The Tale of the Weather Worm

Joe Szabo, John Aycock, Randal Acton, and Jörg Denzinger
Department of Computer Science, University of Calgary
2500 University Drive N.W., Calgary, Alberta, Canada T2N 1N4
{szaboj,aycock,acton,denzinge}@cpsc.ucalgary.ca

ABSTRACT

How humans behave when faced with a disaster, natural or man-made, can be exploited automatically by news-aware malicious software. We introduce *weather worms*, worms that can automatically identify abnormal events and their location, and target computers at that physical location. Such worms could be used to take advantage of poorly-defended computers in a disaster zone, and could amplify the effects of a physical attack. Defenses against weather worms require examination of policy and presentation of information on the Internet.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*invasive software*; K.4.1 [**Computers and Society**]: Public Policy Issues—*human safety*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*invasive software*

General Terms

Security, Human Factors

Keywords

network security, worms, emergencies, disasters, geolocation, information policy

1. INTRODUCTION

The role that humans play in computer security is well-known. Humans are frequently touted as the weakest link in security, and there are always risks of insider attacks, social engineering, and misconfigured and unpatched machines leading to vulnerabilities. These risks result from the behavior of individual humans.

What is not well-known is the effect of large-scale human behavior on security, behavior that large groups of people naturally exhibit at the same time. In previous work, we examined how worldwide combinations of normal business hours and public holidays could yield large windows of vulnerability [8]. This idea can be extended further, as people are also distracted *en masse* by abnormal events like severe weather.

For example, a blizzard could occur in a location which is ill-prepared to deal with one – a blizzard in New York might have the effect of forcing people to stay home for several days. By staying home, people who are normally on-site, maintaining and patching corporate computers may be less able to do so, and their ability to respond to an attack may be limited as well. Other events might have similar effects; tornadoes, tsunamis, earthquakes, riots, and even transit strikes could result in similar circumstances, either directly by preventing access to computers or indirectly by providing considerable distraction. More sinister is the threat of terrorism, where a physical attack is accompanied by a virtual attack, automatically targeting computers in the affected area.

Obviously these attacks would not always be feasible. For example, a severe weather event may cut power and communications to potential targets (although some large data centers may have their own power sources). This is also recognized in military operations, where a physical attack destroys the infrastructure required to conduct electronic warfare [10]. However, relying on the loss of power and communications as a defense is hardly prudent, and the possibility of the attacks we describe must be considered seriously.

In this paper, we present *weather worms*, worms which are able to automatically detect disruptive events like severe weather, and locate and attack computers in the region. The novel parts of a weather worm, which are not malicious by themselves, have been tested as proofs of concept.

We stress that even though we use weather as a running example, the worm mechanism we describe is *not* limited to this scenario. Any abnormal event that affects humans can be automatically targeted, a vital point because an effective response to an public emergency relies on communication. A worm attacking already-stressed infrastructure would only add to the havoc, especially as our dependence on the Internet increases, such as VoIP phone services. While these sorts of attacks could be performed manually, the automation we demonstrate allows the attacker to be separated from the attack itself – in some ways, it is a sophisticated form of a logic bomb. The automation is also a logical step in the progression of software tools, both legitimate and malicious, which in the case of a worm makes the technology increasingly available to "script kiddies."

SAC'08 March 16-20, 2008, Fortaleza, Ceará, Brazil

Sections 2 and 3 discuss weather worm implementation and defense, respectively. Related work is discussed in Section 4; Section 5 concludes.

2. WEATHER WORMS

A "traditional" worm can be thought of as having four main components: target identification (often called scanning), target infection, trigger, and payload. A weather worm is a variant of a traditional worm, which may manifest itself in two main ways:

Event-based target identification. After its initial seeding, a weather worm would not spread further, but wait for a major event to occur. Once a major event happened, the weather worm would try to infect machines geographically-located in the affected area. This type of weather worm would try to take advantage of machines being less-maintained while people are distracted by an abnormal event.

