Defaming Botnet Toolkits:
A Bottom-Up Approach to Mitigating the Threat

Thomas Ormerod, Lingyu Wang, Mourad Debbabi,
Amr Youssef, Hamad Binsalleeh, Amine Boukhtouta, and Prosenjit Sinha
National Cyber-Forensics and Training Alliance CANADA
Computer Security Laboratory, Concordia University
Montreal, Quebec, Canada, H3G 2W1
Email: {t_ormero,wang,debbabi,youssef,h_binsal,a_boukh,p_sinh}@ciise.concordia.ca

Abstract—Botnets have become one of the most prevailing threats to today's Internet partly due to the underlying economic incentives of operating one. Botnet toolkits sold by their authors allow any layman to generate his/her own customized botnet and become a botmaster; botnet services sold by botmasters allow any criminal to steal identities and credit card information; finally, such stolen credentials are sold to end-users to make unauthorized transactions. Many existing botnet countermeasures meet inherent difficulties when they choose to target the botmasters or authors of toolkits, because those at the highest levels of this *food chain* are also the most technology-savvy and elusive. In this paper, we propose a different, bottom-up approach. That is, we defame botnet toolkits through discouraging or prosecuting the end-users of the stolen credentials. To make the concept concrete, we present a case study of applying the approach to a popular botnet toolkit, *Zeus*, with two methodologies, namely, reverse engineering and behavioural analysis.

Keywords-botnet; network security, identity theft; reverse engineering; Zeus.

I. INTRODUCTION

Botnets, networks of compromised machines controlled by botmasters, have become the leading threat to Internet security by providing an ideal platform for DDoS extortion, click fraud, SPAM, identity theft, and so on [1]. The success of botnets is partly due to a strong economic incentive underlying every level of the *food chain*: botnet toolkits sold by their authors allow any layman to generate his/her own customized botnet and become a botmaster; botnet services sold by botmasters allow any criminal to steal identities and credit card information; finally, such stolen credentials are sold to end-users to make unauthorized transactions.

The defence against botnets has drawn significant attention. Both network and host-based intrusion detection approaches have shown some success in detecting botnets (a more detailed review of related work will be given in Section II). However, misuse detection can be evaded by altering signature patterns, and anomaly detection suffers from high false positives when botnets mimic ordinary system behaviours. Although some preliminary efforts also exist on botmaster trace back, a countermeasure meets sig-

nificantly more difficulties when the objective is to capture the botmaster or authors of a toolkit, because those at the highest levels of the food chain are also the most technology-savvy and elusive.

Inspired by the bottom-up control of a biological food chain [2], we propose an approach of defeating a botnet toolkit through discouraging or prosecuting its end-users. We first present a generic framework describing the proposed approach and its two variations that defame a botnet toolkit from the security and profitability perspective, respectively. To make the concepts more concrete, we present a case study of the approach on the *Zeus* botnet toolkit, which is ranked as a top threat in the United States with more than 3.6 million infected systems [3]. For extracting necessary information for the defaming approach, we demonstrate two methodologies, namely, reverse engineering and behavioural analysis in the case study.

The advantage of our approach is that it targets the weakest point of a botnet food chain (the end-users). The cascading effect will eventually affect the top level of the chain (the toolkit author) by diminishing his/her profits when selling updates to existing users and new users. In addition, since we are attacking the business model, malware authors would need to change how they do business to circumvent our attack, which is more difficult than modifying the implementation of their toolkits.

The remainder of this paper is organized as follows. Section II discusses related work. Section III describes the generic framework of our approach. Section IV describes our findings from the case study of Zeus and the reverse engineering and behavioural methodologies. Finally, Section V concludes the paper and discusses future work.

