Using Low-Interaction Honeypots to Study the Abuse of Open Proxies to Send Spam

KLAUS STEDING-JESSEN[1]
NANDAMUDI L. VIJAYKUMAR[2]
ANTONIO MONTES[3]

[1]Brazilian Network Information Center - NIC.br
Computer Emergency Response Team Brazil - CERT.br, São Paulo (SP)
jessen@cert.br
[2]National Institute for Space Research - INPE
Computing and Applied Mathematics Associated Laboratory - LAC, São José dos Campos (SP)
vijay@lac.inpe.br
[3]Ministry of Science and Technology - MCT
Renato Archer Research Center - CenPRA, Campinas (SP)
antonio.montes@cenpra.gov.br

Abstract. One of the main problems in creating effective mechanisms to mitigate the spam problem is the lack of more precise data on it. This paper describes the design and implementation of a honeypot-based architecture to study the abuse of open proxies to send spam. The sensors were installed in Brazilian broadband networks and they captured more than 500 million emails over a period of 15 months. In this paper we present the analysis of these data and describe some contributions to the spam capture field that were implemented.

Keywords: network security, honeypot, spam, open proxy.

(Received November 01, 2007 / Accepted December 04, 2007)

1 Introduction

Spam is one of the Internet abuses that has increased the most, being responsible for a significant amount of the total number of emails in circulation today [10, 21]. Spam has also been used for sending phishing-related messages (an email trying to induce an user to give personal of financial data) and for disseminating malicious code [22].

The word spam is used to describe the act of sending unsolicited or inappropriate messages, specially in bulk and with commercial content [10, 28, 7]. The term became popular to describe sending email in bulk quantities after an incident in April 1994, when an unsolicited message was sent to six thousand Usenet Newsgroups [31, 10]. It is important to note, however, that the concern about unsolicited emails is not new and dates back to 1975 [24].

With the growth of the Internet popularity, the access to email, initially restricted to smaller groups like the academic community, grew considerably among the general public. And, because the cost of sending emails is very low, compared to the cost of regular mail, this has been a major incentive for sending commercial emails in huge quantities [2].

The spam software efficiency has also increased, making existing spam blocking techniques even less effective and making the spammer (person who sends spam) tracking more difficult [4, 22]. As an example, the usage of machines infected by malicious code like bots (program that is able to propagate itself by infecting vulnerable computers and can be controlled remotely by an attacker) for sending spam and phishing [22] is increasing, allowing spammers to remain anonymous.

Currently, one of the main problems in developing effective mechanisms to mitigate spam is the lack of more precise data about the dimension of the problem and about the methods in use for the dissemination of spam and malicious code.

Although there is a notion that spam corresponds to a significant amount of the total number of emails in circulation and plays an important role in the dissemination of malicious code and phishing, most public numbers come from informal research or from companies selling anti-spam products or services [10].

This paper describes the design and implementation of an architecture to study the spam problem, in particular the abuse of open proxies. Open proxies have traditionally been used for sending spam and perpetrating other malicious activities [11, 17]. This kind of abuse happens mainly in end-user machines, connected via broadband, and with the potential of being infected by malicious code and controlled remotely to send spam [14, 27, 12].

The architecture implemented is comprised of a set of sensors based on honeypots. The data captured was then collected by a central server. This paper also describes the extensions implemented in Honeyd [25] in order to expand the protocols supported and the recorded information.

The sensors were installed in Brazilian broadband networks and captured more than 500 million spams over 15 months. The data captured allowed a better analysis of the spam problem and also a better understanding of how spam is being sent. This analysis also helps the creation of better mechanisms to fight spam.

This paper is organized as follows. In section 2 some concepts about honeypots and abuse of services for sending spam are explained. In section 3 we discuss some related work using honeypots to capture and study spam. In section 4 we describe the architecture implemented for spam capture and the services being emulated on the honeypots. We also present in this section the results obtained. Conclusions and future work are discussed in section 5.

2 Definitions

In this section we review some concepts in the honeypot area, as well as abuse of services for sending spam.

