Your Botnet is My Botnet: Analysis of a Botnet Takeover

Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydlowski,
Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna

University of California, Santa Barbara
[bstone,marco,sullivan,rgilbert,msz,kemm,chris,vigna]@cs.ucsb.edu

ABSTRACT

Botnets, networks of malware-infected machines that are controlled by an adversary, are the root cause of a large number of security problems on the Internet. A particularly sophisticated and insidious type of bot is Torpig, a malware program that is designed to harvest sensitive information (such as bank account and credit card data) from its victims. In this paper, we report on our efforts to take control of the Torpig botnet and study its operations for a period of ten days. During this time, we observed more than 180 thousand infections and recorded almost 70 GB of data that the bots collected. While botnets have been "hijacked" and studied previously, the Torpig botnet exhibits certain properties that make the analysis of the data particularly interesting. First, it is possible (with reasonable accuracy) to identify unique bot infections and relate that number to the more than 1.2 million IP addresses that contacted our command and control server. Second, the Torpig botnet is large, targets a variety of applications, and gathers a rich and diverse set of data from the infected victims. This data provides a new understanding of the type and amount of personal information that is stolen by botnets.

1. INTRODUCTION

Malicious code (or malware) has become one of the most pressing security problems on the Internet. In particular, this is true for bots [5], a type of malware that is written with the intent of taking over a large number of hosts on the Internet. Once infected with a bot, the victim host will join a botnet, which is a network of compromised machines that are under the control of a malicious entity, typically referred to as the botmaster. Botnets are the primary means for cyber-criminals to carry out their nefarious tasks, such as sending spam mails [36], launching denial-of-service attacks [29], or stealing personal data such as mail accounts or bank credentials [16, 39]. This reflects the shift from an environment in which malware was developed for fun, to the current situation, where malware is spread for financial profit.

Given the importance of the problem, significant research effort has been invested to gain a better understanding of the botnet phenomenon.

CCS'09, November 9–13, 2009, Chicago, Illinois, USA.

One approach to study botnets is to perform *passive analysis* of secondary effects that are caused by the activity of compromised machines. For example, researchers have collected spam mails that were likely sent by bots [47]. Through this, they were able to make indirect observations about the sizes and activities of different spam botnets. Similar measurements focused on DNS queries [34, 35] or DNS blacklist queries [37] performed by bot-infected machines. Other researchers analyzed network traffic (netflow data) at the tier-1 ISP level for cues that are characteristic for certain botnets (such as scanning or long-lived IRC connections) [24]. While the analysis of secondary effects provides interesting insights into particular botnet-related behaviors, one can typically only monitor a small portion of the Internet. Moreover, the detection is limited to those botnets that actually exhibit the activity targeted by the analysis.

A more *active* approach to study botnets is via *infiltration*. That is, using an actual malware sample or a client simulating a bot, researchers join a botnet to perform analysis from the inside. To achieve this, honeypots, honey clients, or spam traps are used to obtain a copy of a malware sample. The sample is then executed in a controlled environment, which makes it possible to observe the traffic that is exchanged between the bot and its command and control (C&C) server(s). In particular, one can record the commands that the bot receives and monitor its malicious activity. For some botnets that rely on a central IRC-based C&C server, joining a botnet can also reveal the IP addresses of other clients (bots) that are concurrently logged into the IRC channel [4, 11, 35]. While this technique worked well for some time, attackers have unfortunately adapted, and most current botnets use stripped-down IRC or HTTP servers as their centralized command and control channels. With such C&C infrastructures, it is no longer possible to make reliable statements about other bots by joining as a client.

Interestingly, due to the open, decentralized nature of peer-to-peer (P2P) protocols, it is possible to infiltrate P2P botnets such as Storm. To this end, researchers have developed crawlers that actively search the P2P network for client nodes that exhibit bot-like characteristics. Such crawls are the basis for studying the number of infected machines [18, 21] and the ways in which criminals orchestrate spam campaigns [23]. Of course, the presented techniques only work in P2P networks that can be actively crawled. Thus, they are not applicable to a majority of current botnets, which rely mostly on a centralized IRC or HTTP C&C infrastructure.

To overcome the limitations of passive measurements and infiltration – in particular in the case of centralized IRC and HTTP botnets – one can attempt to *hijack* the entire botnet, typically by taking control of the C&C channel. One way to achieve this is to directly seize the physical machines that host the C&C infrastructure [8]. Of course, this is only an option for law enforcement agencies. Alternatively, one can tamper with the domain name ser-

vice (DNS), as bots typically resolve domain names to connect to their command and control infrastructure. Therefore, by collaborating with domain registrars (or other entities such as dynamic DNS providers), it is possible to change the mapping of a botnet domain to point to a machine controlled by the defender [6]. Finally, several recent botnets, including Torpig, use the concept of *domain flux*. With domain flux, each bot periodically (and independently) generates a list of domains that it contacts. The bot then proceeds to contact them one after another. The first host that sends a reply that identifies it as a valid C&C server is considered genuine, until the next period of domain generation is started. By reverse engineering the domain generation algorithm, it is possible to pre-register domains that bots will contact at some future point, thus denying access to the botmaster and redirecting bot requests to a server under one's own control. This provides a unique view on the entire infected host population and the information that is collected by the botmasters.

In this paper, we describe our experience in actively seizing control of the Torpig (*a.k.a.* Sinowal, or Anserin) botnet for ten days. Torpig, which has been described in [40] as "one of the most advanced pieces of crimeware ever created," is a type of malware that is typically associated with bank account and credit card theft. However, as we will see, it also steals a variety of other personal information.

As mentioned previously, the Torpig botnet makes use of domain flux to locate active C&C servers. To take over this botnet, we leveraged information about the domain generation algorithm and Torpig's C&C protocol to register domains that the infected hosts would contact. By providing a valid response, the bots accepted our server as genuine, and volunteered a wealth of information, which we collected and analyzed. This is an approach that is similar to botnet takeover attempts of the Kraken [1] and Conficker [32] botnets. However, in contrast to previous takeovers, we observe that Torpig has certain properties that make our analysis particularly interesting.

First, Torpig bots transmit identifiers that permit us to distinguish between individual infections. This is different from other botnets such as Conficker. The presence of unique identifiers allows us to perform a precise estimate of the botnet size. Moreover, we can account for DHCP churn and NAT effects, which are well-known problems when computing botnet sizes. In addition, we compare our results to IP-based techniques that are commonly used to estimate botnet populations.

Second, Torpig is a data harvesting bot that targets a wide variety of applications and extracts a wealth of information from the infected victims. Together with the large size of the botnet (we observed more than 180 thousand infections), we have access to a rich data set that sheds light on the quantity and nature of the data that cyber-criminals can harvest, the financial profits that they can make, and the threats to the security and privacy of bot victims. The availability of this rich data set is different from previous work where the authors could not send valid responses to the bots (because C&C messages are authenticated [32]) or where the bots were simply not collecting such information [1].

