A Systematic Characterization of IM Threats Using Honeypots

Spiros Antonatos, Iasonas Polakis, Thanasis Petsas and Evangelos P. Markatos
Institute of Computer Science,
Foundation for Research and Technology Hellas
{antonat, polakis, petsas, markatos}@ics.forth.gr

Abstract

The popularity of instant messaging (IM) services has recently attracted the interest of attackers that try to send malicious URLs or files to the contact lists of compromised instant messaging accounts or clients. This work focuses on a systematic characterization of IM threats based on the information collected by HoneyBuddy, a honeypot-like infrastructure for detecting malicious activities in IM networks. HoneyBuddy finds and adds contacts to its honeypot messengers by querying popular search engines for IM contacts or by advertising its accounts on contact finder sites. Our deployment has shown that with over six thousand contacts we can gather between 50 and 110 malicious URLs per day as well as executables. Our experiments show that 21% of our collected executable samples were not gathered by other malware collection infrastructures, while 93% of the identified IM phishing domains were not recorded by popular blacklist mechanisms. Furthermore, our findings show that the malicious domains are hosted by a limited number of hosts that remain practically unchanged throughout time.

1 Introduction

Instant messaging is one of the most popular Internet activities. According to an older survey [7], more than 82 million people in Europe and 69 million people in North America use an instant messenger. A more recent study by Leskovec et al. [29] reveals that the number of MSN messenger (the most popular IM client) users has reached 240 million, with 7 billion exchanged messages per day. Reports estimate over 400 million registered Skype users [18], and 2.1 billion instant messages sent per day by AIM users[2].

This large user-base and the fact that IM is a near real-time form of communication, in contrast to other forms such as e-mail, make IM networks an attractive platform for attackers to launch their campaigns. Attackers either exploit vulnerabilities of the IM client software, or steal account information through phishing schemes. Once a user account has been compromised, the attack propagates by targeting the victim's contacts. The attack vectors are either file transfers or instant messages that contain URLs of websites controlled by the attacker. As users tend to trust content sent from their contacts, the probability of users accepting the transfer or clicking the URL is higher than in the case of traditional phishing campaigns or malicious websites.

This work focuses on the characterization and detection of attacks against IM users. Our proposed architecture, called HoneyBuddy, is based on the concept of honeypots. Honeypots are closely monitored decoy machines that are not used by a human operator but rather wait to be attacked [35]. In a similar fashion, we have deployed IM honeypots: decoy IM accounts with hundreds of contacts on their friend list, that wait for compromised IM accounts to send them malicious URLs or files. Unlike traditional honeypots which wait for attackers passively, HoneyBuddy follows a more active approach. By crawling the web or by advertising the accounts on several sites, we have managed to find accounts of real users and invite them to be our friends. Our decoy accounts have over six thousand users in their contact lists and receive between 50 and 110 unsolicited URLs per day.

In this paper, our main goal is to present an in-depth analysis and characterization of the collected URLs and malware. Our results show that 93% of the phishing URLs caught by HoneyBuddy are not present in other popular existinging blacklist mechanisms, such as the Google blacklist. Additionally, as much as 87% of all malicious URLs collected by our infrastructure is incorrectly flagged as safe by commercial anti-phishing products, such as Norton Safe Web. We also cross-reference the malware caught by our system with other collection infrastructures and find that 21% of our samples are zero-day malware instances. During our analysis of the top-level domains that host malicious URLs we trace the phishing domains back to a small number of IP addresses, revealing a large network of collaborating attackers. We also found that 10 of those domains belong to fast-flux networks. Our study reveals that scams that propagate through IM networks are specifically crafted for this communication channel and are different from those found in email spam.

We provide an attacker profile based on our findings and describe two different strategies for deploying spam campaigns. We argue that different technical aspects of the IM attacks lead to radically different spamming strategies. Next, we examine the effectiveness of IM attack campaigns based on the launching of our own (benign) attack campaigns. This experiment reveals that 12% of the users visit URLs sent to them by our dummy accounts, and 4% are willing to run an executable of unknown origin. Finally, we deploy the prototype implementation of our myMSNhoneypot service, an early detection service that can inform users if their accounts or IM clients have been compromised.

The remainder of the paper is structured as follows. In section 2, an overview of related work is presented. Section 3 describes the various attacks that target IM users. Section 4 provides a detailed description of the HoneyBuddy architecture, while sections 5 and 6 are an analysis of the data collected by our infrastructure. In section 7 we provide an attacker profile based on our findings, and in section 8 we describe our own *benign* campaign and present the results. In section 9 we propose defenses against IM attack campaigns. Finally, we summarize and conclude in section 10.

2 Related Work

Xie et al. propose HoneyIM [39], a system that uses decoy accounts in users' contact lists, to detect content sent by IM malware. HoneyIM can be deployed in a enterprise network and alert network administrators of malicious content, provide attack information, and perform network-wide blocking. HoneyIM has a limited view of the IM attack landscape due to its passive architecture and enterprise deployment. To overcome these disadvantages, HoneyBuddy is an active architecture that constantly adds new "buddies" to its decoy accounts, transcending the narrow confines of an enterprise level deployment, and monitors a variety of instant messaging users for signs of contamination. Furthermore, the use of pidgin [15] prevents their system from detecting attacks that exploit vulnerabilities in dominating instant messaging software such as the MSN live messenger [11].

Trivedi et al. address the problem of instant messaging spam (spim) and how to utilize honeypots to extract network and content characteristics of spim[37]. They set up an open SOCKS proxy that only allows outbound connections to IM servers. The analysis of the collected data reveals several characteristics of spim campaigns. An interesting result is that advertised URLs lead to a small number of websites, something that is confirmed by our findings. However, there are several major differences with our work. While they focus on spim campaigns, our honeypot detects all types of instant messaging threats mentioned in section 3, and also handles malicious file transfers. Furthermore, they propose a passive architecture that waits for spimmers to connect to their open proxy while our system actively broadens its view by connecting with a diverse and wide-spread set of IM users. Finally, their approach will not work with encrypted instant messaging traffic, such as Skype traffic.

Mannan et al. conduct a survey and provide an overview of threats against instant messaging users and existing security measures[33]. Several scenarios of attacks against IM users are presented, as well as the weaknesses of default security and privacy features provided by IM client software. They conclude that existing public and enterprise IM systems fail to provide sufficient security and protect users from existing IM threats. Hindocha[28] provides an overview of several IM clients and protocols, threats to instant messaging like worms and trojans, and issues regarding IM blocking.

Liu et al. [31] propose an architecture, for detecting and filtering spim, that incorporates widely deployed spam-filtering techniques and new techniques specific to spim based on the analysis of spim characteristics. In follow-up publications [32, 30], the authors focus on instant messaging worms. In [32] worm propagation is modeled and traced through multicast event tree tracing, while in [30] a formal IM worm modeling based on branching process is presented. Williamson et al. [38] apply virus throttling as a mitigation measure against viruses and worms that spread through instant messaging. They explore how several throttle parameters delay propagation without interfering with normal traffic.

Provos et al. [36] follow a different approach than ours for locating URLs that distribute malicious content.