2.1 Honeypots

A honeypot is a security device designed to be probed, attacked or exploited [29, 26].

The value of honeypots relies on the fact that everything observed is suspicious and potentially malicious. In this way, they are a security tool with a low number of false-positives and that can provide high value information. They also generate a smaller amount of data to be analyzed, specially if compared to other more traditional security tools like Intrusion Detection Systems (IDS).

Based on the access level to the system and the resources an attacker is given, honeypots can be classified as low-interaction and high-interaction [26, 13].

2.1.1 High-Interaction Honeypots

In high-interaction honeypots attackers interact with real operating systems, services and programs. Once compromised, these honeypots are used to observe the attackers behaviour, their tools, motivations and explored vulnerabilities. This kind of honeypot must have a robust containment mechanism in order to prevent, once compromised, its use to attack other networks [30, 26].

This type of honeypot is justifiable when the main objective is to study the attackers behaviour, their motives and also to study, in detail, the tools being used and the vulnerabilities being exploited. Their use, however, is time consuming and requires good containment techniques, as well as specialized personnel to operate them.

2.1.2 Low-Interaction Honeypots

On low-interaction honeypots, tools are installed in order to emulate operating systems and services and then interact with the attackers and malicious code. This kind of honeypot has a small chance of being compromised and is ideal for production networks, when there are no available personnel to administer a honeynet or when the risk of a high-interaction honeypot is not acceptable.

Typical use of low-interaction honeypots include: port scan identification, generation of attack signatures, trend analysis and malware collection [26].

In this paper we will focus on using low-interaction honeypots emulating open proxy and open relay machines to capture spam.

2.2 Abuse of Services for Sending Spam

The SMTP protocol (Simple Mail Transfer Protocol) is usually associated with port 25/TCP (Transmission Control Protocol), and its main goal is the email transport in a reliable and efficient way [16]. An SMTP relay is a system that receives an email from an SMTP client and retransmits it to another SMTP server for its final delivery or for additional transmission. Normally, this relay service is selective and only retransmits emails for authorized clients. Misconfigured SMTP servers, usually called open relays, allow the delivery of messages from any source to any recipient [19]. This type of server is abused to send spam because it makes the detection of the real origin more difficult. It can also be used to bypass mail blocking lists.

A proxy is a server that works as an intermediary between a client and another server. Normally it is used to access certain services with increased performance or to allow more than one machine to connect to the Internet sharing the same IP (Internet Protocol) address. A proxy acts as a intermediary, making connections on behalf of other clients [5].

A misconfigured proxy allows connections to be made from any origin to any destination IP address or port. This type of proxy is called an open proxy. Open proxies are also intentionally installed by malicious code, bots and trojan horses.

Spammers continuously scan the Internet searching for open proxies [17]. Once located, these computers are then used to make connections to the spam recipients' SMTP servers. These open proxies are used to deliver spam in a more anonymous way [3, 17].

3 Related Work

In this section we discuss related work using honeypots to study or fight spam.

3.1 Honeyd

`Honeyd` [25] is a network daemon which implements a framework for creating virtual honeypots with the purpose of emulating several computers, operating systems and network services.

This software has become very popular to implement low-interaction honeypots. It can also be used, although in a limited way, for the study and prevention of spam. Two `Honeyd` modules can be used in this way:

Open relay emulator: it emulates the characteristics of an open relay SMTP server. The email messages received are all stored and never delivered to its recipients.

Open proxy emulator: it emulates the behaviour of an open HTTP (Hypertext Transfer Protocol) proxy, implementing a subset of the HTTP protocol [5], in particular the GET and CONNECT commands. GET requests are answered with a generic or error page in HTML (hypertext Markup Language). Attempts to use the CONNECT command to connect to the SMTP port of a remote server are redirected to a local SMTP emulator, described above.

By using these two emulators it is possible to capture emails for further studies and also for sending them to collaborative projects creating spam filter signatures [25].

