**Reihe Informatik. TR-2008-006**

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

Thorsten Holz Markus Engelberth Felix Freiling

University of Mannheim
Laboratory for Dependable System
{holz|engelberth|freiling}@informatik.uni-mannheim.de

December 18, 2008

**Abstract**

We study an active underground economy that trades stolen digital credentials.We present a method with which it is possible to directly analyze the amount of data harvested through these types of attacks in a highly automated fashion. We exemplify this method by applying it to 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 the first 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. This helps us better understand the nature and size of these quickly emerging underground marketplaces.

## 1 Introduction

Today, the Internet is driving a new digital economy of services that offer a degree of flexibility unmatched by traditional service providers, like online banking, digital communities, and online trading forums. It is well-known that this economy moves enormous amounts of money and that the market share of digital business is growing rapidly every year [11].

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 [21] 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" [21] is in fact staggering. A large-scale study of similar forms of online activity was later performed by Franklin et al. [12]. 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" [12]. In other words, digital and classical crime are merging. Even in the presence of previous studies [21, 12] it is hard to estimate the real size of the underground economy. This is because previous work has only measured *indirect* effects of underground markets. For example, both studies did not observe real trading, but only *announcements* of trading and *offers* of stolen credentials in public IRC channels. Another question raised by these studies is 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 present a method that allows us to observe the *actual kind and amount* of data that is stolen by attackers from compromised machines, not only the results of trading this data. These

are the goods that can be traded at an underground market. This data gives us a much better basis for estimating the size of the underground economy. 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 malware running on a compromised machine sends all stolen credentials to the dropzone, where the attacker can pick them up and start to abuse them [10, 31, 32]. Our method to analyze keylogger-based attacks is based on executing malware within an instrumented environment [36], extracting the location of the dropzone, and harvesting the keylogger data on our own (if possible). All this is done in a highly automated manner and we applied it to two different classes of keyloggers to demonstrate the practical feasibility of our method.

We present in this paper the first empirical study of this phenomenon, giving many details about these 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. As an additional contribution 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 earn hundreds of dollars per day.

The analysis method presented in this paper is not restricted to keylogger-based attacks, it can be applied to all attacks in which an attacker steals authentication credentials of a victim after some initial form of contact. We call these types of attacks *impersonation attacks*. This class covers a wide range of real-world attacks including many different forms of phishing, certain forms of sending spam, or online-fraud based on identity theft.

## 1.1 Related Work

In the field of phishing prevention and mitigation, there has been some work specific to attacks based on email and fake websites [5, 13]. Chandrasekaran et al. [5] generate fake input and investigate a site's response to detect phishing sites. Gajek and Sadeghi [13] use fake credentials to track down phishers. Our work is complementary to this work: we study the actual dropzone, i.e., the site where the stolen credentials are stored, and infer from this data more information about the extent and size of the attack.

Jakobsson introduces *phishing graphs* to model phishing attacks [16]. In such a graph, nodes correspond to knowledge or access rights, and (directed) edges correspond to means of obtaining information or access rights from already possessed information or access rights. This enables the study of different kinds of phishing attacks and an economic analysis on the viability of attacks. In contrast, our model is more general and captures the whole class of impersonation attacks, for which phishing is an instantiation. Furthermore, we focus on the goal of the attacker, i.e., stealing of credentials.

Impersonation attacks 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 [9], or Transport Login Protocol [6] can protect against certain forms of impersonation 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 keylogger also defeat simple tricks to conceal the entered password as proposed by Herley and Florêncio [14]. However, malware prevention methods and systems that protect confidential information can defend against this kind of attacks [22, 34].

## 1.2 Summary of Contributions

To summarize, our work presented in this paper makes the following two main contributions:

1. We present a method to analyze a large class of credential-stealing attacks (impersonation attacks) in an highly automated fashion.

2. We apply this method to the case of keylogging attacks based on dropzones and provide a detailed analysis of the collected data, giving a novel and first-hand insight into the underground economy of Internet criminals from a unique and novel viewpoint.

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. Together with previous 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.** The paper is structured as follows: We first define the class of impersonation attacks in Sect. 2. Then in Sect. 3 we perform a case study of a particular form of impersonation attack: stealing credentials using keyloggers and dropzones. We analyze the collected data in Sect. 4 and briefly conclude the paper in Sect. 5.