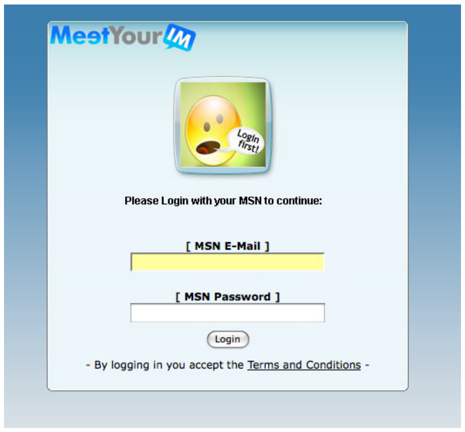
They actively scan a large number of URLs to locate malicious actions and focus only on drive-by downloads, while we passively collect URLs from spam messages in IM traffic. A very interesting fact is that their findings show that there is a difference between the domains of the frontend servers that contain URLs that exploit vulnerabilities in users' browsers or plugins, and the domains of the backend servers that distribute the malware. However, our results based on URLs collected by the HoneyBuddy infrastructure do not reveal any such frontend servers. All malware samples were downloaded from the same domain without redirection to a different domain. This highlights a different approach to malware distribution between drive-by downloads and phishing campaigns.

3 Attacks on Instant Messaging networks

The high population of IM networks makes them an attractive target for attackers that try to exploit them for malicious purposes, such as spreading malware and scamming. We identify four different scenarios of attacks on IM networks.

Malware infection. Recent malware instances [27] can attach to a victim's instant messaging client and start sending URLs that point to malicious websites, or spread themselves by sending executables. In the most common case the malware instance logs in to the IM network, randomly selects users from the victim's contact list, sends the malicious URLs or files and then immediately logs out. In order to be more appealing to potential victims, the URLs point to domains whose name contains the username of the recipient, for example `http://contact_username.party-pics.com`. The vast majority of the attack campaigns we have detected send messages in English. However, we believe that attackers will soon shift towards localized messages, as is the case with one localized phishing site that we have detected.

Compromised accounts. Attackers can also use compromised credentials to log in as several different users and flood the victims' contact lists. Many services, like MSN, use unified credentials for e-mail and instant messaging, making life easier for attackers. Attackers can harvest IM accounts by setting up phishing sites for the service, by planting key-loggers or through social engineering. A relatively known attack campaign is that of websites advertising a service that can reveal to users if someone has blocked them. If the user enters her IM credentials in the website, she is redirected to a page from another domain where nothing happens.



Please Login with your MSN to continue:

[MSN E-Mail]

[MSN Password]

Login

- By logging in you accept the Terms and Conditions -

Figure 1. Screenshot from an MSN phishing site.

Later on, the phishing site owner logs in as the user and sends messages to the victim's contact list. A screenshot of such a phishing site is displayed in Figure 1. A study of the phishing domains is presented in section 5.

Exploiting weak privacy settings. Even in the absence of malware infection or stolen credentials, some messengers provide the option to allow incoming messages from people who are not in the user's contact list. We tested the latest client versions of the most popular IM services: MSN live messenger (version 14.0.8089), Skype (version 4.1), Yahoo (version 10) and AIM (version 7.1). MSN live messenger is the only IM client we tested that has a privacy setting enabled by default that blocks messages from accounts not contained in the contact list. Skype, Yahoo and AIM by default allow anyone to send instant messages to our account, but this setting can be opted-out. Attackers exploit these settings to send unsolicited messages to IM users.

Exploiting client software. IM client software suffers from the problem of monocultures. Once an exploit is discovered, then automatically millions of clients can be infected immediately [26]. While in the case of malware infection exploits take advantage of the IM client to spread, this case involves the attack where the IM client is used to infect the rest of the machine.

4 Design and implementation

HoneyBuddy was designed taking into consideration the four attack scenarios described in section 3. In contrast to previous work[39], HoneyBuddy does not use modified versions of open source alternatives. It rather uses the latest version of the original clients, the same software most users install. The main reason for this choice is that direct attacks on IM client software will be detected. The basic concept behind HoneyBuddy is to add random accounts to a decoy IM account and monitor the incoming connections. As HoneyBuddy is in fact a honeypot, any incoming communication is by default suspicious. For our prototype we chose the MSN service due to its popularity. However, the design of HoneyBuddy is generic enough to allow the fast implementation of other services as well, like AIM, Skype and Yahoo messengers. Furthermore, MSN live messenger 2009 inter-operates with Yahoo, and is planned to introduce interoperability with Google Talk, AIM and other services, rendering our architecture deployable for all major instant messaging services[1]. All deployed messengers run in a fully patched Windows XP SP3 system.

4.1 Architecture

HoneyBuddy has three main components; a harvesting module, a script-based engine that handles the MSN messenger clients and the inspection module.

The harvesting module is responsible for gathering accounts that will later be added to the decoy accounts. All harvested accounts are inserted in CTT files (MSN contact files) that are imported in the messengers and all accounts listed are automatically invited. Another way is to search for e-mail addresses that belong to the @hotmail.com and @live.com domains. Other potential sources are sites where users advertise their MSN account, such as [10]. A more advanced method is to harvest account names from popular social networking sites.

The script-based engine starts the messengers and invites all contacts gathered from the harvesting module. Based on the AutoIt software [3] , we can automatically start the application, import CTT files and invite other accounts to our friend list. The AutoIT software allows the manipulation of the windows of an application the same way a user would manually click, monitor the status of the application and check for new windows (in order to check for incoming messages). After encountering an attacker that waited for a reply to his initial message before sending the malicious URL, we modified our system to send a random automated response to each incoming message. When an incoming message comes and includes a request for a file transfer, the engine automatically accepts the transfer. As each messenger can only have a limited number of friends in its contact list, it is preferable to run multiple messengers. For resource efficiency reasons, we used MSN Polygamy [12] in order to run multiple MSN messengers on a single platform without the need of additional virtual machines.

The inspection module monitors the logs of the messengers for malicious URLs. It additionally checks the default download folder for new file transfers. An interesting finding is that we received URLs and malware in the Hotmail inboxes of our accounts. Thus, we extended the inspection module to also fetch and analyze e-mails, so as to extract URLs and executable attachments. All malicious URLs are stored in a database and are queried every one hour to check their uptime status.

4.2 Contact sources

We used two major sources for finding and adding contacts. The first one was queries for contact files and e-mail accounts belonging to the @hotmail.com and @live.com domains. Simple queries like "filetype:ctt msn" or "inurl:'@hotmail.com" were able to provide us with thousands of contacts. We also harvested e-mail accounts from other popular sites like `buddyfetch.com`[4], from which we extracted 38,000 hotmail addresses. Overall, we have invited 14,912 contacts to become friends with our accounts. 3,012 of those (20%) accepted our invitation. The exact number of invitations and acceptances per decoy account is displayed in Figure 2. The five decoy accounts denoted in Figure 2 as decoy accounts 14 to 18, sent a thousand invitations each, to addresses extracted from `buddyfetch.com`. We plan on adding the remaining accounts to our system in the near future. More advanced methods of harvesting [17] can be based on popular social networking sites like Facebook. By crawling such networks, one can collect IM accounts from accessible profile pages.