The modules mentioned above, however, are restricted to HTTP proxy emulation. This fact severely impacts the amount of emails that can be captured. A contribution of this work was the development of a SOCKS proxy emulator to be used with `Honeyd` to capture spam. The details of this new module can be seen in section 4.

3.2 HoneySpam

HoneySpam [1] is a framework based on honeypots that has the objective of fighting spam at its origin, instead of doing so in the destination. The system architecture is designed to be used as a spam identification and blocking tool on a corporate network. Its main functionalities are described below.

Harvesting interference: harvesting is a process of collecting valid email addresses in Web pages. This module creates Web pages with a large number of links between them, slowing down the process.

Block list creation: the HTML pages generated in the previous module contain email addresses dynamically created pointing to a local SMTP server. As soon as these addresses start receiving emails, this information can be used to create email block lists based on the source IP addresses sending spam.

Another objective of generating invalid, dynamically generated email addresses is to pollute the spammer's email databases, decreasing its usefulness.

Open proxy and open relay emulation: it also uses the open relay and open HTTP proxy implemented by `Honeyd` [25], described in section 3.1. Connections to these emulated services are registered and used to block traffic from the source IP addresses.

Because this framework main goal is to block spam using filters, it is not the right tool to collect spam for further studies.

4 Honeypot Network to Capture Spam

In this section we present the architecture used to implement a set of sensors, based on honeypots, to capture spam. We also present some preliminary results based on the analysis of the data collected.

4.1 Architecture

The implemented architecture, shown in Figure 1, used 10 low-interaction honeypots for capturing spam. These honeypots were deployed in 5 Brazilian broadband networks (both cable and ADSL – Asymmetric Digital Subscriber Line), in 4 Brazilian cities for 15 months. They were deployed at volunteers' homes, to be exposed to the same conditions a typical home user computer with broadband connectivity would be.

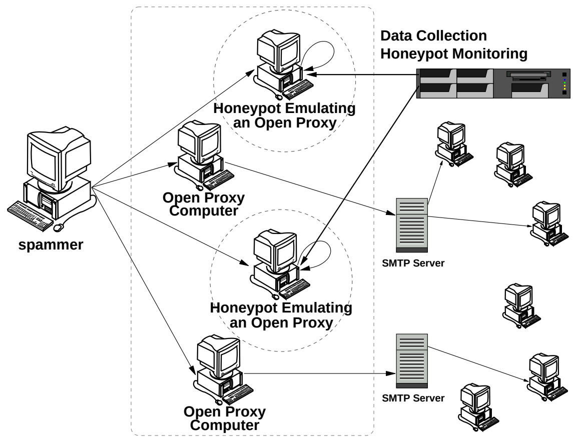

Figure 1: Diagram of the implemented architecture.

Table 1: TCP ports being emulated by the honeypots.

protocol	TCP port
SMTP	25
HTTP	80, 81, 2282, 3128, 3332, 3382, 3802, 4480, 5490, 6588, 8000, 8080, 8090, 11120, 57123, 63809, 65506
SOCKS	559, 1029, 1080, 1202, 1813, 1978, 1979, 2280, 2425, 3127, 3380, 3800, 4471, 4777, 4894, 5748, 6042, 7531, 9938, 10000, 10001, 10232, 11117, 15859, 19086, 24971, 24972, 24973, 30021, 30022, 35612, 38994, 40934, 41457, 57123, 63808

In each ISP we installed a residential connection, with dynamic IP address and less bandwidth, and a commercial one, usually with a static IP address and more bandwidth.

The honeypots were configured to emulate open proxy computers. Any spammer trying to use one of these honeypots to send spam would tend to believe that the emails were delivered successfully.

Another part of the architecture was a central server, configured to collect all the spam captured, as well as to periodically monitor the honeypots.

In the following sections we describe these components in more detail.

4.1.1 Honeypots Basic Configuration

Each honeypot was configured using an i386-based computer, with OpenBSD as its operating system and with the following hardware characteristics: 2.66 GHz Intel Celeron, 512 MB of RAM, 80 GB of disk and a 10/100 network interface.

