

Learning More About the Underground Economy: A Case-Study of Keyloggers and Dropzones

Thorsten Holz^{1,2} Markus Engelberth¹ Felix Freiling¹

¹ Laboratory for Dependable Distributed Systems, University of Mannheim, Germany
<holz, engelberth, freiling@informatik.uni-mannheim.de>

² Secure Systems Lab, Vienna University of Technology, Austria

Abstract. We study an active underground economy that trades stolen digital credentials. In particular, we investigate keylogger-based stealing of credentials via *dropzones*, anonymous collection points of illicitly collected data. Based on the collected data from more than 70 dropzones, we present an empirical study of this phenomenon, giving many first-hand details about the attacks that were observed during a seven-month period between April and October 2008. We found more than 33 GB of keylogger data, containing stolen information from more than 173,000 victims. Analyzing this data set helps us better understand the attacker’s motivation and the nature and size of these emerging underground marketplaces.

1 Introduction

With the growing digital economy, it comes as no surprise that criminal activities in digital business have lead to a digital underground economy. Because it is such a fast-moving field, tracking and understanding this underground economy is extremely difficult. Martin and Thomas [20] gave a first insight into the economy of trading stolen credit card credentials over open IRC channels. The “blatant manner” in which the trading is performed with “no need to hide” [20] is in fact staggering. A large-scale study of similar forms of online activity was later performed by Franklin et al. [10]. The result of this study is that Internet-based crime is now largely profit-driven and that “the nature of this activity has expanded and evolved to a point where it exceeds the capacity of a closed group” [10]. In other words, digital and classical crime are merging.

In general, it is hard to estimate the real size of the underground economy. This is because the only observable evidence refers to *indirect* effects of underground markets. For example, both previous studies [10,20] did not observe real trading, but only *announcements* of trading and *offers* of stolen credentials in public IRC channels. It is in fact a valid question how much of the offered data really belongs to online scams—rather than being just the result of “poor scum nigerians and romanians try[ing] to make 20\$ deals by ripping eachother off” [4].

In this paper, we report on measurements of the *actual kind and amount* of data that is stolen by attackers from compromised machines, i.e., we *directly* observe the goods that can be traded at an underground market. Obviously, this data gives us a much better basis for estimating the size of the underground economy and also helps to understand the attacker’s motivation.

It may seem as if direct observations of illicitly traded goods are much harder to obtain than indirect ones. In this paper we show that this must not be the case. In particular, we focus on the newly emerging threat of keyloggers that communicate with the attacker through so-called *dropzones*. A dropzone is a publicly writable directory on a server in the Internet that serves as an exchange point for keylogger data. The attack is visualized in Figure 1. The attacker A first infects victims V_1 , V_2 and V_3 with keylogging malware. This malware secretly collects credentials that victims use to authenticate to online services like a bank P_1 or a webmailer P_2 . After collecting these credentials, the malware running on a compromised machine sends them to the dropzone, where the attacker can pick them up and start to abuse them [9,30,31,32,34].

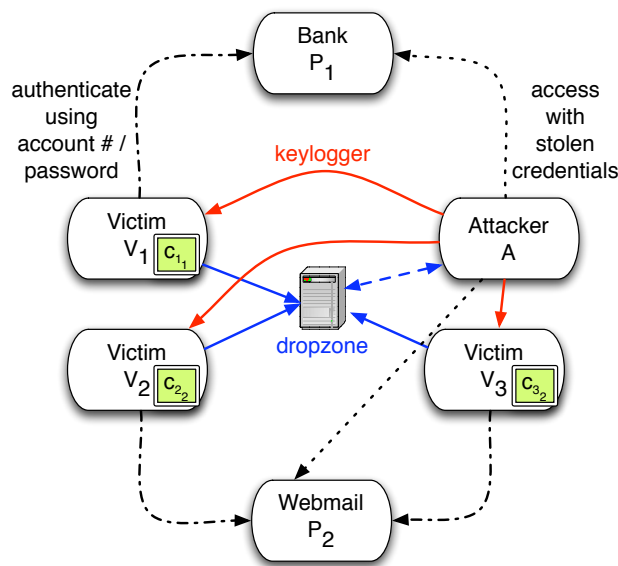


Fig. 1: Schematic overview of keylogger-based attacks using dropzones.

We analyzed these keylogger-based attacks by first collecting keyloggers with different techniques such as honeypots [27] or spamtraps, and then executing them within an instrumented environment [37], thereby extracting the location of the dropzone. By accessing the dropzone directly, we harvested the keylogger data just like the attacker would have done this. We perform our case study using two different classes of keyloggers called *Limbo/Nethell* and *Zeus/Zbot/Wsnpoem*. We give details of attacks we observed during a seven-month period between April and October 2008. In particular, we were able to harvest a total of 33 GB of keylogger data from more than 70 unique dropzones, resulting in information about stolen credentials from more than 173,000 compromised machines. We present the results of a statistical analysis of this data. To our knowledge, this is the first time that it has been possible to perform such an analysis on *stolen* data on such a large scale. It gives rather credible answers to questions

about the type and the amount of data criminals steal, which allows us to study the underground economy since these stolen credentials are marketable goods. For example, we recovered more than 10,700 stolen online bank account credentials and over 149,000 stolen email passwords, potentially worth several million dollars on the underground market. Our analysis shows that this type of cybercrime is a profitable business, allowing an attacker to potentially earn hundreds or even thousands of dollars per day.

1.1 Related Work

Besides the related work discussed previously, this paper touches on a several related research areas. In the field of phishing prevention and mitigation, there has been some work specific to attacks based on email and fake websites [5,11]. Chandrasekaran et al. [5] generate fake input and investigate a site's response to detect phishing sites. Gajek and Sadeghi [11] use fake credentials to track down phishers. Our work is complementary to this work: we study the actual dropzone and infer from this data more information about the extent and size of the attack.

Recently Kanich et al. studied the *conversion rate* of spam, i.e., the probability that an unsolicited e-mail will ultimately elicit a *sale* [15]. This is another example of a direct observation of the underground economy and provides a different point of view into the market mechanisms behind cybercrime.