Other potential sources are sites where users advertise their MSN account, such as `messengerfinder` [10]. The `messengerfinder` site contains more than 25,000 active messenger contacts that are advertised by their owners for social networking purposes. We advertised our accounts on this site and instructed our honeypot messengers to accept any friend request. So far, we have added 3,505 contacts while this number increases daily. The exact number of contacts per decoy account is shown in Figure 3.

4.3 Hardening against direct attacks

HoneyBuddy runs the latest version of the original

[1]An experimental deployment of Skype and Yahoo honeypots collected too few URLs to extract any conclusions.

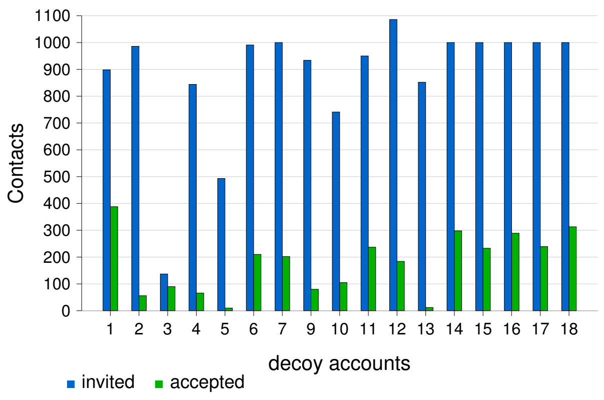


Figure 2. Number of friend invitations sent and number of accepted invitations, per decoy account.

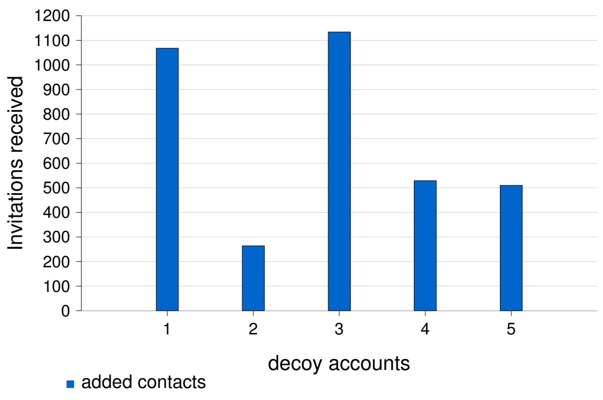


Figure 3. Number of invitations our decoy accounts received and accepted after being advertised on messengerfinder.com

IM client software and is, thus, vulnerable to infections. In order to prevent infections that will render our honeypot useless and make it a platform for further attacks, the honeypot messengers run inside Argos[34]. Argos is a containment environment that is based on memory tainting techniques. In a nutshell, Argos marks all bytes coming from the network as dirty and tracks their flow in the system memory. If a dirty byte is passed to the processor for execution, then we have an exploitation attempt since honeypots never download software to execute. When an exploit is detected, the application under attack is restarted and all attack information is logged to an alert file. This way, our system cannot be used as an attack platform as it is immune to remote exploits. A disadvantage of containment environments is that applications run 20 to 40 times slower than in a vanilla system. However, in the case of HoneyBuddy this is not a problem as messengers are hardly demanding in terms of computing power and memory requirements, since most of the time they are idle and wait for incoming messages. In our experiments, each MSN client's memory footprint is 50-80 MB and consumes negligible CPU resources.

We have to note, however, that during our experiments we have not yet encountered any attempts from attackers trying to exploit the IM client software. While this type of attack may not be common now, we believe that when IM attacks become more widespread, our implementation will provide useful information for such attacks.

5 Collected data analysis

In this section we provide an analysis of data collected by the HoneyBuddy infrastructure, from the 27th of February to the 16th of September 2009. Despite the technical simplicity of our system, we were surprised by the fact that popular defense mechanisms had not detected the majority of our collected data. During the collection period, the HoneyBuddy infrastructure collected 6,966 unique URLs that belong to 742 unique top-level domains.

During the first weeks of Honeybuddy operation we were able to fetch all URLs through the *wget* tool. However, malicious sites changed their behavior to avoid these fetches. Their pages now serve an obfuscated piece of Javascript code that changes the window location to a URL like `http://www.malicious.com/?key=<randomkeyhere>`. If a user has not visited the page with the key, then all subsequent requests are ignored and eventually her IP address is blocked for 24 hours. This behavioral change has forced us to fetch URLs through the Crowbar [6] environment that allows running javascript scrapers against a DOM to automate web sites scraping.

Our first step was to provide a simple classification for those URLs. Our five major categories were phishing, porn, dating, adware [2] and malware. The results are summarized in Figure 4. 1,933 of the URLs in 142 top-level domains were phishing for MSN accounts, 1,240

[2]We characterize sites that promote third-party addons for the MSN messenger (like extra winks, emoticons etc.) as adware sites

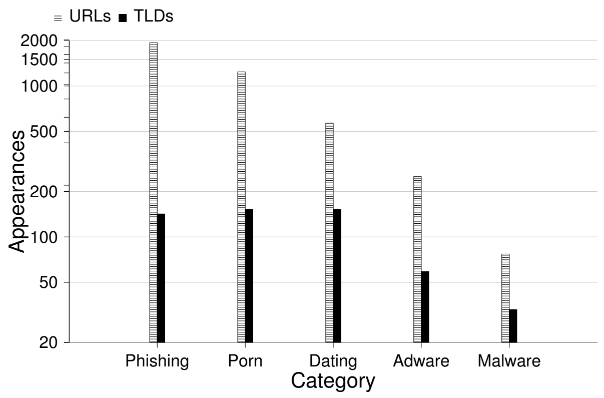


Figure 4. Classification of collected URLs

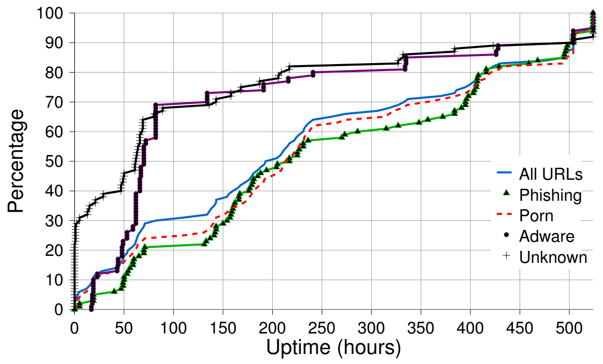


Figure 5. CDF of uptime of URLs per category

were porn, 567 were dating services and 251 sites were adware. While porn, dating and adware can be considered as harmless, the phishing sites pose a security danger for users. Furthermore, 77 URLs redirected to executable files or to pages that contained a direct link to a ".exe" or ".scr" file. We classify these URLs as malware.

We also spotted several sites that advertise subscription-based services for mobile phones. When the users enter their mobile phone number, they receive a subscription request. Once they subscribe to the service, they get charged for receiving SMS messages. These sites claim to give away free mobile devices to the subscribers of the service or promote quiz games that may attract victims, such as love calculators etc. These sites are highly localized. We visited them from different geographic locations using the Planetlab infrastructure [16] and got different pages in the language of the origin country. An interesting fact is that when the site cannot find the geolocation of the user, it redirects her to an MSN phishing site.