The honeypot used the `pf` packet filter [8] blocking all incoming traffic, with the exception of traffic used for its management and traffic related to the TCP ports being emulated by `Honeyd`. The list of TCP ports emulated is shown in Table 1.

The honeypot's internal clock was synchronized using NTP (Network Time Protocol) [23]. The timezone used in each honeypot was GMT (Greenwich Mean Time), for standartization purposes and also to remain independent from local daylight saving changes.

4.1.2 Capture

The email capture on the honeypots was implemented using `Honeyd` together with the SMTP, HTTP and SOCKS proxy emulation subsystems.

Although `Honeyd` already had both SMTP and HTTP proxy subsystems, it does not have support for SOCKS protocol versions 4 and 5. In this way, a SOCKS emulator was developed as part of this work and to complement the other emulators. Additionally, the existing `Honeyd` modules were also modified.

The description of the emulation subsystems follows.

SMTP emulation: this `Honeyd` module receives traffic directed to port 25/TCP and emulates the responses of a typical SMTP server. For each message received, it behaves as an open relay server, accepting the message. The message, however, is stored locally and never delivered to its recipients.

HTTP proxy emulation: this `Honeyd` module is configured to run on several TCP ports normally associated with HTTP proxy services: 80/TCP, 8080/TCP, 3128/TCP, among other TCP ports listed in Table 1. When the spammer connects to this module, he typically requests to the proxy a connection to the IP address of the victim's SMTP server, on port 25/TCP. The proxy emulator, instead of granting the request, makes a connection to its own local SMTP server, which sends an SMTP response back to the spammer. This response gives an SMTP banner similar to the one that would be given by the requested SMTP server. The spammer then starts sending his emails, convinced that he is connected to the originally requested SMTP server.

SOCKS proxy emulation: this emulator was developed as part of this work. It emulates a SOCKS [18] (versions 4 and 5) proxy, receiving network traffic to the TCP ports normally associated to this service, as shown in Table 1. This emulator acts like a proxy that does not require authentication and allows connections coming from any IP address. After connecting to this emulator, the spammer requests a connection to an outside TCP address and port. If the requested TCP port is different from 25/TCP (SMTP), the emulator returns an error message to the client. If the requested port is 25/TCP, redirects the spammer to the local SMTP emulator, as described above in the HTTP proxy emulator.

Every transaction made by the `Honeyd` modules is logged, including timestamp, client IP, destination IP and TCP port as well as protocol version requested.

A description of the modifications made to the existing `Honeyd` modules follows.

Storage structure: the SMTP module is responsible for storing the emails received on disk, in a directory structure that takes into account the origin IP address and the message itself. We made some modifications to this structure scheme to include the date and the TCP port used for the message delivery, in order to help process the information.

Additional information logged: the HTTP proxy module has been modified to log more information, like the connection method, origin IP address, destination IP address, TCP port and status. The SMTP module log generation has also been modified in order to use the storage structure changes explained previously.

Data compression: data compression support has been added to the SMTP module, using the `zlib` library. This change allowed us to keep every email received in a compressed form, saving disk space.

4.1.3 Spam Collection

The spam captured on the honeypot was collected, at regular intervals, by a central server. This collection mechanism was implemented using the remote copy and synchronization program `rsync`, through an encrypted SSH (Secure Shell) tunnel.

Each honeypot had the ability to keep the captured spams for several days. However, after a successful synchronization with the central server, the local data was deleted from the honeypot to save disk space.

4.1.4 Honeypot Status Monitoring

In order to guarantee that every honeypot was working correctly, a status monitoring scheme was implemented. By executing this monitoring function several times a day, it was possible to quickly check if every honeypot was operating as expected.

This status monitoring procedure was specially important because of the nature of the connectivity and physical location of the honeypots. Broadband connections typically have less quality of service when compared to dedicated connections. As the honeypots were installed in residential environments, stability of electric power supply was also expected to have more problems than those in controlled environments, like Internet Data Centers.