The keylogger-based attacks we study in this paper can be stopped using different kinds of techniques, for example multi-factor authentication, biometrics, or special hardware or software. While techniques like SpoofGuard [7], Dynamic Security Skins [8], or Transport Login Protocol [6] can protect against certain forms of these attacks, e.g., classical phishing attacks, they can not stop keylogger-based attacks that we study in this paper. Preventing this kind of attacks is harder since the user machine itself is compromised, which allows the malicious software to steal credentials directly as the victim performs the login procedure. Modern keyloggers also defeat simple tricks to conceal the entered password as proposed by Herley and Florêncio [12]. However, malware prevention methods and systems that protect confidential information can defend against this kind of attacks [21,35].

1.2 Summary of Contributions

To summarize, our work presented in this paper makes the following contributions: We investigate keylogging attacks based on dropzones and provide a detailed analysis of the collected data, giving a first-hand insight into the underground economy of Internet criminals from a unique and novel viewpoint. We believe that our method can be generalized to many other forms of credential-stealing attacks, such as phishing attacks.

We argue that combined with prices from the underground economy, our study gives a more precise estimate of the dangers and potential of the black market than indirect measures performed previously [10,20]. Together with these prior studies, we hope that our results help to relinquish the common mindset we often see with politicians and commercial decision-makers that we do not need to track down and prosecute these criminals because it is too costly. We feel that the sheer size of the underground economy now and in the future will not allow us to neglect it.

Paper Outline. We describe in Section 2 in more detail how keylogging based attacks work and introduce two different families of keyloggers. In Section 3, we introduce our analysis setup and present statistics for the dropzones we studied during the measurement period. We analyze the collected data in Section 4 using five different categories and briefly conclude the paper in Section 5 with an overview of future work.

Data Protection and Privacy Concerns. The nature of data analyzed during this study is very sensitive and often contains personal data of individual victims. We are not in a position to inform each victim about the security breach and therefore decided to hand over the full data set to AusCERT, Australia’s National Computer Emergency Response Team. This CERT works together with different banks and other providers to inform the victims. We hope that the data collected during this study can help to recover from the incidents and more damage is prevented.

2 Background: Keylogger-based Attacks

Figure 1 provides a schematic overview of keylogger-based attacks using dropzones. Each victim V_i has a specific credential c_{i_j} to authenticate at provider P_j to use the service. For example, P_1 is an online banking website and V_1 uses his account number and a password to log in. The attacker A uses different techniques to infect each victim V_i with a keylogger. Once the victim V_i is infected, the keylogger starts to record all keystrokes: A defines in advance which keystrokes should be logged and the malware only records these. For example, A can specify that only the login process of an online banking website should be recorded. The malware then observes the values entered in input fields on the website and sends this information to a dropzone. This dropzone is the central collection site for all harvested information. The attacker can access the dropzone, extract the stolen credentials, and use them to impersonate at P_j as V_i .

2.1 Studying the Attack

The practical challenge of our approach is to find a way to access the harvested information so that it can be used for statistical analysis. To study this attack, we use the concept of *honeypots*, i.e., information system resources whose value lies in unauthorized or illicit use of that resource [27]. We play the role of a victim V_i and react on incoming attacks in the same way a legitimate victim would do. For example, we use *spamtraps*, i.e., email accounts used to collect spam, and open email attachments to emulate the infection process of malware that propagates with the help of spam. Furthermore, we also visit links contained in spam mails with client-side honeypots to examine whether or not the spammed URL is malicious and the website tries to install a keylogger via a drive-by download [28,36]. Using these techniques, our honeypot can be infected with a keylogger in an automated way and we obtain information about the attack vector.

After a successful infection, we extract the sample from the honeypot for further analysis. We perform dynamic analysis based on an analysis tool called CWSandbox [37] since static analysis can be defeated by malware using many different techniques [17,24,26]. CWSandbox executes the malware in a controlled environment and

analyzes the behavior of the sample during runtime by observing the system calls issued by the sample. As a result, we obtain an analysis report that includes for example information about changes to the filesystem or the Windows registry, and all network communication generated by the sample during the observation period.

When executing, the keylogger typically first contacts the dropzone to retrieve configuration information. The configuration file commonly includes a list of websites that should be monitored for credentials and similar customization options for the malware. From an attacker's perspective, such a modus operandi is desirable since she does not have to hardcode all configuration options during the attack phase, but can dynamically re-configure which credentials should be stolen after the initial infection. This enables more flexibility since the attacker can configure the infected machines on demand. By executing the keylogger within our analysis environment and closely monitoring its behavior, we can identify the dropzone in an automated way since the keylogger contacts the dropzone at an early stage after starting up.

However, certain families of keylogger already contain all necessary configuration details and do not contact the dropzone: only after keystrokes that represent a credential are observed by these keyloggers, they send the harvested information to the dropzone. In order to study this in a more automated fashion, we need some sort of *user simulation* to actually simulate a victim V . Note that we do not need to generically simulate the full behavior of a user, but only simulate the aspects of user interaction that are *relevant* for keyloggers, e.g., entering credentials in an online banking application or logging into a webmail account. The keylogger then monitors this behavior and sends the collected information to the dropzone, and we have successfully identified the location of a dropzone in an automated way. More information about the actual implementation of user activity simulation is provided in Section 3.1.

2.2 Technical Details of Analyzed Keyloggers

To exemplify a technical realization of the methodology introduced in this paper, we analyzed in detail two different families of keyloggers that are widespread in today's Internet: *Limbo/Nethell* and *Zeus/Zbot/Wsnpoem*. We provide a short overview of both families in this section. More details and examples are available in a technical report [13].