Our second step was to analyze the uptime of the collected URLs. The uptime graph can be seen in Figure 5. On average, a site is functional approximately for 240 hours (10 days). We also plotted the uptime graph for each category. We notice that porn and MSN phishing sites present much higher uptime than adware and unclassified sites. Half of the MSN phishing sites were alive for up to 250 hours (ten and a half days), while adware present a shorter lifetime of up to 80 hours (three and a half days).

5.1 MSN phishing

Attackers try to gather MSN credentials by tricking the user into entering her MSN e-mail and password in a bogus site. These sites falsely advertise a service that will reveal to the user which accounts from her contact list have blocked her. To validate that these phishing sites actually steal user credentials, we created several MSN accounts and entered them into the phishing sites. Each account had one of our decoy accounts as a friend. The decoy account received messages from the stolen MSN accounts that advertised the phishing site. However, the attackers did not change the passwords of any of the compromised accounts. After the attackers had started using our victim accounts, we submited "friend requests" towards these accounts. We wanted to see whether attackers will interfere with such matters and automatically accept requests. None of the submitted requests were accepted.

All phishing sites we visited shared one of three different "looks". A screenshot of such a site is shown in Figure 1. We analyzed the source HTML code of all the three "looks" and there was absolutely zero difference among the pages with the same look. This means the phishing pages with the same look had the exact same size and contained the same images and forms. This indicates that the majority of the different phishing campaigns might be deployed by a number of collaborating attackers. We also detected a localized phishing site which contained translated content, a technique used in e-mail spam campaigns[20]. The number of syntactical and grammatical errors revealed that the text translation was done automatically. For the time being, simple pattern matching for specific text segments is efficient for detecting these sites. Another detection mechanism is to query the various URL blacklists.

We queried the Google blacklist through the Google Safe Browsing API [8] to check if it included the phishing sites we discovered. From the 142 unique top-level domains (TLD) that hosted phishing sites and were detected by HoneyBuddy, only 11 were listed by Google blacklist. That means that 93% of the domains captured by HoneyBuddy were not listed elsewhere on their day of detection, making HoneyBuddy an attractive solution for MSN phishing detection. The average delay from when our system detected one of the 11 sites until it was included in the Google blacklist was around two weeks, leaving a time window of 15 days for attackers to trick users. Firefox, one of the most popular browsers uses the Google Safe Browsing API as an anti-phishing measure. We also compared our findings with the blacklist maintained by SURBL [22] and URLblacklist.com [24]. SURBL detected only 1 out of the 142 MSN phishing domains (0.7%) and none of the adware domains. None of the phishing or adware sites were listed by URLblacklist.com.

A very interesting fact was that when resolved, all the unique top level domains translated to a much smaller number of unique IP addresses. This fact confirms our initial theory that all these phishing campaigns lead to a limited number of collaborating attackers. To further investigate this behaviour, we conducted an experiment for a period of almost two months, presented in Section 6.

At the time of writing this paper, our infrastructure collected two new types of malicious URLs that demonstrate an evolution in the behaviour of IM scam campaigns. The first type is a phishing site that instead of advertising a service for MSN users, has recreated the exact "look and feel" of the Windows Live login page. The second type is a site that offers a Java applet to convert hosted images into a slideshow. The Java applet is not signed from a trusted source and requires unrestricted access to the victim's machine. Our disassembly showed that the applet downloads a PE32 executable from another domain and executes it.

5.2 Malware sample analysis

In this section we provide an analysis of the malware collected by the HoneyBuddy infrastructure, from the 1st to the 31st of March 2009. Our infrastructure collected 19 unique malware samples. We distinguish the malware collected by the HoneyBuddy infrastructure into two categories, the direct malware set and the indirect malware set. We present the two categories and proceed to further analyze the collected samples.

The first category contains malware samples collected either through direct file transfers (uncommon case) or by visiting URLs that were redirected to executable files. In the case of the URLs, the e-mail account of the victim was always appended as a parameter to make it look more realistic. In some cases attackers used popular keywords, like Facebook.

The second category, the indirect one, contains malware samples collected in two types of cases. In the first case, users are presented with a web page that alerts them that they need to download the latest version of the "adobe flash plugin" so as to play a certain video strip, and are prompted to download the installer which is, obviously, malware. In the second case, users are redirected to a page prompting them to install a screen saver. This ".scr" file they are prompted to download and install is a malicious file that infects the machine upon execution.

Due to the small volume of files, we were able to manually check these files using the Anubis analysis center [1]. All of them were characterized as dangerous , while some of them were bots that connected to an IRC C&C server. By joining the IRC network, we downloaded even more malware samples (not listed in this section).

In order to verify how original our samples are, we submitted them to the VirusTotal [25] service. VirusTotal is a large malware collection center with the primary goal of providing a free online virus and malware scan report for uploaded samples. The reason we chose to use VirusTotal is twofold. First, VirusTotal receives around 100,000 samples every day from a multitude of sources resulting in a large database of malware samples. The second and most important reason is that VirusTotal collaborates with a large number of well known anti-virus vendors[3] and uses their anti-virus engines and, therefore, can provide us with an accurate picture of the efficiency of up-to-date, state-of-the-art defense solutions for home users.

Four collected samples had not been seen by VirusTotal before, that is 21% of our samples were previously unseen malware instances. Figure 6 shows the relative detection delay compared to the date the samples entered the VirusTotal database. The base bar of the stack graph (solid white) shows how many samples were detected with a delay of one or more days, the middle bar (solid black) displays the number of samples that were

[3]For a complete list refer to `http://www.virustotal.com/sobre.html`

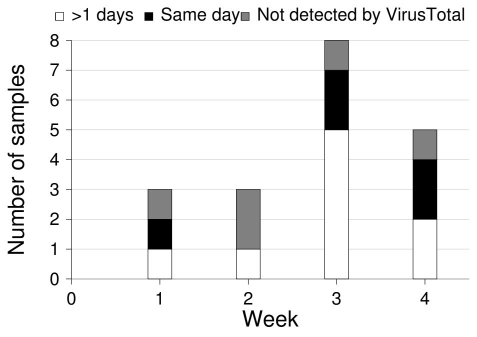


Figure 6. Detection delay of collected samples compared to the VirusTotal database. 21% of the samples were previously unseen, while 26% were collected the same day they entered the VirusTotal database.

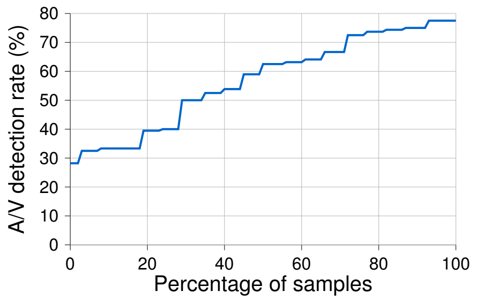


Figure 7. Cumulative distribution function of detection rate for collected samples based on VirusTotal reports. 42% of the samples were detected by 50% of the anti-virus engines.

detected the same day as VirusTotal while the top bar shows the number of samples not included in the VirusTotal database. Five samples (26%) were collected the same day they entered the VirusTotal database, while the maximum detection delay was five days.

