The Spammer, the Botmaster, and the Researcher: on the Arms Race in Spamming Botnet Mitigation - Major Area Exam

Gianluca Stringhini

December 16, 2011

1 Introduction

Spam, or *Unsolicited Bulk Email*, is a big problem in nowadays internet. Recent studies report that spam accounts for more than 90% of the worldwide email traffic [40]. Spam is not only annoying for users, who receive content they did not request, but is also a burden for the whole email delivery infrastructure, that needs to keep delivering legitimate emails with a short delays, but also make sure that unsolicited messages are detected and blocked.

Spam can have different goals, from carrying out scams to spread malware with malicious email attachments. However, one of the most common types of spam is the one that promotes e-commerce sites selling illicit goods, such as pharmaceutical products or counterfeit watches and accessories. The e-commerce sites selling such products present the same functionalities of popular legitimate sites (e.g., Amazon), and offer customer service, daily deals, and even refunds if the user is not satisfied. The reason of this is that the cybercriminals behind such sites want to appear as legitimate as possible. Also, unlike what people commonly think, these sites are not scamming their users, and the purchased goods are actually delivered to the recipients [30, 36]. Whether the drugs produces by these companies are equivalent to the branded ones or not is an open question.

Large spam e-commerce sites offer affiliate programs (*partnerka* in Russian) [51]. People who join these affiliate programs send spam through their own email delivery infrastructures, and receive a cut of the revenue in exchange for their services. Past research showed that spam e-commerce sites are quite profitable, with estimates going from $300,000 to $1 Million a month for a large affiliate site [29, 30]. The operations of spam e-commerce involve parties located in most parts of the world, from the domain registrars, to the hosting providers, to the banks processing the payments [36]. For this reason, although many countries have good anti-spam laws, effectively fight them on the law side is hard.

Nowadays, most of worldwide spam is sent by *botnets*, which are networks of compromised computers that act under the control of a single entity, the so called *botmaster*. Recent reports show how botnets are responsible for 85% of worldwide spam [59]. Botnets provide a convenient infrastructure for cybercriminals, because they combine the best of two worlds: *Internet worms* and *IRC bots* [45]. First generation botnets worked just like Internet worms. They would infect a machine by exploiting a known vulnerability, and start scanning the network for more vulnerable hosts. Also, similarly to IRC bots, infected machines would join an IRC server where the botmaster could give them orders. Botnets are also used for a variety of malicious activities, such as stealing personal information, running denial of service attacks, and solving captchas.

Security researchers have spent a lot of efforts in disrupting botnets and detecting spam. On the other hand, spammers and botmasters constantly come up with more sophisticated techniques that allow them to avoid detection. In the rest of this paper, I analyze this arms race, focusing on the efforts by both communities. I also conclude look at some unexplored research areas that might give good results on fighting spamming botnets in the future.

2 The evolution of botnets

Since they first became a threat, in the mid-2000s, botnets evolved a lot, to keep up with the security research being conducted to disrupt them, and make sure their operation could continue. This evolution happened in two fields: the structure of the botnets, and the way they propagate their infections.

2.1 The evolution of botnet structures

IRC botnets As mentioned before, the first botnets borrowed characteristics from IRC bots. Therefore, after being infected, a bot would connect to an IRC server, join a specific channel, and wait for orders [17, 18, 45].

These botnets did not use a lot of sophistication for hiding their actions. They usually used a password to protect the IRC channel where the botmaster would give the orders, but both the server name and the password would be in clear in the malicious binary. Therefore, researchers could retrieve this information, and join the channel to learn important information about the botnet. As an alternative, researchers could sinkhole the DNS traffic asking for the malicious domains, so that the infected machines would connect to them instead of going to the Command and Control server (C&C from now on) [45]. Another weakness of this model is that the C&C traffic used the IRC protocol, which is easy to detect and monitor.

Proprietary botnets To avoid using known protocols and make their activity less evident, botmasters started to use proprietary, encrypted protocols for their C&C traffic [56]. This makes botnet infiltration harder, but researchers can still reverse engineer the protocol, create software that implements it, and join the botnet. Also, since the infrastructure still uses a single domain, sinkholing is still possible.

