Spam Zombies from Outer Space

John Aycock & Nathan Friess
Department of Computer Science
University of Calgary

About the Authors

John Aycock is an assistant professor at the University of Calgary in the Department of Computer Science. He received a B.Sc. from the University of Calgary, and an M.Sc. and Ph.D. from the University of Victoria. He researches computer security and compilers, and conceived and taught the University's "Computer Viruses and Malware" and "Spam and Spyware" courses.

Nathan Friess is completing his B.Sc. in Computer Science at the University of Calgary. His research interests include computer security and wireless networking.

Mailing address: Department of Computer Science, University of Calgary, 2500 University Drive N.W., Calgary, Alberta, Canada T2N 1N4; Phone: +1 403 210 9409; E-mail: {aycock,friessn}@cpsc.ucalgary.ca.

Keywords

Zombies, botnets, spam, spyware, email worms

Spam Zombies from Outer Space

Abstract

How can better email worms be created? How can spyware claim more victims? How can better spam be sent? It turns out that these three questions are all related. We look at the future threat posed by email worms, spyware, and spammers that use zombie machines in a new way: sophisticated data mining of their victims' saved email. The end result skirts many defenses, and could trick even experienced users. We present new defenses that can be deployed now against this threat.

Introduction

In both advertising and social engineering, the common goal is to convince a targeted human to perform an action. The exact nature of the action varies with the domain: email worm authors may try to have the target run an executable attachment; spyware authors may want to direct the target to a specific web site for a drive-by download; spammers may want to have the target visit their web site or that of an affiliate. We will refer to email worm authors, spyware authors, and spammers using the generic term "adversary" since they share this common goal. In each case, the goal can be accomplished by sending out bulk email – spam – to potential targets.

Adversaries have one other goal, too. They want their spam to be as effective as possible. How can they do this? A flippant answer is for adversaries to use spelling checkers and both upper *and* lower case. A full answer is more complicated. The effectiveness of an adversary's spam can be characterized using the following formula:

$$effectiveness = \underbrace{sent \times (1 - P_{filtered})}_{\text{pre-delivery}} \times \underbrace{P_{read} \times P_{clickthrough}}_{\text{post-delivery}}$$

This formula measures effectiveness as a function of how many targeted people eventually perform the action that the adversary wants them to perform. The formula can be broken into two parts, one part describing what happens before the spam gets delivered to the target's mailbox, one part describing what happens post-delivery. The other terms are:

- *sent*, the amount of spam email messages sent by the adversary.
- $P_{filtered}$, the percentage of the spam that is intercepted *en route* by anti-spam defenses. The place where the filtering occurs is not relevant to effectiveness, and may be near the sending machine, the receiving machine, or done on a per-user basis.
- P_{read}, the percentage of the spam that is delivered to a potential target, and that is read by the target. By itself, this is a measure of the effectiveness of the From: and Subject: email headers used by the spam, as targeted users use the information in these fields (as displayed in their mail reader) to decide whether to read or delete the email.
- $P_{clickthrough}$, the percentage of targeted users that click on what the adversary supplies in their spam, such as a URL or an attachment. This describes how convincing the spam's body is.

The bulk of effort by adversaries, when trying to increase the effectiveness of their spam, has been directed at the pre-delivery part of the formula. Spammers increase the number of messages sent, to try and reach a wider audience of potential targets, or alter the messages in bizarre ways to try and evade filters. The latter tactic is particularly counterproductive, as it often renders the message unintelligible to the target.

Some efforts have been made at improving the post-delivery percentages, too. Prepending “Re:” onto a Subject: line to suggest that the spam is really a reply to some previous message was a clear attempt to improve P_{read}. As another example, the Mydoom.bb email worm tried to improve upon $P_{clickthrough}$ by using social engineering, and claiming to be from the IT staff (McAfee, 2005). These efforts are not very advanced yet, however. Ironically, if post-delivery attacks were more advanced, spam volume could actually *decrease* while yielding the same overall effectiveness.