Each status check performed is described below:

- Honeypot connectivity. Tests if the honeypot is accessible from the central server.
- System uptime. Particularly useful test to check for unexpected reboots, caused, for example, by electrical power problems.
- System load. Measures the number of processes in the operating system execution queue in the last 1, 5 and 15 minutes. The honeypot load is related to the amount of processes running in a given moment.
- Honeypot disk usage. Very important check to make sure that each honeypot has enough disk space to store the captured spams.
- Clock synchronization. This test evaluates the honeypot clock difference in comparison to an external reference, using NTP. The correct clock synchronization on the honeypots is very important to generate reliable log information.
- Critical processes running on the honeypot. This test checks if certain critical processes, important to capture spam, are running on the honeypot.
- Bandwidth usage and email capture rate. Measures the bandwidth usage (inbound and outbound, both in packets/s and bytes/s) and the email capture rate in the last 15 minutes.
- Data synchronization with the central server. Checks the last time the data synchronization with the central server occurred.

4.2 Results

In this section we show some of the results from the analysis made on the captured emails, taking into account their origin and TCP ports used.

As shown in Table 2, this project collected spams for 15 months, capturing approximately 525 million spams that would have been delivered to 4.8 billion recipients. The daily average amount of emails collected, combining all 10 honeypots, was approximately 1.1 million spams. These email messages came from 216,888 different IP

Table 2: General statistics related to the captured emails.

Data collection start	2006-06-10
Data collection end	2007-09-18
Days of collected data	466
Total emails collected	524,585,779
Total recipients	4,805,521,964
Average recipients per spam	9.16
Average emails per day	1,125,720
Unique IPs sending spam	216,888
Unique ASes sending spam	3,006
Unique Country Codes	165

Table 3: Top Country Codes originating spams.

#	CC	Emails	%
01	TW	385,189,756	73.43
02	CN	82,884,642	15.80
03	US	29,764,293	5.67
04	CA	6,684,667	1.27
05	JP	5,381,192	1.03
06	HK	4,383,999	0.84
07	KR	4,093,365	0.78
08	UA	1,806,210	0.34
09	DE	934,417	0.18
10	BR	863,657	0.16

addresses, allocated to 165 different countries (country codes, as defined by ISO 3166 [15]).

It is possible to note a significant concentration on the spam's country of origin. 97% of all spam captured by the project originated from only 5 countries: Taiwan, China, United States, Canada and Japan. This result can be seen in Table 3. Taiwan contribution in terms of the spam origin is also very significant, with 73.43% of all the messages sent. If we analyze the top 10 countries originating spam to the project's honeypots, this number is 99.50% of all messages.

In Figure 2 it is possible to see the percentage of spams sent by country code, during the project duration.

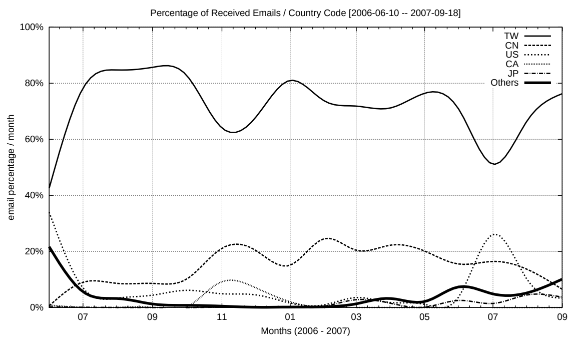

Figure 2: Percentage of received emails by country code.

Table 4: Top ASes originating spam.

#	ASN	AS Name	Emails	%
01	9924	TFN-TW (TW)	170,998,167	32.60
02	3462	HINET (TW)	131,381,486	25.04
03	17623	CNCGROUP (CN)	65,214,192	12.43
04	4780	SEEDNET (TW)	54,430,806	10.38
05	9919	NCIC-TW (TW)	9,186,802	1.75
06	4837	CHINA169 (CN)	9,025,142	1.72
07	33322	NDCHOST (US)	8,359,583	1.59
08	4134	CHINANET (CN)	7,287,251	1.39
09	18429	EXTRALAN (TW)	6,746,124	1.29
10	7271	LOOKAS (CA)	5,599,442	1.07

An AS (Autonomous System) is an connected group of one or more network blocks with a well-defined routing policy [9]. Table 4 shows the spam distribution by originating AS. Again it is possible to see that the activity is concentrated: the top 4 ASes originating spam were responsible for approximately 80% of all activity recorded.