Multiple tier botnets To make their infrastructure more resilient to attacks, botmasters started developing multiple-tier botnets [55]. In this architecture, the bots, instead of contacting the C&C server directly, contact one of many proxies, that then forward their request to the C&C server. By doing this cybercriminals make sure that, even if researchers took control of a limited number of proxies, the botnet would still be operational.

In addition, botmasters developed a new technique that gives them even more reliability on their C&C infrastructure: *Fast Flux* [26, 41]. This technique is similar to the ones involved in *Round Robin DNS* and the ones used by *Content Delivery Networks*, and its goal is to assign a fast-changing number of IP addresses to the domains used by the botnet C&C infrastructure. By doing this, the botmaster ensures that even if a very small number of proxies would survive a takedown, the botnet would still be operational.

Even if the C&C infrastructure now uses multiple domains, it is still possible for the researchers to sinkhole or blacklist them.

Domain Generation Algorithms To mitigate the disadvantages of having a limited number of domains for the C&C infrastructure, cybercriminals developed algorithms that allow both the bots and the C&C to generate domains on the fly. These algorithms, called *Domain Generation Algorithms (DGA)*, are typically time-sensitive, and, at any point in time, tell bots at which domain to find the C&C server (or one of the proxies associated to it) [55]. What the botmaster has to do is registering the domains that will be used in the future, and make sure his infrastructure will respond to the bots at the right time.

Of course, researchers could reverse engineer the domain generation algorithm, and register the domains before the botmaster does. A countermeasure to this is making the DGA non-deterministic, for example by using information taken from social network trending topics.

Peer-to-peer botnets Another evolution is to make the botnet peer-to-peer [33, 54]. In this architecture, another level of relayers is deployed between the bots and the proxies. Typically, those bots that don't have a public IP (i.e., are behind a NAT) act as regular bots, while those that have a public IP act as relayer bots. Regular bots find the nearest relayer by implementing some sort of *Overnet* protocol, which is typical of peer-to-peer networks (e.g., *Kademlia*. A problem of this approach is that the botmaster gives up the control on a critical part of his infrastructure (i.e., the relayers). Researchers could reverse engineer the C&C protocol and infiltrate the botnet by pretending to be a relayer. This would allow them to collect a wealth of information about the botnet, for example what spam templates are delivered to the bots.

2.2 The evolution of the botnet infection model

Not only the botnet structure changed over time, but also the infection model the botmaster uses to gain control of more machines evolved.

At the beginning, bots were behaving like Internet worms [22]. This means that an infected machine would scan for more vulnerable hosts in her network, and try to propagate. This approach became less and less used over time, with Conficker being the last large botnet using it in 2008.

The next step has been to use bots that did not propagate on their own anymore. In this phase, the main channels of infections were two:

- Sending malicious binaries through spam emails, and luring victims into clicking on them
- Setting up malicious web pages that tried to exploit vulnerabilities in the victim's browser and download the malicious binary (i.e., a *drive-by-download attack*).

Nowadays, the trend botmasters follow to deploy their bots is to use third party services. These services typically use pre-existing botnets to download additional components (i.e., bots) on the infected machines, for a fee [7].

3 The evolution of Botnet and Spam mitigation

After studying how botnets evolved to avoid detection, let's have a look at what techniques researchers developed to detect infected machines and malicious servers, and mitigate their effects.

There are a number of vantage points researchers can leverage to make their analysis, each one providing different information, and each one allowing to take different countermeasures. We analyze each one of them in the following paragraphs.

Host based detection A vantage point researchers can leverage is the victim host. By looking at the binaries that get installed, one can try to infer whether they are malicious or not.

Traditional anti-viruses build signatures (i.e., regular expressions) from known malware, and look for the presence of those signatures in the binaries the user downloads. This technique is not very robust, and previous work showed how the detection can be fooled by simple obfuscations such as inserting NOP instructions and performing code transpositions [11, 13].

Remaining in the field of static analysis, a better approach is to extract semantic information from malware, and look for the same semantics in new samples while performing detection [11, 12]. The problem here is that program equivalence is an undecidable problem. Therefore, even if the proposed systems can cover a number of variations that model the same behavior, it is not guaranteed that this will work for any possible sample. In addition, modern malware comes packed and decrypts itself at runtime, and this makes static analysis difficult.