II. RELATED WORK

In recent years there have been many approaches to detect and mitigate botnets and their related attacks. Ramachandran *et al.* [4] proposed using DNS blacklist counter-intelligence to determine botnet membership. However, this is limited to certain categories of SPAM botnets. Yen and Reiter [5] proposed TAMD, a system to detect a botnet by aggregating

network traffic. In a similar work Gu *et al.* [6] proposed BotMiner, which correlates traffic but on different characteristics to TAMD. BotHunter, also proposed by Gu *et al.* [7], is a system to passively detect individual bots by correlating IDS alerts to a predefined infection model. Goebel and Holz [8] created a system, Rishi, to detect IRC botnets through signature detection on known IRC nicknames. The main difficulty to these approaches is that botnets are constantly changing their behaviour to circumvent such solutions.

There are several works in the research field of identity theft analysis, detection and mitigation. Franklin *et al.* [9] presented two attacks on the trust and rating system of the Internet black market. However, their approach did not handle the case where credentials are not sold on the black market. Chandrasekaran *et al.* [10] detected phishing sites by submitting fake credentials and monitoring site behaviour; Holz *et al.* [11] analyzed data seized from 70 dropzones and provided metrics on the wealth of the underground economy; Birk *et al.* [12] used honeytokens [13] to track phishers. Our work is complementary to this; we submit honeytokens through botnets to attack the toolkit that created them and incorporate legal prosecution of the end-users.

Recent work has been proposed that attacks the price equilibrium of the Internet black market. Ford *et al.* [14] presented an attack at the ad revenue stream generated by botnets. This work justified the viability of an attack against the botnet business model. Li *et al.* [15] used honeypots to create uncertainty in the necessary rental size for a DDoS attack. These attacks are intended to affect the profit margins of botmasters by reducing the effectiveness of their services. While these attacks do show promise, they have not addressed a fundamental parameter in the existence of botnets, which is the malware authors who have created them. Our approach extends the economic attack to lower the author's profits by targeting the botnet toolkits they sell.

The closest work to ours is a framework proposed by Li and Schmitz [16]. They utilize spam traps to submit credentials to phishing sites and also use phoneybots to submit credentials to botnets such as Zeus. However, their target is the end-users, or money mules, while our intent is to discredit a particular toolkit and the malware authors behind it. This approach will reduce the variety of toolkits available to potential botmasters and directly affect the widespread use of them to perform botnet related crimes. In addition, we utilize reverse engineering results from our analysis of the Zeus botnet toolkit to describe in technical detail the method to inject falsified information into the black market.

III. Proposed Framework

The goal of the proposed framework is to discredit a botnet toolkit and ultimately reduce its sales. We attack credibility on two fronts: profitability of the toolkit in respect to the use and sale of identity information and security of the toolkit in respect to its ability to protect

its users from prosecution. We propose two variants to this approach: reducing a toolkit's profitability by flooding it with false information, and making the toolkit insecure by submitting honeytokens to the botnets that aid in arresting and prosecuting the end-users of the stolen information. The framework is illustrated in Figure 1, which shows the two variants and the surrounding process.

A. First Variant

The first variant's intent is to dilute the stolen identity information from a botnet with false credentials. It works as follows:

1) Monitor current botnet toolkit trends and select the leading toolkit used for identity theft as our target.
2) Acquire as many samples of botnet binaries as possible from honeynets, antivirus companies, financial institutions and security forums.
3) Determine targeted web sites and targeted fields by analyzing malware samples.
4) Generate false identity information.
5) Submit identity information to the botmasters through the framework.

At this stage the framework ends and the fraudulent credentials are propagated through the black market economy. The result of diluting the information has a cascading effect. Each subsequent party discredits the next party in the chain. Firstly, the information sold on the black market will generate less than expected profits for the buyer as many of the credentials are false. Unprofitable sales will discredit the seller, which will have an impact on future sales. The sellers of the information, who are also the botmasters, will attribute their defamation to their botnet, and more specifically, the malware toolkit used to create it. Their response will be to discredit the toolkit they used and no longer purchase new revisions.