Limbo/Nethell. This family of malware typically uses malicious websites and drive-by download attacks as attack channel to infect the victims who are lured by social engineering tricks to visit these websites. The malware itself is implemented as a *browser helper object* (BHO), i.e., a plugin for Internet Explorer that can respond to browser events such as navigation, keystrokes, and page loads. With the help of the interface provided by the browser, Limbo can access the Document Object Model (DOM) of the current page and identify sensitive fields which should be monitored for credentials (*form grabbing*). This enables the malware to monitor the content of these fields and defeats simple tricks to conceal the entered password as proposed by Herley and Florêncio [12]. The malware offers a flexible configuration option since the sites to be monitored can be specified during runtime in a configuration file. Upon startup, the malware contacts the dropzone to retrieve the current configuration options from there. Furthermore, this malware has the capability to steal cookies and to extract information

from the Protected Storage (*PStore*). This is a mechanism available in certain versions of Windows which provides applications with an interface to store user data [22] and many applications store credentials like username/password combinations there.

Once a credential is found, the harvested information is sent to the dropzone via a HTTP request to a specific PHP script installed at the dropzone, e.g., `http://example.org/datac.php?userid=21102008_110432_2025612`. This example depicts the initial request right after a successful infection with which the keylogger registers the newly compromised victim. The `userid` parameter encodes the infection date and time, and also a random victim ID. By observing the network communication during the analysis phase, we can automatically determine the network location of the dropzone. The dropzone itself is implemented as a web application that allows the attacker amongst other tasks to browse through all collected information, search for specific credentials, or instruct the victims to download and execute files. We found that these web applications often contain typical configuration errors like for example world-readable directory listings that lead to insecure setups, which we can take advantage of to obtain access to the full data set.

Zeus/Zbot/Wsnpoem. The attack channel for this family of malware is spam mails that contain a copy of the keylogger as an attachment. The emails use common social engineering tricks, e.g., pretending to be an electronic invoice, in order to trick the victim into opening the attachment. In contrast to Limbo, which uses rather simple techniques to steal credentials, Zeus is technically more advanced: the malware injects itself into all user space processes and hides its presence. Once it is successfully injected into Internet Explorer, it intercepts HTTP POST requests to observe transmitted credentials. This malware also steals information from cookies and the Protected Storage. All collected information is periodically sent to the dropzone via HTTP requests. The dropzone itself is implemented as a web application and the stolen credentials are either stored in the filesystem or in a database. Again, insecure setups like world-readable directory listings enable the access to the full dropzone data, allowing us to monitor the complete operation of a certain dropzone.

Similar to Limbo, Zeus can also be dynamically re-configured: after starting up, the malware retrieves the current configuration file from the dropzone. The attacker can for example specify which sites should be monitored (or not be monitored) for credentials. Furthermore, the malware can create screenshot of 50×50 pixels around the mouse pointer taken at every left-click of the mouse for specific sites. This capability is implemented to defeat *visual keyboards*, i.e., instead of entering the sensitive information via the keyboard, they can be entered via mouse clicks. This technique is used by different banks and defeats typical keyloggers. However, by taking a screenshot around the current position of the mouse, an attacker can also obtain these credentials. In addition, the configuration file also specifies for which sites man-in-the-middle attacks should be performed: each time the victim opens such a site, the request is transparently redirected to another machine, which hosts some kind of phishing website that tricks the victim into disclosing even more credentials. Finally, several other configuration options like DNS modification on the victim's machine or update functionality are available.

3 Studying Keylogger-based Attacks

In this section, we introduce the analysis and measurement setup, and present general statistics about the dropzones. The next section then focusses on the results of a systematic study of keylogger-based attacks using keylogger and a dropzone as outlined in the previous sections. All data was collected during a seven-month measurement period between April and October 2008.

3.1 Improving Analysis by Simulating User Behavior

We developed a tool called *SimUser* to simulate the behavior of a victim V_i after an infection with a keylogger. The core of SimUser is based on *AutoIt*, a scripting language designed for automating the Windows GUI and general scripting [1]. It uses a combination of simulated keystrokes, mouse movement, and window/control manipulation in order to automate tasks. We use AutoIt to simulate arbitrary user behavior and implemented SimUser as a frontend to enable efficient generation of user profiles. SimUser itself uses the concept of *behavior templates* that encapsulate an atomic user task, e.g., opening a website and entering a username/password combination in the form fields to log in, or authenticating against an email server and retrieving emails. We implemented 17 behavior templates that cover typical user tasks which require a credential as explained before. These templates can be combined in an arbitrary way to generate a profile that simulates user behavior according to specific needs.

In order to improve our analysis, we execute the keylogger sample for several minutes under the observation of CWSandbox. During the execution, SimUser simulates the behavior of a victim, which browses to several websites and fills out login forms. In the current version, different online banking sites, free webmail providers, as well as social networking sites are visited. Furthermore, CWSandbox was extended to also simulate certain aspects of user activity, e.g., generic clicking on buttons to automatically react on user dialogues. We also store several different credentials in the Windows Protected Storage of the analysis machine as some kind of *honeypot*. By depositing some credentials in the Protected Storage, we can potentially trigger on more keyloggers.

Simulating user behavior enables us to learn more about the results of a keylogger infection, e.g., we can detect on which sites it triggers and what kind of credentials are stolen. The whole process can be fully automated and we analyzed more than 2,000 keylogger samples with our tools as explained in the next section. Different families of keyloggers can potentially use distinct encodings to transfer the stolen credentials to the dropzone and the dropzone itself uses different techniques to store all stolen information. In order to fully analyze the dropzone and the data contained there, we thus need to manually analyze this communication channel once per family. This knowledge can then be used to extract more information from the dropzone for all samples of this particular family. To provide evidence of the feasibility of this approach, we analyzed two families of keyloggers in detail, as we explain next. Note that even if we cannot fully decode the malware's behavior, we can nevertheless reliably identify the network location of the dropzone based on the information collected during dynamic analysis. This information is already valuable since it can be used for mitigating the dropzone, the simplest approach to stop this whole attack vector.