If we take into account the countries that are associated to these ASes, again Taiwan and China play an important role. Also, the top 2 ASes, TFN-TW and HINET, both from Taiwan, originated 58% of all spam captured.

Another aspect analyzed was the TCP port used by the spammers to inject spams. Although each honeypot was configured to emulate several TCP ports, as seen in Table 1, only 11 TCP ports were used. Table 5 shows the set of TCP ports used for sending spam, as well as the protocol (HTTP or SOCKS) and the service normally associated with each port.

The development of the SOCKS support, as part of this work, was very important as port 1080/TCP, which is the default port associated to this protocol, was the port that collected the most spam during the period.

Some of the TCP ports abused by the spammers are associated with popular services — examples are 80/TCP, 8000/TCP and 8080/TCP, normally used by the `http` service. Other TCP ports, like 6588/TCP and 4480/TCP (`AnalogX` and `Proxy+`, proxies normally used by home users) show an attempt by spammers to explore misconfigured home proxies. It is also important to note the activity on TCP ports associated with proxies installed by malicious code, like `MyDoom` and `Sobig.f`, indicating that spammers are actively using previously infected machines to send spam.

Figure 3 shows the percentage of emails received by TCP port, during the collecting period.

The requests made to the HTTP and SOCKS proxy modules were also analyzed. These results are shown in Table 6. For the HTTP module the requests were divided as follows: outgoing connections to port 25/TCP, outgoing connection to other TCP ports, "GET" requests

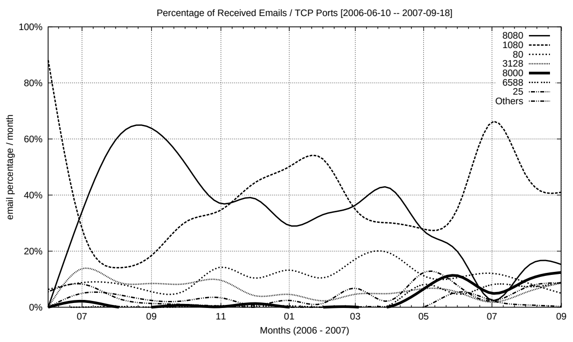

Figure 3: Percentage of emails received by TCP port.

(attempts to use the proxy to access a Web server) and errors (invalid commands). For the SOCKS module we made a similar separation: connections to port 25/TCP, connections to other ports and errors. It is important to note that the majority of the requests (97.62% for HTTP and 87.31% for SOCKS) were attempts to connect to port 25/TCP on third party machines. This makes it clear that the main objective of the majority of the connections to these honeypots was to deliver spam, and not other type of proxy abuse.

Table 5: TCP ports abused.

#	Port	Protocol	Usual Service	%
01	1080	SOCKS	socks	37.31
02	8080	HTTP	alternate http	34.79
03	80	HTTP	http	10.92
04	3128	HTTP	Squid	6.17
05	8000	HTTP	alternate http	2.76
06	6588	HTTP	AnalogX	2.29
07	25	SMTP	smtp	1.46
08	4480	HTTP	Proxy+	1.38
09	3127	SOCKS	MyDoom Backdoor	1.00
10	3382	HTTP	Sobig.f Backdoor	0.96
11	81	HTTP	alternate http	0.96

Table 6: Requests made to the HTTP and SOCKS modules.

Module	Type	Requests	%
HTTP	connection to 25/TCP	89,496,969	97.62
	connection to other	106,615	0.12
	get	225,802	0.25
	errors	1,847,869	2.01
	total	91,677,255	100.00
SOCKS	connection to 25/TCP	46,776,884	87.31
	connection to other	1,055,081	1.97
	errors	5,741,908	10.72
	total	53,573,873	100.00

5 Conclusions

This paper described the design and implementation of an architecture, based on honeypots, for the study of the spam problem, in particular the abuse of open proxies. It also described some extensions to the current spam capture technology and showed that this architecture, in place for 15 months, proved to be very efficient in capturing spam.