Here we now have an effect not just on the economy of the identity theft market, but on the incomes of the malware authors. If future sales of the toolkit can be sufficiently reduced, then the authors may decide that supporting their toolkit is no longer profitable. We believe that the programmers of these toolkits are the minority of Internet criminals and if, through these attacks, we can dissuade them from creating botnet toolkits for material gain, we will reduce the amount of innovation these botnets possess in future attacks. In addition, this approach may frustrate potential botmasters from purchasing toolkits by continually reducing the effectiveness of the botnets that these toolkits create. The framework repeats, re-evaluating the currently most prevalent and dangerous botnet toolkit and reapplying the approach. As a toolkit loses its popularity, the framework adjusts to tackle the next leading threat.
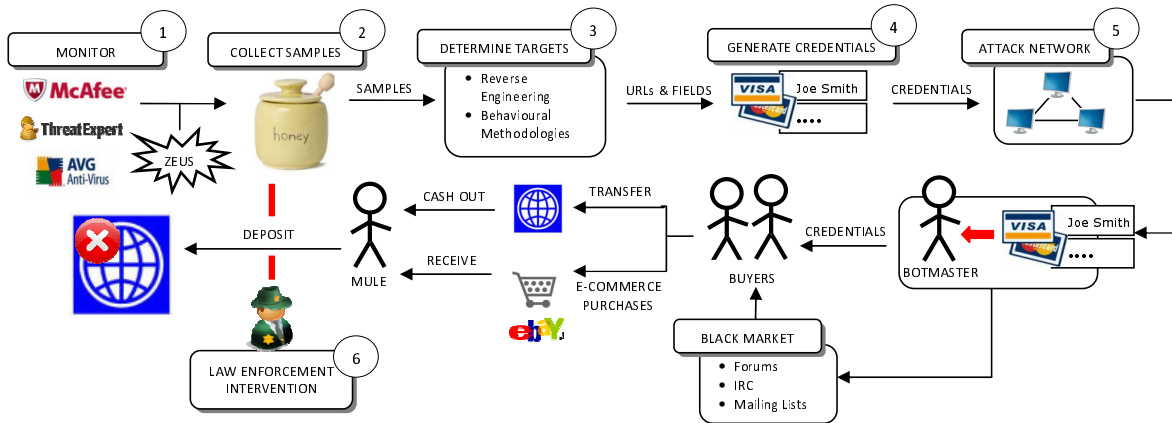
Figure 1. Botnet toolkit defamation process.

B. Second Variant

The second variant of our framework is an extension of the first one. The intent of this approach is to discredit the security for the end-users of the stolen information. The approach contains one additional step that occurs after the credentials propagate through the Internet black market:

6) Monitor account usage and make arrests when funds withdrawn and purchases received.

Coordination between law enforcement and the various financial institutions is paramount for this approach to succeed. The criminals must be convinced that the accounts are valid and the transactions are working. This can be accomplished by providing funds to the accounts and through coordinating the transfers between institutions. Law enforcement's role is to monitor these accounts and attempt to arrest individuals as they try to extract the funds. With foreknowledge of the accounts being used, the success rate of apprehending account holders should be higher.

The impact of apprehension and prosecution of end-users of the stolen information, further adds to discrediting the involved parties. Firstly, any arrests will remove some end-users from further involvement in the black market. Reduction in supply of potential end-users, as well as increased concern over security will raise the cost of "hiring" them, which in turn reduces the future profits of the purchasers of the stolen identities. The purchasers of stolen identity information will continue to propagate the effects of arrests within their ranks, by attacking the credibility of the identity information they purchased. This impugns the reputations of the sellers of the stolen identity information and drives down the market price for the information. Finally, sellers of stolen identity information, who are also botmasters, will discredit the toolkits they used to create their botnets.