3.2 Measurement Setup

With the help of CWSandbox, we analyzed more than 2,000 unique Limbo and Zeus samples collected with different kinds of spamtraps and honeypots, and user submissions at `cwsandbox.org`, in the period between April and October 2008. Based on the generated analysis reports, we detected more than 140 unique Limbo dropzones and 205 unique Zeus dropzones. To study these dropzones, we implemented a monitoring system that periodically collects information like for example the configuration file.

For 69 Limbo and 4 Zeus dropzones we were able to *fully* access all logfiles collected at that particular dropzone. This was possible since these dropzones were configured in an insecure way by the attackers, enabling unauthenticated access to all stolen credentials. The remaining dropzones had access controls in place which prevented us from accessing the data. We periodically collected all available data from the open dropzones to study the amount and kind of stolen credentials to get a better understanding of the information stolen by attackers. In total, our monitoring system collected 28 GB of Limbo and 5 GB of Zeus logfiles during the measurement period.

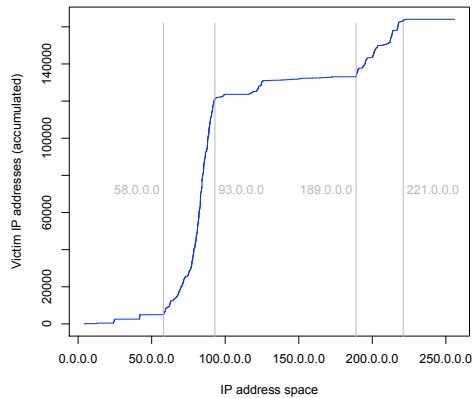
3.3 Analysis of Limbo Victims

To understand the typical victims of keylogger attacks, we performed a statistical analysis of the collected data. The number of unique infected machines and the amount of stolen information per Limbo dropzone for which we had full access is summarized in Table 1. The table is sorted by the number of unique infected machines and contains a detailed overview of the top four dropzones. In total, we collected information about more than 164,000 machines infected with Limbo. Note that an infected machine can potentially be used by many users, compromising the credentials of many victims. Furthermore, the effective number of infected machines might be higher since we might not observe all infected machines during the measurement period. The numbers are thus a lower bound on the actual number of infected machines for a given dropzone. The amount of information collected per dropzone greatly varies since it heavily depends on the configuration of the keylogger (e.g., what kind of credentials should be harvested) and the time we monitored the server. The dropzones themselves are located in many different Autonomous Systems (AS) and no single AS dominates. The country distribution reveals that many dropzones are located in Asia or Russia, but we found also many dropzones located in the United States.

We also examined the *lifetime* for each dropzone and the *infection lifetime* of all victims, i.e., the total time a given machine is infected with Limbo. Each logfile of a dropzone contains records that include a unique victim ID and a timestamp, which indicates when the corresponding harvesting process was started. As the infection lifetime of a victim we define the interval between the timestamp of the last and first record caused by this particular victim. This is the lower bound of the total time of infection since we may not be able to observe all log files from this particular infection and thus underestimate the real infection time. The interval between the last and the first timestamp seen on the whole dropzone is defined as the lifetime of this dropzone. Using these definitions, the average infection time of a victim is about 2 days. This is only a coarse lower bound since we often observe an infected machine only a limited amount of time.

Table 1: Statistical overview of largest Limbo dropzones, sorted according to the total number of infected machines.

Dropzone	# Infected machines	Data amount	AS #	Country	Lifetime in days
webpinkXXX.cn	26,150	1.5 GB	4837	China	36
coXXX-google.cn	12,460	1.2 GB	17464	Malaysia	53
77.XXX.159.202	10,394	503 MB	30968	Russia	99
finXXXonline.com	6,932	438 MB	39823	Estonia	133
<i>Other</i>	108,122	24.4 GB			
Total	164,058	28.0GB			61



(a) Cumulative distribution of IP addresses infected with Limbo.

Country	# Machines	Percentage
Russia	26,700	16,3%
United States	23,704	14,4%
Spain	20,827	12,7%
United Kingdom	19,240	11,7%
Germany	10,633	6,5%
Poland	8,598	5,4%
Australia	6,568	4,0%
Turkey	5,328	3,2%
Brazil	4,369	2,7%
India	3,980	2,4%
Ukraine	2,674	1,6%
Egypt	2,302	1,4%
Italy	1,632	0,9%
Thailand	1,356	0,8%
<i>Other</i>	26,147	16,0%

(b) Distribution of Limbo infections by country.

Fig. 2: Analysis of IP addresses for machines infected with Limbo and their regional distribution.

The maximum lifetime of a Limbo victim we observed was more than 111 days. In contrast, the average lifetime of a dropzones is approximately 61 days.

Figure 2a depicts the cumulative distribution of IP addresses for infected machines based on the more than 164,000 Limbo victims we detected. The distribution is highly non-uniform: The majority of victims are located in the IP address ranges between 58.* – 92.* and 189.* – 220.*. Surprisingly, this is consistent with similar analysis of spam relays and scam hosts [3,29]. It could indicate that these IP ranges are often abused by attackers and that future research should focus on securing especially these ranges.

We determined the geographical location of each victim by using the Geo-IP tool Maxmind [18]. The distribution of Limbo infections by country is shown in Figure 2b. We found a total of 175 different countries and almost one third of the infected machines are located in either Russia or the United States.

3.4 Analysis of ZeuS Victims

We performed a similar analysis for the ZeuS dropzones and the victims infected with this malware. Figure 3a lists the top five countries in which the dropzones are located based on 205 dropzones we identified with our method. Most ZeuS dropzones can be found in North America, Russia, and East Asia — a results that also applies to the Limbo dropzones. We also found that the dropzones are located in many different Autonomous Systems (68 different AS in total), but several AS host a larger percentage of ZeuS dropzones: The three most common AS host 49% of all dropzones, indicating that there are some providers preferred by the attackers. Presumably those providers offer *bullet-proof hosting*, i.e., takedown requests are not handled properly by these providers or the providers even tolerate certain abusive behavior.

Country	# Dropzones	Percentage
United States	34	17%
Russia	29	14%
Netherlands	16	8%
Malaysia	14	7%
China	8	4%

(a) Top countries in which ZeuS dropzones are located.