In the remainder of this paper, we present a new method adversaries could use to improve spam effectiveness. The following sections explain the improved spam technique, the current defenses it avoids, our proof-of-concept implementation, and new defenses. We then look at further possibilities for this technique, along with related work.

Improving Spam

Improved spam will come from zombie machines. At first blush, this isn't terribly original, but the difference is in how the zombies are used. A zombie machine is not just a throwaway resource, a launching pad for DDoS attacks and spam; a zombie contains a wealth of data.

There are two key reasons why spam is suspicious to anti-spam filters and human targets alike. First, it often comes from an unrecognized source. Second, it doesn't *look* right. The evolution of spam zombies will change this. These new zombies will mine corpora of email they find on infected machines, using this data to automatically forge and send improved, convincing spam to others. In addition to the adversary, there are two other parties involved here: the *victim*, who owns a zombie machine, and whose saved email the adversary will be mining; the *target*, currently uninfected, that the adversary wants to click on something.

What features can be easily extracted from an email corpus? There are four categories:

1. Email addresses. The victim's email address and any other email aliases they have can be extracted, as can the email addresses of people with whom the victim corresponds. These addresses could be seen as fodder for harvesting by spammers, but we will show an alternate way to use these.
2. Configuration. This is information unrelated to individual messages, but related to the victim's email program and its configuration. For example, the User-Agent: email header added by the mail program to identify the sending program, the message encoding as text and/or HTML, and any automatically-appended signature file. The quoting style used by the victim's email program for replies and forwarded messages is a configuration issue too.
3. Vocabulary. The normal vocabulary used by the victim and the people with whom they correspond.
4. Email style. There are a number of distinctive stylistic traits, including:
    - Line length, as some people never break lines;[1]
    - Capitalization, or lack thereof;
    - Manually-added signatures, often the victim's name;
    - Abbreviations, e.g., “u” for “you”;

[1] The email program and its configuration may also play a role here.

- Misspellings and typos;
- Inappropriate synonyms, e.g., “there” instead of “their”;
- Replying above or below quoted text in replies.

Email style should be correlated with the email recipient(s), because the victim may use both formal and informal email styles depending on whom they are sending to.

The adversary would begin by releasing malware that would infect machines and create an initial pool of zombie machines. The malware would mine any email corpora on the infected machines to determine as many of the victim-specific email features above as possible. Any of the features may change over time – a former co-worker the victim emailed may have left the company, or the victim may have switched email programs – so a prudent adversary would only use recent messages in a corpus.

Next, the adversary would supply a message to send out, with proper spelling, grammar, and capitalization. While the message could consist of multiple sentences, the adversary would supply it without line breaks. It is irrelevant whether the message was built in to the malware, or whether the adversary has a mechanism to communicate a fresh message to the zombie network. The malware would transform the message using simple filters, so that the message is rendered in the style the victim uses for a given target. For example, if Vic the victim normally uses an informal style to email Margaret the target, then the malware might determine the following parameters about Vic's prior email to Margaret:

Average line length	30 characters
Capitalization	none
Manually-added signature	“-vic”
Abbreviations	“u” for “you”, “r” for “are”

And the message supplied by the adversary:

```
Hey, have you seen this web site yet? It has some cool v
ideos: http://evil.example.com
```

Would be transformed into:

```
hey, have u seen this web site
yet? it has some cool videos:
http://evil.example.com

- vic
```

At this point, the malware could simply spam out the message to Margaret and other targets. The transformed message still looks “spammy” by itself, however, even though it will come in the correct style from a recognized source. We think that more effective spam will not be sent like this, but in one of the following forms:

- Replies to incoming email. Here, the malware monitors legitimate incoming email addressed to the victim. When a new message arrives, its sender is targeted; before the victim has a chance to reply to the email, the malware composes and sends a reply in the victim's style:

```
Sorry, I won't be able to get to your email for
a while.  However, this should keep you amused
in the meantime: http://evil.example.com
```

This type of reply is consistent with observed human behavior in email interactions (Tyler & Tang, 2003). The deception can be further extended if the malware doesn't generate a reply when the victim is actively using the infected computer, because the victim may actually be replying to the email.