C. Technical Approach

We now discuss three methods to determine the targeted web sites. The first method is to reverse engineer malware samples to determine what sites they are targeting and in particular what identity information they are after. In the case of botnet toolkits, malware samples generated by the same version of the toolkit will have a similar code structure even though they may appear completely different at first glance. This will allow a reverse engineering analysis to be compiled into scripts that may be run on each malware sample to repeat the reverse engineering steps. In Section IV we describe our results from such an analysis. In this section we also discuss the two other methods to determine the targeted web sites that consist of a behaviour methodologies approach that stemmed from our analysis of Zeus and have roots in side-channel attacks [17]. The first methodology is to look for anomalous filesystem activity related to the browser process to indicate when web page harvesting occurs. In particular we are trying to determine when the malware stores identity information on the hard drive prior to sending it to the botmaster. The second methodology is to look for anomalous network activity when a user browses certain web pages. This method should better handle malware that does not use an intermediate file store to save the stolen credentials.

IV. ZEUS CASE STUDY

We now illustrate the technical details of the framework through examples of how it would be applied to the Zeus botnet toolkit. This section includes the reverse engineering steps needed to infiltrate the botnet and inject the false information into the black market, and the behavioural methodologies that could be applied to future toolkits.

Zeus is an identity theft malware that can be purchased in the underground economy. It is a fully fledged software toolkit that is configurable, contains a user manual, and is
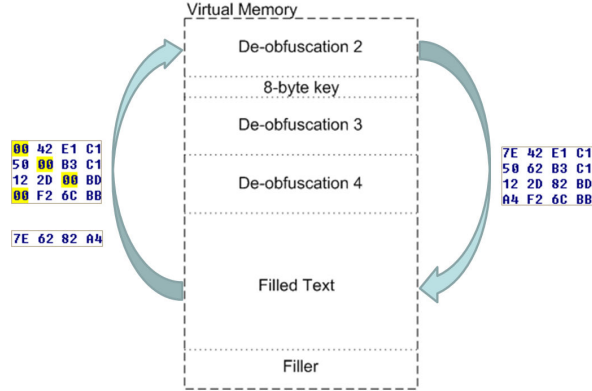
Figure 2. The result from the second deobfuscation routine

an ideal fit for our framework. The Zeus toolkit contains a builder executable that creates a Zeus bot executable using the user supplied configuration files. We begin by performing a reverse engineering analysis on the generated Zeus executables. In this paper we focus on extracting the encryption key and extracting the targeted website URLs for use in our framework. With these two pieces of information we are able to join an existing Zeus botnet, pretend to be an active member, and submit the falsified identity information.

To join a botnet we use scripts written during the reverse engineering analysis to automatically extract the encryption key and the configuration information embedded in the bot binary. This static configuration directs us to a URL, which contains the dynamic configuration for the botnet. This file is a list of targeted URLs and their respective extraction rules. We then use the encryption key to decrypt the configuration file to determine the web sites that are being targeted and the identity information sought.

In order to extract the encryption/decryption key and the static configuration structure we first need to remove the obfuscation layers that Zeus employs to hide its internal code structure.

A. Deobfuscation Process

The generated Zeus binary contains four segments: a text/code segment, an imports segment, a resources segment, and a data segment. We begin our analysis at the malware Entry Point (EP), which resides in the text/code segment. The initial analysis of the disassembly showed that only a small portion of the text/code block contains valid computer instructions. This indicates that most of the binary is obfuscated, which means the computer cannot use this code directly and must first use procedures in the valid code blocks of the binary to unravel these sections.

Using the IDA Pro debugger [18], we are able to debug the malware and step through the instructions to analyze and understand the logic of the deobfuscation routines. Each routine reveals some information that is used by the other

routines until all obfuscation layers are removed. The first deobfuscation routine contains a 4-byte long decryption key and a one-byte long seed value. These two values are used to decrypt a block of data from the text/code segment using an add operation and then write the decrypted data into virtual memory. The result of the first deobfuscation routine reveals three new deobfuscation routines. We now explain the main logic of the first of these new routines, which we refer to as the second deobfuscation routine.