Event-based trigger. A weather worm could identify and infect targets through normal means; infected machines would look for a major event to trigger the payload. The payload itself, possibly a DDoS attack, would be focused on machines in the geographic area where the major event occurred. Weather worms of this type would be trying to add to the confusion of a major event by disrupting Internet traffic in the region. As a special case, one specific physical location (e.g., a world capital) could be targeted by looking for an event only in that location. DDoS attacks like the recent ones in Estonia [23] could be launched automatically by weather worms looking for rioting events.

It is important to note the distinction between weather worms and malware like Small.DAM, a.k.a. Storm Worm [7]. In Small.DAM, manually-constructed email subject lines, one even related to a severe storm in Europe [19], are used to entice users to run a malicious email attachment. A weather worm, in contrast, *automatically* identifies important news events and their location, and *automatically* finds targets at the location to attack.

There are two novel technical aspects to a weather worm. First, Internet information sources are used to identify events and locations, resulting in what we call *news-aware* malicious software. Second, IP addresses of potential targets in a given location are found through a process we call *inverse geolocation*. In both cases, we have looked for lightweight solutions that might be used by a worm author, which gives us some insight into the possible scope of such a threat and effective defenses. This section discusses these lightweight solutions in detail.

2.1 Identifying Events and Locations

The Internet offers a plethora of data sources that might be useful for identifying events and locations, though not all sources are suitable for this particular task. If we take weather as an example, depending on the location, the same two weather events may have a drastically different impact. A blizzard can be a regular occurrence in northern Canada, and the event would have far less impact than the same blizzard occurring in New York. While online weather sources would report both events, news sources would only choose to report the more significant of the two, making news services a better source for this particular application. In addition, using multiple news sources allows measurement of the importance of a particular event. A large number of news sources reporting on the same event might indicate that the event in question is actually very important and could possibly impact a large number of people. News sources also permit the identification of events to extend beyond weather. We see malicious software that makes use of news sources, i.e., news-aware malicious software, as a potential threat to computer security.

In general, our problem of event identification and location is an information retrieval application (see [13]). For our purposes, information retrieval is an area that tries to achieve some limited understanding of (English) texts without doing any real natural language processing. In the following, we describe a system that uses only straightforward instantiations of information retrieval techniques that are easy to implement. While these instantiations come with known problems, more sophisticated techniques that do not suffer from these problems may require more heavyweight implementations.

Our prototype system, NewsWatcher, requires input from three separate files: a "seed" URL list, an event list, and a location list containing city names. The seed URL list specifies starting URLs where NewsWatcher can begin its news search.[1] The event list provides event names for the system to search for, and match to locations provided in the location list.

2.1.1 URL Processing

NewsWatcher operates by processing a queue of URLs. Initially, the queue consists of the seed URLs. Each URL in the queue is processed in three steps:

Step 1: downloading. The URL content is downloaded, but only if the content size is under 150 K; this prevents NewsWatcher from downloading large media files instead of web pages. HTML metadata is discarded, because it was found to list many event types that would otherwise cause false matches.

Step 2: link detection. Each HTML page is scanned for links to add to the URL queue. Some links are ignored outright, like self-referencing URLs and URLs that use JavaScript. URLs are also ignored that have already been seen by NewsWatcher.

Subject to these constraints, NewsWatcher traverses all links found on the seed URL pages. Links on subsequent pages are only followed if the (human-readable) URL link text contains an event name. This allows traversing a second level of links, but only when the link appears to be of interest.

At no time does NewsWatcher follow more than two levels of links from a seed URL, and may only traverse one level deep. This happens for two reasons. First, important news stories will not be buried in a news website. Second, our experiments showed that regularly traversing beyond one level of links yielded an unmanageably large number of new URLs.