In summary, the main contribution of this paper is a comprehensive analysis of the operations of the Torpig botnet. For ten days, we obtained information that was sent by more than 180 thousand infected machines. This data provides a vivid demonstration of the threat that botnets in general, and Torpig in particular, present to today's Internet. For our paper, we study the size of the botnet and compare our results to alternative ways of counting botnet populations. In addition, the analysis of the rich and diverse collection of user data provides a new understanding of the type and amount of personal information that is stolen by botnets.

2. BACKGROUND

Torpig is a malware that has drawn much attention recently from the security community. On the surface, it is one of the many Trojan horses infesting today's Internet that, once installed on the victim's machine, steals sensitive information and relays it back to its controllers. However, the sophisticated techniques it uses to steal data from its victims, the complex network infrastructure it relies on, and the vast financial damage that it causes set Torpig apart from other threats.

So far, Torpig has been distributed to its victims as part of Mebroot. Mebroot is a rootkit that takes control of a machine by replacing the system's Master Boot Record (MBR). This allows Mebroot to be executed at boot time, before the operating system is loaded, and to remain undetected by most anti-virus tools. More details on Mebroot can be found in [9, 12, 25]. In this paper, we will focus on Torpig, introducing Mebroot only when necessary to understand Torpig's behavior. In particular, hereinafter, we present the life cycle of Torpig and the organization of the Torpig botnet, as we observed it during the course of our analysis. We will use Figure 1 as a reference.

Victims are infected through drive-by-download attacks [33]. In these attacks, web pages on legitimate but vulnerable web sites (1) are modified with the inclusion of HTML tags that cause the victim's browser to request JavaScript code (2) from a web site (the *drive-by-download server* in the figure) under control of the attackers (3). This JavaScript code launches a number of exploits against the browser or some of its components, such as ActiveX controls and plugins. If any exploit is successful, an executable is downloaded from the drive-by-download server to the victim machine, and it is executed (4).

The downloaded executable acts as an installer for Mebroot. The installer injects a DLL into the file manager process (`explorer.exe`), and execution continues in the file manager's context. This makes all subsequent actions appear as if they were performed by a legitimate system process. The installer then loads a kernel driver that wraps the original disk driver (`disk.sys`). At this point, the installer has raw disk access on the infected machine. The installer can then overwrite the MBR of the machine with Mebroot. After a few minutes, the machine automatically reboots, and Mebroot is loaded from the MBR.

Mebroot has no malicious capability *per se*. Instead, it provides a generic platform that other modules can leverage to perform their malicious actions. In particular, Mebroot provides functionality to manage (install, uninstall, and activate) such additional modules. Immediately after the initial reboot, Mebroot contacts the *Mebroot C&C server* to obtain malicious modules (5). These modules are saved in encrypted form in the `system32` directory, so that, if the user reboots the machine, they can be immediately reused without having to contact the C&C server again. The saved modules are timestamped and named after existing files in the same directory (they are given a different, random extension), to avoid raising suspicion. After the initial update, Mebroot contacts its C&C server periodically, in two-hour intervals, to report its current configuration (i.e., the type and version number of the currently installed modules) and to potentially receive updates. All communication with the C&C server occurs via HTTP requests and responses and is encrypted using a sophisticated, custom encryption algorithm [9]. Currently, no publicly available tool exists to circumvent this encryption scheme.

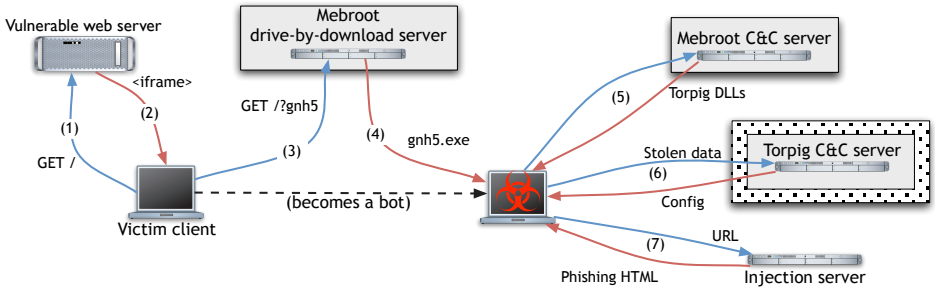

Figure 1: The Torpig network infrastructure. Shaded in gray are the components for which a domain generation algorithm is used. The component that we "hijacked" is shown with dotted background.

During our monitoring, the C&C server distributed three modules, which comprise the Torpig malware. Mebroot injects these modules (i.e., DLLs) into a number of applications. These applications include the Service Control Manager (`services.exe`), the file manager, and 29 other popular applications, such as web browsers (e.g., Microsoft Internet Explorer, Firefox, Opera), FTP clients (CuteFTP, LeechFTP), email clients (e.g., Thunderbird, Outlook, Eudora), instant messengers (e.g., Skype, ICQ), and system programs (e.g., the command line interpreter `cmd.exe`). After the injection, Torpig can inspect all the data handled by these programs and identify and store interesting pieces of information, such as credentials for online accounts and stored passwords.

Periodically (every twenty minutes, during the time we monitored the botnet), Torpig contacts the *Torpig C&C server* to upload the data stolen since the previous reporting time (6). This communication with the server is also over HTTP and is protected by a simple obfuscation mechanism, based on XORing the clear text with an 8-byte key and base64 encoding. This scheme was broken by security researchers at the end of 2008, and tools are available to automate the decryption [20]. The C&C server can reply to a bot in one of several ways. The server can simply acknowledge the data. We call this reply an `okn` response, from the string contained in the server's reply. In addition, the C&C server can send a configuration file to the bot (we call this reply an `okc` response). The configuration file is obfuscated using a simple XOR-11 encoding. It specifies how often the bot should contact the C&C server, a set of hard-coded servers to be used as backup, and a set of parameters to perform "man-in-the-browser" phishing attacks [14].

Torpig uses phishing attacks to actively elicit additional, sensitive information from its victims, which, otherwise, may not be observed during the passive monitoring it normally performs. These attacks occur in two steps. First, whenever the infected machine visits one of the domains specified in the configuration file (typically, a banking web site), Torpig issues a request to an *injection server*. The server's response specifies a page on the target domain where the attack should be triggered (we call this page the *trigger page*, and it is typically set to the login page of a site), a URL on the injection server that contains the phishing content (the *injection URL*), and a number of parameters that are used to fine tune the attack (e.g., whether the attack is active and the maximum number of times it can be launched). The second step occurs when the user visits the trigger page. At that time, Torpig requests the injection URL from the injection server and injects the returned content into the user's browser (7). This content typically consists of an HTML form that asks the user for sensitive information, for example, credit card numbers and social security numbers.

These phishing attacks are very difficult to detect, even for attentive users. In fact, the injected content carefully reproduces the style and look-and-feel of the target web site. Furthermore, the injection mechanism defies all phishing indicators included in modern browsers. For example, the SSL configuration appears correct, and so does the URL displayed in the address bar. An example screen-shot of a Torpig phishing page for Wells Fargo Bank is shown in Figure 2. Notice that the URL correctly points to `https://online.wellsfargo.com/signon`, the SSL certificate has been validated, and the address bar displays a padlock. Also, the page has the same style as the original web site.