**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 Impersonation Attacks

## 2.1 Terminology

We focus on a specific class of attacks that we call *impersonation attacks*. In such attacks, there are three different actors (see Fig. 1 for a schematic overview): The first actor is the *provider* $P$ of some online service like an online bank, an online trading platform (like eBay or Amazon), an Internet service provider, or even a social network (like Facebook). The second actor is the *victim* $V$, a registered user of the service provided by $P$. To ensure only authorized access to its services, $P$ performs authentication prior to granting access to its services. For this reason, $P$ has granted $V$ authentication credentials $c$ after registration. Using $c$ it is possible to authenticate as user $V$ towards $P$. The third (and final) actor is the *attacker* $A$. The goal of $A$ is to enjoy the service of $P$ in an unauthorized way. More specifically, $A$ wants to use $P$'s service by pretending to be $V$. To do this, $A$ needs $V$'s credentials $c$. So to mount a successful impersonation attack, $A$ has to get hold of $c$. This means that 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 impersonation attack. We call this channel the *attack channel*.

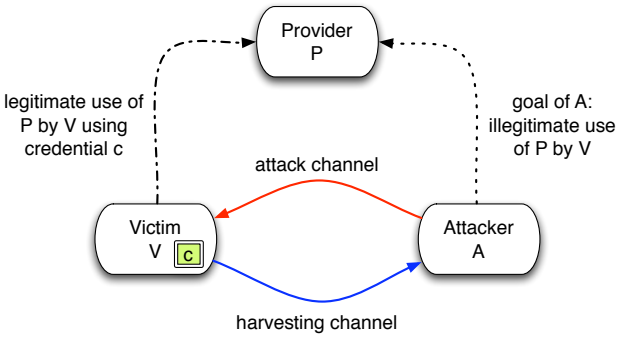



Figure 1: Structure of impersonation attacks.

## 2.2 Examples

Impersonation attacks cover a wide variety of real-world attacks. The classic example is *phishing*: In classical phishing, $P$ is a bank and $c$ are login credentials to an online banking website like an account number and password, possibly together with one or more transaction numbers. The attack channel is usually a spoofed email asking $V$ to check or correct his login credentials and pointing him to a particular (fake) website. This website is part of the harvesting channel. The user enters $c$ into the website, disclosing his credentials. This information is then either stored locally on the webserver or directly sent to $A$, e.g., via email to an account the attacker controls.

Another form of attack channel involves exploiting cross-site scripting errors in websites which allow, for example, to redirect victims to other websites where they enter (and thus disclose) their credentials. A more basic attack channel is using any form of social engineering to persuade $V$ to send $c$ to $A$ (e.g., by having $A$ call $V$ on the phone).

Variants of classical phishing involve different types of attack and harvesting channels. For example, the attack channel can consist of the infection of $V$ by a piece of malicious software crafted by $A$. This can for example be achieved with the help of drive-by downloads via malicious websites [35] or sending the software to the victim in an email attachment. The malicious software itself can contain keylogging functionality that saves $V$'s keystrokes (including the credentials $c$) to a file which is sent to $A$ by means of email or HTTP POST requests in regular intervals (harvesting channel). A popular harvesting channel today is called a *dropzone*. Dropzones are publicly writable directories on servers in the Internet. Instead of sending keylogger data directly to $A$, the malware at $V$ "drops" the data into the directory at regular intervals. The attacker regularly accesses the dropzone, downloads this data, and extracts $c$.

In a variant of phishing via keyloggers, the provider $P$ is an Internet service provider offering a mail relay server to send email. The credentials $c$ there consist of $V$'s username and password for the mail relay. These are used by $A$ to send spam disguising as $V$.

## 2.3 Analysis Method

Our goal is to analyze impersonation attacks with a general method. The method we propose in this paper is based on the analysis of the harvesting channel. To do this, the harvesting channel must be identified first. To identify the harvesting channel we basically have two options. The first option consists of collaborating with $V$. For example, $V$ could provide access to its network interface which we can monitor to extract information about the harvesting channel. The second option is to play the role of $V$ ourselves. For example, we can setup a dedicated environment in which we imitate $V$. In this environment, we first need to emulate the infection process, i.e., fall for the attacks targeting $V$ over the attacker channel.

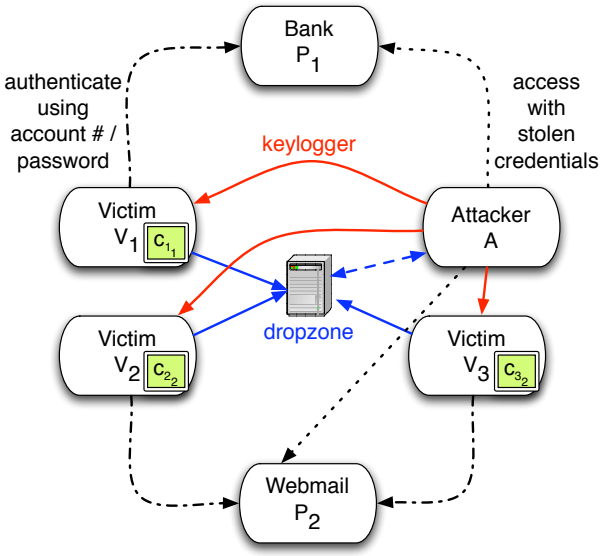



Figure 2: Schematic overview of impersonation attacks which use keyloggers as attack channel and a dropzone as harvesting channel.