Dynamic analysis makes this kind of analysis easier, because one can look at the program once the decryption has happened. The techniques that have been proposed include modelling the behavior of a program based on the system calls it executes [32], monitoring programs accessing sensitive information while they should not [67], or looking at the buffers allocated by a malware sample to reconstruct the C&C protocol it uses [8]. The problem of dynamic analysis are, again, that program equivalence is an undecidable problem. In addition, running large amounts of malware samples takes time and resources.

Malicious web pages detection Another approach researchers used is looking at malicious web pages that try to compromise the victim's browser and download a piece of malware. These attacks are typically performed by malicious Javascript scripts. To detect such scripts, various approaches have been used:

- Using machine learning to detect legitimate and malicious pages. The features researchers used include how many redirections the web page uses, or how much Javascript code is obfuscated. Detection can be done either offline, by visiting the page with an emulated browser [15], or online, by instrumenting the victim browser, and stopping executing a script once one detects it's malicious [16, 25].
- Another approach is to look at the changes in the victim's system when she visits a malicious page. The creation of files or the changes of registry keys are indicators of a compromise [43].
- The last possibility is to look at typical attack patterns and flag as malicious any script that shows those patterns [49].

The problem with these techniques is mostly that they rely on a static model to detect malicious scripts. Therefore, they could miss newer attacks cybercriminals might come up with.

Command and Control based detection Other work operated at the Command and Control level. The goal is typically to learn important information about the botnet, or to attempt a takedown.

For older botnets, that were using IRC, infiltration was easy. All that researchers had to do was joining the IRC server and a particular channel, and look for the commands being issued by the botmaster [45]. Nowadays, this is not possible anymore, and researchers need to reverse engineer the C&C protocol first. A way of doing that is by active probing [10]. This type of techniques enable botnet infiltration, which means that researchers can create a piece of software that behaves exactly like a bot, but does not execute any malicious activity[9, 33, 54] .

Another possibility, for those botnets that use it, is to reverse engineer the DGA used by the botnet. By doing this, researchers can register the C&C domains before the botmasters do, and impersonate the Command and Control server [55].

Another possible way of interacting with the C&C infrastructure is by setting up honeypots. Honeypots are virtualized environments where it is possible to run malware, and monitor its activity [28]. Monitoring the activity of the bots can give insights on what the

Command and Control IP addresses and domains are. This information can be used for blacklisting them [57] or for performing botnet takedowns [56].

Honeypots, however, come with some problems. First of all, malware might detect that it is running in a virtualized environment, and do perform any malicious activity [4]. On the other hand, while running malware, researchers need to make sure that their environment is constrained enough that they cannot damage anybody.

DNS based detection As most Internet services, botnets use the DNS infrastructure to easily connect the different components of their infrastructure (i.e., the bots and the Command and Control server). Therefore, by looking at the interaction between bots and DNS servers one can learn important information about the botnet, such as what IP addresses are associated to infected machines. This can be done by sinkholing the domains used by a botnet's C&C infrastructure. By doing this, the infected machines will contact the researchers instead of the botmaster, and it will be possible to enumerate them [18]. Another option is to look in local DNS servers for the presence of cached results associated to malicious domains [45]. If such records are found, that is an indicator of the presence of infected machines in the network.

DNS activity can also be used to detect domains that use Fast Flux. Such domains present very different features than legitimate ones. For example, the IP addresses returned for a query to a Fast Flux domain will belong to very different networks in various parts of the world, the TTL will be low, and two subsequent queries will return different results. Previous work focused on building classifiers to detect such domains [26, 27, 41]. Once a Fast Flux domain is detected, network administrators can blacklist it to avoid their machines to connect to the C&C infrastructure.

Another approach is to detect malicious domains by observing patterns in which domains are queried. Proposed systems use local data from Recursive DNS servers [2, 6], or a more comprehensive view by looking at the Top Level Domain (TLD) level [3].

SMTP based detection Another vantage point researchers can operate at are SMTP servers. Once spamming bots sent their emails, various techniques can be used on the SMTP server to figure out whether that email is spam or not.