OS version	# Infected Machines	%
Windows XP SP2	6,629	70.2 %
Windows XP SP0	1,264	13.1 %
Windows XP SP1	1,146	12.1 %
Windows 2000 SP4	285	3.0 %
Other	156	1.6 %

(b) Distribution of operating system for machines infected with ZeuS.

Fig. 3: General statistics for ZeuS dropzones and victims.

The four dropzones we had full access to contained information stolen from about 9,480 infected machines. Based on this data, we can determine the operating system version of each infected machine since the keylogger also extracts this information. Figure 3b provides an overview of the operating system running on the infected machines. The majority of victims is using Windows XP with Service Pack 2, thus they are not on the latest patch level (Service Pack 3 was released on May 6, 2008). A large fraction of machines run on even older version of the operating system. Only a minority of all victims have the latest service pack installed or are running Windows Vista. We also examined the language version of the operating system. Most infected machines have either English (53.8%) or Spanish (20.2%) as language. Consistent to the machines infected with Limbo, the majority of ZeuS infections can be found in the two network ranges 58.* – 92.* (56.9%) and 189.* – 220.* (25.8%).

As explained in Section 2.2, ZeuS can be dynamically re-configured by the attacker via a configuration file. The most frequent configurations are shown in Table 2. Websites that should be logged are listed in the first part of the table and the second part enumerates the websites that should be logged and where a screenshot should be taken. Online banking websites clearly dominate this statistic and indicate that these attacks aim at stealing credentials for bank accounts. Finally, websites where no keystrokes should be recorded are listed at the end of the table. This excluding of websites from the harvesting process is presumably done in order to minimize the logged data.

Table 2: Overview of top four websites a) to be logged, b) to be logged including a screenshot, and c) not to be logged.

	Website	# Appearances (205 dropzones)
a)	https://internetbanking.gad.de/*portal?bankid=*	183
	https://finanzportal.fiducia.de/*?rzid=*&rzbk=*	177
	https://www.vr-networld-ebanking.de/	176
	https://www.gruposantander.es/*	167
b)	@*/login.osmp.ru/*	94
	@*/atl.osmp.ru/*	94
	@https://*.e-gold.com/*	39
	@https://netteller.tsw.com.au/*/ntv45.asp?wci=entry	29
c)	!http://*.myspace.com*	132
	!*microsoft.com/*	98
	!http://*.odnoklassniki.ru/*	80
	!http://vkontakte.ru/*	72

4 Analysis of Stolen Credentials

Based on the data collected by our monitoring system, we analyzed what kind of credentials are stolen by keyloggers. This enables a unique point of view into the underground market since we can study what goods are available for the criminals from a first-hand perspective. We mainly focus on five different areas: online banking, credit cards, online auctions, email passwords, and social networks. At first sight, the last two areas do not seem to be very interesting for an attacker. However, especially these two kinds of stolen credentials can be abused in many ways, e.g., for identity theft, spear phishing, spamming, anonymous mail accounts, and other illicit activities. This is also reflected in the market price for these two types of goods as depicted in Table 3 based on a study by Symantec [33].

Identifying which credentials are stolen among the large number of collected data is a challenge. The key insight is that credentials are typically sent in HTTP POST requests from the victim to the provider. To find credentials, we thus need to pin-point

Table 3: Breakdown of prices for different goods and services available for sale on the underground market according to a study by Symantec [33]. *Percentage* indicates how often these goods are offered.

Goods and services	Percentage	Range of prices
Bank accounts	22%	\$10 – \$1000
Credit cards	13%	\$0.40 – \$20
Full identities	9%	\$1 – \$15
Online auction site accounts	7%	\$1 – \$8
Email passwords	5%	\$4 – \$30
Drop (request or offer)	5%	10% – 50% of total drop amount
Proxies	5%	\$1.50 – \$30

which requests fields are actually relevant and contain sensitive information. We use a trick to identify these fields: when a victim enters his credential via the keyboard, Limbo stores this information together with the current URL. Based on the collected data, we can thus build provider-specific models M_{P_i} that describe which input fields at P_i contain sensitive information. For example, $M_{\text{login.live.com}} = \{\text{login}, \text{passwd}\}$ and $M_{\text{paypal.com}} = \{\text{login_email}, \text{login_password}\}$. These models can then be used to search through all collected data to find the credentials, independent of whether the victim entered the information via the keyboard or they were inserted by a program via the Protected Storage. In total, we generated 151,070 provider-specific models. These models cover all domains for which keystrokes were logged by all infected machines. For our analysis, we only used a subset of all provider-specific models that are relevant for the area we analyzed.

We also need to take typing errors into account: if a victim makes a typing error during the authentication process, this attempt is not a valid credential and we must not include it in our statistics. We implement this by keeping track of which credentials are entered by each victim and only counting each attempt to authenticate at a specific provider once. During analysis, we also used methods like pattern matching or heuristics to find specific credentials as we explain below.

4.1 Banking Websites

We used 707 banking models that cover banking sites like Bank of America or Lloyds Bank, and also e-commerce business platforms like PayPal. These models were chosen based on the ZeuS configuration files since this keylogger aims specifically at stealing banking credentials. In total, we found 10,775 unique bank account credentials in all logfiles. Figure 4a provides an overview of the top five banking websites for which we found stolen credentials. The distribution has a long tail: for the majority of banking websites, we found less than 30 credentials.

Banking Website	# Stolen Credentials
PayPal	2,263
Commonwealth Bank	851
HSBC Holding	579
Bank of America	531
Lloyds Bank	447

(a) Overview of top five banking websites for which credentials were stolen.

Credit Card Type	# Stolen Credit Cards
Visa	3,764
MasterCard	1,431
American Express	406
Diners Club	36
<i>Other</i>	45

(b) Overview of stolen credit card information.

Fig. 4: Analysis of stolen banking accounts and credit card data.