In the second phase, we need to automatically analyze the attacks observed on the attack channel to extract information about the harvesting channel. We will use this approach later in Sect. 3 to identify and automatically analyze the harvesting channel and show that the process can be automated to a high degree with very little human interaction needed.

# 3 Case Study: Impersonation Attacks Using Keyloggers

In this section, we show that the methodology we introduced in Sect. 2 can be applied to different kinds of impersonation attacks and is thus very general. In particular, we show how to efficiently analyze attack schemes that use malware in the form of *keyloggers* as attack channel and *dropzones* as harvesting channel. Fig. 2 provides a schematic overview of this kind of attacks. 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. This can for example be with the help of a spam mail that contains a copy of the keylogger in an attachment, installing the keylogger with the help of a drive-by download on a malicious website, or other attack vectors. Once the victim $V_i$ is infected, the keylogger starts to record all relevant 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 banking 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 $V_i$.

## 3.1 Studying the Attack and Harvesting Channel

The practical challenge of the methodology presented in this paper is to automate the analysis of the attack and harvesting channel as much as possible. To study the attack channel, we use the concept of *honeypots*, i.e., information system resources whose value lies in unauthorized or illicit use of that resource [28]. We play the role of a victim $V_i$ and react on incoming attacks 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 [29, 35]. Using these techniques, our honeypot can be infected with a keylogger in an automated way and we obtain information about the attack channel.

After a successful infection, we extract the sample from the honeypot for further analysis of the harvesting channel. We choose dynamic analysis based on a dedicated analysis tool called CWSandbox [36] since static analysis can be defeated by malware using many different techniques [18, 25, 27]. 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 of 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, we can thus detect the harvesting channel and identify the dropzone in an automated way since the keylogger contacts the dropzone at an early stage after starting.

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. Furthermore, we do not only want to detect the harvesting channel, but also learn more about it. So in order to study the harvesting channel 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.

## 3.2 Improving Detection of Harvesting Channel 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. 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 of the harvesting channel, 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 (*PStore*) of the analysis machine as some kind of *honeytoken*. PStore provides applications with an interface to store user data [23] and many applications store credentials like username/password combinations there. By depositing some credentials in PStore, we can potentially trigger on more keyloggers: by offering more bait, we can learn more about the malware.

Simulating user behavior enables us to learn more about the results of a keylogger infection and we can also analyze some properties of the harvesting channel, 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 Sect. 4. 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 the harvesting channel once per family to improve our understanding of the harvesting channel. 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 in the next section. Note that even if we cannot fully decode the harvesting channel, 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 close the harvesting channel.

## 3.3 Technical Details for 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*. Both families use distinct attack and harvesting channels and we provide a short overview of both families in this section.

**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 [14]. 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 extract information from the Protected Storage and to steal cookies.

Once a credential is found, the harvested information is sent to the dropzone. The harvesting channel is implemented via HTTP requests 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 thus automatically determine the dropzone. The dropzone itself is implemented as a web application that allows the attacker amongst other

```
entry "WebFilters"
 "https://www.gruposantander.es/*"
 "https://internetbanking.gad.de/*/portal
 "!http://*myspace.com*"
 "@https://caionline.cai.es/*"
end
entry "WebFakes"
 "https://sitekey.bankofamerica.com/sas/"
    "http://192.168.2.1/fk/US/bofa.php"
end
entry "DnsMap"
 ;127.0.0.1 downloads1.kaspersky-labs.com
 ;127.0.0.1 www.symantec.com
end
```

Figure 3: Excerpt of ZeuS configuration file.

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 exploit to obtain access to the full data set. An example of (sanitized) information stolen by Limbo infections is shown in Appendix A.

**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, which forms the harvesting channel. 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 enable the access to the full dropzone data. An example of (sanitized) information stolen by ZeuS infections is shown in Appendix B.

Similar to Limbo, ZeuS can also be dynamically re-configured: after starting up, the malware retrieves the current configuration file from the dropzone. Fig. 3 provides an excerpt of such a configuration file. The *WebFilters* section configures which sites should be monitored (or not be monitored with the `!` sign). The `@` sign denotes websites for which a screenshot of 50x50 pixel around the mouse pointer should be taken at every left-click of the mouse. 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. The *WebFakes* section configures man-in-the-middle attacks: each time the victim opens the first URL, the request is transparently redirected to the second URL, which hosts some kind of phishing website that tricks the victim into disclosing even more credentials. Several other configuration options like DNS modification on the victim's machine or update functionality are available.