We also checked the VirusTotal analysis reports for the collected samples. 42% of the samples were detected by half of the anti-virus engines, while the maximum detection rate was 77%. However, the dates of the analysis reports were one month after the collection date as we did not submit the samples the day they were captured. The one month delay means higher detection rates for the anti-virus engines as they update their signature files daily. Even in that case, it can be observed that there are samples that are recognized by only one third of the anti-virus products. The cumulative distribution function of detection rates can be seen in Figure 7.

5.3 Mailbox analysis

In this section we present an analysis of the emails we found in the mailboxes of our decoy accounts. Our analysis focuses on two aspects of the incoming emails. First, whether the body of the email contains any URLs and, second, whether the email contains any attachments. The decoy accounts received a total of 4,209 emails, 403 of which contained a total of 1,136 attachments. The emails contained 5,581 URLs which were passed to our classifier. The goal of the classification was to only identify phishing URLs and URLs that downloaded malware. 26 of the received URLs belonged to phishing domains while 7 downloaded malware samples.

While the majority of the attachments were pictures, several were windows media files and office documents. We checked the VirusTotal database for the MD5 hashes of the files but found no matches. This was expected, since hotmail scans incoming emails for malware and blocks executables. The most interesting attachments were two ".zip" files. Once extracted, the zip files returned a ".lnk" file. Upon inspection, we found that the files were command line scripts that connect to an FTP site and download and execute malicious software. For a more detailed analysis of the file refer to this report by F-Secure [9].

5.4 Comparison to email spam

An important aspect of IM based attacks that we wanted to explore was whether they are scams that use other commmunication channels as well or if they are unique to this attack medium. In particular, we wanted to explore similarities between the campaigns collected by HoneyBuddy and scams that propagate through emails that are collected by spam traps. We obtained 458,615 spam emails collected by Spam Archive [19] from the 27th of February to the 16th of September 2009. From these emails we extracted 467,211 unique URLs that belonged to 52,000 unique TLDs. We compared them to our 6,966 unique URLs and found only one common instance of a well-known benign website. This result was expected since URLs in IM attacks usu-

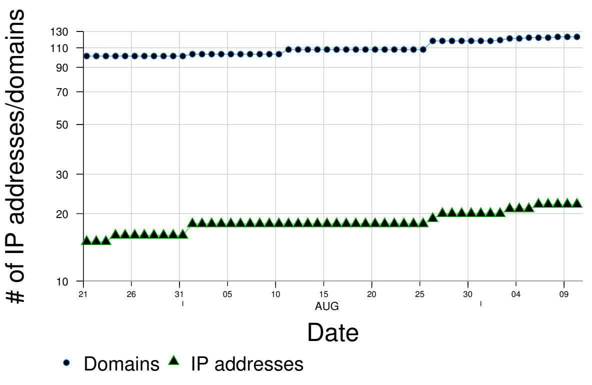


Figure 8. Number of distinct phishing domains and the IP addresses they resolve to over time.

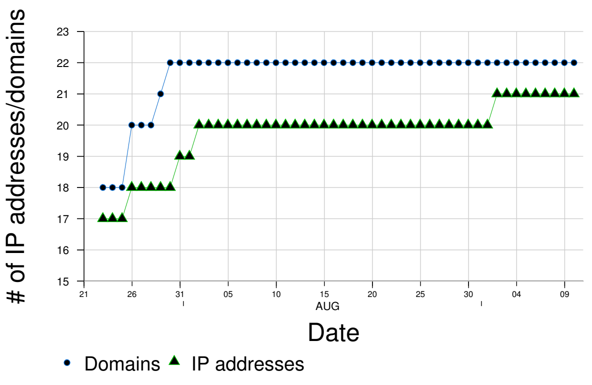


Figure 9. Number of distinct malware-distributing domains and the IP addresses they resolve to over time.

ally contain the target's username. Next, we compared the unique TLDs and found 21 common domains, 9 of which were popular and benign domains. From the 12 suspicious domains, 3 were classified as porn, 3 hosted malware, 2 were for dating, 1 was for adware and 3 were not assigned to a known category. None of the common TLDs hosted msn phishing scams, the prevalent threat of IM networks. Therefore, we can conclude that attackers have crafted scams specific to this new attack medium that are fundamentally different to existing email spam campaigns.

The results of this comparison are a strong indication that scams that propagate through IM networks are new campaigns and not email spam campaigns that utilized a new attack channel. They present their own unique properties that differentiate them from traditional scams that propagate through spam emails.

5.5 Comparison to commercial anti-phishing product

Multiple anti-virus vendors provide software products designed to protect users from malicious sites. To verify whether the URLs we collect have already been seen by large vendors that offer anti-phishing products, we evaluated our URLs using such a service. We used the Norton Safe Web service [13] provided by Symantec [23], where users can submit a URL and receive a report that contains information regarding possible threats from the specific website. We submitted the 2,010 phishing and malware URLs collected by our infrastructure after the collection period. Submitting the URLs after the collection period and not each URL individually upon collection, results in a potentially higher detection rate for the Norton Safe Web service. Even so, Norton Safe Web flagged only 13% of the submitted URLs as dangerous or suspicious. Specifically, 246 phishing and 10 malware-distributing URLs were reported as malicious while all other pages were characterized as safe. That means that over 87% of the malicious URLs collected by HoneyBuddy had not been properly categorized as dangerous by one of the largest security vendors.

6 Hosting analysis

The fact that all the top-level domains of the URLs collected from our initial experiment translated to a very small number of IP addresses, urged us to conduct a new experiment that might reveal more information. For a period of 50 days during July and August of 2009, we periodically ran *nslookup*[14] for each of the unique top level domains our system had collected up to that moment, in order to gather more information regarding how and where attackers host phishing and malware-distributing domains. Here we present the results for each category of domains separately and highlight their particular behaviour.

The experiment gave us further insight in regards to the small number of IP addresses that host a multitude of phishing campaigns. All top level domains translated to one or two IP addresses, while 98% of them translated to only one. Furthermore, ten of the top level domains belonged to fast-flux networks and translated to a differ-

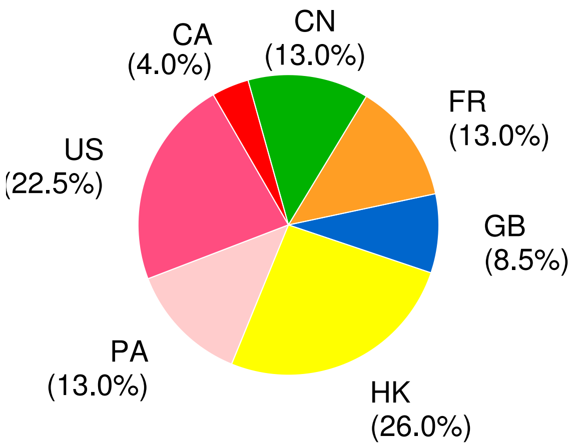


Figure 10. Breakdown of countries that host the phishing domains.