The analysis of the received spam made it clear that the origin of the IP addresses exploring open proxies for sending spam was very concentrated: approximately 97% of all email came from only 5 different country codes. Approximately 89% of all messages came from two countries: Taiwan and China. When the origin was analyzed based on source Autonomous System a similar concentration was observed. The top 4 ASes (out of a total of 3,006) were responsible for almost 80% of all the emails observed. This high concentration of activity, both in terms of country codes and Autonomous Systems, could be helpful to fight the problem: if acceptable use and anti-spam policies were enforced in these networks, it could have a significant impact in reducing spam.

The results also show that most of the requests received by the proxy modules were directed to the SMTP port (25/TCP) of third party machines. This shows that the main objective of the majority of these connections was, in fact, spam delivery. To mitigate the delivery of spam via the abuse of residential and dynamic IP networks, a method that could be very effective is port 25 management [20]. The main objective is to differentiate email client submission traffic from email transport between SMTP servers. Traffic between mail servers would continue to use port 25/TCP, but email submission traffic from clients to servers would use an alternate submission mechanism: the authenticated Message Submission for Email, on port 587/TCP [6]. With this mechanism in place it is possible to filter outgoing port 25/TCP traffic in residential and dynamic IP networks, reducing the amount of spam leaving these networks and making the problem more traceable for ISPs [20].

After this initial phase of operation and validation of the technology in use, some directions for future work are possible.

All data was analyzed only based on its origin, not on its content. As a future work it could be possible to conduct a more detailed analysis of the collected emails, taking into account the email body, recipient addresses, among others.

It is also important to propose best practices recommendations to Brazilian ISPs in order to reduce the abuse of their networks for sending spam.

Another possible line of work would be the deployment of sensors in other countries to have a more global view of the problem.

Acknowledgements

The authors would like to thank the Brazilian Internet Steering Committee (CGI.br), and the Brazilian Network Information Center (NIC.br), for their financial support regarding equipment and connectivity costs. We would also like to thank the CERT.br team for their enthusiasm and dedication.

References

[1] Andreolini, M., Bulgarelli, A., Colajanni, M., and Mazzoni, F. Honeyspam: Honeypots fighting spam at the source. In *Proceedings of the USENIX Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI '05)*, pages 77–83, Cambridge, MA, USA, July 2005.

[2] Cerf, V. G. Spam, spim, and spit. *Commun. ACM*, 48(4):39–43, 2005.

[3] Chuvakin, A. Honeynets: High Value Security Data – Analysis of real attacks launched at a honeypot. *Elsevier Network Security*, 2003(8):11–15, August 2003. ISSN: 1353-4858.

[4] Cranor, L. F. and LaMacchia, B. A. Spam! *Commun. ACM*, 41(8):74–83, 1998.

[5] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. RFC 2616: Hypertext Transfer Protocol – HTTP/1.1. `http://www.ietf.org/rfc/rfc2616.txt`, June 1999.

[6] Gellens, R. and Klensin, J. RFC 4409: Message Submission for Mail. `http://www.ietf.org/rfc/rfc4409.txt`, April 2006.

[7] Hambridge, S. and Lunde, A. RFC 2635: DON'T SPEW A Set of Guidelines for Mass Unsolicited Mailings and Postings (spam*). `http://www.ietf.org/rfc/rfc2635.txt`, June 1999.

[8] Hartmeier, D. Design and performance of the openbsd stateful packet filter (pf). In *Proceedings of the FREENIX Track: 2002 USENIX Annual Technical Conference (FREENIX '02)*, Monterey, CA, USA, June 2002.

[9] Hawkinson, J. and Bates, T. RFC 1930: Guidelines for creation, selection, and registration of an Autonomous System (AS). `http://www.ietf.org/rfc/rfc1930.txt`, March 1996.