### 3.4 Studying Other Types of Impersonation Attacks

Besides impersonation attacks which use keylogger and dropzones, there are many other forms of this attack that can also be analyzed with the method we propose in this paper. For example, to study classical phishing attacks, we need to analyze phishing mails since they constitute the attack channel. To learn more about the harvesting channel, we open the spammed URL in a honeyclient and can thus automatically identify the phishing website. A recent study by Cova et al. shows that phishing sites commonly use email as dropping technique [8]. By analyzing the phishing kit used during a specific attack, we can identify this part of the harvesting channel and identify the email account that is used to store the stolen credentials. Typically it is then not possible to fully study the dropzone since we commonly cannot access this email account. We thus obtain a limited view into the harvesting channel, but the information about the account can already be used for mitigation purposes.

There are keylogger-based impersonation attacks that use email as the harvesting channel. For this kind of attacks, we can study the attack channel with the methods outlined above. Again, the harvesting channel itself can be automatically identified by using dynamic analysis: when starting the keylogger in CWSandbox and simulating user activity with SimUser, we potentially trigger the submission of logged credentials. We can thus automatically identify the email account used as drop account. Similar to classical phishing with email as drop technique, we can not obtain a detailed insight into the harvesting channel, but can nevertheless reliably identify the drop email account which can then be mitigated.

## 4 Results

In this section, we report on the results of a systematic study of impersonation attacks using keylogger and a dropzone as outlined in the previous sections. All data was collected during a seven-month period between April and October 2008.

### 4.1 General Statistics

With the help of CWSandbox, we analyzed more than 2,000 unique keylogger 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 the harvesting channel, i.e., we had access to all logfiles collected at that particular dropzone. We also periodically collected this information to study the amount and kind of stolen credentials to get a better understanding of the data stolen by attackers during such impersonation attacks. In total, our monitoring system collected 28 GB of Limbo and 5 GB of ZeuS logfiles during the measurement period.

**Limbo.** 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 Tab. 1. In total, we collected information about more than 164,000 infected machines. The table is sorted by the number of unique infected machines and contains a detailed overview of the top four dropzones. 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 US.

| 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 | Russian | 99 |
| finXXXonline.com | 6,932 | 438 MB | 39823 | Estonia | 133 |
| *Other* | 108,122 | 24.4 GB | | | |
| Total | 164,058 | 28.0GB | | | 61 |

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

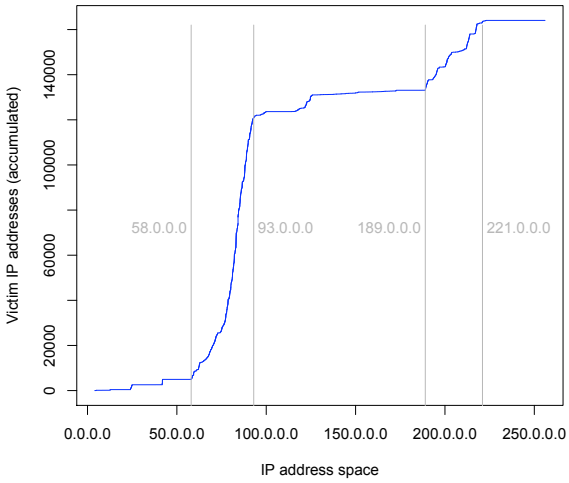
We also examined the *lifetime* for each dropzone and the *infection lifetime* of all victims, i.e., the total time a given machine is infected with Limbo. Each logfile of a dropzone contains records that include a unique victim ID and a timestamp, which indicates when the corresponding harvesting process 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. 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.

Fig. 4a 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, 30]. 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 [19]. The distribution of Limbo infections by country is shown in Fig. 4b. 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.

**ZeuS.** We performed a similar analysis for the ZeuS dropzones and the victims infected with this malware. Fig. 5a 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.

We were able to collect information about 9,480 infected machines from the four dropzones we had full access to. Based on this data, we can determine the operating system version of each infected machine since the keylogger also extracts this information. Fig. 5b provides an overview of the operating system running on the infected machines. The majority of victims run with Windows XP with Service Pack 2, thus they are not on the latest patch level. A large fraction of machines run on even older version of the operating system. 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




(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% |
| *Other* | 26,147 | 16,0% |

(b) Distribution of Limbo infections by country.

Figure 4: Analysis of IP addresses for machines infected with Limbo and their regional distribution.

| 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.

Figure 5: General statistics for ZeuS dropzones and victims.

with Limbo, the majority of ZeuS infections can be found in the network ranges 58.* – 92.* (56.9%) and 189.* – 220.* (25.8%).