1) First, two binary blocks from the text/code segment are concatenated together and then written to virtual memory. This new block has two sections; the first contains data with many zero value bytes and the second is used to fill these zero bytes.
2) Next, the routine scans through every byte of the first segment and when it encounters a hole (zero byte) in the data, it will overwrite with the next available byte in the filler text block. This is repeated until all holes are filled (See Figure 2).

Creating the filled text segment is the main purpose of the second deobfuscation routine. However, this text segment is still not machine readable computer instructions. The third deobfuscation routine is used to decrypt the filled text. This routine is similar to the first deobfuscation routine with two differences: it utilizes an 8-byte key, and uses an eXclusive-OR (XOR) operation as the decryption algorithm. Finally, the last deobfuscation routine contains heavy computations to initialize and prepare some parameters for the rest of the malware operations. It uses the decrypted bytes revealed by the previous routines to modify the rest of the text/code segment. After this routine completes execution, we have access to the original malware machine code in its entirety. In our extraction scripts, this portion of the deobfuscation routine is an emulation of the disassembled code and is not represented in a high level format.

Algorithm IV.1: DECRYPT_URL(enc_url)

String url = new String(enc_url.length());
for $i \leftarrow 0$ **to** enc_url.length()
do $\begin{cases} \textbf{if } (i\%2 == 0) \\ \quad \textbf{then } \text{url}[i] = (\text{enc\_url}[i] + \text{0xF6} - i * 2) \% \text{0xFF}; \\ \\ \quad \textbf{else } \text{url}[i] = (\text{enc\_url}[i] + \text{0x7} + i * 2) \% \text{0xFF}; \end{cases}$
return (url)

B. Key and Configuration Extraction

As we learned from Section IV, the Zeus botnet has a static configuration structure that includes an encryption key and a dynamic configuration URL. During the deobfuscation process, this structure is recovered at memory location (0x416000) (See Figure 3). All information in the

structure is completely deobfuscated except for two URLs: "url_compip" and "url_config". These URLs can be deobfuscated using Algorithm IV.1. The string "url_config" is the web location to download the dynamic configuration file for the botnet. This file contains the targeted URLs. The static configuration structure also contains an RC4 substitution table that was generated from an encryption key provided by the botmaster when the Zeus payload was generated. This substitution table is valuable because it can be used to decrypt communications of this Zeus botnet as well as being able to decrypt the dynamic configuration file.

We begin the process by downloading the dynamic configuration file from the web location "url_config". Once downloaded, the dynamic configuration file can be decrypted using the RC4 algorithm with the substitution table provided. Next, we extract the URLs that are being targeted from this file. These URLs satisfy step 3 of our attacks and are the input to the identity information generator in step 4. We now use a machine infected with the binary sample from, which we extracted the URLs, and submit our generated identity information to those same URLs. The Zeus bot will detect the website and the submitted credentials and send them to the botmaster. Figure 1 outlines the rest of the propagation process. After the botmaster receives the credentials, they will sell the information on the black market to buyers, or they themselves will act as buyers. The buyers next perform transactions with other criminals, referred to as money mules, using the credentials. In the case of credit card numbers they will purchase goods from e-commerce websites, in the case of online bank credentials they will perform Internet transfers to other accounts. In the attack the receiver of the goods can be arrested from their pickup location, and the receiver of the transferred funds can be arrested when they cash out the accounts.

C. Behavioural Methodologies

Reverse engineering provides us with exact results for the targeted URLs. However, it can be a time consuming process as malware authors utilize obfuscation techniques to prevent binary snooping. Since we are interested in applying our method to all makes of malware not just Zeus, we have developed behavioural methodologies for determining the affected URLs to handle the general malware case.

