume that IM malware always attempts to infect all online contacts by either initiating a file transfer or sending a malicious URL during its spread. This hit-all propagation strategy, however, might not always be used. For example, "smart" IM malware may send malicious URLs or files only to the active online contacts, i.e., those contacts that the infected IM client is talking to; or the propagation is activated only after the infected client receives a message. Taking the non-hit-all strategy, IM malware might not hit the decoy contact even if the contact list of the infected IM user includes the decoy accounts.

IM malware can realize the non-hit-all propagation strategy by either intentionally or randomly selecting a part of all online contacts as targets. To prevent decoys from being easily distinguished, we can enhance HoneyIM with interaction functionality. As a countermeasure, HoneyIM uses the interaction functionality to mimic human users for decoys by initiating chat sessions with normal users, making it much harder for IM malware to tell decoys from others. The chat content can be important security notices or other user interested information. We readily agree that IM malware can still avoid decoy contacts even with the interaction functionality, for example, by infecting the most active contacts. However, the spread of this type of IM malware could be significantly reduced. According to a recent IM traffic measurement [34], IM users only contact a small portion of users in their contact lists. On average an AIM user chats with only 1.9 users and an MSN user chats with 5.5 users.

The random selection of infection targets may also help IM malware bypass decoy contacts. To study the effect of the random selection on HoneyIM, we conduct the following experiments based on the previous simulation for HoneyIM. We apply a probabilistic propagation strategy to the

experiments. That is, when IM malware propagates, it will send malicious content to each contact with a probability p. With the probabilistic infection, the number of users that malware will contact becomes $p \times n$ on average, where n is the total number of the online contacts of the infected user.

We test and compare the effects of random target selection on HoneyIM with three different probabilities $p = 1, 0.5, 0.25$ on the three IM networks, respectively. Here $p = 1$ refers to the aforementioned deterministic infection. The comparison is displayed in Figure 6, in which the curve of $p = 0.5$ is above the curve of $p = 1$ but below the curve of $p = 0.25$. It indicates that with the decrease of the probability value, the average number of infected users becomes larger. However, the difference among three curves quickly becomes negligible with the increase of the coverage. In general, the random target selection has little effect on HoneyIM.

7. Conclusion

In this paper we have proposed HoneyIM, a novel detection and suppression mechanism to defend against IM malware for enterprise-like networks. Distinct from all previous defense schemes, HoneyIM introduces decoy users for IM malware detection. It exploits the basic spreading characteristics of IM malware and guarantees almost zero false positive. With accurate detection, the suppression of HoneyIM achieves instant network-wide blocking. Moreover, HoneyIM notifies network administrators of the infected machines and the infection features of IM malware in real-time. The generic design of HoneyIM enables its flexible realization on a network that uses either enterprise IM services or public IM services. We have built a prototype of HoneyIM that works with public IM services using open-source IM client `Pidgin` and client honeypot `Capture`. The simulation studies demonstrate that even with a small portion of IM users equipped with decoy accounts, HoneyIM can still detect and block IM malware in the early stage of its spread. The real experiments on the prototype further demonstrate that HoneyIM is competently capable of detecting and suppressing the spread of IM malware.

Acknowledgments

We are very grateful to the anonymous reviewers for their insightful comments. This work was supported by NSF grants CNS-0627339 and CNS-0627340.

References

[1] Pidgin. `http://pidgin.im/`, 2007.

[2] T. Bu and D. Towsley. On Distinguishing Between Internet Power Law Topology Generators. In *Proceedings of the 2002 IEEE INFOCOM*, pages 638–647, New York, NY, June 2002.

[3] A. Gostev. Social engineering: the latest chapter. `http://www.viruslist.com/en/weblog?weblogid=168136245`, August 2005.

[4] M. Hicks. Reuters suspends im service due to kelvir worm. `http://www.eweek.com/article2/0,1759,1786151,00.asp`, Apri 2005.

[5] N. Hindocha and E. Chien. Malicious Threats and Vulnerabilities in Instant Messaging. `http://www.symantec.com/avcenter/reference/malicious.threats.instant.me%ssaging.pdf`, 2003.

[6] IBM. Lotus Sametime. `http://www-142.ibm.com/software/sw-lotus/sametime`.

[7] Kaspersky. IM-Worm.Win32.Bropia.aj. `http://www.viruslist.com/en/viruses/encyclopedia?virusid=72841`.

[8] Kaspersky. IM-Worm.Win32.Opanki.d. `http://www.viruslist.com/en/viruses/encyclopedia?virusid=84950`.