As explained in Sec. 3.3, ZeuS can be dynamically re-configured by the attacker via a configuration file. The most frequent configurations are shown in Tab. 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.

## 4.2 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-hands 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,

| | 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 |
| | @*/login.osmp.ru/* | 94 |
| b) | @*/login.osmp.ru/* | 94 |
| | @*/atl.osmp.ru/* | 94 |
| | @https://*.e-gold.com/* | 39 |
| | @https://netteller.tsw.com.au/*/ntv45.asp?wci=entry | 29 |
| | @https://net.dataaction.com.au/802254v45/ntv45.asp?wci=entry | 28 |
| c) | !http://*myspace.com* | 132 |
| | !*.microsoft.com/* | 98 |
| | !http://*odnoklassniki.ru/* | 80 |
| | !http://vkontakte.ru/* | 72 |
| | !*microsoft* | 18 |

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

| 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 |

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.

anonymous mail accounts, and other illicit activities. This is also reflected in the market price for these two types of goods as depicted in Tab. 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 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 thus we must not include it in our statistics. We implemented this by keeping

track of which credentials are entered by each victim and only counting each attempt to authenticate at a specific provider once. For analysis, we also used other methods like pattern matching or heuristics to find specific credentials as we explain below.

**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. Fig. 6a 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 |
| *Other* | 45 |

(b) Overview of stolen credit card information.

Figure 6: Analysis of stolen banking accounts and credit card data.

ZeuS has the capability to parse specific online banking website to extract further 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.

**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 [20], 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. Fig. 6b 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 $298.00 per card as reported in the 2007 Internet Crime Complaint Center's Internet Crime Report [15]. If we assume that all credit cards we detected are abused by the attacker, we obtain an estimated loss of funds of almost $1,700,000.

**Email Passwords.** Large portals and free webmail provider 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. Fig. 7a provides an overview of the distribution for all stolen email credentials.

| 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 |
| *Other* | 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 |
| *Other* | 33,476 |

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

Figure 7: Analysis of stolen credentials from free webmail providers and social networking sites.

**Social Networks.** 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 Fig. 7b provides an overview of the distribution. Such credentials can for example be used by the attacker for spear phishing attacks.

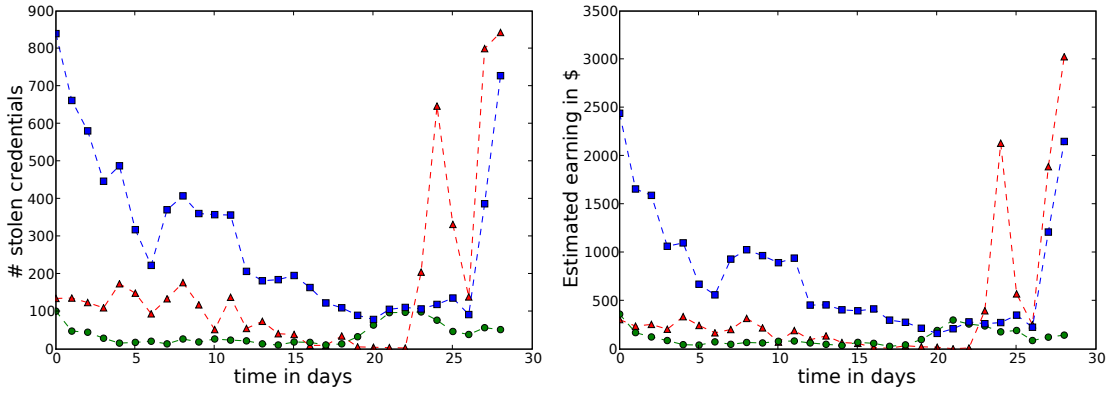
**Online Trading Platforms.** The final type of stolen credentials we analyze are online trading platforms. We used provider-specific models for the big four platforms: eBay, Amazon, Allegro.pl (third biggest platform world-wide, popular in Poland), and Overstock.com. In total, we found 7,105 credentials 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.

## 4.3 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 [12, 21]. 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. Tab. 4 summarizes the results of this computation. These results are based on market prices as reported by Symantec [33]. 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 | 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 |
| *Total* | 224,485 | n/a | \$793,318 – \$16,604,605 |

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




(a) Number of stolen credentials per day

(b) Estimated daily earning for attackers

Figure 8: 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.

stolen credentials. These results indicate that the information collected during our measurement period is potentially worth several millions of dollars. 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) suggests 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 Fig. 8 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. Fig. 8a 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, Fig. 8b 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 Fig. 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 hundred dollars per day based on impersonation attacks with keyloggers – a seemingly lucrative business.