There are other advantages for the adversary. The malware is able to quote the incoming message, lending more credence to the spam. Also, by replying, the malware has a valid Message-Id: to use in the In-Reply-To: and References: email headers – a target using an email program that shows threads will link the malware's reply to the original mail.

∉ Forwarded email. The forwarding of messages, attachments, and links is a common occurrence in email, and is arguably a form of gift-giving (Smith, Ubois, & Gross, 2005). However, forwarding by humans is done bearing in mind the recipient's interests (Smith et al., 2005). A target's interests are not only hard for malware to automatically determine, but they are highly unlikely to coincide with the spam the adversary has to send.

Effective spam can shy away from the problem of targeting specific interests by using humor, a common denominator. An adversary can lure targets with the promise of funny pictures, sound clips, or animations; these can be easily copied from the Internet onto an adversary's web site (which also sports a drive-by download) or attached to spam. Having such a real lure acts as a Trojan horse, and can delay suspicion by the target.

The other problem is making the forwarded email come from an authentic-looking source. Malware can mine information from the victim's email corpora about groups of people that the victim usually CCs or sends email to; this is more precise targeting than using a victim's address book, because it reflects the victim's actual, current email practice. This information can be used for several purposes:

- Sending supposedly-forwarded spam from the victim to an authentic social group of targets;
- Making reference to a common acquaintance in the spam;
- Spamming fake “e-card” announcements from the victim to a plausible group of recipients.

As a special case, some people with a close, informal relationship simply forward URLs in the Subject: line, with a blank message body. The malware can look for these in the victim's email corpus, and exploit the trusted relationship to send URLs to the target similarly.

None of these schemes will work if the malware sends multiple copies of the spam to a target, or if the target receives one of these spams for every message they send to the victim. The malware must keep a database of spam attempts, and only send one spam per target. Further suspicion can be avoided if spam is not sent to already-infected machines; a zombie can covertly hide an infection indicator in outbound email with some crude steganography (Wayner, 2002). For example, the last line of all email from a zombie machine (whether a spam message or not) may have thirteen spaces appended onto it.

Malware can also avoid sending email at unusual times. As with other features, a victim's email corpus can be mined to determine those times when they usually respond to email. This also contributes to making the spam look more normal.

The path traveled by more effective spam is worth noting. A topologically-aware worm (Staniford, Paxson, & Weaver, 2002) is a worm that scans using information about network topology, to avoid the wasted effort in random or quasi-random scanning. More effective spam travels along the topology of *social* networks instead of computer networks, like an IM worm that only tries to infect people on “buddy lists”. The “small world” hypothesis holds that any two people are separated from each other by a short chain of acquaintances (Milgram, 1967; Travers & Milgram, 1969); this idea has recently been applied to searching in computer networks (e.g., Simsek & Jensen, 2005). One theory is that a social network works this way due to the presence of some highly-connected people (Gladwell, 2002) scattered throughout the network – intuitively, some people are more popular than others. This would mean that more effective spam only has to fool a handful of targets, enough to turn a highly-connected target into a victim, in order to be widely dispersed.

Defenses Avoided

The more effective spam described in the last section avoids many defensive measures that targets may have in place. Here, we look at anti-spam defenses responsible for $P_{filtered}$; detecting any malicious content run or downloaded as a result of the victim clicking what the adversary supplies is a separate issue, and is beyond the scope of this paper. For each defense, we give a brief description of it and an explanation of how more effective spam can evade it. Specific references are given where appropriate; Zdziarski (2005) is a good general introduction to the area.

∉ Whitelisting. Incoming email is allowed only from those senders who are on a special list, called a “whitelist”. Effective spam is sent from the victim, *as* the victim, to the target; if the victim is not whitelisted already, then no legitimate email from the victim can get through to the target.

∉ Blacklisting. Mail is refused if the sender is on a special “blacklist”. Again, if the victim is blacklisted, then legitimate mail from the victim could not be getting through to the target, and thus blacklisting is ineffective here.