Step 3: event searching. The HTML page is searched for all events provided in the event list. If an event is found

[1]RSS feeds from news sources could be used similarly, but would present the same problems identifying events and their locations; we thus omit further mention of RSS without loss of generality. Also, some news sources' RSS feeds (e.g., `cnn.com`) regularly include the news headline but no article content, so a web page fetch would be needed anyway.

in the page, NewsWatcher defines a *search field* around that event name, 100 characters on each side, in which it searches for any matching location names. If periods are found within the search field on either side of the event name, then the search field is truncated, because we only want to examine the sentence in which the event name is found. A location found in the search field causes the event and its location to be stored for later output.

Once these steps are complete, the URL is fully processed, and NewsWatcher continues onto the next queued URL until the queue is empty.

2.1.2 Problems

Several problems were identified during the design and development of NewsWatcher.

Problem 1. *How is the system to discern whether a news story is reporting an event that is currently happening, or that had occurred a year earlier (i.e., a retrospective anniversary)?*

This issue was mostly addressed by ignoring event matches whose corresponding search field were found to contain the words "ago" or "anniversary." Further problems of this nature were also averted by ignoring URLs that contained the string "archive." This problem is difficult to overcome entirely, as some news stories reference prior events while reporting current events.

Problem 2. *In cases such as hurricanes, where news services are able to report the event long prior to it actually occurring, will it be possible to distinguish between event buildup and the event itself?*

Over the course of an event, we found that the output from the prototype can be used to identify event buildup and cool-down periods. Events that started with one or two URL matches would have many more URL matches in later runs. Later still, there would again be few URL matches for the event.

This can be partially addressed using thresholds (Section 2.1.3). Without examining web pages for event occurrence times and dates, it may not be generally possible to identify such periods in single runs of NewsWatcher without jeopardizing its lightweight nature.

Problem 3. *Name confusion has to be accounted for. The system must be able to discern whether names found in news stories such as the Carolina Hurricanes or Calgary Flames (both professional hockey teams) are actually events.*

An easy solution is to require an exact match with the names in the event list and to avoid the use of event names like "Hurricanes" or "Flames."

Problem 4. *The system must be as small and lightweight as possible, to compare meaningfully to one that might be bundled in the (small) package of a worm.*

The set of input files for NewsWatcher contains 6416 location names, 38 event names, and 24 starting URLs. The total size of these text files is under 64 K, 29 K if compressed. The size can be reduced further by having fewer locations in the location list; in any case the size is insignificant.

Problem 5. *Events can span many degrees of severity. Is it possible for the system to be able to determine if an event is still in its infancy, or if an event is too great that all computers in the corresponding location would be either destroyed or unable to infect because of the extreme circumstances? In both cases, having the event reported would be of little use.*

This problem is very similar to that of identifying event buildup and cool-down periods and those observations also apply here.

To further address the issue, some modifications to NewsWatcher could be made. It is possible to have an additional input list containing "blacklist" events. Finding a blacklisted event would cause any other event matches in the search field to be ignored.

Suppose "volcano erupted" was contained in the event list and "buildings destroyed by lava" was contained in the event blacklist. If a news article read "the volcano erupted on Tuesday, and the resulting devastation left hundreds of buildings destroyed by lava," the "volcano" event would be ignored. Similarly, if "hurricane" was in the event list and "expected to arrive" were in the event blacklist, no events would be matched in the text "hurricane Bob is expected to arrive next week."

Problem 6. *Some event names are also names of certain objects, and in some contexts it can be impossible to distinguish between actual events and other non-related reports.*

Problem 7. *Some event names can be used to metaphorically describe totally unrelated events.*

The last two problems have been encountered during testing. Consider this actual news headline:

> 'War games: Tornado jets zoom out of Calgary
> as British troops train for combat' [6]

from which NewsWatcher identifies the event "Tornado" at location "Calgary," and

> 'Tuesday's electoral earthquake triggered an equally
> seismic reaction in Washington yesterday, . . . ' [3]

mistakenly finds an "earthquake" in "Washington."