## 4.4 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 examined 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

In this paper, we introduced a method to analyze impersonation attacks. The technical challenge of the approach is to automate the analysis process as much as possible and to analyze the large amount of data collected in this fashion. Based on empirical measurements we showed that attackers steal thousands of credentials from the infected machines. This stolen data can then be traded on the underground market and our work provides an insight into this market from a unique point of view.

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, and thus we usually trigger the harvesting phase. 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 [24] or a combination of dynamic and static analysis [17]. Another limitation is that we do not exactly determine which credentials are stolen. Techniques from the area of taint tracking [26, 37] 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 Sect. 4. In the future, we plan to extend the current analysis system to include techniques that automate the analysis phase to an even greater extent.


## 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 it's presentation. Frank Boldewin helped in analyzing the ZeuS configuration files and the AusCERT team was very helpful in notifying the victims.


# References


[1] AutoIt Script Home Page. Internet: `http://www.autoitscript.com/`, 2008.

[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 *Proceedings of the 16th 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 *Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, 2006.

[6] Taehwan Choi, Sooel Son, Mohamed Gouda, and Jorge Cobb. Pharewell to phishing. In *Proceedings of 10th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, November 2008.

[7] Neil Chou, Robert Ledesma, Yuka Teraguchi, and John C. Mitchell. Client-side defense against web-based identity theft. In *Proceedings of the 11th Network and Distributed System Security Symposium (NDSS'04)*, 2004.

[8] Marco Cova, Christopher Kruegel, and Giovanni Vigna. There is No Free Phish: An Analysis of "Free" and Live Phishing Kits. In *Proceedings of USENIX Workshop on Offensive Technologies (WOOT'08)*, 2008.

[9] Rachna Dhamija and J. D. Tygar. The battle against phishing: Dynamic Security Skins. In *Proceedings of the Symposium on Usable Privacy and Security (SOUPS'05)*, pages 77–88, 2005.

[10] Finjan. Malicious Page of the Month. `http://www.finjan.com/Content.aspx?id=1367`, April 2008.

[11] Forrester Research, Inc. The state of retailing online 2008: Merchandising and web optimization report, August 2008.

[12] Jason Franklin, Vern Paxson, Adrian Perrig, and Stefan 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)*, pages 375–388, 2007.

[13] Sebastian Gajek and Ahmad-Reza Sadeghi. A forensic framework for tracing phishers. In *Proceedings of Third 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.

[14] Cormac Herley and Dinei Florencio. How To Login From an Internet Cafe Without Worrying About Keyloggers. In *Proceedings of the 2006 Symposium on Usable Privacy and Security (SOUPS'06)*, 2006.

[15] Internet Crime Complaint Center (IC3). 2007 Internet Crime Report, April 2008. `http://www.ic3.gov/media/annualreports.aspx`.

[16] Markus Jakobsson. Modeling and Preventing Phishing Attacks. In *Proceedings of 9th International Conference on Financial Cryptography and Data Security*, 2005.

[17] Engin Kirda, Christopher Kruegel, Greg Banks, Giovanni Vigna, and Richard Kemmerer. Behavior-based Spyware Detection. In *Proceedings of 15th Usenix Security Symposium*, 2006.

[18] Cullen Linn and Saumya Debray. Obfuscation of Executable Code to Improve Resistance to Static Disassembly. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS'03)*, pages 290–299, 2003.

[19] MaxMind LLC. MaxMind GeoIP. http://www.maxmind.com/app/ip-location, August 2008.

[20] Hans P. Luhn. Computer for Verifying Numbers, August 1960. U.S. Patent 2,950,048.

[21] Jerry Martin and Rob Thomas. the underground economy: priceless. *USENIX ;login:*, 31(6), December 2006.

[22] Jonathan M. McCune, Adrian Perrig, and Michael K. Reiter. Bump in the Ether: A Framework for Securing Sensitive User Input. In *Proceedings of the USENIX Annual Technical Conference*, pages 17–17, 2006.

[23] Microsoft. Protected Storage (Pstore), August 2008. Microsoft Developer Network (MSDN).

[24] Andreas Moser, Christopher Kruegel, and Engin Kirda. Exploring Multiple Execution Paths for Malware Analysis. In *Proceedings of 2007 IEEE Symposium on Security and Privacy*, 2007.

[25] Andreas Moser, Christopher Kruegel, and Engin Kirda. Limits of Static Analysis for Malware Detection. In *Proceedings of 23rd Annual Computer Security Applications Conference (ACSAC'07)*, pages 421–430, 2007.

[26] James Newsome and Dawn Xiaodong Song. Dynamic taint analysis for automatic detection, analysis, and signaturegeneration of exploits on commodity software. In *Proceedings of the 12th Network and Distributed System Security Symposium (NDSS'05)*, 2005.