∉ Greylisting. Mail from first-time senders is initially refused, on the premise that spammers won't retry transmission of email, but legitimate senders will. Senders that do retry are temporarily whitelisted to prevent future retransmission delays (Levine, 2005). More effective spam can piggyback on the victim's normal email system, which (presumably) handles retries properly.

∉ Rate limiting. Rate limiters, or throttles, assume that spam is being sent in large quantities, and slow the rate of outbound email to suppress spam (Williamson, 2003). This assumption is not valid for more effective spam. Reply-based spam will operate only as fast as legitimate incoming email to the victim; forward-based spam will not be convincing if sent in large quantities, so a smart adversary would limit the rate of forward-based spam themselves.

∉ Tarpits. A tarpit is a special mail transport program that replies… very… slowly… to incoming spam transmissions, to slow down overall spam transmission or at least annoy spammers. There would be no way to apply a tarpit against more effective spam without harming the victim's legitimate email, and again, this defense assumes large quantities of spam transmission.

∉ Spam traps. Spam traps are email addresses made public (e.g., posted to a web page) with the intention that they be harvested by spammers. A spam trap does not correspond to a real person, however, so any email sent to that address must be spam. More effective spam is sent to addresses from legitimate incoming email, or to legitimate addresses mined from a victim's saved email, thus avoiding spam traps.

∉ Challenge/response systems. A first-time sender is issued a challenge, which the sender must reply to prior to their email being delivered. Essentially, the goal is to try and prove that the sender is a human being. A successful response to a challenge results in the sender being whitelisted. Our reply-based spam would not be subject to a challenge; correctly-designed challenge/response systems would automatically whitelist the victim, because the target initiated the email exchange. Forward-based spam is sent to people named in the victim's saved email, for which the victim would have already replied to the challenge.

∉ Proof of work. Proof-of-work systems have the sender demonstrate that they have spent some nontrivial amount of computer time computing a value (Dwork & Naor, 1993). The idea is that this would impede spammers from sending mass email by imposing a tangible "cost" on them. Yet again, this defense assumes large volumes of spam, and is inapplicable to more effective spam.

∉ Authentication systems. Mail is verified to be sent from a legitimate mail sender for the domain stated in the email (e.g., Sender Policy Framework, Wong & Schlitt, 2005), *or* mail is verified to be sent by the stated domain and unaltered in transmission (e.g., DomainKeys, Delany, 2005). More effective spam is sent from the victim's machine in an authorized manner, and is unaltered *en route*, so sender authentication does not help as a defense.

∉ Checksum-based filtering. A collaborative scheme whereby a checksum, preferably one insensitive to minor text alterations, is computed for each incoming message. The checksum is sent to a central server that stores the checksums of known, reported spam; a hit from the server indicates that the email is spam. More effective reply-based spam would not be vulnerable to detection, because the target's quoted text in the reply would throw the checksum off. Forward-based spam could be caught, if the adversary's malware only introduces minor variations to the spam. However, the low-volume, deceptive nature of more effective spam may result in either the spam (and its checksum) not being reported, or it falling short of a "bulk email" threshold used by the central checksum server.

∉ Statistical filtering. Statistical filters, such as Bayesian filters (Graham, 2002), predict the likelihood of email being spam or ham based on how often the words[2] in the email appeared in corpora of previous spam and ham. Reply-based spam includes the target's quoted text to push the email in the "ham" direction. In a sense, this is a sophisticated form of "word salad", where spammers try to slip by statistical filters by including random words or phrases in their spam (Graham-Cumming, 2004). Forward-based spam is more vulnerable to detection, though, because its message is more exposed.

∉ Pattern-based and heuristic filtering. Pattern-based filters detect spam by looking for a match in a predetermined set of "this is spam" patterns. Heuristic filters combine a large number of heuristic tests (which may include all of the above types of defense) to make a spam/ham judgment about email. Both cases are similar to statistical filtering: reply-based

[2] "Word" is used in a very loose sense here.

spam is likely to avoid detection, forward-based spam may or may not be caught. Because more effective spam deceives by looking like legitimate email, the risk of trying to increase an anti-spam filter's detection rate is an increased number of false positives.