Though a possible solution might involve an event blacklist (blacklisting "jets" would have prevented the first problem), it might be necessary to use more complicated sentence or pattern recognition techniques from information retrieval (see [13]). As for metaphors, even relatively recent work in computational linguistics 'is far from being able to recognize metaphoric language in general' [14, p. 43].

2.1.3 Interpreting NewsWatcher Output

NewsWatcher's output is a list of triples, abstractly speaking, where each triple identifies an event, a location, and the number of URLs the match was found at. How should this output be interpreted?

If a greater number of event matches are desired, at the expense of some incorrect matches, the most straightforward interpretation of the output is to simply regard the events reported as correct and target all of the locations listed in the output. Unfortunately, NewsWatcher is not accurate enough to allow its output to be safely regarded in this way. For example, "violence in Washington" showed up frequently, but these were news articles reporting on people in Washington discussing violence elsewhere in the world.

While some incorrect matches can be tolerated, it is generally the case that very few URL matches for an event implies the event itself is either not very important, or is not an actual event and may have been erroneously detected.

To address this problem, a threshold value can be selected: only the events having a number of URLs higher than the

Table 1: Top three events by URLs matched

	Event	Location	# URLs
Slow	tornado	Carolina	17
news	violence	Baghdad	16
day	floods	Kabul	11
Moderate	flooding	Nairobi	38
news	violence	Washington	21
day	violence	Baghdad	19
Busy	fires	Oaxaca	60
news	violence	Baghdad	36
day	violence	Washington	25

threshold value would be considered correct. Choosing a single, static threshold value is difficult, though, because there is a normal variation in news volumes. Intuitively, this can be thought of as how "busy" the news day is.

Table 1 illustrates the difference in terms of the number of URLs reported. A static threshold of 12 would eliminate much of the noise regardless of news volume. Unfortunately, a static threshold might result in very few event matches and might actually filter out wanted data. Using the "slow news day" as an example, a threshold of 12 eliminates the "floods for location Kabul" event (because only 11 URLs were found reporting the event), whereas this event is detected correctly and is potentially of interest.

An alternative to using a static threshold is implementing a dynamic threshold with respect to the output, such as only using the top k triples with the largest number of URLs. Again, this has problems, because even the most-reported events on a slow news day are probably not cataclysmic enough to result in the human distraction the weather worm tries to take advantage of.

We conjecture that the likeliest implementation would use a combination of static and dynamic thresholds. A static threshold would help filter spurious matches and weed out correctly-detected but minor events; this would probably be no lower than 50 URLs. A dynamic threshold would pick out the top remaining major events, if any. Invariably some tuning of the static threshold would be necessary.

NewsWatcher clearly demonstrates that it is possible to detect news events and their locations in a lightweight fashion. While its detection is not perfect, malware does not need to operate with complete precision, and we conclude that this aspect of a weather worm is technically feasible.

2.2 Inverse Geolocation

By identifying events and locations, a worm author could build an automated version of the "Storm Worm" [7]. No need to manually distribute worm variants with new, enticing headlines: the worm itself would identify important headlines and modify its infectious emails accordingly. Indeed, this is a possible next step in worm evolution.

The situation is worse, however, if a full weather worm is developed. This would allow worm activity to be targeted to an event-affected region using information that is already available to worm authors. We have published this work elsewhere [1], but we summarize the results here for completeness.

Geolocation is the process of mapping an IP address into a physical location. We refer to the inverse process as *inverse geolocation*: mapping a physical location into IP addresses. In terms of a weather worm, when an event is detected and located, what are the IP addresses of potential targets?

There are two main ways we have demonstrated that inverse geolocation is possible:

1. A database intended for geolocation was transformed for inverse geolocation. By permitting small errors (which would be acceptable for malware), and limiting the locations to cities with over 1 M people, our inverse geolocation database was just over 700 K compressed.
2. Large lists of IP addresses for specified locations were produced using well-known Internet services: a search engine and the *whois* database.