Behavioural methodologies allow us to determine the URLs for any malware type, malware version, and individual build of the malware, as they detect the URLs independently of the binary structure and the obfuscation techniques employed by the malware. We employ two techniques that cover a broader range of malware: detecting anomalous filesystem activity and detecting anomalous network activity.

1) Anomalous filesystem activity: To detect website harvesting activity for certain types of malware we can employ anomalous filesystem activity detection. This method proves effective against Zeus, as Zeus does not immediately
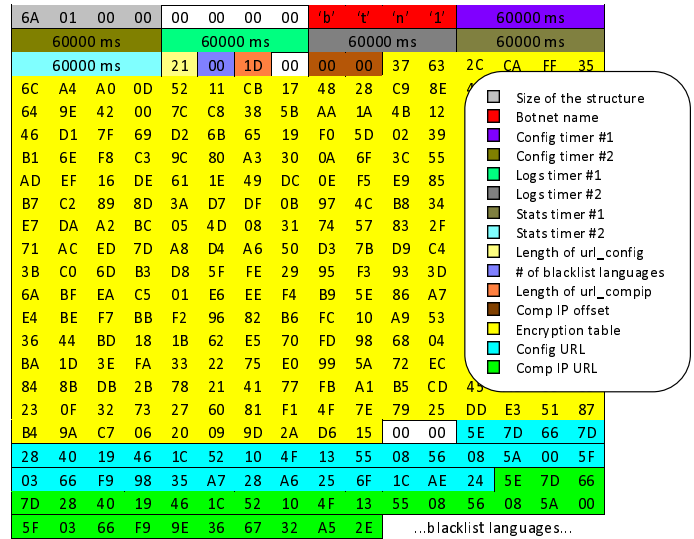
Figure 3. Static configuration structure

send the stolen identity information to the botmaster, but first stores the credentials on the local hard drive. After a configurable time interval, Zeus will send all harvested information.

We can use this behaviour to detect when malware harvests page information to determine the targeted URLs. Using the Windows Performance Monitor tool [19] we monitor the "IO Write Bytes/sec" counter for the browser process on both infected and clean machines to detect irregular disk activity as an indicator that page harvesting occurred. The method is performed on a variety of popular e-commerce and Internet banking sites for each of the acquired malware samples. This allows us to both group malware samples together as potentially belonging to the same botmaster and to determine what false identity information to generate and submit to the botnets the malware samples were created for.

2) Anomalous network activity: Malware may not store the harvested information, but instead transmit the information as it is gathered. By monitoring outgoing network activity and detecting anomalous transmissions we can determine when page information has been harvested. We, again, interact with popular e-commerce and online banking sites on both infected and clean machines and look for anomalous network activity to create a list of targeted websites. This approach may best work on malware that does not use an intermediary file to store credentials.

V. Conclusion and Future Work

In this paper we introduced a framework to combat identity theft toolkits. The technical challenge of the approach is to determine the websites that are targeted by instances of a botnet toolkit. We discussed reverse engineering results from an analysis we performed on Zeus that allows us to

automate this process for Zeus binary samples. In addition, two behavioural methodologies were proposed that may handle the general malware case.

One problem with our approach is we do not know if we are flooding many botnets with identity information from many botmasters or many botnets from one or a few botmasters. Even if we collect many binaries that use different drop zones, they all may belong to the same botmaster. Flooding the botnets that belong to our collected samples may only be affecting one or a few botmasters business models while other botmasters that use the same toolkit are unaffected. Currently we rely on an assumption that the most active botnets will likely compose our acquired malware samples.

Future work will be to research additional behavioural methodologies for use in our framework. In addition we will look at evaluation methods to determine how much false identity information is necessary to have a major effect on the toolkit market. We will also discuss any legal barriers, such as extortion laws, to our approach.

ACKNOWLEDGMENT

REFERENCES

[1] N. Kshetri, "The simple economics of cybercrimes," *IEEE Security & Privacy*, vol. 4, no. 1, pp. 33–39, Jan. 2006.