Of course, there are many variations and combinations of these defenses; this should be taken as an exhausting, but not exhaustive, list. The main point is that existing defenses are not enough to combat more effective spam.

Proof-of-Concept Implementation

We performed some experiments mining data out of email corpora, to determine how easy it would be to mount our proposed spam attack, and to see how convincing the result would be. Although they could be forged by the adversary's malware, a number of the features mentioned previously will be automatically incorporated into outgoing mail if the malware uses the victim's email program to send its mail, including From:, User-Agent:, preferred encoding (text and/or HTML), and any automatic signature file.

For our experiments, we focused on mining capitalization, commonly used abbreviations, common misspellings, manually entered signatures, and social groups from two corpora. The first corpus we used was a small dataset that we manually generated ourselves; the second was the public corpus of Enron email (CALO Project, 2004). In each of the cases, except for the social groups, we only examined the victim's outgoing email folder so that we are only examining text that the victim actually wrote. The implementation was comprised of several Perl scripts, each of which mined one of the listed features from the corpus. Each script did the following for each email message:

1. Strip off the automatically-appended signature, if any. This signature was found by searching the email program's settings.
2. Search for and remove any quoted text, in case this message was a reply or forward.
3. Perform the data mining (e.g., capitalization, manual signature).
4. Save the statistically significant results for later use in generating spam.

When mining the victim's capitalization habits, the script merely split the text into individual words, counting the number of words that were completely lower case, completely upper case, or mixed case. The decision as to which form of capitalization the user used for any given message was based on a percentage of which of the three forms were encountered. Although we simply estimated that 20% of words should be capitalized before we declare that the victim was using mixed case, a proper analysis of English text could be used to fine-tune this number. Once the script had processed all of the emails, our spam generator mimicked the capitalization most used by the victim.

Mining abbreviations was quite similar to capitalization, except that our script started with a list of commonly used abbreviations such as:

u	you
r	are
ur	your
cya	see you
ic	I see
afaik	as far as I know
thru	through

Each time one of the abbreviated forms of a word was encountered, the script incremented a counter for that type of abbreviation. In the end, if we found more than five occurrences of an abbreviation, our generator also used that abbreviation, although this threshold could be fine-tuned as well.

The script for mining common misspellings sent each word to a spell checker (Aspell), and noted which words were flagged as incorrect, as well as the suggested replacement. Again, if we found more than five occurrences of a spelling mistake, and the spell checker provided a suggested replacement word, then our generator performed the reverse replacement when creating the spam email.

When mining manually entered signatures, rather than looking at each word of the text, the script examined the last non-blank line of the text. The script kept track of each unique signature, and how many times it was encountered throughout the corpus. Similar to the other transformations, if a signature was found more than five times, it was added to the pool of signatures that were randomly added to generated spam.

The script that mined social groups was slightly different in that it examined all of the headers of an email, as well as the entire body, including quoted text. The script built a list of all of the email addresses found in the message for two purposes. The first reason was to count the number of times any given pair of addresses appeared in the same message. The second was to create a map between email addresses and real names of individuals. Both of these results were also used by our spam generator.

Finally, our spam generator used all of the above results to transform any given text into a spam email. The input for the generator was an email to reply to and some text to add to the reply, which would presumably contain a URL for our target to click on. The generator then output an email message that could be sent to the target.

The results of our experimentation were very positive (for the adversary). For the Enron corpus, our scripts were able to mine almost all of the mentioned transformations successfully. In one mailbox, we found over 670 occurrences of the abbreviation “r” and over 160 occurrences of “u”. Furthermore, even though the formatting of the emails was quite erratic, our scripts also successfully found the victim's manual signature “Vince”, with 1470 occurrences, far above the next best guess at 216 occurrences. Mining common spelling mistakes was less successful, as the spell checker had a tendency to correct things like people's names when it shouldn't have. This result is not entirely surprising. However, the spell checker did find minor nits such as “p.m.” versus “PM”.