Using either approach, a weather worm could build a list of potential targets in a relatively lightweight fashion. We conclude that this aspect of a weather worm is also technically feasible.

3. WEATHER WORM DEFENSES

There are four primary avenues to consider for defending against weather worms and other news-aware malicious software.

First, there is the usual suspect. Assuming a weather worm sample had been acquired, anti-virus software is probably the only defense common enough to eradicate an established weather worm prior to it triggering.

In the event of a highly-targeted attack by news-aware malicious software, such as coordinating virtual and physical terrorist attacks, capturing a sample before the triggering event may be critical. A sample may reveal the site of a planned physical terrorist attack by virtue of an unusually-short location list, allowing proper precautions to be taken.

Second, reliance on human maintenance of computers must be reduced. For example, automatic patching of machines might be considered [24]. Normally this is suggested because of concern over worms which spread extremely quickly, like "flash worms" [22, 21]; humans are unable to react to such worms in time. The weather worm concept suggests that less reliance on humans is good not because humans can't react in time, but because they can be distracted by a major event and rank other priorities (e.g., family, personal survival) more highly then computer security.

Third, the weather worm can be prevented from gathering the information it needs to operate. The weather worm primarily draws upon news web sites; HTTP access is extremely unlikely to be blocked, and there are no telltale search engine queries to detect and block either, so any defense has to be from the information source itself.

Information sources can be blocked from fully-automated use. This is currently accomplished using CAPTCHAs [25], tests which are easy for a human to solve but hard for a computer to solve, in theory. But use of CAPTCHAs for accessing news websites would be incredibly annoying and would break legitimate automatic news programs, so we do not see CAPTCHAs being an effective defense.

The results from information sources can be made harder to interpret automatically. As Section 2.1 alluded to, the current version of NewsWatcher has limitations that render it ineffective against web pages having any of the following characteristics:

- The page identifies event and location names in separate sentences, or at least 200 characters apart.

- The page contents are greater than 150 K in size (larger files are ignored by NewsWatcher).
- Event or location names are spelled with s p a c e s in them, or spelled incorrectly.

These are defenses that exploit specific, known aspects of NewsWatcher. To defend against news-aware malicious software in general, as a news source, the following techniques can be useful:

- Using frames to display the event and location names from distinct HTML source pages.
- Displaying data in a non-text form, representing events and locations using image files or Flash.
- Using any form of HTML obfuscation [9].
- Changing the words and phrases used to describe events, and using more ambiguous words and metaphor.

But all these techniques come with costs and there is already known research in information retrieval that can make the techniques useless. The first three general techniques result in news pages that are larger, thus putting more strain on networks. In addition, they make it more difficult for the human user to use tools that help with the classification of news for non-malicious purposes. This is also true for the fourth general technique, which might make it difficult for the user to understand the information easily. In fact, a common expectation of human users is to get news in a concise, clear, and easy-to-understand form from news web sites.

Technical and usability considerations aside, there is currently no compelling business reason for news sources to alter their output to be hostile to weather worms. We therefore would not rely on this as a defensive measure in practice.

The fourth and final avenue of defense is on the side of the targeted computers. Countermeasures must include the possibility of a news-aware attack in the emergency plans that people, businesses, and other organizations have. If it is not possible to shut down the machines during the period of an emergency then the following measures could be taken:

- Transfer necessary offered services to better supervised computers in unaffected areas.
- Arrange for intensified remote supervision of the computers in the emergency area.
- Use stricter security policies for the duration of the emergency that might deny some legitimate requests but block the attack or its effects.

A thorough disaster recovery plan should require little change to adapt to news-aware malicious software attacks. The key realization is simply that such attacks are possible, and to be prepared for them.