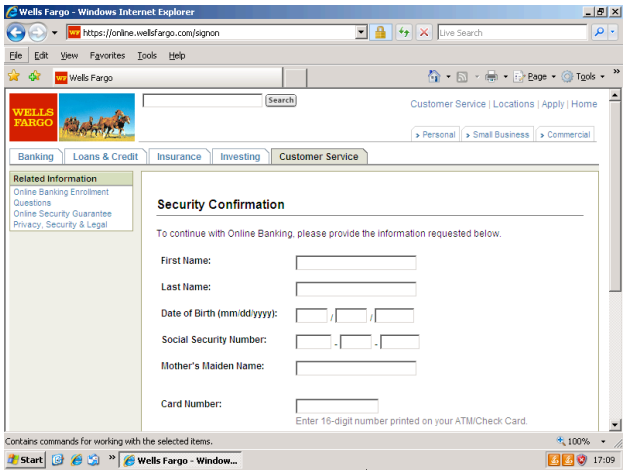

Figure 2: A man-in-the-browser phishing attack.

Communication with the injection server is protected using the standard HTTPS protocol. However, since Torpig does not check the validity of the server's certificate and blindly accepts any self-signed certificate, it is possible to mount a man-in-the-middle attack and recover the data exchanged with the injection server.

In summary, Torpig relies on a fairly complex network infrastructure to infect machines, retrieve updates, perform active phishing attacks, and send the stolen information to its C&C server. However, we observed that the schemes used to protect the communication in the Torpig botnet (except those used by the Mebroot C&C) are insufficient to guarantee basic security properties (confidentiality, integrity, and authenticity). This was a weakness that enabled us to seize control of the botnet.

3. DOMAIN FLUX

A fundamental aspect of any botnet is that of coordination; i.e., how the bots identify and communicate with their C&C servers. Traditionally, C&C hosts have been located by their bots using their

```
suffix =  ["anj", "ebf", "arm", "pra", "aym", "unj",
    "ulj", "uag", "esp", "kot", "onv", "edc"]

def generate_daily_domain():
    t = GetLocalTime()
    p = 8
    return generate_domain(t, p)

def scramble_date(t, p):
    return (((t.month ^ t.day) + t.day) * p) +
        t.day + t.year

def generate_domain(t, p):
    if t.year < 2007:
        t.year = 2007
    s = scramble_date(t, p)
    c1 = (((t.year >> 2) & 0x3fc0) + s) % 25 + 'a'
    c2 = (t.month + s) % 10 + 'a'
    c3 = ((t.year & 0xff) + s) % 25 + 'a'
    if t.day * 2 < '0' ||  t.day * 2 > '9':
        c4 = (t.day * 2) % 25 + 'a'
    else:
        c4 = t.day % 10 + '1'
    return c1 + 'h' + c2 + c3 + 'x' + c4 +
        suffix[t.month - 1]
```

Listing 1: Torpig daily domain generation algorithm.

IP address, DNS name, or their node ID in peer-to-peer overlays. In the recent past, botnet authors have identified several ways to make these schemes more flexible and robust against take-down actions, e.g., by using IP fast-flux techniques [17]. With fast-flux, the bots would query a certain domain that is mapped onto a set of IP addresses, which change frequently. This makes it more difficult to take down or block a specific C&C server. However, fast-flux uses only a single domain name, which constitutes a single point of failure.

Torpig solves this issue by using a different technique for locating its C&C servers, which we refer to as *domain flux*. With domain flux, each bot uses a domain generation algorithm (DGA) to compute a list of domain names. This list is computed independently by each bot and is regenerated periodically. Then, the bot attempts to contact the hosts in the domain list in order until one succeeds, i.e., the domain resolves to an IP address and the corresponding server provides a response that is valid in the botnet's protocol. If a domain is blocked (for example, the registrar suspends it to comply with a take-down request), the bot simply rolls over to the following domain in the list. Domain flux is also used to contact the Mebroot C&C servers and the drive-by-download servers. Domain flux is increasingly popular among botnet authors. In fact, similar mechanisms were used before by the Kraken/Bobax [1] and the Srizbi bots [46], and, more recently, by the Conficker worm [32].

In Torpig, the DGA is seeded with the current date and a numerical parameter. The algorithm first computes a "weekly" domain name, say *dw*, which depends on the current week and year, but is independent of the current day (i.e., remains constant for the entire week). Using the generated domain name *dw*, a bot appends a number of TLDs: in order, *dw*.com, *dw*.net, and *dw*.biz. It then resolves each domain and attempts to connect to its C&C server. If all three connections fail, Torpig computes a "daily" domain, say *dd*, which in addition depends on the current day (i.e., a new domain *dd* is generated each day). Again, *dd*.com is tried first, with fallbacks to *dd*.net and *dd*.biz. If these domains also fail, Torpig attempts to contact the domains hardcoded in its configuration file (e.g., `rikora.com`, `pinakola.com`, and `flippibi.com`). Listing 1 shows the pseudo-code of the routines used to generate the daily domains *dd*. The DGA used in Torpig is completely deterministic; i.e., once the current date is determined, all bots generate the same list of domains, in the same order.

From a practical standpoint, domain flux generates a list of "rendezvous points" that *may* be used by the botmasters to control their bots. Not all the domains generated by a DGA need to be valid for the botnet to be operative. However, there are two requirements that the botmasters must satisfy to maintain their grip on the botnet. First, they must control at least one of the domains that will be contacted by the bots. Second, they must use mechanisms to prevent other groups from seizing domains that will be contacted by bots before the domains under their control.

In practice, the Torpig controllers registered the weekly .com domain and, in a few cases, the corresponding .net domain, for backup purposes. However, they did not register all the weekly domains in advance, which was a critical factor in enabling our hijacking.

The use of domain flux in botnets has important consequences in the arms race between botmasters and defenders. From the attacker's point of view, domain flux is yet another technique to potentially improve the resilience of the botnet against take-down attempts. More precisely, in the event that the current rendezvous point is taken down, the botmasters simply have to register the next domain in the domain list to regain control of their botnet. On the contrary, to the defender's advantage, domain flux opens up the possibility of sinkholing (or "hijacking") a botnet, by registering an available domain that is generated by the botnet's DGAs and returning an answer that is a valid C&C response (to keep bots from switching over to the next domain in the domain list). As we mentioned, Torpig allowed both of these actions: C&C domain names were available for registration, and it was possible to forge valid C&C responses.

The feasibility of these sinkholing attacks depends not only on technical means (e.g., the ability to reverse engineer the botnet protocol and to forge a valid C&C server's response), but also on economic factors, in particular the cost of registering a number of domains sufficient to make the sinkholing effective. Since domain registration comes at a price (currently, from about $5 to $10 per year per .com and .net domain name), botmasters could prevent attacks against domain flux by making them economically infeasible, for example, by forcing defenders to register a disproportionate number of names. Unfortunately, this is a countermeasure that is already in use. Newer variants of Conficker generate 50,000 domains per day and introduce non-determinism in their generation algorithm [32]. Taking over all the domains generated by Conficker at market prices would cost between $91.3 million and $182.5 million per year. Furthermore, the domain flux arms race is clearly in favor of the malware authors. Generating thousands more domains requires an inexpensive modification to the bot code base, while registering them costs time and money.