Furthermore, the generator was able to create a convincing reply to an email in our custom made corpus. This can be demonstrated through an example. Given the following data from our fictitious victim, Jane Doe:

Abbreviations	“u” for “you”
Capitalization	mixed case
Manually-added signature	“Jane D”
Social groups	`tim@bigcorp.domain`, `rick@bigcorp.domain`
Real names	`rick@bigcorp.domain` ♥ Rick CoWorker

If Jane Doe had received an email from Tim Boss with the text:

```
Hi Jane,

When you get this message, I'm expecting a reply.
Please reply to me as soon as possible.

Thanks,

T Boss

--
Tim Boss
The Big Manager
Big Corporation
```

And given an intended spam message of:

```
Hi,

I just talked with [INSERT NAME HERE] and you should tak
e a quick look at http://some.bad/url
```

Our generator was able to create a reply to Tim Boss with the text below. (Note that the style of quoted text and the automatically-appended signature were added by the victim's email program.)

```
Tim Boss wrote:
>Hi Jane,
>
>When you get this message, I'm expecting a reply.
>Please reply to me as soon as possible.
>
>Thanks,
>
>T Boss
>
>--
>Tim Boss
>The Big Manager
>Big Corporation
>

Hi,

I just talked with Rick CoWorker and u should take a
quick look at http://some.bad/url
Jane D

--
Jane Doe
An Employee
The Big Corporation
```

Similarly, our generator can forward an existing message from the victim's inbox to a target. Continuing with the previous example, if Jane Doe had an email from Tim Boss in her inbox, our generator could forward this email to Rick CoWorker (part of Jane's social group) and add spam text. This forwarded email might look like:

```
Hi Rick,

I thought u might be interested in knowing about this.
Also, take a quick look at http://some.bad/url for more
details.
-------- Original Message --------
Subject: That message
Date: Fri, 06 Aug 2005 10:01:14 -0600
From: Tim Boss <tim@bigcorp.domain>
To: jane@bigcorp.domain


Hi Jane,

When you get this message, I'm expecting a reply.
Please reply to me as soon as possible.

Thanks,

T Boss

--
Tim Boss
The Big Manager
Big Corporation

--
Jane Doe
An Employee
The Big Corporation
```

Since we had the Enron corpus at our disposal, we analyzed it to determine approximately how many people our spamming technique would have to fool in order to infect any given target within a group of connected people. The Enron corpus contains 517,431 messages from 151 employees' mailboxes. We examined each message, noting who had sent email to whom, using the From:, To:, and CC: fields. This defines a graph where each email address is a node, and each connection (as given by a pair of addresses found in the From-To or From-CC fields) is an edge. Using this graph, we then computed the shortest path between each pair of nodes. (In order to reduce the amount of computation required, we reduced the graph to only consider messages sent between addresses within the `enron.com` domain.) According to our analysis, the maximum path length is 15 hops, with a median of 8 hops. Using this graph we also found that 89 nodes had a degree of 500 or more, which intuitively corresponds to the number of highly connected people in the Enron social group. This "small world" helps support our claim that using a social network is an effective way to distribute more effective spam.

We also analyzed our Enron graph to find pairs of people with close relationships, where two people have exchanged a lot of mail with each other. Such people may fall prey more easily to the special case of forwarded email we mentioned, where the adversary's URL is in the Subject: line and the message body is empty. Looking at both single- and multiple-recipient email messages in the corpus where 500 or more messages have been sent between two people, there are 506 close relationships. If we lower our threshold from 500 to 100 messages, then the number of close relationships climbs to 4511. In context, this means that out of every 46 relationships found in the Enron graph, we find one close relationship. We would conclude that this special case of forwarded email would not yield enough victims by itself for an adversary, yet it may be effective as a complement to other techniques.

New Defenses

User education is often viewed as a key defense against malware and spam. However, this is of limited value against the more effective spam we describe, because the spam bears all the normal hallmarks that the legitimate sender would display. This implies that a strong reliance must be placed on technical defensive measures. We suggest the following:

∉ Encrypt email corpora. Encryption will delay malware from mining email corpora, but will not indefinitely prevent it. Malware on the zombie machine can eventually steal the decryption key and operate as before.