[10] Hayes, B. Spam, spam, spam, lovely spam. *American Scientist*, 91(3):200–204, May–June 2003.

[11] Hoepers, C., Steding-Jessen, K., and Chaves, M. H. P. C. Projeto e Desenvolvimento de um Sistema de Controle e Acompanhamento de Notificações de Spam. In *Anais do V Simpósio sobre Segurança em Informática (SSI'2003)*, São José dos Campos, SP, Novembro 2003.

[12] Holz, T. A short visit to the bot zoo. *IEEE Security & Privacy*, 3(3):76–79, May-June 2005. `http://www-pi1.informatik.uni-mannheim.de/publications/show/13`.

[13] Honeynet Project. Know Your Enemy: Honeynets. `http://www.honeynet.org/papers/honeynet/`.

[14] Ianelli, N. and Hackworth, A. Botnets as a vehicle for online crime. `http://www.cert.org/archive/pdf/Botnets.pdf`, December 2005. CERT Coordination Center, Carnegie Mellon University.

[15] International Organization for Standardization. ISO 3166: Codes for the representation of names of countries and their subdivisions – Part 1: Country codes. `http://www.iso.org/iso/country_codes.htm`, November 2006.

[16] Klensin, J. RFC 2821: Simple Mail Transfer Protocol. `http://www.ietf.org/rfc/rfc2821.txt`, April 2001.

[17] Krawetz, N. Anti-honeypot technology. *IEEE Security & Privacy*, 2(1):76–79, January–February 2004.

[18] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and Jones, L. RFC 1928: SOCKS Protocol Version 5. `http://www.ietf.org/rfc/rfc1928.txt`, March 1996.

[19] Lindberg, G. RFC 2505: Anti-Spam Recommendations for SMTP MTAs. `http://www.ietf.org/rfc/rfc2505.txt`, February 1999.

[20] Messaging Anti-Abuse Working Group (MAAWG). Managing Port 25 for Residential or Dynamic IP Space. `http://www.maawg.org/port25/MAAWG_Port25rec0511.pdf`, Dec 2005.

[21] Messaging Anti-Abuse Working Group (MAAWG). Email Metrics Program: Report #5 – First Quarter 2007. `http://www.maawg.org/about/MAAWG20071Q_Metrics_Report.pdf`, June 2007.

[22] Milletary, J. Technical trends in phishing attacks. http://www.cert.org/archive/pdf/Phishing_trends.pdf, October 2005. CERT Coordination Center, Carnegie Mellon University.

[23] Mills, D. RFC 4330: Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI. http://www.ietf.org/rfc/rfc4330.txt, January 2006.

[24] Postel, J. RFC 706: On the junk mail problem. http://www.ietf.org/rfc/rfc0706.txt, November 1975.

[25] Provos, N. A Virtual Honeypot Framework. In *Proceedings of 13th USENIX Security Symposium*, pages 1–14, San Diego, CA, USA, August 2004. http://www.usenix.org/publications/library/proceedings/sec04/tech/provos.html.

[26] Provos, N. and Holz, T. *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. Addison-Wesley Professional, 1st edition, July 2007. ISBN-13: 978-0321336323.

[27] Sauver, J. S. Spam zombies and inbound flows to compromised customer systems. In *Proceedings of the MAAWG General Meeting*, March 2005. http://darkwing.uoregon.edu/~joe/zombies.pdf.

[28] Shirey, R. RFC 2828: Internet Security Glossary. http://www.ietf.org/rfc/rfc2828.txt, May 2000.

[29] Spitzner, L. *Honeypots: Tracking Hackers*. Addison-Wesley Professional, 1st edition, September 2002. ISBN: 0321108957.

[30] The Honeynet Project. *Know Your Enemy: Learning about Security Threats*. Addison-Wesley Professional, 2nd edition, May 2004. ISBN: 0321166469.

[31] Zakon, R. RFC 2235: Hobbes' Internet Timeline. http://www.ietf.org/rfc/rfc2235.txt, November 1997.