ZeuS has the capability to parse the content of specific online banking website to extract additional information from them, e.g., the current account balance. We found 25 unique victims whose account balance was disclosed this way. In total, these 25 bank accounts hold more than \$130,000 in checking and savings (mean value is \$1,768.45,

average is \$5,225). Based on this data, we can speculate that the attackers can potentially access millions of dollars on the more than 10,700 compromised bank accounts we recovered during our analysis.

4.2 Credit Card Data

To find stolen credit card data, the approach with provider-specific models cannot be used since a credit card number can be entered on a site with an arbitrary field name. For example, an American site might use the field name `cc_number` or `cardNumber`, whereas a Spanish site could use `numeroTarjeta`. We thus use a pattern-based approach to identify credit cards and take the syntactic structure of credit card numbers into account: each credit card has a fixed structure (e.g., MasterCard numbers are 16 digits and the first two digits are 51-55) that we can identify. Furthermore, the first six digits of the credit card number are the Issuer Identification Number (IIN) which we can also identify. For each potential credit card number, we also check the validity with the Luhn algorithm [19], a checksum formula used to guard against one digit errors in transmission. Passing the Luhn check is only a necessary condition for card validity and helps us to discard numbers containing typing errors.

With this combination of patterns and heuristics, we found 5,682 valid credit card numbers. Figure 4b provides an overview of the different credit card types we found. To estimate the potential loss due to stolen credit cards we use the median loss amount for credit cards of \$223.50 per card as reported in the 2008 Internet Crime Complaint Center's Internet Crime Report [14]. If we assume that all credit cards we detected are abused by the attacker, we obtain an estimated loss of funds of almost \$1,270,000.

4.3 Email Passwords

Large portals and free webmail providers like Yahoo!, Google, Windows Live, or AOL are among the most popular websites on the Internet: 18 sites of the Alexa Top 50 belong to this category [2]. Accordingly, we expect that also many credentials are stolen from these kinds of sites. We used 37 provider-specific models that cover the large sites of this category. In total, we found 149,458 full, unique credentials. We detected many instances where the attackers could harvest many distinct webmail credentials from just one infected machine. This could indicate infected system in public places, e.g., schools or Internet cafes, to which many people have access. Figure 5a provides an overview of the distribution for all stolen email credentials.

4.4 Social Networks and Online Trading Platforms

Another category of popular sites are social networks like Facebook and MySpace, or other sites with a social component like YouTube. Of the Alexa Top 50, 14 sites belong to this category. To analyze stolen credentials from social networks, we used 57 provider-specific models to cover common sites in this category. In total, we found 78,359 stolen credentials and Figure 5b provides an overview of the distribution. Such credentials can for example be used by the attacker for spear phishing attacks.

Webmail Provider	# Stolen Credentials
Windows Live	66,540
Yahoo!	27,832
mail.ru	17,599
Rambler	5,379
yandex.ru	5,314
Google	4,783
<i>Other</i>	22,011

(a) Overview of stolen credentials from portals and webmail providers.

Social Network	# Stolen Credentials
Facebook	14,698
hi5	8,310
nasza-klasa.pl	7,107
odnoklassniki.ru	5,732
Bebo	5,029
YouTube	4,007
<i>Other</i>	33,476

(b) Overview of stolen credentials from social networking sites.

Fig. 5: Analysis of stolen credentials from free webmail providers and social networking sites.

The final type of stolen credentials we analyze are online trading platforms. We used provider-specific models for the big four platforms: eBay, Amazon, Allegro.pl (third biggest platform world-wide, popular in Poland), and Overstock.com. In total, we found 7,105 credentials that were stolen from all victims. Of these, the majority belong to eBay with 5,712 and Allegro.pl with 885. We found another 477 credentials for Amazon and 31 for Overstock.com. This kind of credentials can for example be used for money laundering.

4.5 Underground Market

The analysis of stolen credentials also enables us to estimate the total value of this information on the underground market: each credential is a marketable good that can be sold in dedicated forums or IRC channels [10,20]. If we multiply the number of stolen credentials with the current market price, we obtain an estimate of the overall value of the harvested information. Table 4 summarizes the results of this computation. These results are based on market prices as reported by Symantec [33,34]. Other antivirus vendors performed similar studies and their estimated market prices for these goods are similar, thus these prices reflect – to the best of our knowledge – actual prices paid on the underground market for stolen credentials. These results indicate that the information collected during our measurement period is potentially worth several millions of dollars.

Table 4: Estimation of total value of stolen credentials recovered during measurement period. Underground market prices are based on a study by Symantec [33].

Stolen credentials	Amount	Range of prices	Range of value
Bank accounts	10,775	\$10 – \$1000	\$107,750 – \$10,775,000
Credit cards	5,682	\$0.40 – \$20	\$2,272 – \$113,640
Full identities / Social Networks	78,359	\$1 – \$15	\$78,359 – \$1,175,385
Online auction site accounts	7,105	\$1 – \$8	\$7,105 – \$56,840
Email passwords	149,458	\$4 – \$30	\$597,832 – \$4,483,740
<i>Total</i>	224,485	n/a	\$793,318 – \$16,604,605

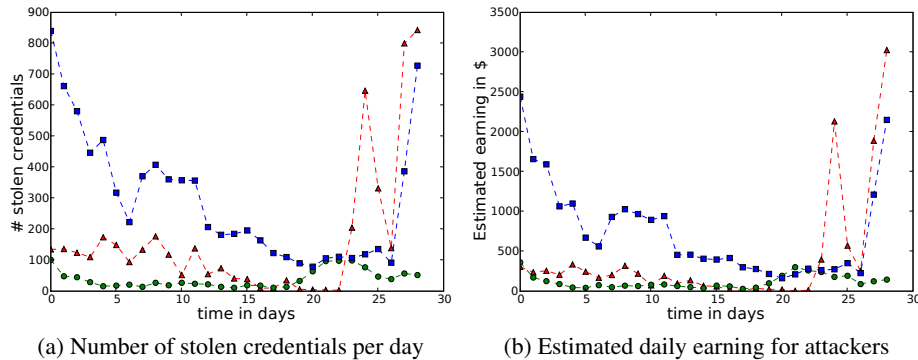


Fig. 6: Number of unique stolen credentials and estimated amount of money earned per day due to harvested keylogger data for three Limbo dropzones. Other dropzones have a similar distribution.