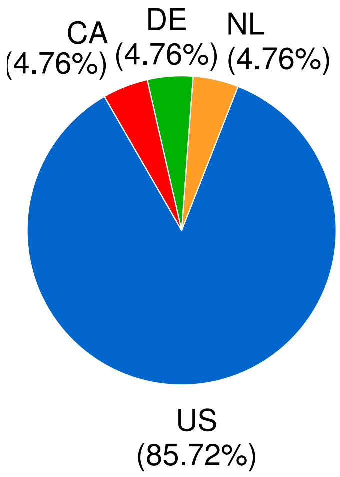


Figure 11. Breakdown of countries that host the malware-distributing domains.

ent set of IP addresses each time. In Figure 8 we can see that during the first days of the experiment, all 101 top-level domains translated to only 14 different IP addresses. The TLDs that belonged to fast-flux networks are excluded from the graphs. This behaviour is consistent throughout the duration of the experiment and as new domains were added only a small number of new unique IP addresses appeared.

Next we wanted to track down the country where the domains were hosted. We used the MaxMind [4] database. In Figure 10 we can see the breakdown of the percentages of the countries that hosted the top level domains. Honk Kong ranks first hosting 26% of the domains, while the United states follow with 22%. A surprising result is that only 13% of the domains were hosted in China, which is quite lower than what we would expect based on reports [21].

Next we present the results from the experiment regarding domains that distribute malware. Our initial goal was to investigate whether the top level domains of the malware-distributing websites also translate to only a small number of IP addresses.

In Figure 9 we present the results from this experiment. We can see that in the case of the URLs that contain malware, the top level domains translated to different IP addresses. Unlike the phishing domains, here each top level domain translated to a different IP address, and only one to three IP addresses overlapped at each moment in time. The IP address that hosted three malware-distributing domains also hosted one of the phishing domains and was located in the United States. None of the other IP addresses hosted both a malware-distributing and a phishing domain. The nslookup operation for all the top level domains returned only one IP address, and only one of the domains belonged to a fast-flux network pointing to a different address each time. Since the IP addresses that host phishing domains are more likely to be blacklisted, this result is not surprising. Another interesting fact is that none of the top level domains is in both of the sets, meaning that none of the domains hosted a phishing site and simultaneously distributed malware.

Similarly to the phishing domains, we wanted to trace the country where the malware-distributing domains were hosted. In Figure 11 we can see the breakdown of the percentages of the countries. The United States were responsible for hosting the majority of the domains that distribute malware through IM traffic, reaching almost 86%. The remaining three countries, Canada Germany and the Netherlands, hosted an equal amount of domains. Once again, it is surprising that China did not host any of the domains caught by our infrastructure.

7 Attacker profile

In this section we present statistics and observations in an effort to outline the behaviour of IM attackers and recognize different strategy patterns.

[4] http://www.maxmind.com/app/geolitecity

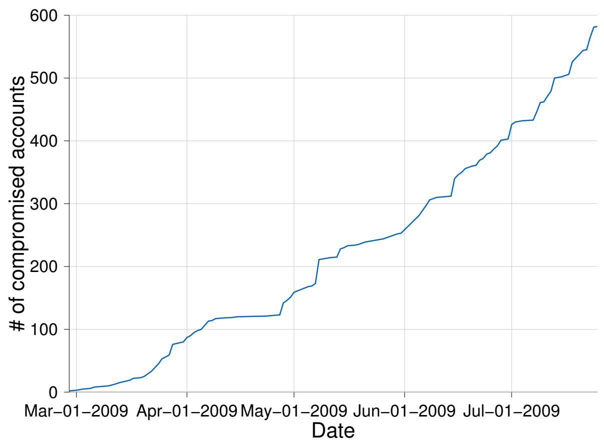


Figure 12. Number of compromised accounts that contacted our decoy accounts over time.

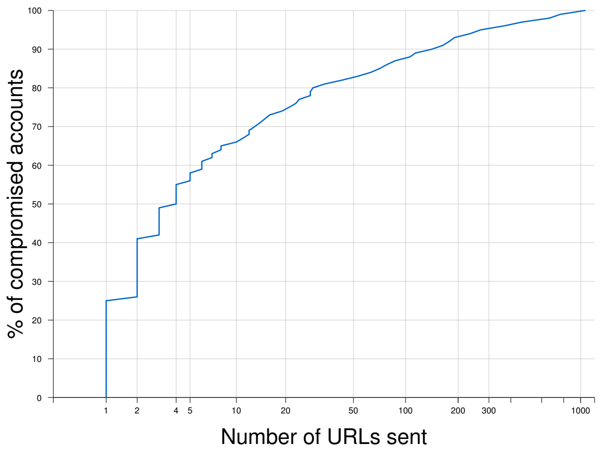


Figure 13. CDF of the number of URLs sent by compromised accounts to our decoy accounts.

First of all, in Figure 12 we present the number of unique compromised accounts that had sent URLs to our decoy accounts over time. We can see that the plot line follows a sub-linear curve, with almost 100 new accounts contacting us each month. This indicates that over time legitimate users still follow malicious URLs sent by compromised "buddies" and in turn get infected too.

In Figure 13 we can see the CDF plot of the number of URLs sent by each compromised account to our infrastructure throughout the duration of the experiment. One should note that approximatelly 25% of the compromised accounts sent only one URL and 40% up to two URLS. Based on the numbers we can identify one possible strategy that attackers choose to follow.

Eventhough some of these accounts/hosts may have been dis-infected before sending us another URL, it is improbable to assume that all of them were "cleaned up". Therefore, this might indicate a cautious behaviour on behalf of the attackers. With 55% of the compromised accounts sending up to 4 URLs and 75% sending less than 20, it is evident that one strategy that attackers follow is to avoid aggressive spamming behaviours so as not to raise suspicions among the compromised accounts' contacts. Such aggressive behaviours could alert the user and lead to the dis-infection of the account/machine. However, this cautious behaviour could also be attributed to technical reasons. If the attack is propagated through a worm that infects the client, then a low rate of worm propagation would be used so as not to trigger antivirus or intrusion detection systems.

Furthermore, approximately 12% of the attackers sent at least 100 URLs to our decoy accounts. This aggressive strategy of massively dispatching spam messages, indicates a category of attackers that don't try to remain beneath a certain threshold. This can also be attributed to technical reasons. Specifically, amongst the top ten compromised accounts that sent us the largest number of URLs, we found all the victim accounts whose credentials we had entered in phishing sites. Therefore, the attackers use tools to send messages from these compromised accounts without relying on worms that infect the IM client software. Thus, we can recognize a second more aggressive strategy, where it is not necessary for attackers to adopt a stealthy propagation rate. Finally, it is interesting to note that attackers send URLs from all the categories, as well as malware, and do not focus on one specific type.

8 Real-case Evaluation

We were interested in investigating the potential effectiveness of an IM attack campaign. To do so, we decided to launch our own benign campaign targeting the contacts of one of our honeypots. There were two factors to be taken into consideration. The first one was localization. Several users would get suspecious if we sent them messages that were not in their native language.