In short, the idea of combating domain flux by simply acquiring more domains is clearly not scalable in the long term, and new approaches are needed to tilt the balance away from the botmasters. In particular, the security community should build a stronger relationship with registrars. Registrars, in fact, are the entity best positioned to mitigate malware that relies on DNS (including domain flux), but, with few exceptions, they often lack the resources, incentives, or culture to deal with the security issues associated with their roles. In addition, rogue registrars (those known to be a safe haven for the activity of cyber-criminals) should lose their accreditation. While processes exist to terminate registrar accreditation agreements (a recent case involved the infamous EstDomains registrar [2]), they should be streamlined and used more promptly.

4. TAKING CONTROL OF THE BOTNET

In this section, we describe in more detail how we obtained control over the Torpig botnet. We registered domains that bots would resolve and setup a server to which bots would connect to find their C&C. Moreover, we present our data collection and hosting infrastructure and review a timeline of events during our period of control.

The behavior of the botmasters was to not register many of the future Torpig C&C domains in advance. Therefore, we were able to register the .com and .net domains that were to be used by the botnet for three consecutive weeks from January 25th, 2009 to February 15th, 2009. However, on February 4th, 2009, the Mebroot controllers distributed a new Torpig binary that updated the domain algorithm. This ended our control prematurely after ten days. Mebroot domains, in fact, allow botmasters to upgrade, remove, and install new malware components at any time, and are tightly controlled by the criminals. It is unclear why the controllers of the Mebroot botnet did not update the Torpig domain algorithm sooner to thwart our sinkholing.

4.1 Sinkholing Preparation

We purchased service from two different hosting providers that are well-known to be unresponsive to abuse complaints, and we registered our .com and .net domains with two different registrars. This provided redundancy so that if one domain registrar or hosting provider suspended our account, we would be able to maintain control of the botnet. This proved to be useful when our .com domain was suspended on January 31, 2009 due to an abuse complaint. Fortunately, we owned the backup .net domain and were able to continue our collection unabated during this period until we could get our primary domain reinstated.

On our machines, we set up an Apache web server to receive and log bot requests, and we recorded all network traffic[1]. We then automated the process of downloading the data from our hosting providers. Once a data file was downloaded, we removed it from the server on the hosting provider. Therefore, if our servers were compromised, an attacker would not have access to any historical data. During the ten days that we controlled the botnet, we collected over 8.7GB of Apache log files and 69GB of pcap data.

We expected infected machines to connect to us on January 25th, which was the day when bots were supposed to switch to the first weekly domain name that we owned. However, on January 19th, when we started our collection, we instantly received HTTP requests from 359 infected machines. This was almost a week before the expected time. After analyzing the geographical distribution of these machines and the data they were sending, we concluded that these were probably systems that had their clock set incorrectly.

4.2 Data Collection Principles

During our collection process, we were very careful with the information that we gathered and with the commands that we provided to infected hosts. We operated our C&C servers based on previously established legal and ethical principles [3]. In particular, we protected the victims according to the following:

PRINCIPLE 1. *The sinkholed botnet should be operated so that any harm and/or damage to victims and targets of attacks would be minimized.*

PRINCIPLE 2. *The sinkholed botnet should collect enough information to enable notification and remediation of affected parties.*

[1]All the collected traffic was encrypted using 256-bit AES.

There were several preventative measures that were taken to ensure Principle 1. In particular, when a bot contacted our server, we always replied with an `okn` message and never sent it a new configuration file. By responding with `okn`, the bots remained in contact only with our servers. If we had not replied with a valid Torpig response, the bots would have switched over to the .biz domains, which had already been registered by the criminals. Although we could have sent a blank configuration file to potentially remove the web sites currently targeted by Torpig, we did not do so to avoid unforeseen consequences (e.g., changing the behavior of the malware on critical computer systems, such as a server in a hospital). We also did not send a configuration file with a different HTML injection server IP address for the same reasons. To notify the affected institutions and victims, we stored all the data that was sent to us, in accordance with Principle 2, and worked with ISPs and law enforcement agencies, including the United States Department of Defense (DoD) and FBI Cybercrime units, to assist us with this effort. This cooperation also led to the suspension of the current Torpig domains owned by the cyber-criminals.

5. BOTNET ANALYSIS

As mentioned previously, we have collected almost 70GB of data over a period of ten days. The wealth of information that is contained in this data set is remarkable. In this section, we present the results of our data analysis and important insights into the size of botnets and their victims.

5.1 Data Collection and Format

All bots communicate with the Torpig C&C through HTTP POST requests. The URL used for this request contains the hexadecimal representation of the bot identifier and a submission header. The body of the request contains the data stolen from the victim's machine, if any. The submission header and the body are encrypted using the Torpig encryption algorithm (base64 and XOR). The bot identifier (a token that is computed on the basis of hardware and software characteristics of the infected machine) is used as the symmetric key and is sent in the clear.

After decryption, the *submission header* consists of a number of key-value pairs that provide basic information about the bot. More precisely, the header contains the time stamp when the configuration file was last updated (`ts`), the IP address of the bot or a list of IPs in case of a multi-homed machine (`ip`), the port numbers of the HTTP and SOCKS proxies that Torpig opens on the infected machine (`hport` and `sport`), the operating system version and locale (`os` and `cn`), the bot identifier (`nid`), and the build and version number of Torpig (`bld` and `ver`). Figure 3 shows a sample of the header information sent by a Torpig bot.

Data Type	Data Items (#)
Mailbox account	54,090
Email	1,258,862
Form data	11,966,532
HTTP account	411,039
FTP account	12,307
POP account	415,206
SMTP account	100,472
Windows password	1,235,122

Table 1: Data items sent to our C&C server by Torpig bots.

The request body consists of zero or more *data items* of different types, depending on the information that was stolen. Table 1 shows the different data types that we observed during our monitoring. In

```
POST /A15078D49EBA4C4E/qxoT4B5uUFFqw6c35AKDYFpdZHdKLCNn...AaVpJGoSZG1at6E0AaCxQg6nIGA

ts=1232724990&ip=192.168.0.1:&sport=8109&hport=8108&os=5.1.2600&cn=United%20States&
                        nid=A15078D49EBA4C4E&bld=gnh5&ver=229
```

Figure 3: Sample URL requested by a Torpig bot (top) and the corresponding, unencrypted submission header (bottom).

```
[gnh5_229]
[MSO2002-MSO2003:pop.smith.com:John Smith:
    john@smith.com]
[pop3://john:smith@pop.smith.com:110]
[smtp://:@smtp.smith.com:25]
```

```
[gnh5_229]
POST /accounts/LoginAuth
Host:  www.google.com
POST_FORM:
Email=test@gmail.com
Passwd=test
```

Figure 4: Sample data sent by a Torpig bot: a mailbox account on the left, a form data item on the right.