Given the fact that we studied just two families of keyloggers and obtained detailed information about only 70 dropzones (from a total of more than 240 dropzones that we detected during our study), we can argue that the overall size of the underground market is considerably larger.

We also studied the estimated revenue of the individual dropzones. For each dropzone, we computed the total number of credentials stolen per day given the five categories examined in this paper. Furthermore, we use the range of prices reported by Symantec [33] to estimate the potential daily earnings of the operator of each dropzone. The results of this analysis are shown exemplarily in Figure 6 for three different Limbo dropzones. These dropzones were chosen since we were able to obtain continuous data for more than four weeks for these sites. However, the distribution for other dropzones is very similar. Figure 6a depicts the number of unique stolen credentials per day. This number varies greatly per day, presumably due to the fact that the malware has a certain rate at which new victims are infected and this rate also varies per day. We also observe that there is a steady stream of fresh credentials that can then be traded at the underground market. On the other hand, Figure 6b provides an overview of the estimated value of stolen credentials for each particular day. We obtain this estimate by multiplying the number of credentials stolen per day with the *lowest* market price according to the study by Symantec [33] (see Figure 3). This conservative assumption leads to a lower bound of the potential daily income of the attackers. The results indicate that an attacker can earn several hundreds of dollars (or even thousands of dollars) per day based on attacks with keyloggers — a seemingly lucrative business.

4.6 Discussion

Besides the five categories discussed in this section, ZeuS and Limbo steal many more credentials and send them back to the attacker. In total, the collected logfiles contain more than three million unique keystroke logs. With the provider-specific models ex-

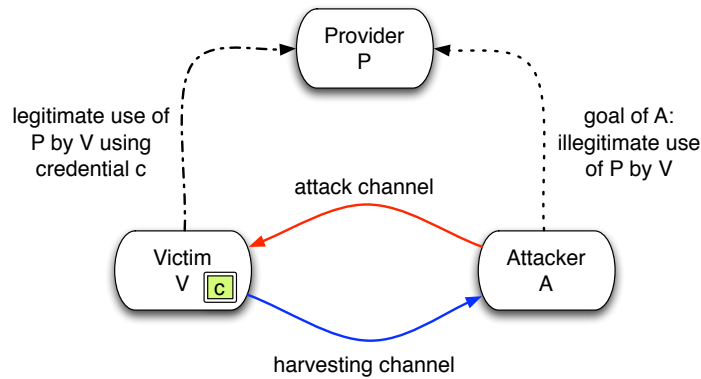


Fig. 7: Structure of attacks susceptible to our method.

amined in the five categories, we only cover the larger types of attacked sites and high-profile targets. Many more types of stolen sensitive information against small websites or e-commerce companies are not covered by our analysis. As part of future work, we plan to extend our analysis and also include an analysis of stolen cookies and the information extracted from the Protected Storage of the infected machines.

5 Conclusion and Future Work

Our user simulation approach is rather ad-hoc and does not allow us to study all aspects of keyloggers. The main limitation is that we do not know exactly on which sites the keylogger becomes active and thus we may miss specific keyloggers. Our empirical results show that keyloggers typically target the main online banking websites and also extract information from the Protected Storage. Nevertheless, we may miss keyloggers that only steal credentials from a very limited set of sites. This limitation could be circumvented by using more powerful malware analysis techniques like multi-path execution [23] or a combination of dynamic and static analysis [16]. Another limitation is that we do not exactly determine which credentials are stolen. Techniques from the area of taint tracking [25,38] can be added to our current system to pinpoint the stolen credentials. Despite these limitation, the ad-hoc approach works in practice and enables us to study keyloggers as we showed in Section 3 and 4.

The approach we took in this paper works for keylogger-based attacks, but it can in fact be generalized to other attacks as well, for example classical phishing. The abstract schema behind the class of attacks that can be analyzed is shown in Figure 7. There, a provider P offers some online service like an online bank or an online trading platform (like eBay or Amazon). The victim V is a registered user of the service provided by P and uses credentials c to authenticate as a legitimate user towards P . The attacker A wants to use P 's service by pretending to be V . To do this, A needs V 's credentials c . So for a successful attack, there must exist a (possibly indirect) communication channel from V to A over which information about c can flow. We call this channel the *harvesting channel*. Apart from the harvesting channel there also exists another (possibly

indirect) communication channel from A to V . This channel is used by the attacker to initiate or trigger an attack. We call this channel the *attack channel*. The generalization of our approach presented in this paper involves an analysis of the harvesting channel. This is a hard task, which together with more automation is a promising line for future work in this area.

Acknowledgements. We would like to thank Carsten Willems for extending CWSandbox such that certain aspects of user simulation such as generic clicking are directly implemented within the sandbox. Jan Göbel provided valuable feedback on a previous version of this paper that substantially improved its presentation. Frank Boldewin helped in analyzing the ZeuS configuration files and the AusCERT team was very helpful in notifying the victims. This work has been supported by the WOMBAT and FORWARD projects funded by the European Commission.