[9] N. Leavitt. Instant messaging: A new target for hackers. *Computer*, 38(7):20–23, July 2005.

[10] Z. Liu and D. Lee. Coping with instant messaging worms - statistical modeling and analysis. In *Proceedings of the 15th IEEE Workshop on Local and Metropolitan Area Networks*, Princeton, NJ, June 2007.

[11] Z. Liu, G. Shu, N. Li, , and D. Lee. Defending against instant messaging worms. In *Proceedings of IEEE GLOBECOM 2006*, pages 1–6, San Francisco, CA, Nov. 2006.

[12] 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*, pages 69–77, Fredericton, NB, Canada, 2004.

[13] M. Mannan and P. C. van Oorschot. On Instant Messaging Worms, Analysis and Countermeasures. In *Proceedings of WORM 2005*, pages 2–11, Fairfax, VA, Nov. 2005.

[14] Microsoft. Office Live Communications Server. `http://www.microsoft.com/office/livecomm/prodinfo/default.mspx`.

[15] C. D. Morse and H. Wang. The Structure of An Instant Messenger Network and Its Vulnerability to Malicious Codes. In *Proceedings of ACM SIGCOMM 2005 Poster Session*, Philadelphia, PA, Aug. 2005.

[16] Netfilter. iptables project. `http://www.netfilter.org/projects/iptables/`.

[17] J. Newsome and D. Song. Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. In *Proceedings of the 12th NDSS*, San Diego, CA, Feb. 2005.

[18] Norman. Norman sandbox whitepaper. `http://download.norman.no/whitepapers/whitepaper_Norman_SandBox.pdf`.

[19] G. Portokalidis, A. Slowinska, and H. Bos. Argos: an emulator for fingerprinting zeroday attacks. In *Proceedings of the EUROSYS 2006*, Leuven, Belgium, April 2006.

[20] C. Raiu. The IM worms armada. `http://www.viruslist.com/en/weblog?weblogid=203678309`, October 2006.

[21] R. Schouwenberg. Kelvir changes its approach. `http://www.viruslist.com/en/weblog?weblogid=162243612`, April 2005.

[22] R. Schouwenberg. Do you like photos? `http://www.viruslist.com/en/weblog?weblogid=199354341`, Sept. 2006.

[23] R. Schouwenberg. MSN filter bypassing - part 2. `http://www.viruslist.com/en/weblog?weblogid=199850358`, Sept. 2006.

[24] R. D. Smith. Instant Messaging as a Scale-Free Network. `http://arxiv.org/abs/cond-mat/0206378v2`, 2002.

[25] Sophos. Troj/Kelvir-M. `http://www.sophos.com/virusinfo/analyses/trojkelvirm.html`.

[26] Sophos. W32/Jitux-A. `http://www.sophos.com/virusinfo/analyses/w32jituxa.html`.

[27] Sophos. W32/Kelvir-F. `http://www.sophos.com/virusinfo/analyses/w32kelvirf.html`.

[28] Sophos. W32/Kelvir-Q. `http://www.sophos.com/virusinfo/analyses/w32kelvirq.html`.

[29] R. Steenson and C. Seifert. Capture: A high interaction client honeypot. `http://www.nz-honeynet.org/capture.html`.

[30] The Honeynet Project. *Know Your Enemy: Learning about Security Threats (2nd Edition)*. Addison-Wesley Professional, May 2004.

[31] A. J. Trivedi, P. Q. Judge, and S. Krasser. Analyzing Network and Content Characteristics of Spim Using Honeypots. In *Proceedings of the 3rd USENIX SRUTI*, Santa Clara, CA, June 2007.

[32] Y.-M. Wang, D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen, and S. King. Automated web patrol with strider honeymonkeys: Finding web sites that exploit browser vulnerabilities. In *Proceedings of the 13th NDSS*, San Diego, CA, Feb. 2006.

[33] M. M. Williamson, A. Parry, and A. Byde. Virus throttling for instant messaging. Technical report, HP Lab Bristol, May 2004.

[34] Z. Xiao, L. Guo, and J. Tracey. Understanding Instant Messaging Traffic Characteristics. In *Proceedings of the 27th ICDCS*, Toronto, Canada, June 2007.

[35] C. C. Zou, D. Towsley, and W. Gong. Modeling and Simulation Study of the Propagation and Defense of Internet Email Worm. *IEEE Transactions on Dependable and Secure Computing*, 4(2):105–118, April-June 2007.