∉ Salt email corpora. Email corpora can be salted with false information for malware to find, such as spam trap addresses or manually-added signatures like "Help, my machine is a spam zombie!" This would provide some warning of malware infection.

∉ Sandbox the browser. URLs followed from email should be viewed in a sandboxed browser, to limit the risk from drive-by downloads. More generally, this precaution applies to any application launched from an email reader.

∉ Adjust anti-spam filters. Anti-spam filters can be attuned to our more effective spam in four ways:

1. Watch for known infection indicators, if the adversary's malware uses one.
2. Place extra emphasis on URLs sent via email. This is unfortunately not always a trivial check, as an adversary may use innocuous-looking jump pages in their spam that redirect to the real web site (Spammer-X, 2004).
3. Canonicalize incoming email before filtering, undoing known transformations that the adversary's malware performs, before looking for known spam. The obvious disadvantage is the emphasis on known information, leaving lots of leeway for an adversary to slip new spam messages through before defenses have caught up.
4. Examine alleged reply messages both with and without any quoted text from the original message, to try and expose reply-based spam by stripping off its legitimate cloak.

For threats sent by an adversary as attachments, anti-virus software is still the best defense.

Other Possibilities

There are further possibilities for more effective spam. We have limited ourselves to relatively simple transformations of the adversary's original spam message. We conjecture that to scale the length of the spam (e.g., to the length of a typical 419 scam email) in a manner that would be convincing to a target would require a fairly sophisticated language model. This does not preclude an adversary from doing this, merely makes it more difficult. Furthermore, if excessive time or space were required to do this, malware could transmit necessary information from a victim's corpora to the adversary's computer for offline processing.

Zombies could also be used for targeted marketing by spammers. Currently, spammers can enter keywords describing their target market into programs like Atomic Email Hunter (AtomPark Software, 2005). The program will search for those keywords on a web search engine, then mine the returned web pages for email addresses. These keyword queries could be done just as easily in a victim's email corpora, with the advantage that the social networks that the corpora describe tend to connect people with like interests. Further keyword information is available if the adversary's malware automatically locates and searches a victim's homepage (Culotta, Bekkerman, & McCallum, 2004).

Specific vulnerabilities in mail programs, either mail readers or mail transport programs, could be exploited by mining email corpora too. Malware could be instructed to look for vulnerable versions of software in saved email headers, such as the User-Agent: and Received: lines, in order to identify targets. One defense would simply be to not save these email headers, but this defense would need to be widely-adopted to be effective, and a zombie machine could easily intercept the header information on new incoming email.

One hurdle adversaries face when evading targets' statistical spam filters is that there is no easy way of determining which words are indicative of ham and spam (Graham-Cumming, 2003). A victim's email corpus, however, details the words that have gotten by a target's statistical filter in the past. It also documents a target's own vocabulary, which is likely a strong indicator as to which words a target considers normal, or “hammy”.

Related Work

> ‘Directed attacks can, in theory, work, but require so much work on the spammer's part that the spam-filtering community hasn't seen a true attempt of these in the wild.’ – Zdziarski (2005, p. 137)

The quote above refers to attacks on statistical spam filters, and is also echoed by Graham-Cumming (2003) – the assumption is that an adversary will have an uphill battle, manually guessing at per-user ham words in the absence of feedback from the target. Our work suggests the contrary: that such directed attacks, using a victim's email corpus, are very likely, can be automated, and are scalable.

More effective spam can be both simpler *and* richer than predicted. The number one defense suggested by Graham-Cumming (2003) is to only accept plain-text email, yet our more effective spam will work in both plain-text and HTML formats. Graham (2002) refers to one-line messages with URLs as ‘the spam of the future’, yet we have shown that these emails can be enhanced with mined contextual information in a believable fashion, and this is only the tip of the iceberg.

The closest work to ours is the ‘context-aware’ phishing attacks suggested by Jakobsson (2005). There, a target is tricked into believing a phishing email by exploiting context information, in conjunction with a well-timed attack. While he gives some specific phishing-related examples, there is no scheme proposed for automating such attacks or generalizing them beyond phishing.