We queried the profile pages of most of the contacts we had at our disposal but unfortunately we could not retrieve country information for most of them, so we decided not to localize the messages sent. The second one was whether a conversation would take place before actually sending the URL. A message containing a URL without any prior conversation might get flagged as suspicious immediately.

Nonetheless, we decided to launch a simple spam campaign that imitated the ones caught by HoneyBuddy, that would not provide biased results. We logged in one of our honeypot accounts and sent the following message to the online contacts: "Hey! Check this out: `http://mygallery.webhop.net/gallery1/photo1.jpg`". The URL pointed to a web server controlled by us and redirected the user to a web page that asked the user to download a (benign) executable. The executable was a harmless program that just requested another URL, again from our own web server. That way, we were able to track if and when a user actually executed the file she downloaded. We contacted each online contact only once for the whole duration of the experiment.

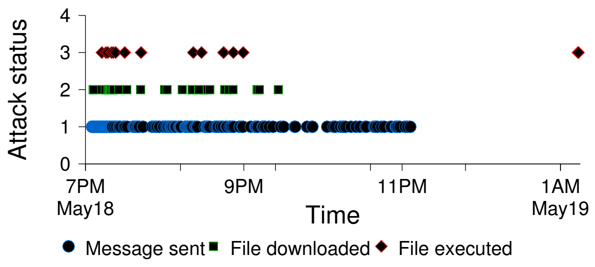


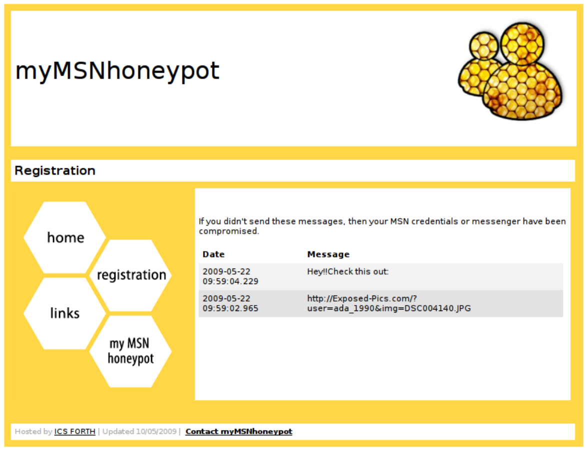
Figure 14. Time series of events that occurred during our benign campaign.

The results of the campaign are summarized in Figure 14. The bottom series plots the timestamps when the message was sent to an online user. The middle series plots the timestamps when a contact visited the URL contained in the sent message and downloaded the executable. The top series displays the timestamps when a contact actually ran the downloaded executable. During our campaign we sent 231 messages. 27 unique users (11.6%) visited the URL and downloaded the file, while 9 of them (4%) executed the downloaded file. We repeated the same experiment with two other accounts and the results were almost the same.

9 MyIMhoneypot, a detection service

In this section we present an overview of existing defense measures, and propose a service for the early detection of attacks targeting instant messaging networks. The existing defense mechanisms deployed by instant messaging service providers and other vendors, are insufficient for protecting users from the threats presented in Section 3. Anti-virus products that scan files received from instant messaging file-transfers fail to identify all malware used by IM attackers, as shown by our findings. Anti-virus vendors could provide more up-to-date signatures for IM malware by deploying HoneyBuddy for the early collection of such malware. Furthermore, as shown in Section 5.5, anti-virus products designed to protect users from phishing attacks fail to detect 87% of the malicious URLs collected by our infrastructure. Pop up messages from IM client software that alert users of phishing, that are triggered by all messages that contain a URL even if it is benign, are ineffective since users tend to ignore warnings that are presented even for well-known benign URLs. We propose that IM clients should correlate received URLs with blacklists and alert users only when they belong to malicious domains. We present our client-side mechanism that is orthogonal to existing defense mechanisms; myIMhoneypot, an early detection service that can inform users if their accounts or IM clients have been compromised. IM attacks try to spread through the victim's contact list by sending either URLs or files to the victim's friends. Any user that wants to check if her account is compromised registers with the myIMhoneypot service. Upon registration, the service creates a unique IM honeypot account (for example, a new MSN account that will be used as a decoy account) and informs the user to add that honeypot account to her contact list. As the user will never start a conversation with the honeypot account but an IM attacker will (with great probability), the user can check if something is wrong by visiting the website of the service and checking the conversation logs with her unique honeypot account. If there are entries in the conversation log of her decoy account like the example in Figure 15, then there is a strong indication that her IM client or credentials have been compromised.

The reason that a unique IM account must be created per user is twofold. First, if the service has only one or a few honeypot accounts then they can be easily blacklisted (recall that anyone can subscribe to the service, including attackers). The attacker should not be able to distinguish whether a contact is a decoy account or not. The service creates accounts with human-like nick-


myMSNhoneypot

Registration

If you didn't send these messages, then your MSN credentials or messenger have been compromised.

Date	Message
2009-05-22 09:59:04.229	Hey!!Check this out:
2009-05-22 09:59:02.965	http://Exposed-Pics.com/?user=ada_1990&img=DSC004140.JPG

Hosted by ICS FORTH | Updated 10/05/2009 | Contact myMSNhoneypot

Figure 15. A screenshot of the log presented to a user whose IM account has been compromised.

names. Second, the attacker can try to hack into the service's accounts once she knows the user is a subscriber. Using a unique honeypot per user makes the attacker's life a lot harder. The attacker cannot correlate common friends across accounts and has to try to compromise all the accounts in the user's contact list. Even if she does that, most IM services (at least MSN and AIM) do not keep conversation logs at the server side so she cannot find her spam messages in the logs of decoy accounts.

The attacker could guess the decoy accounts by checking the locally stored conversation logs. Normally, a user will have conversations with all members of her contact list except the honeypot account. Therefore, the attacker could avoid sending messages to accounts for which no conversation logs were found. This attack can be easily circumvented by planting a fake conversation log on the user's side.

The myIMhoneypot service has a limitation. For each registered user, a new IM account must be created in order to be used as a decoy. This process involves the solution of CAPTCHAs [5] which prevents us from making it completely automatic. Although we could claim that MyIMhoneypot is a legal case for laundering CAPTCHAs, we did not implement it for obvious reasons. For the time being, we have to manually create decoy accounts. However, we propose that this service should be implemented by each IM provider as a means of protection for its users. We implemented a prototype of myIMhoneypot for the MSN platform. We call it myMSNhoneypot and it can be found at `www.honeyathome.org/imhoneypot.`

We also provide a service that does not require user registration. Users can submit URLs they receive in instant messages to correlate them with our database. As mentioned before, suspicious URLs usually contain the target's username and, thus, searching for an identical URL in our database would rarely result in a match. Therefore, the service searches our database for any URL that has the same top level domain with that of the submitted URL, which is an indication that they might belong to the same campaign. If a match is found the user is presented with a small report containing the date the URL was first caught by HoneyBuddy and the category it was assigned by our classifier. Based on our findings in Section 5 concerning the uptime of each of collected TLDs, we assign the submitted URLs with a value of how likely they are to still pose a threat to users depending on the time window between being collected by HoneyBuddy and being submitted by the user.