The first way of detecting spam is by looking at the content of the emails. Traditional spam detection relies on rules. Similarly to what happens for anti-viruses, these rules are not robust enough, since the nature of spam changes often. Also, having a binary decision value (i.e., spam or ham) does not help in many cases. The reason is that having a misclassified ham email affects the user way more than having a misclassified spam email. For this reason, introducing a confidence level, instead of a binary decision, allows the server administrator to set how sensitive one wants to be while doing spam detection. Method based on a confidence measure include *Bayesian filtering* [1, 50] and using *Support Vector Machines* (SVM) [19]. These approaches consider an email as spam if it contains words that are highly discriminative of spam content. The problem of these techniques is that attackers can include words that are typical of good email in their spam messages, and bypass the filters [31, 38].

Other content analysis techniques include building reputation systems according to features in the received emails [24], building templates from bot-generated spam and use them for spam detection [42], or studying the URLs contained in spam emails [65, 39, 61]. The problem with all these approaches is that content analysis is expensive, and mailservers cannot afford to run these techniques on every email they receive. For this reason, several approaches that try to prioritize email, and apply content analysis only when it is strictly necessary, have been proposed [60, 62, 63].

A more lightweight spam detection technique is IP blacklisting. The most common type of services providing this functionality are *DNS-based blacklists*. Mail servers can query these services over DNS to know whether an individual IP is a know spammer or not. The problem with these services is mainly their low coverage. Previous work showed that the coverage of DNS-based blacklists spans from 20% to 90% [46, 52]. One of the reasons, is that most bot machines have dynamic IPs and, therefore, blacklisting them is ineffective as soon as the bots obtain a new IP. Better approaches looked at assigning an IP reputation to networks based on how many blacklisted IPs are in a certain network [47, 53, 44], or learning the behavior of blacklisted IPs to flag other IPs behaving in a similar way [48].

Another type techniques that can be employed on the SMTP server side is based on policies. *Greylisting* is a popular technique that relies on the fact that bots will not retry sending an email when they receive a non-critical error from the server, while legitimate clients will [37]. Other approaches bring digital signatures to email headers, to make sure that the address that sent an email is really who it claims to be [35].

Social network based detection As social networks became more popular, malicious users started using them to spread spam. A common trend on social networking sites is to see fake accounts that are used by bots to post malicious content. Previous work focused on detecting these fake profiles by looking at typical features such as how many friends a certain profile has, or how similar the content it posts is [5, 34, 58, 66].

A new trend for botmasters, however, is to compromise legitimate accounts and use them to spread malicious content. Recent work shows how this trend is growing [20], and this will be the next field to be tackled by researchers for sure.

Network edge based detection The last vantage point that can be leveraged by researchers is the network edge. By observing the communication between infected machines inside the network and Command and Control servers outside the network, it is possible to learn important information about botnets, and develop effective countermeasures.

A direction researchers looked at is detecting successful infections by monitoring network traffic [22]. In this research work, infections are modeled as a set of flows that picture the different steps of the infection. Although interesting, this model cannot be applied anymore today. The reason is that years ago botnet infections followed a well defined, worm-like behavior (i.e., scanning for victims, exploitation, download of an egg, connection to the command and control), which is not widely used anymore.

More recent research proposed to look at the correlation between Command and Control messages and malicious activity. The idea is that any time a bot will receive a command, it will perform a malicious activity. By looking at this correlation, it is possible to detect bots without any previous knowledge of the botnet [21]. The problem is how to identify Command and Control traffic. Older approaches were looking for commonly-misused, well known protocols (e.g., IRC) [23]. However, this type of techniques are not applicable anymore. More recent work looks for malicious activity first, and then looks for the interaction that came before to find the actual commands [64].

Another proposed techniques is to look at what servers are contacted by machines the network administrator knows are infected, and looking for more machines contacting the same servers to have a broader knowledge of the bot population in the network [14].

4 Conclusions

In this writeup, I talked about the arms race between cybercriminals and security researchers. In the future, this arms race will continue. Future botnets will likely be more sophisticated than the current ones, and detecting the hosts involved in malicious activity will be harder. However, a functional botnet has to interact with legitimate services (e.g., DNS or SMTP). Botmasters cannot obfuscate this communication, since it has to be understood by a legitimate third party. In addition, bots interact with legitimate services in an automated fashion. Therefore, it is likely possible to distinguish between bots and legitimate users using a service, and leverage this information for botnet detection and mitigation.