[27] Igor V. Popov, Saumya K. Debray, and Gregory R. Andrews. Binary obfuscation using signals. In *Proceedings of 16th USENIX Security Symposium*, pages 1–16, 2007.

[28] The Honeynet Project. *Know Your Enemy: Learning About Security Threats*. Addison-Wesley Longman, 2nd edition, May 2004.

[29] Niels Provos, Panayiotis Mavrommatis, Moheeb A. Rajab, and Fabian Monrose. All Your iFRAMEs Point to Us. In *Proceedings of the 17th USENIX Security Symposium*, 2008.

[30] Anirudh Ramachandran and Nick Feamster. Understanding the Network-Level Behavior of Spammers. *SIGCOMM Comput. Commun. Rev.*, 36(4):291–302, 2006.

[31] SecureWorks. PRG Trojan. `http://www.secureworks.com/research/threats/prgtrojan/`, June 2007.

[32] SecureWorks. Coreflood Report. `http://www.secureworks.com/research/threats/coreflood-report/`, August 2008.

[33] Symantec. Global Internet Security Threat Report, April 2008. Trends for July – December 07.

[34] XiaoFeng Wang, Zhuowei Li, Ninghui Li, and Jong Youl Cho. PRECIP: Towards Practical and Retrofittable Confidential Information Protection. In *Proceedings of 15th Network and Distributed System Security Symposium (NDSS'08)*, 2008.

[35] 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 *Proceedings of 13th Network and Distributed System Security Symposium (NDSS'06)*, 2006.

[36] Carsten Willems, Thorsten Holz, and Felix Freiling. Toward Automated Dynamic Malware Analysis Using CWSandbox. *IEEE Security & Privacy Magazine*, 5(2):32–39, March 2007.

[37] Heng Yin, Dawn Song, Manuel Egele, Christopher Kruegel, and Engin Kirda. Panorama: Capturing System-wide Information Flow for Malware Detection and Analysis. In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS'07)*, pages 116–127, 2007.

# A Example of Information Stolen by Limbo/Nethell

Fig.9 contains three examples of credentials stolen by Limbo. The examples are sanitized so that they do not allow one to draw any conclusions about specific attacks against a particular system, and protect the identity and privacy of those involved. The first parts shows an example of stolen credentials stored in the Protected Storage: ZeuS extracts the username/password combination for two specific sites. The provider-specific model for Windows Live is $M_{\text{login.live.com}} = \{\text{login}, \text{passwd}\}$ and the second part of Fig.9 depicts an example for a match for this model. The last part presents an example for stolen cookies for the domain `www.google.com`.

```
a) *******PROTECTED STORAGE*******
   Resource: https://ssologon.bankofamerica.com/
   Description: AutoComplete Passwords
   Username: XXXk8p3
   Password: kissXXX

   Resource: http://www.myspace.com/
   Description: AutoComplete Passwords
   Username: cutelaXXX@live.com
   Password: alleXXX

b) Timestamp:08.07.2008 22:12:25
   [http://login.live.com/login.srf?wa=wsignin1.0[...]
   login=KEYLOGGED:leear2XXX.edu KEYSREAD:leear@XXX.edu
   [http://login.live.com/login.srf?wa=wsignin1.0[...]
   passwd=KEYLOGGED:04uciXX KEYSREAD:04uciXX
   [https://login.live.com/ppsecure/post.srf?wa=wsignin1.0[...]
   Sign In
   PPSX=Pass
   PwdPad=IfYouAreReadingThisYouHaveTooMuchF
   login=leear@XXX.edu
   passwd=04uciXX
   LoginOptions=2
   PPFT=Bzy3u0imy4JP6WjmRQISrSb3fwF7WDxCHfhHEHZTg74Hzktqk
   IP=XXX.181.104.92
   ID=08052008_155842_32185906

c) Timestamp:08.07.2008 18:07:54
   ************COOKIES************
   http://www.google.com/ig
   PREF=ID=a0567315adb4edXX:TB=5:TM=1137730467:LM=1208466064:
   S=4RBtzgEjOVgwH9wP; NID=12=IsSRM11qplYNeeFneEaVGt1F5LUF[...]
```

Figure 9: Sanitized examples of information stolen by Limbo infections.