4. RELATED WORK

The related work can be divided into three parts. We discuss work related to NewsWatcher and the overall idea of weather worms in this section; work related to inverse geolocation may be found in [1].

4.1 NewsWatcher-Related Work

NewsWatcher can be viewed as performing very simplified web crawling, related to Internet search engines like Google (see [4]). NewsWatcher can also be seen as computing a crude news summary, and [16] provides users with summaries of news articles.

NewsWatcher achieves the coordination of a group of malicious programs by having them process the same news information. The potential of using the Internet for coordinating software was suggested in [12]. NewsWatcher adds to this by not requiring precisely-defined event descriptions, essentially using event types and being opportunistic in its target selection. That communication between programs can be avoided if the programs share the exact same information about their (perceived) environment and use the same decision making algorithms was pointed out in [18].

Several techniques have been previously suggested for identifying real-world locations referred to by documents, including web pages. Special attention is paid to resolve place name "disambiguation" [27]: resolving the difference between ambiguous place names such as the city named Batman in Turkey, or the Batman comic-book character. This is also referred to as geo/non-geo ambiguity [2]. To further complicate the problem, one must also address geo/geo ambiguity, or the ambiguity caused by different places sharing the same name. Another closely related issue is 'the problem of indexing and navigation of web resources by geospatial criteria,' [15, p. 221] and the problem of location aliasing (for example, L.A. instead of Los Angeles).

NewsWatcher's lightweight approach precludes carrying the amount of data or performing the level of computation required by other proposed solutions. The NewsWatcher prototype location list contains only locations having large populations, which removes most ambiguous location names. Beyond that, we found that the best way to address ambiguity was to manually remove offending location names, making these locations immune to NewsWatcher.

The problem of geo/geo ambiguity is ignored completely; NewsWatcher is concerned with gathering location *names*, and not the actual whereabouts of the location in question. The location aliasing problem rarely occurs on news web sites; during prototype testing, news sites that were encountered always tended to use the full location names (although state names were commonly abbreviated).

4.2 Weather Worm-Related Work

Byers et al. describe a specific attack scenario through/on the physical postal service, launched from the Internet using publicly-available information [5]. Although they do not cause a direct physical effect, weather worms are an example of an attack using publicly-available information, which can focus on Internet-only targets, or amplify the effect of preexisting physical events.

HTML obfuscation is of course well-known to spammers [20]. Using HTML obfuscation and CAPTCHAs as a *defense* has been suggested previously [5, 12], although we doubt their overall usefulness against weather worms.

The abuse of search engines and the information they provide was pointed out by Byers et al. [5]. Weaver et al.'s worm taxonomy noted the possibility of using search engines to identify vulnerable targets [26], an idea expanded upon by Provos et al.'s "search worms" [17]. We know of only one example of this occurring in the wild: Santy [11]. However,

Santy and the previous work deals with searching for targets exploitable using some specific vulnerability.

While worm activation through human activity has been mentioned [26], the activities noted were those of individual humans. The only work on large-scale human behavior we know of is our own [8].

5. CONCLUSION

The weather worm shows that news-aware malicious software is possible: software that automatically notes a major event, identifies the location, and finds targets at the physical location of the event. Human behavior, the mass distraction of a major event, might thus be exploited maliciously.

While defenses, both specific and general, can be deployed by information providers, it is unlikely that they will be deployed unless news-aware malware becomes a frequent threat. Instead, the onus is on public- and private-sector organizations – potential targets – to incorporate countermeasures into their disaster plans.

Generally, we are forced to conjecture that any information provided for legitimate purposes can be used for malicious purposes. The very information and infrastructure we rely upon as a society is the information and infrastructure through which society can be attacked.

6. ACKNOWLEDGMENTS

The second author's research is supported in part by a grant from the Natural Sciences and Engineering Research Council of Canada. Nathan Friess performed the inverse geolocation database experiments.