References

[1] I. Androutsopoulos, J. Koutsias, K. Chandrinos, and C. Spyropoulos. An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *international ACM SIGIR conference on Research and development in information retrieval*, 2000.

[2] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster. Building a dynamic reputation system for DNS. In *USENIX Security Symposium*, 2010.

[3] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou, and D. Dagon. Detecting Malware Domains at the Upper DNS Hierarchy. In *USENIX Security Symposium*, 2011.

[4] D. Balzarotti, M. Cova, C. Karlberger, C. Kruegel, K. Engin, and G. Vigna. Efficient detection of split personalities in malware. In *Symposium on Network and Distributed System Security (NDSS)*, 2010.

[5] F. Benvenuto, G. Magno, T. Rodrigues, and V. Almeida. Detecting spammers on twitter. In *Conference on Email and Anti-Spam (CEAS)*, 2010.

[6] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi. Exposure : Finding malicious domains using passive dns analysis. In *Symposium on Network and Distributed System Security (NDSS)*, 2011.

[7] J. Caballero, C. Grier, C. Kreibich, and V. Paxson. Measuring Pay-per-Install: The Commoditization of Malware Distribution. *USENIX Security Symposium*, 2011.

[8] J. Caballero, P. Poosankam, C. Kreibich, and D. Song. Dispatcher: Enabling active botnet infiltration using automatic protocol reverse-engineering. In *ACM Conference on Computer and Communications Security (CCS)*, 2009.

[9] C. Cho, J. Caballero, C. Grier, V. Paxson, and D. Song. Insights from the inside: A view of botnet management from infiltration. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2010.

[10] C. Y. Cho, D. Babic, and D. Song. Inference and Analysis of Formal Models of Botnet Command and Control Protocols. In *ACM Conference on Computer and Communications Security (CCS)*, 2010.

[11] M. Christodorescu and S. Jha. Static analysis of executables to detect malicious patterns. In *USENIX Security Symposium*, 2003.

[12] M. Christodorescu, S. Jha, S. Seshia, D. Song, and R. Bryant. Semantics-aware malware detection. In *IEEE Symposium on Security and Privacy*, 2005.

[13] M. Christodoresu and S. Sha. Testing malware detectors. *International Symposium on Software Testing and Analysis (ISSTA)*, 2004.

[14] B. Coskun and S. Dietrich. Friends of An Enemy : Identifying Local Members of Peer-to-Peer Botnets Using Mutual Contacts Categories and Subject Descriptors. In *Annual Computer Security Applications Conference (ACSAC)*, 2010.

[15] M. Cova, C. Kruegel, and G. Vigna. Detection and analysis of drive-by-download attacks and malicious JavaScript code. In *International Conference on World Wide Web (WWW)*, 2010.

[16] C. Curtsinger, B. Livshits, B. Zorn, and C. Seifert. ZOZZLE: Fast and Precise In-Browser JavaScript Malware Detection. 2011.

[17] D. Dagon, G. Gu, C. P. Lee, and W. Lee. A Taxonomy of Botnet Structures. In *Annual Computer Security Applications Conference (ACSAC)*, 2007.

[18] D. Dagon, C. Zou, and W. Lee. Modeling botnet propagation using time zones. In *Symposium on Network and Distributed System Security (NDSS)*, 2006.

[19] H. Drucker, D. Wu, and V. N. Vapnik. Support vector machines for spam categorization. In *IEEE transactions on neural networks*, 1999.

[20] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Zhao. Detecting and characterizing social spam campaigns. In *Internet Measurement Conference (IMC)*, 2010.

[21] G. Gu, R. Perdisci, J. Zhang, and W. Lee. BotMiner: clustering analysis of network traffic for protocol-and structure-independent botnet detection. In *USENIX Security Symposium*, 2008.

[22] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee. Bothunter: Detecting malware infection through ids-driven dialog correlation. In *USENIX Security Symposium*, 2007.

[23] G. Gu, J. Zhang, and W. Lee. BotSniffer: Detecting botnet command and control channels in network traffic. In *Symposium on Network and Distributed System Security (NDSS)*, 2008.