[2] J. Shurin, D. Gruner, and H. Hillebrand, "All wet or dried up? real differences between aquatic and terrestrial food webs," *Proceedings of the Royal Society B: Biological Sciences*, vol. 273, no. 1582, pp. 1–9, 01 2006.

[3] P. Coogan. (2009, August) Zeus, king of the underground crimeware toolkits. Symantec. Last accessed: 19/04/2010. [Online]. Available: http://www.symantec.com/connect/blogs/zeus-king-underground-crimeware-toolkits

[4] A. Ramachandran, N. Feamster, and D. Dagon, "Revealing botnet membership using dnsbl counter-intelligence," in *Proceedings of USENIX SRUTI06*, July 2006, pp. 49–54.

[5] T.-F. Yen and M. K. Reiter, "Traffic aggregation for malware detection," in *Proceedings of the Fifth GI International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA08)*, 2008 2008, pp. 207–227.

[6] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection," in *Proceedings of the USENIX Security Symposium*. Berkeley, CA, USA: USENIX Association, August 2008, pp. 139–154.

[7] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "Bothunter: detecting malware infection through ids-driven dialog correlation," in *SS'07: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*. Berkeley, CA, USA: USENIX Association, 2007, pp. 1–16.

[8] J. Goebel and T. Holz, "Rishi: Identify bot contaminated hosts by irc nickname evaluation," in *Proceedings of USENIX HotBots07*. Berkeley, CA, USA: USENIX Association, 2007.

[9] J. Franklin, V. Paxson, A. Perrig, and S. Savage, "An inquiry into the nature and causes of the wealth of internet miscreants," in *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS'07)*, 2007, pp. 375–388.

[10] M. Chandrasekaran, R. Chinchani, and S. Upadhyaya, "Phoney: Mimicking user response to detect phishing attacks," in *Proceedings of the 2006 International Symposium on the World of Wireless, Mobile and Multimedia Networks*, 2006, pp. 5pp.–672.

[11] T. Holz, M. Engelberth, and F. Freiling, "Learning more about the underground economy: A case-study of keyloggers and dropzones," University of Mannheim, Tech. Rep. Reihe Informatik TR-2008-006, December 2008.

[12] D. Birk, S. Gajek, F. Gröbert, and A.-R. Sadeghi, "A forensic framework for tracing phishers," in *IFIP Summer School on The Future of Identity in the Information Society*, Karlstad, Sweden, 2007.

[13] L. Spitzner. (2003, July) Honeytokens: The other honeypot. Symantec. Last accessed: 19/04/2010. [Online]. Available: http://www.symantec.com/connect/articles/honeytokens-other-honeypot

[14] R. Ford and S. Gordon, "Cent, five cent, ten cent, dollar: Hitting botnets where it really hurts," in *Proceedings of New Security Paradigms Workshop*, 2006, pp. 3–10.

[15] Z. Li, Q. Liao, and A. Striegel, "Botnet economics: Uncertainty matters," in *Proceedings of the 7th Workshop on the Economics of Information Security (WEIS'08)*, 2008.

[16] S. Li and R. Schmitz, "A novel anti-phisihng framework based on honeypots," in *Proceedings of the 4th annual Anti-Phishing Working Groups eCrime Researchers Summit*, September 2009.

[17] D. Molnar, M. Piotrowski, D. Schultz, and D. Wagner, "The program counter security model: Automatic detection and removal of control-flow side channel attacks," in *Proceedings of the 8th International Conference on Information Security and Cryptology*, Seoul, Korea, December 2005, pp. 156–168.

[18] Idapro - multi-processor disassembler and debugger. Last accessed: 19/04/2010. [Online]. Available: http://www.hex-rays.com/idapro/

[19] M. E. Russinovich and D. A. Solomon, *Microsoft Windows Internals: Windows Server 2003, Windows XP, and Windows 2000*, 4th ed. Microsoft Press, 2004.