10 Conclusions

In this paper we propose HoneyBuddy, an active honeypot infrastructure designed to detect malicious activities in instant messaging services. HoneyBuddy automatically finds user accounts that belong to a supported IM service and adds them to its contact list. Our system monitors decoy accounts for incoming messages and file transfers, and extracts suspicious executables and URLs. The suspicious data gathered by HoneyBuddy is correlated with existing blacklists, and malware collection center databases. Despite the simplicity of our system, deployment for the MSN service showed that 93% of the identified phishing domains were not listed by popular blacklist mechanisms and 87% of all malicious URLs were incorrectly flaged as safe by a commercial "web-safety" product. Furthermore, 21% of collected malware samples were also not listed by other infrastructures. These findings confirm that existing security measures of instant messaging services are insufficient, and also indicate the effectiveness of our system as a complementary detection infrastructure. We further inspected the top level domains that host the phishing URLs and found that they translate to a very small number of IP addresses suggesting the existence of a large network of collaborating attackers. On the other hand, domains that distribute malware do not follow the same tactics and translate to a different set of IP addresses. We located domains that belong to fast flux-networks in both cases, however they are more common in the case of the phishing domains, which have a higher probability of being blacklisted. Based on the results from the analysis of the IM attacks we caught, we provided a profile of the attackers and their spamming strategies. An interesting aspect of IM attacks that could not be measured by our infrastructure was how successful an MSN phishing campaign can be. To get an estimation, we deployed our own *benign* campaign and found that almost 12% of the users followed the URL and 4% ran the executable it redirected to.

We also deployed myMSNhoneypot, a prototype implementation of a service that is open to the public and creates dedicated IM honeypots for users. This service provides an early alerting mechanism for users whose IM accounts or clients are compromised. It provides decoy accounts for users that register with the service to add to their contact list. A message from the user to a decoy account is an indication that the user's credentials or IM client are compromised, as the user would never initiate a conversation with the decoy contact. We propose this type of service to be adopted and deployed by instant messaging vendors. Finally, we offer a service where users can submit a URL and receive a report indicating if the top level domain has been classified as dangerous.

Acknowledgments

This work was supported in part by the project CyberScope, funded by the Greek Secretariat for Research and Technology under contract number PENED 03ED440. We thank the anonymous reviewers for their valuable comments. Iasonas Polakis, Thanasis Petsas and Evangelos P. Markatos are also with the University of Crete.

References

[1] Anubis: Analyzing unknown binaries. `http://anubis.iseclab.org/`.

[2] AQABA Search Engine Demographics. `http://http://www.aqaba-sem.com/search_ed.htm/`.

[3] AutoIt. `http://www.autoitscript.com/autoit3/index.shtml/`.

[4] BuddyFetch. `http://buddyfetch.com/`.

[5] CAPTCHA: Telling Humans and Computers Apart Automatically. `https://captcha.net/`.

[6] Crowbar. `http://simile.mit.edu/wiki/Crowbar`.

[7] Europe surpasses north america in instant messenger users, comscore study reveals. `http://www.comscore.com/press/release.asp?press=800`.

[8] Google safe browsing api. `http://code.google.com/apis/safebrowsing/`.

[9] H1N1 Shortcut Malware. `http://www.f-secure.com/weblog/archives/00001738.html`.

[10] MessengerFinder, Find people online. `http://messengerfinder.com/`.

[11] Msn messenger. `http://messenger.live.com/`.

[12] MSN Polygamy. `http://www.softpedia.com/get/Internet/Chat/Instant-Messaging/MSN-Messenger-7-8-Polygamy.shtml/`.

[13] Norton safe web from symantec. `http://safeweb.norton.com/`.

[14] nslookup. `http://en.wikipedia.org/wiki/Nslookup`.

[15] Pidgin, the universal chat client. `http://www.pidgin.im/`.

[16] Planetlab, an open platform for developing, deploying and accessing planetary-scale services. `http://www.planet-lab.org`.

[17] Scraping facebook email addresses. `http://kudanai.blogspot.com/2008/10/scraping-facebook-email-addresses.html`.

[18] Skype Fast Facts, Q4 2008. `http://ebayinkblog.com/wp-content/uploads/2009/01/skype-fast-facts-q4-08.pdf`.

[19] Spam Archive. `http://untroubled.org/spam/`.

[20] The state of spam a monthly report august 2007. `http://www.symantec.com/avcenter/reference/Symantec_Spam_Report_-_August_2007.pdf`.

[21] StopBadware Blog : China Hosts Majority of Badware Sites. `http://blog.stopbadware.org/2008/06/24/china-hosts-majority-of-badware-sites`.

[22] Surbl. `http://www.surbl.org`.

[23] Symantec. `http://www.symantec.com/index.jsp`.

[24] Urlblacklist.com. `http://www.urlblacklist.com/`.

[25] Virustotal, online virus and malware scan. `http://www.virustotal.com/`.

[26] Vulnerability in PNG Processing Could Allow Remote Code Execution. `http://www.microsoft.com/technet/security/bulletin/MS05-009.mspx`.

[27] W32.Bropia. `http://www.symantec.com/security_response/writeup.jsp?docid=2005-012013-2855-99&tabid=2`.

[28] N. Hindocha. Threats to instant messaging. *Symantec Security Response*, 2003.

[29] J. Leskovec and E. Horvitz. Planetary-Scale Views on a Large Instant-Messaging Network. In *Proceedings of WWW 2008*, April 2008.

[30] Z. Liu and D. Lee. Coping with instant messaging worms - statistical modeling and analysis. pages 194–199, June 2007.

[31] Z. Liu, W. Lin, N. Li, and D. Lee. Detecting and filtering instant messaging spam - a global and personalized approach. pages 19–24, Nov. 2005.

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

[33] M. Mannan and P. Van Oorschot. Secure public instant messaging: A survey. In *Proceedings of the 2nd Annual Conference on Privacy, Security and Trust (PST04)*, pages 69–77.

[34] G. Portokalidis, A. Slowinska, and H. Bos. Argos: an Emulator for Fingerprinting Zero-Day Attacks. In *Proceedings of ACM SIGOPS Eurosys 2006*, April 2006.

[35] H. Project. Know your enemy: Learning about Security Threats. *Pearson Education, Inc.*, 2004.

[36] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose. All your iframes point to us. In *SS'08: Proceedings of the 17th conference on Security symposium*, pages 1–15, Berkeley, CA, USA, 2008. USENIX Association.

[37] A. Trivedi, P. Judge, and S. Krasser. Analyzing network and content characteristics of spim using honeypots. In *Proceedings of the 3rd USENIX SRUTI*, 2007.

[38] M. Williamson, A. Parry, and A. Byde. Virus throttling for instant messaging. In *Virus Bulletin Conference*, pages 38–4, 2004.

[39] M. Xie, Z. Wu, and H. Wang. HoneyIM: Fast detection and suppression of instant messaging malware in enterprise-like networks. In *Proceedings of the 2007 Annual Computer Security Applications Conference (ACSAC07)*.