References

1. AutoIt Script Home Page. Internet: <http://www.autoitscript.com/>, 2009.
2. Alexa, the Web Information Company. Global Top Sites, September 2008. http://alexa.com/site/ds/top_sites?ts_mode=global.
3. David S. Anderson, Chris Fleizach, Stefan Savage, and Geoffrey M. Voelker. Spamscatter: Characterizing Internet Scam Hosting Infrastructure. In *USENIX Security Symposium*, 2007.
4. Anonymous. Comment about posting “Good ol’ #CCpower” on honeyblog. Internet: <http://honeyblog.org/archives/194-CCpower-Only-Scam.html>, June 2008.
5. Madhusudhanan Chandrasekaran, Ramkumar Chinchani, and Shambhu Upadhyaya. PHONEY: Mimicking User Response to Detect Phishing Attacks. In *Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2006.
6. Taehwan Choi, Soeul Son, Mohamed Gouda, and Jorge Cobb. Pharewell to Phishing. In *Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, 2008.
7. Neil Chou, Robert Ledesma, Yuka Teraguchi, and John C. Mitchell. Client-Side Defense Against Web-Based Identity Theft. In *Network and Distributed System Security Symposium (NDSS)*, 2004.
8. Rachna Dhamija and J. D. Tygar. The Battle Against Phishing: Dynamic Security Skins. In *Symposium on Usable Privacy and Security (SOUPS)*, 2005.
9. Finjan. Malicious Page of the Month. <http://www.finjan.com/Content.aspx?id=1367>, April 2008.
10. Jason Franklin, Vern Paxson, Adrian Perrig, and Stefan Savage. An Inquiry Into the Nature and Causes of the Wealth of Internet Miscreants. In *Conference on Computer and Communications Security (CCS)*, 2007.
11. Sebastian Gajek and Ahmad-Reza Sadeghi. A Forensic Framework for Tracing Phishers. In *IFIP WG 9.2, 9.6/11.6, 11.7/FIDIS International Summer School on The Future of Identity in the Information Society*, Karlstad University, Sweden, August 2007.
12. Cormac Herley and Dinei Florencio. How To Login From an Internet Cafe Without Worrying About Keyloggers. In *Symposium on Usable Privacy and Security (SOUPS)*, 2006.
13. Thorsten Holz, Markus Engelberth, and Felix Freiling. Learning More About the Underground Economy: A Case-Study of Keyloggers and Dropzones. Technical Report TR-2008-006, University of Mannheim, 2008.
14. Internet Crime Complaint Center (IC3). 2008 Internet Crime Report, March 2009. <http://www.ic3.gov/media/annualreports.aspx>.

15. Chris Kanich, Christian Kreibich, Kirill Levchenko, Brandon Enright, Geoffrey M. Voelker, Vern Paxson, and Stefan Savage. Spamalytics: An Empirical Analysis of Spam Marketing Conversion. In *Conference on Computer and Communications Security (CCS)*, 2008.
16. Engin Kirda, Christopher Kruegel, Greg Banks, Giovanni Vigna, and Richard Kemmerer. Behavior-based Spyware Detection. In *USENIX Security Symposium*, 2006.
17. Cullen Linn and Saumya Debray. Obfuscation of Executable Code to Improve Resistance to Static Disassembly. In *Conference on Computer and Communications Security (CCS)*, 2003.
18. MaxMind LLC. MaxMind GeoIP. <http://www.maxmind.com/app/ip-location>, August 2008.
19. Hans P. Luhn. Computer for Verifying Numbers, August 1960. U.S. Patent 2,950,048.
20. Jerry Martin and Rob Thomas. the underground economy: priceless. *USENIX ;login.*, 31(6), December 2006.
21. Jonathan M. McCune, Adrian Perrig, and Michael K. Reiter. Bump in the Ether: A Framework for Securing Sensitive User Input. In *USENIX Annual Technical Conference*, 2006.
22. Microsoft. Protected Storage (Pstore), August 2008. Microsoft Developer Network (MSDN).
23. Andreas Moser, Christopher Kruegel, and Engin Kirda. Exploring Multiple Execution Paths for Malware Analysis. In *IEEE Symposium on Security and Privacy*, 2007.
24. Andreas Moser, Christopher Kruegel, and Engin Kirda. Limits of Static Analysis for Malware Detection. In *Annual Computer Security Applications Conference (ACSAC)*, 2007.
25. James Newsome and Dawn Xiaodong Song. Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploits on Commodity Software. In *Network and Distributed System Security Symposium (NDSS)*, 2005.
26. Igor V. Popov, Saumya K. Debray, and Gregory R. Andrews. Binary Obfuscation Using Signals. In *USENIX Security Symposium*, 2007.
27. The HoneyNet Project. *Know Your Enemy: Learning About Security Threats*. Addison-Wesley Longman, 2nd edition, May 2004.
28. Niels Provos, Panayiotis Mavrommatis, Moheeb A. Rajab, and Fabian Monrose. All Your iFRAMEs Point to Us. In *USENIX Security Symposium*, 2008.
29. Anirudh Ramachandran and Nick Feamster. Understanding the Network-Level Behavior of Spammers. *SIGCOMM Comput. Commun. Rev.*, 36(4):291–302, 2006.
30. SecureWorks. PRG Trojan. <http://www.secureworks.com/research/threats/prgtrojan/>, June 2007.
31. SecureWorks. Coreflood Report. <http://www.secureworks.com/research/threats/coreflood-report/>, August 2008.
32. Mika Stahlberg. The Trojan Money Spinner. In *Virus Bulletin Conference*, 2007.
33. Symantec. Global Internet Security Threat Report: Trends for July – December 07, April 2008.
34. Symantec. Report on the Underground Economy July 07 – June 08, November 2008.
35. XiaoFeng Wang, Zhuowei Li, Ninghui Li, and Jong Youl Cho. PRECIP: Towards Practical and Retrofittable Confidential Information Protection. In *Network and Distributed System Security Symposium (NDSS)*, 2008.
36. Yi-Min Wang, Doug Beck, Xuxian Jiang, Roussi Roussev, Chad Verbowski, Shuo Chen, and Samuel T. King. Automated Web Patrol with Strider HoneyMonkeys: Finding Web Sites That Exploit Browser Vulnerabilities. In *Network and Distributed System Security Symposium (NDSS)*, 2006.
37. Carsten Willems, Thorsten Holz, and Felix Freiling. Toward Automated Dynamic Malware Analysis Using CWSandbox. *IEEE Security & Privacy Magazine*, 5(2):32–39, March 2007.
38. Heng Yin, Dawn Song, Manuel Egele, Christopher Kruegel, and Engin Kirda. Panorama: Capturing System-wide Information Flow for Malware Detection and Analysis. In *Conference on Computer and Communications Security (CCS)*, 2007.