particular, *mailbox account* items contain the configuration information for email accounts, i.e., the email address associated with the mailbox and the credentials required to access the mailbox and to send emails from it. Torpig obtains this information from email clients, such as Outlook, Thunderbird, and Eudora. *Email* items consist of email addresses, which can presumably be used for spam purposes. According to [45], Torpig initially used spam emails to propagate, which may give another explanation for the botmasters' interest in email addresses. *Form data* items contain the content of HTML forms submitted via POST requests by the victim's browser. More precisely, Torpig collects the URL hosting the form, the URL that the form is submitted to, and the name, value, and type of all form fields. These data items frequently contain the usernames and passwords required to authenticate with web sites. Notice that credentials transmitted over HTTPS are not safe from Torpig, since Torpig can access them before they are encrypted by the SSL layer (by hooking appropriate library functions). *HTTP account*, *FTP account*, *POP account*, and *SMTP account* data types contain the credentials used to access web sites, FTP, POP, and SMTP accounts, respectively. Torpig obtains this information by exploiting the password manager functionality provided by most web and email clients. SMTP account items also contain the source and destination addresses of emails sent via SMTP. Finally, the *Windows password* data type is used to transmit Windows passwords and other uncategorized data elements. Figure 4 shows a sample of the data items sent by a Torpig bot.

5.2 Botnet Size

In this section, we address the problem of determining the size of the Torpig botnet. More precisely, we will be referring to two definitions of a botnet's size as introduced by Rajab et al. [34]: the botnet's *footprint*, which indicates the aggregated total number of machines that have been compromised over time, and the botnet's *live population*, which denotes the number of compromised hosts that are simultaneously communicating with the C&C server.

The size of botnets is a hotly contested topic, and one that is widely, and sometimes incorrectly, reported in the popular press [7, 13, 26–28, 30]. Several methods have have been proposed in the past to estimate the size of botnets. These approaches are modeled after the characteristics of the botnet under study and vary along different axes, depending on whether they have access to direct traces of infected machines [6] or have to resort to indirect measurements [11, 34, 35, 37], whether they have a complete or partial view of the infected population, and, finally, whether individual bots are identified by using a network-level identifier (typically, an IP address) or an application-defined identifier (such as a bot ID).

In particular, we briefly compare our measurement technique to those described by Rajab et al. [34] and Kanich et al. [23], who have discussed in detail the methodological aspects of measuring a botnet's size.

Rajab et al. focus on IRC-based botnets. They propose to query DNS server caches to estimate the number of bots that resolved the name of a C&C server and to infiltrate IRC C&C channels with trackers that record the channel activity, in particular, the IDs of channel users. Both methods rely on indirect measurements of bot traffic and are based on active querying and probing. DNS cache querying is partial, since, in its basic form, it only determines if a network contains infected bots, while IRC monitoring can potentially reveal all the bots that connect to a given channel. Finally, the authors observe that IRC identifiers (i.e., nicknames) were found to overestimate the actual size of the botnet,

Kanich et al. focus on P2P botnets. In particular, they measure the size of the Storm network by active probing and crawling the Overnet distributed hash table (DHT). They confirm that the Storm botnet is not ideal for measuring its footprint and live population due to many factors such as protocol aliasing between infected and non-infected Overnet hosts and adversarial aliasing where nodes purposely poison the network to disrupt or impair its operation. The authors also caution that the application IDs used in Overnet were not a good bot identifier, due to a bug in the way they were generated by Storm.

In comparison to these studies, the Torpig C&C's architecture provides an advantageous perspective to measure the botnet's size. In fact, since we centrally and directly observed *every* infected machine that normally would have connected to the botmaster's server during the ten days that we controlled the botnet, we had a complete view of the machines belonging to the botnet. In addition, our collection methodology was entirely passive and, thus, it avoided the problem of active probing that may have otherwise polluted the network that was being measured. Finally, Torpig generates and transmits unique and persistent IDs that make for good identifiers of infected machines.

In the next section we discuss the characteristics of the botnet that enabled us to determine an overall range for the number of infected machines. We will then compare different methodologies to count the Torpig botnet's footprint and live population.

5.2.1 Counting Bots by nid

As a starting point to estimate the botnet's footprint, we analyzed the `nid` field that Torpig sends in the submission header. Our hypothesis was that this value was unique for each machine and remained constant over time, and that, therefore, it would provide an accurate method to uniquely identify each bot.

By reverse engineering the Torpig binary we were able to reconstruct the algorithm used to compute this 8-byte value. In particular, the algorithm first queries the primary SCSI hard disk for its model

and serial numbers. If no SCSI hard disk is present, or retrieving the disk information is unsuccessful, it will then try to extract the same information from the primary physical hard disk drive (i.e., IDE or SATA). The disk information is then used as input to a hashing function that produces the final `nid` value. If retrieving hardware information fails, the `nid` value is obtained by concatenating the hard-coded value of 0xBAD1D222 with the Windows volume serial number.

In all cases, the `nid` depends on (software or hardware) characteristics of the infected machine's hard disk. Therefore, it does not change, unless the hard disk is replaced (in which case the machine would no longer be infected), or the user manually changes the system's volume serial number (which requires special tools and is not likely to be done by casual users). This gave us confidence that the `nid` remains constant throughout the life of an infected machine.

We then attempted to validate whether the `nid` is unique for each bot. Therefore, we correlated this value with the other information provided in the submission header and bot connection patterns to our server. In particular, we were expecting that all submissions with a specific `nid` would report the same values for the `os`, `cn`, `bld`, and `ver` fields. Unfortunately, we found 2,079 cases for which this assumption did not hold.

Therefore, we conclude that counting unique `nid`s underestimates the botnet's footprint. As a reference point, between Jan 25, 2009 and February 4, 2009, 180,835 `nid` values were observed.

5.2.2 Counting Bots by Submission Header Fields

As a more accurate method to identify infected machines, we used the `nid`, `os`, `cn`, `bld`, and `ver` values from the submission header that Torpig bots send. As we have seen, the `nid` value is mostly unique among bots, and the other fields help distinguishing different machines that have the same `nid`. In particular, the `os` (OS version number) and `cn` (locale information) fields are determined by using the system calls `GetVersionEx` and `GetLocaleInfo`, respectively, and do not change unless the user modifies the locale information on her computer or changes her OS. The values of the `bld` and `ver` fields are hard-coded into the Torpig binary.

We decided not to use the `ts` field (time stamp of the configuration file), since its value is determined by the Torpig C&C that distributed the configuration file and not by characteristics of the bot. Also, we discarded the `ip` field, since it could change depending on DHCP and other network configurations, and the `sport` and `hport` fields, which specify the proxy ports that Torpig opens on the local machine, because they could change after a reboot.

By counting unique tuples from the Torpig headers consisting of (`nid`, `os`, `cn`, `bld`, `ver`), we estimate that the botnet's footprint for the ten days of our monitoring consisted of 182,914 machines.

5.2.3 Identifying Probers and Researchers

Finally, we wanted to identify security researchers and other curious individuals who probed our botnet servers. These do not correspond to actual victims of the botnet and, therefore, we would like to identify them and subtract them from the total botnet size.