# B Example of Information Stolen by ZeuS/Zbot/Wsnpoem

Fig.10 contains three (sanitized) examples of credentials stolen by ZeuS. The order of examples is the same as for Limbo: the first part depicts different credentials stolen by abusing the Protected Storage. The second part shows an example of credentials extracted during the login process of a banking website. The two fields `userid` and `password` belong to the provider-specific model for `onlineservices.wachovia.com`. The third part of the example illustrates the logfile for two stolen cookies.

```
a) [0002] VERSION: 1548 ---------------------------------------------
   System time: 09.05.2008 09:37:12, GMT: -4.00, Login time: 00:02:03
   Version: 5.1.2600 SP1, Language: 1033
   Process: C:\WINDOWS\system32\services.exe
   -
   Protected Storage:
   https://login.comcast.net/login = acesharXXX|downdXXX
   https://my.screenname.aol.com/_cqr/login/login.psp = biglamXX|laXX

b) [0739] VERSION: 1289 ---------------------------------------------
   System time: 18.06.2008 03:07:55, GMT: -4.00, Login time: 00:05:43
   Version: 5.1.2600 SP2, Language: 1033
   Process: C:\Program Files\Internet Explorer\iexplore.exe
   -
   https://onlineservices.wachovia.com/auth/AuthService
   Referer: -
   Keys: www.wachovia.com  julXXX  mckinXXX  minimXXX  july2XXX
   Data:
   action=uidLogin
   bi=version%3D1%26pm_fpua%3Dmozilla%2F4.0 %28compatible%3B msie 7.0
     windows nt 5.1%3B .net clr 1.0.3705%3B .net clr 1.1.4322%3Bmedia
     center pc 4.0%29%7C4.0 %28compatible%3B MSIE 7.0%3B Windows[...]
   requestTimestamp=1213758461593
   homepage=yes
   userid=julXXXmckinXXX
   password=july2XXX
   systemtarget=gotoOSH

c) IE Cookies:
   Path: google.com /
   PREF=ID=d9361e071b3ebeXX:TM=1214675597:LM=1214675597:S=COFeRz[...]

   Path: yahoo.com / B=3eju00t46cv1v&b=3&s=au
```

Figure 10: Sanitized examples of information stolen by ZeuS infections.

We also present some more statistical information for the ZeuS dropzones in this section. Table 5 lists the banking websites that the attackers are most interested in (based on the configuration files we

found). Fiducia and OSMP are actually not banks, but German and Russian providers of fare management systems, therefore they are marked in italics. Two different domains are used by Volks- und Raiffeisenbanken – a German bank – for online banking. Many ZeuS dropzones are configured to monitor these two domains, so that the portion of Volks- und Raiffeisenbanken is very high in this statistic: on the one hand, this is `internetbanking.gad.de` (183 times in the configuration files) and on the other hand `www.vr-networld-ebanking.de` (177 times in the configuration files). Furthermore, one large customer of Fiducia is Volks- und Raiffeisenbanken. Similar to the results for the dropzone locations, most banks can be found in North America, Europe, and Asia.

| Bank | # Appearance | Country |
|---|---|---|
| Volks- and Raiffeisenbanken | 360 | Germany |
| *OSMP* | *306* | *Russia* |
| Santander Central Hispano | 256 | Spain |
| *Fiducia* | *196* | *Germany* |
| ABN AMRO Bank | 107 | Netherlands |
| Citigroup | 105 | United States |

Table 5: Top five banking sites that should be monitored for credentials.

Among the possibility to specify websites to log or not to log, ZeuS also allows to specify hostnames in the configuration files, which should be blocked on an infected machine. Often this technique is used in order to affect functional efficiency of locally installed anti-virus engines, as shown in Table 6. Interestingly, the two domains `lem0n.info` and `i-nt-e-r-n-e-t.com` belong to other ZeuS dropzones which indicate rivalry amongst the attackers: on an infected machine, the access to other dropzones is blocked such that multiple infections of a single machine can be prevented.

| Site | # Appearance |
|---|---|
| microsoft.com | 159 |
| kaspersky.com | 126 |
| norton.com | 108 |
| lem0n.info | 93 |
| i-nt-e-r-n-e-t.com | 93 |

Table 6: Top five malicious host-file entries.

In addition to the logging functionality, ZeuS can be configured to perform man-in-the-middle attacks. Inside the configuration file exists one particular record containing several websites. If the user visits one of these sites, ZeuS reroutes this request to another fake site under the control of the attacker. The victim does not notice this rerouting since it is transparent to the user. This enables efficient phishing attacks since the attacker can use additional social engineering tricks at the fake site to trick the victim into disclosing even more credentials. The most frequently rerouted websites are shown in Table 7.

| Site | # Appearance |
|---|---|
| https://sitekey.bankofamerica.com/sas/signon.do | 35 |
| https://www.unicaja.es/PortalServlet*pag=1110902071492* | 33 |
| https://www.gruposantander.es/bog/sbi | 33 |
| https://www.bancaja.*/ControlParticulares | 33 |

Table 7: Top seven sites that are target of a man-in-the-middle attack.