Some facets of our approach have appeared independently, too. Parasitic spam (Swimmer, 2005) would have zombies tag spam onto legitimate outgoing email messages; FitzGerald (2005) mentioned zombies making use of a victim's email program settings.

Conclusion

More effective spam can be sent by using malware on zombie machines to mine data from email corpora. This allows spam to be sent that automatically mimics legitimate email sent by the real owners of the zombie machines, and our proof-of-concept implementation demonstrates that the result can be convincing even to seasoned users. While this more effective spam has not, to our knowledge, been seen in the wild, there are defensive steps that can be taken now to limit its impact when this spam makes its debut.

Acknowledgment

The work of both authors has been supported by grants from the Natural Sciences and Engineering Research Council of Canada. Thanks to the anonymous referees, whose comments were helpful in improving this paper.

References

AtomPark Software (2005). Email extractor [Computer software]. Retrieved 13 January, 2006, from http://www.massmailsoftware.com/extractweb.

CALO Project (2004, March 2). Enron email dataset. Retrieved 1 March, 2006, from http://www.cs.cmu.edu/~enron/enron_mail_030204.tar.gz.

Culotta, A., Bekkerman, R., & McCallum, A. (2004). Extracting social networks and contact information from email and the Web. In First Conference on Email and Anti-Spam.

Delany, M., ed. (2005). Domain-based email authentication using public-keys advertised in the DNS (DomainKeys). Internet draft; work in progress.

Dwork, C., & Naor, M. (1993). Pricing via processing or combatting junk mail. In Proceedings of CRYPTO ’92, pages 139–147.

FitzGerald, N. (2005). Why ‘user authentication’ is a bad idea. In Virus Bulletin Conference, page 226. Abstract only; the comment we refer to was made during his conference presentation.

Gladwell, M. (2002). The Tipping Point: How Little Things Can Make a Big Difference: Little, Brown and Company.

Graham, P. (2002). A plan for spam. Retrieved 7 January, 2006, from http://www.paulgraham.com/spam.html.

Graham-Cumming, J. (2003). Fooling and poisoning adaptive spam filters. Sophos white paper.

Graham-Cumming, J. (2004). How to beat an adaptive spam filter. In MIT Spam Conference.

Jakobsson, M. (2005). Modeling and preventing phishing attacks. In Financial Cryptography ’05. Phishing panel.

Levine, J. R. (2005). Experiences with greylisting. In Second Conference on Email and Anti-Spam.

McAfee, Inc. (2005). W32/Mydoom.bb@MM. In McAfee Virus Information Library. Retrieved 2 March, 2006, from http://vil.nai.com/vil/content/v_131856.htm.

Milgram, S. (1967). The small-world problem. Psychology Today, pages 61–67.

Simsek, Ö. & Jensen, D. (2005). Decentralized search in networks using homophily and degree disparity. In Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, pages 304–310.

Smith, M. A., Ubois, J., & Gross, B. M. (2005). Forward thinking. In Second Conference on Email and Anti-Spam.

Spammer-X (2004). Inside the Spam Cartel: Syngress.

Staniford, S., Paxson, V., & Weaver, N. (2002). How to 0wn the Internet in your spare time. In Proceedings of the 11th USENIX Security Symposium.

Swimmer, D. (2005). The spectre of parasitic spam. Virus Bulletin, pages S2–S4.

Travers, J., & Milgram, S. (1969). An experimental study of the small world problem. Sociometry, 32(4), 425–443.

Tyler, J. R., & Tang, J. C. (2003). When can I expect an email response? A study of rhythms in email usage. In Proceedings of the Eighth European Conference on Computer Supported Cooperative Work, pages 239–258.

Wayner, P. (2002). Disappearing Cryptography: Morgan Kaufmann, second edition.

Williamson, M. M. (2003). Design, implementation and test of an email virus throttle. In 19th Annual Computer Security Applications Conference.

Wong, M., & Schlitt, W. (2005). Sender policy framework (SPF) for authorizing use of domains in E-MAIL, version 1. Internet draft; work in progress.

Zdziarski, J. A. (2005). Ending Spam: No Starch Press.