We used two heuristics to identify probers and (likely) security researchers. First, we observed that the `nid` values generated by infected clients running in virtual machines is constant. This is because the `nid` depends essentially on physical characteristics of the hard disk, and, by default, virtual machines provide virtual devices with a fixed model and serial number. Since virtual machines are often used by researchers to study malware in a contained environment, we assume that these bots in reality correspond to researchers studying the Torpig malware. In particular, we were able to determine the `nid` values generated on a standard configuration of the VMware and QEMU virtual machines and we found 40 hosts using these values. Second, we identified hosts that send invalid requests to our C&C server (i.e., requests that cannot be generated by Torpig bots). For example, these bots used the GET HTTP method in requests where a real Torpig bot would use the POST method. Using this approach, we discounted another 74 hosts. We further ignored background noise, such as scanning of our web server and traffic from search engine bots. After subtracting probers and researchers, our final estimate of the botnet's footprint is 182,800 hosts.

5.2.4 Botnet Size vs. IP Count

It is well-known that, due to network effects such as DHCP churn and NAT, counting the number of infected bots by counting the unique IP addresses that connect to the botnet's C&C server is problematic [34]. In this section, we examine the relationship between the botnet size and the IP counts in more detail.

As we discussed, during our ten days of monitoring, we observed 182,800 bots. In contrast, during the same time, 1,247,642 unique IP addresses contacted our server. Taking this value as the botnet's footprint would overestimate the actual size by an order of magnitude. We further analyzed the difference between IP count and the actual bot count by examining their temporal characteristics. In particular, Figure 5 displays the number of unique IP addresses observed during the ten days that we were in control of the Torpig C&C. After the initial spike when the bots started to contact our server, there was a consistent diurnal pattern of unique IP addresses with an average of 4,690 new IPs per hour. In contrast, the average number of new bots observed was 705 per hour, with a very rapid drop-off after the first peak, as shown in Figure 6. Therefore, the number of cumulative new IP addresses that we saw over time increased linearly, as shown in Figure 7. On the other hand, the aggregate number of new bots observed decayed quickly. Figure 8 shows that more than 75% of all new Torpig bots during the ten-day interval were observed in the first 48 hours.

While the aggregate number of total unique IP addresses distorts the botnet's footprint and live population, the number of IP addresses can be used to closely approximate the botnet's size using other metrics. The median and average size of Torpig's live population was 49,272 and 48,532, respectively. The live population fluctuates periodically, where the peaks correspond to 9:00am Pacific Standard Time (PST), when the most computers are simultaneously online in the United States and Europe. Conversely, the smallest live population occurs around 9:00pm PST, when more people in the United States and Europe are offline. When we compare the observed number of unique bot IDs per hour with the number of unique IP addresses, they are virtually identical (as shown in Figure 9). On average, the bot IDs were only 1.3% less than the number of IP addresses per hour. Thus, the number of unique IPs per hour provides a good estimation of the botnet's live population. The similarity between bot IDs and IPs per hour is a consequence of each infected host connecting to the C&C every 20 minutes, which occurs more frequently than the rate of DHCP churn. Hence, the more often a bot connects to the C&C, the more accurate an IP count will be to the live population on an hourly scale. In comparison, the number of IPs per day does not accurately reflect the botnet's live population (as shown in Figure 10), with a difference of 36.5% between IP addresses and bot IDs. The median number and average number of IPs per day during our ten days of controlling the C&C was 182,058 and 179,866 respectively. Interestingly, both of these statistics provide a reasonable approximation to the botnet's footprint in comparison to the bot IDs.

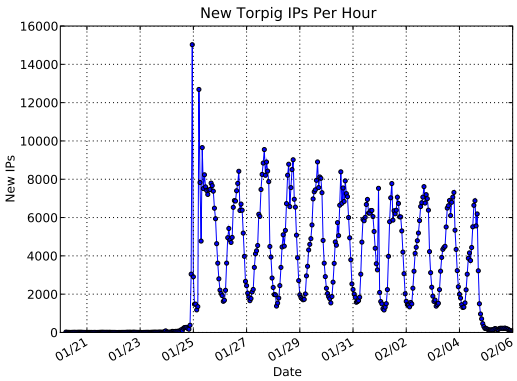

Figure 5: New unique IP addresses per hour.

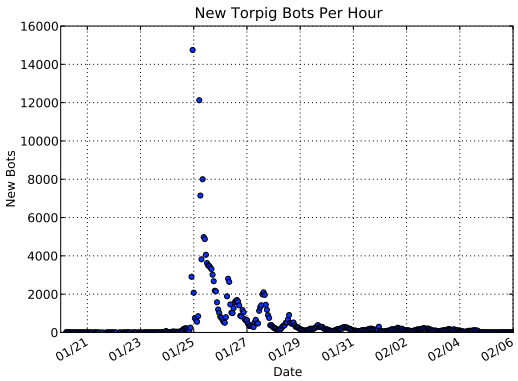

Figure 6: New bots per hour.

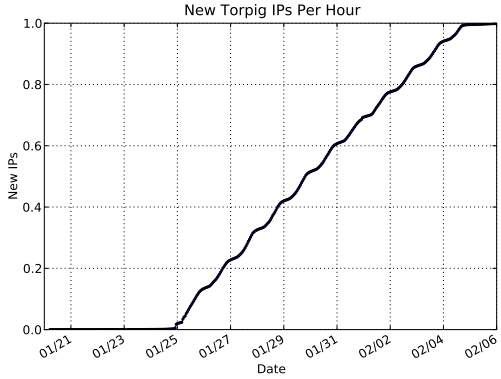

Figure 7: CDF – New unique IP addresses per hour.

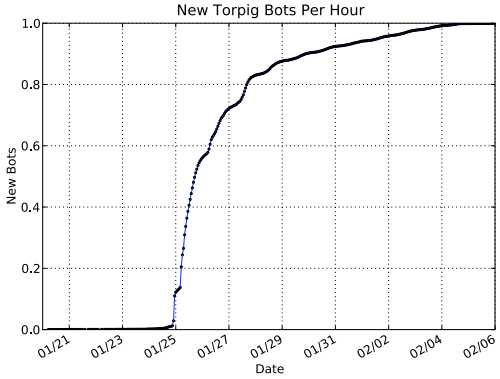

Figure 8: CDF – New bots per hour.

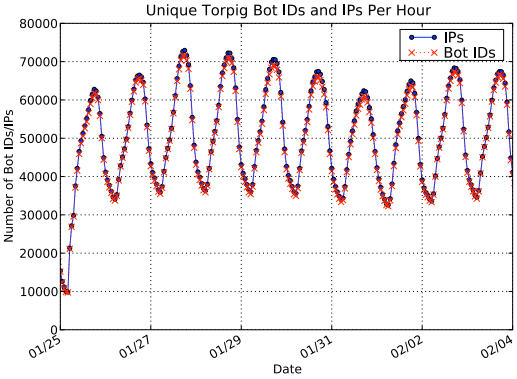

Figure 9: Unique Bot IDs and IP addresses per hour.