[24] S. Hao, N. Syed, N. Feamster, A. Gray, and S. Krasser. Detecting spammers with SNARE: Spatio-temporal network-level automatic reputation engine. In *USENIX Security Symposium*, 2009.

[25] M. Heiderich, T. Frosch, and T. Holz. Iceshield: Detection and mitigation of malicious websites with a frozen dom. *Symposium on Recent Advances in Intrusion Detection (RAID)*, 2011.

[26] T. Holz, C. Gorecki, K. Rieck, and F. Freiling. Measuring and detecting fast-flux service networks. In *Symposium on Network and Distributed System Security (NDSS)*, 2008.

[27] X. Hu, M. Knysz, and K. Shin. Rb-seeker: Auto-detection of redirection botnets. In *Symposium on Network and Distributed System Security (NDSS)*, 2009.

[28] J. John, A. Moshchuk, S. Gribble, and A. Krishnamurthy. Studying spamming botnets using Botlab. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2009.

[29] C. Kanich, C. Kreibich, K. Levchenko, B. Enright, G. Voelker, V. Paxson, and S. Savage. Spamalytics: An empirical analysis of spam marketing conversion. In *ACM Conference on Computer and Communications Security (CCS)*, 2008.

[30] C. Kanich, N. Weaver, D. McCoy, T. Halvorson, C. Kreibich, K. Levchenko, V. Paxson, G. Voelker, and S. Savage. Show Me the Money: Characterizing Spam-advertised Revenue. *USENIX Security Symposium*, 2011.

[31] C. Karlberger, G. Bayler, C. Kruegel, and E. Kirda. Exploiting redundancy in natural language to penetrate bayesian spam filters. *USENIX Workshop on Offensive Technologies (WOOT)*, 2007.

[32] C. Kolbitsch, P. Comparetti, C. Kruegel, E. Kirda, X. Zhou, and X. Wang. Effective and efficient malware detection at the end host. In *USENIX Security Symposium*, 2009.

[33] C. Kreibich, C. Kanich, K. Levchenko, B. Enright, G. Voelker, V. Paxson, and S. Savage. Spamcraft: An inside look at spam campaign orchestration. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2009.

[34] K. Lee, J. Caverlee, and S. Webb. Uncovering social spammers: social honeypots + machine learning. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2010.

[35] B. Leiba. DomainKeys Identified Mail (DKIM): Using digital signatures for domain verification. *Conference on Email and Anti-Spam (CEAS)*, 2007.

[36] K. Levchenko, A. Pitsillidis, N. Chachra, B. Enright, M. Félegyházi, C. Grier, T. Halvorson, C. Kanich, C. Kreibich, H. Liu, and Others. Click Trajectories: End-to-End Analysis of the Spam Value Chain. In *IEEE Symposium on Security and Privacy*, 2011.

[37] J. R. Levine. Experiences with Greylisting. In *Conference on Email and Anti-Spam (CEAS)*, 2005.

[38] D. Lowd and C. Meek. Good word attacks on statistical spam filters. In *Conference on Email and Anti-Spam (CEAS)*, 2005.

[39] J. Ma, L. Saul, S. Savage, and G. Voelker. Beyond blacklists: learning to detect malicious web sites from suspicious urls. In *ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009.

[40] MessageLabs. MessageLabs Intelligence: 2010 Annual Security Report. `http://www.messagelabs.com/mlireport/MessageLabsIntelligence_2010_Annual_Report_FINAL.pdf`, 2010.

[41] E. Passerini, R. Paleari, L. Martignoni, and D. Bruschi. Fluxor: detecting and monitoring fast-flux service networks. In *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, 2008.

[42] A. Pitsillidis, K. Levchenko, C. Kreibich, C. Kanich, G. Voelker, V. Paxson, N. Weaver, and S. Savage. Botnet judo: Fighting spam with itself. In *Symposium on Network and Distributed System Security (NDSS)*, 2010.

[43] N. Provos, P. Mavrommatis, M. Abu Rajab, and F. Monrose. All your iframes point to us. In *USENIX Security Symposium*, 2008.