7. REFERENCES

[1] R. Acton, N. Friess, and J. Aycock. Inverse geolocation: Worms with a sense of direction. In *Malware '07*, pages 487–493, 2007.

[2] E. Amitay, N. Har'El, R. Sivan, and A. Soffer. Web-a-where: geotagging web content. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and development in Information Retrieval*, pages 273–280, 2004.

[3] D. Balz. For Bush's new direction, cooperation is the challenge. http://www.washingtonpost.com/wp-dyn/-content/article/2006/11/08/AR2006110800489.html, Nov. 2006. Page as of 11-11-06.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[5] S. Byers, A. D. Rubin, and D. Kormann. Defending against an Internet-based attack on the physical world. *ACM Transactions on Internet Technology*, 4(3):239–254, 2004.

[6] Calgary Herald. War games: Tornado jets zoom out of Calgary as British troops train for combat. http://www.canada.com/calgaryherald/news/city/-story.html?id=32dad626-2a7d-497b-a369-64e2f68b91a3, Nov. 2006. Page as of 11-11-06.

[7] F-Secure. Small.DAM. F-Secure Trojan Information Pages, 17 January 2007.

[8] N. Friess, R. Vogt, and J. Aycock. Timing is everything. *Computers & Security*, 24(8):599–603, 2005.

[9] J. Graham-Cumming. The spammer's compendium. http://www.jgc.org/tsc/.

[10] Headquarters, Department of the Army. Information operations. Field manual No. 100-6, 27 August 1996. United States Army.

[11] M. Hyponnen. F-Secure virus descriptions: Santy, 2004.

[12] H. H. Lee, E.-C. Chang, and M. C. Chan. Pervasive random beacon in the Internet for covert coordination. In *Information Hiding, 7th International Workshop, IH 2005 (LNCS 3727)*, pages 53–61, 2005.

[13] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2007.

[14] Z. J. Mason. CorMet: A computational, corpus-based conventional metaphor extraction system. *Computational Linguistics*, 30(1):23–44, 2004.

[15] K. S. McCurley. Geospatial mapping and navigation of the Web. In *Proceedings of the 10th international conference on World Wide Web*, pages 221–229, 2001.

[16] K. McKeown and D. R. Radev. Generating summaries of multiple news articles. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 74–82, July 1995.

[17] N. Provos, J. McClain, and K. Wang. Search worms. In *WORM '06*, pages 1–8, 2006.

[18] D. V. Pynadath and M. Tambe. Multiagent teamwork: Analyzing the optimality and complexity of key theories and models. In *Proceedings AAMAS 2002*, pages 873–880, Bologna, 2002.

[19] D. Rising. Storms in Europe kill 46, disrupt travel. Associated Press, 19 January 2007.

[20] Spammer-X. *Inside the SPAM Cartel*. Syngress, 2004.

[21] S. Staniford, D. Moore, V. Paxson, and N. Weaver. The top speed of flash worms. In *Proceedings of the 2004 ACM Workshop on Rapid Malcode*, pages 33–42, 2004.

[22] S. Staniford, V. Paxson, and N. Weaver. How to 0wn the Internet in your spare time. In *Proceedings of the 11th USENIX Security Symposium*, 2002.

[23] The Economist (U.S.). A cyber-riot; Estonia and Russia. 383(8528):55, 2007.

[24] M. Vojnović and A. Ganesh. On the effectiveness of automatic patching. In *Proceedings of the 2005 ACM Workshop on Rapid Malcode*, pages 41–50, 2005.

[25] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In *Proceedings of EUROCRYPT 2003 (LNCS 2656)*, pages 294–311, 2003.

[26] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham. A taxonomy of computer worms. In *WORM '03*, pages 11–18, 2003.

[27] W. Zong, D. Wu, A. Sun, E.-P. Lim, and D. H.-L. Goh. On assigning place names to geography related web pages. In *JCDL '05: Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 354–362, 2005.