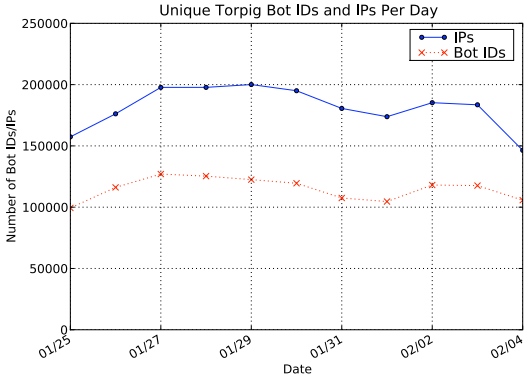

Figure 10: Unique Bot IDs and IP addresses per day.

The difference between IP count and the actual bot count can be attributed to DHCP and NAT effects. In networks using the DHCP protocol (or connecting through dial-up lines), clients (machines on the network) are allocated an address from a pool of available IP addresses. The allocation is often dynamic, that is, a client is not guaranteed to always be assigned the same IP address. This can inflate the number of observed IP addresses at the botnet C&C server. Short leases (the length of time for which the allocation is valid) can further magnify this effect. This phenomenon was very common during our monitoring. In fact, we identified the presence of ISPs that rotate IP addresses so frequently that almost every time that an infected host on their network connected to us, it had a new IP address. In one instance, a single host had changed IP addresses 694 times in just ten days! In some cases, the same host was associated with different IP addresses on the same autonomous systems, but different class B /16 subnets. We observed this DHCP churn on several different networks with the most common being, in descending order: Deutsche Telekom, Verizon, and BellSouth. Overall, there were 706 different machines that were seen with more than one hundred unique IP addresses. At this point, we can only speculate why these ISPs recycle IP addresses so frequently.

Country	IP Addresses (Raw #)	Bot IDs	DHCP Churn Factor
US	158,209	54,627	2.90
IT	383,077	46,508	8.24
DE	325,816	24,413	13.35
PL	44,117	6,365	6.93
ES	31,745	5,733	5.54
GR	45,809	5,402	8.48
CH	30,706	4,826	6.36
UK	21,465	4,792	4.48
BG	11,240	3,037	3.70
NL	4,073	2,331	1.75
Other	180,070	24,766	7.27
Totals:	1,247,642	182,800	6.83

Table 2: Top 10 infected hosts by country.

Furthermore, by comparing the number of bots we observed and their IP addresses, we can determine the effect of DHCP churn at a country level. Interestingly, the IP address count significantly overestimates the infection count in some countries, because the ISPs in those regions recycle IP addresses more often in comparison to others as shown in Table 2. For instance, a naïve estimate per country would consider Italy and Germany to have the largest number of infections. However, the ISPs in those countries assign IP addresses *much* more frequently than their U.S. counterparts. In fact, Germany had less than half the number of infected hosts, yet double the number of IP address connections. Furthermore, the ratio of IPs to hosts in Germany was four times higher than that of the United States. Because Torpig spreads through drive-by-download web sites, we believe the clustering by country reflects that most of the malicious sites use English, Italian, or German, since these are the top affected countries.

The information provided in the Torpig headers also allows us to estimate the impact of NAT, which is commonly used to enable shared Internet access for an entire private network through a single public access (masquerading). This technique reduces the number of IPs observed at the C&C server, since all the infected machines in the masqueraded network would count as one. By looking at the IP addresses in the Torpig headers we are able to determine that 144,236 (78.9%) of the infected machines were behind a NAT, VPN, proxy, or firewall. We identified these hosts by using the non-publicly routable IP addresses listed in RFC 1918: 10/8, 192.168/16, and 172.16-172.31/16. We observed 9,336 distinct bots for 2,753 IP addresses from these infected machines on private networks. Therefore, if the IP address count was used to determine the number of hosts it would underestimate the infection count by a factor of more than 3 times.

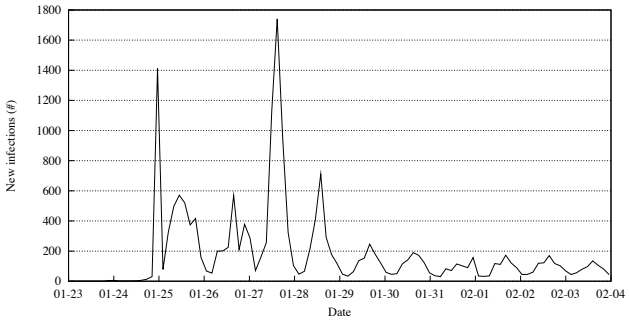

Figure 11: New infections over time.

5.2.5 New Infections

The Torpig submission header provides the time stamp of the most recently received configuration file. We leveraged this fact to approximate the number of machines newly infected during the period of observation by counting the number of distinct victims whose initial submission header contains a time stamp of 0. Figure 11 shows the new infections over time. In total, we estimate that there were 49,294 new infections while the C&C was under our control. New infections peaked on the 25th and the 27th of January. We can only speculate that, on those days, a popular web site was compromised and redirected its visitors to the drive-by-download servers controlled by the botmasters.

5.3 Botnet as a Service

An interesting aspect of the Torpig botnet is that there are indications that different groups would be dividing (and profiting from) the data it steals. Torpig DLLs are marked with a *build* type represented by the `bld` field in the header. This value is set during the drive-by download (the build type is included in the URL that triggers the download) and remains the same during the entire life cycle of an infection. The build type does not seem to indicate different feature sets, since different Torpig builds behave in the same way. However, Torpig transmits its build type in all communications with the C&C server, and, in particular, includes it in both the submission header (as the `bld` parameter) and in each data item contained in a submission body (for example, in Figure 3 the build type was `gnh5`). Therefore, the most convincing explanation of the build type is that it denotes different "customers" of the Torpig botnet, who, presumably, get access to their data in exchange for a fee. If correct, this interpretation would mean that Torpig is actually used as a "malware service", accessible to third parties who do not want or cannot build their own botnet infrastructure.

During our study, we observed 12 different values for the `bld` parameter: `dxtrbc`, `eagle`, `gnh1`, `gnh2`, `gnh3`, `gnh4`, `gnh5`, `grey`, `grobin`, `grobin1`, `mentat`, and `zipp`. Not all builds contribute equally to the amount of data stolen. The most active versions are `dxtrbc` (5,432,528 submissions), `gnh5` (2,836,198), and `mentat` (1,582,547).

6. THREATS AND DATA ANALYSIS

In this section, we will discuss the threats that Torpig poses and will turn our attention to the actual data that infected machines sent

Country	Institutions (#)	Accounts (#)
US	60	4,287
IT	34	1,459
DE	122	641
ES	18	228
PL	14	102
Other	162	1,593
Total	410	8,310

Table 3: Accounts at financial institutions stolen by Torpig.

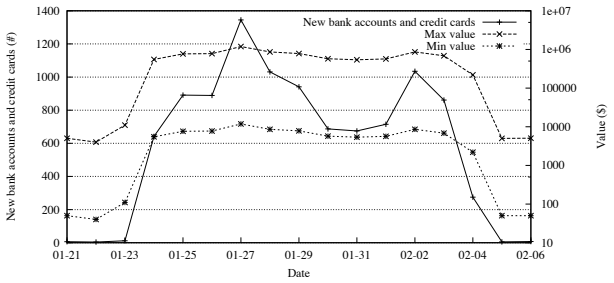

Figure 12: The arrival rate of financial data.

to our C&C server. We will see that Torpig creates a considerable potential for damage due not only to the shear volume of data it collects, but also to the amount of computing resources the botnet makes available.