[44] Z. Qian, Z. Mao, Y. Xie, and F. Yu. Investigation of Triangular Spamming: a Stealthy and Efficient Spamming Technique. In *IEEE Symposium on Security and Privacy*, 2010.

[45] A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A multifaceted approach to understanding the botnet phenomenon. In *ACM SIGCOMM Conference on Internet Measurement*, 2006.

[46] A. Ramachandran, D. Dagon, and N. Feamster. Can DNS-based blacklists keep up with bots? In *Conference on Email and Anti-Spam (CEAS)*, 2006.

[47] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. *ACM SIGCOMM Computer Communication Review*, 2006.

[48] A. Ramachandran, N. Feamster, and S. Vempala. Filtering spam with behavioral blacklisting. In *ACM Conference on Computer and Communications Security (CCS)*, 2007.

[49] P. Ratanaworabhan and B. Lishvits, B. amd Zorn. Nozzle: A defense against heap-spraying code injection attacks. In *USENIX Security Symposium*, 2011.

[50] M. Sahami, S. Dumais, D. Heckermann, and E. Horvitz. A Bayesian approach to filtering junk e-mail. *Learning for Text Categorization*, 1998.

[51] D. Samosseiko. THE PARTNERKA - WHAT IS IT, AND WHY SHOULD YOU CARE? In *Virus Bulletin Conference*, 2009.

[52] S. Sinha, M. Bailey, and F. Jahanian. Shades of grey: On the effectiveness of reputation-based "blacklists“. In *International Conference on Malicious and Unwanted Software*, 2008.

[53] S. Sinha, M. Bailey, and F. Jahanian. Improving spam blacklisting through dynamic thresholding and speculative aggregation. *Symposium on Network and Distributed System Security (NDSS)*, 2010.

[54] B. Stock, J. Göbel, M. Engelberth, F. Freiling, and T. Holz. Walowdac - Analysis of a Peer-to-Peer Botnet. *European Conference on Computer Network Defense*, 2009.

[55] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna. Your botnet is my botnet: Analysis of a botnet takeover. In *ACM Conference on Computer and Communications Security (CCS)*, 2009.

[56] B. Stone-Gross, T. Holz, G. Stringhini, and G. Vigna. The Underground Economy of Spam: A Botmaster's Perspective of Coordinating Large-Scale Spam Campaigns. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2011.

[57] B. Stone-Gross, C. Kruegel, K. Almeroth, A. Moser, and E. Kirda. FIRE: FInding Rogue nEtworks. In *Annual Computer Security Applications Conference (ACSAC)*, 2009.

[58] G. Stringhini, C. Kruegel, and G. Vigna. Detecting Spammers on Social Networks. In *Annual Computer Security Applications Conference (ACSAC)*, 2010.

[59] Symantec Corp. State of spam & phishing report. `http://www.symantec.com/business/theme.jsp?themeid=state_of_spam`, 2010.

[60] B. Taylor. Sender reputation in a large webmail service. In *Conference on Email and Anti-Spam (CEAS)*, 2006.

[61] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song. Design and evaluation of a real-time URL spam filtering service. In *IEEE Symposium on Security and Privacy*, 2011.

[62] R. D. Twining, M. Williamson, M. Mowbray, and M. Rahmouni. Email Prioritization : reducing delays on legitimate mail caused by junk mail. *Technical Report*, 2004.

[63] S. Venkataraman, S. Sen, O. Spatscheck, P. Haffner, and . Song. Exploiting network structure for proactive spam mitigation. In *USENIX Security Symposium*, 2007.

[64] P. Wurzinger, L. Bilge, T. Holz, J. Goebel, C. Kruegel, and E. Kirda. Automatically generating models for botnet detection. *European Symposium on Research in Computer Security (ESORICS)*, 2010.

[65] Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov. Spamming botnets: Signatures and characteristics. *ACM SIGCOMM Computer Communication Review*, 2008.

[66] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Zhao, and Y. Dai. Uncovering Social Network Sybils in the Wild. In *ACM SIGCOMM Conference on Internet Measurement*, 2011.

[67] H. Yin, D. Song, M. Egele, C. Kruegel, and E. Kirda. Panorama: capturing system-wide information flow for malware detection and analysis. In *ACM Conference on Computer and Communications Security (CCS)*, 2007.