6.1 Financial Data Stealing

Consistent with the past few years' shift of malware from a for-fun (or notoriety) activity to a for-profit enterprise [10, 15], Torpig is specifically crafted to obtain information that can be readily monetized in the underground market. Financial information, such as bank accounts and credit card numbers, is particularly sought after. For example, the typical Torpig configuration file lists roughly 300 domains belonging to banks and other financial institutions that will be the target of the "man-in-the-browser" phishing attacks described in Section 2.

Table 3 reports the number of accounts at financial institutions (such as banks, online trading, and investment companies) that were stolen by Torpig and sent to our C&C server. In ten days, Torpig obtained the credentials of 8,310 accounts at 410 different institutions. The top targeted institutions were PayPal (1,770 accounts), Poste Italiane (765), Capital One (314), E*Trade (304), and Chase (217). On the other end of the spectrum, a large number of companies had only a handful of compromised accounts (e.g., 310 had ten or less). The large number of institutions that had been breached made notifying all of the interested parties a monumental effort. It is also interesting to observe that 38% of the credentials stolen by Torpig were obtained from the password manager of browsers, rather than by intercepting an actual login session. It was possible to infer that number because Torpig uses different data formats to upload stolen credentials from different sources.

Another target for collection by Torpig is credit card data. Using a credit card validation heuristic that includes the Luhn algorithm and matching against the correct number of digits and numeric prefixes of card numbers from the most popular credit card companies, we extracted 1,660 unique credit and debit card numbers from our collected data. Through IP address geolocation, we surmise that 49% of the card numbers came from victims in the US, 12% from Italy, and 8% from Spain, with 40 other countries making up the balance. The most common cards include Visa (1,056), MasterCard (447), American Express (81), Maestro (36), and Discover (24).

While 86% of the victims contributed only a single card number, others offered a few more. Of particular interest is the case of a single victim from whom 30 credit card numbers were extracted. Upon manual examination, we discovered that the victim was an agent for an at-home, distributed call center. It seems that the card numbers were those of customers of the company that the agent was working for, and they were being entered into the call center's central database for order processing.

Quantifying the value of the financial information stolen by Torpig is an uncertain process because of the characteristics of the underground markets where it may end up being traded. A report by Symantec [43] indicated (loose) ranges of prices for common goods and, in particular, priced credit cards between $0.10–$25 and bank accounts from $10–$1,000. If these figures are accurate, in ten days of activity, the Torpig controllers may have profited anywhere between $83K and $8.3M.

Furthermore, we wanted to determine the rate at which the botnet produces *new* financial information for its controllers. Clearly, a botnet that generates all of its value in a few days and later only recycles stale information is less valuable than one where fresh data is steadily produced. Figure 12 shows the rate at which new bank accounts and credit card numbers were obtained during our monitoring period. In the ten days when we had control of the botnet, new data was continuously stolen and reported by Torpig bots.

6.2 Proxies

As we mentioned previously, Torpig opens two ports on the local machine, one to be used as a SOCKS proxy, the other as an HTTP proxy. 20.2% of the machines we observed were publicly accessible. Their proxies, therefore, could be easily leveraged by miscreants to, for example, send spam or navigate anonymously. In particular, we wanted to verify if spam was sent through machines in the Torpig botnet. We focused on the 10,000 IPs that contacted us most frequently. These, arguably, correspond to machines that are available for longer times and that are, thus, more likely to be used by the botmasters. We matched these IPs against the ZEN blocklist, a well-known and accurate list of IP addresses linked to spamming, which is compiled by the Spamhaus project [44]. We found that one IP was marked as a verified spam source or spam operation and 244 (2.45%) were flagged as having open proxies that are used for spam purposes or being infected with spam-related malware. While we have no evidence that the presence of these IPs on the ZEN blocklist is a consequence of the Torpig infection, it is clear that Torpig has the potential to drag its victims into a variety of malicious activities. Furthermore, since most IPs are "clean", they can be used for spamming, anonymous navigation, or other dubious enterprises.

6.3 Denial-of-Service

To approximate the amount of aggregate bandwidth among infected hosts, we mapped the IP addresses to their network speed, using the ip2location[2] database. This information is summarized in Table 4. Unfortunately the database does not contain records for about two-thirds of the IP addresses, but from the information that it provides, we can see that cable and DSL lines account for 65% of the infected hosts. If we assume the same distribution of network speed for the unknown IP addresses, there is a tremendous amount of bandwidth in the hands of the botmaster, considering that there

[2]http://www.ip2location.com

Network Speed	IP Addresses (Raw #)	Bot IDs	DHCP Churn Factor
Cable/DSL	356,428	50,535	7.05
Dial-up	129,493	9,923	13.05
Corporate	40,818	17,217	2.37
Unknown	677,434	105,125	6.44

Table 4: Network speed of infected hosts.

were more than 70,000 active hosts at peak intervals. In 2008, the median upstream bandwidth in the United States was 435 kbps for DSL connections [42]. Since the United States ranks as one of the slowest in terms of broadband speeds, we will use 435 kbps as a conservative estimate for each bot's upstream bandwidth. Thus, the aggregate bandwidth for the DSL/Cable connections is roughly 17 Gbps. If we further add in corporate networks, which account for 22% of infected hosts, and consider that they typically have significantly larger upstream connections, the aggregate bandwidth is likely to be considerably higher. Hence, a botnet of this size could cause a massive distributed denial-of-service (DDoS) attack.

6.4 Password Analysis

A recent poll conducted by Sophos in March 2009 [41], reported that one third of 676 Internet users neglect the importance of using strong passwords and admitted that they reused online authentication credentials across different web services. While it is reasonable to trust the results of a poll, it is also important to cross-validate these results, as people may not always report the truth. Typically, this validation task relies on the presence of ground truth, which is generally missing or very hard to obtain.

Interesting enough, our effort to take over the Torpig botnet over a ten-days period offered us the rare opportunity to obtain the necessary ground truth to validate the results of the Sophos poll. The benefits of the credential analysis we performed are twofold. First, it is possible to rely on real data, i.e., data that had been actually collected, and not on user-provided information, which could be fake. Second, the data corpus provided by the Torpig-infected machines was two orders of magnitude bigger than the one used in the Sophos poll, and results derived from a large data corpus are usually less prone to outliers and express trends in a better way than those performed on a smaller one.

Torpig bots stole 297,962 unique credentials (username and password pairs), sent by 52,540 different Torpig-infected machines, over the period we controlled the botnet. The stolen credentials were discovered as follows. For each infected host $\mathcal{H}$, we retrieved all the unique username and password pairs c submitted by $\mathcal{H}$. Afterward, the number w_c of distinct web services where a credential c was used was obtained. Finally, we concluded that c had been reused across w_c different web services, if w_c was greater than or equal to 2.

Our analysis found that almost 28% of the victims reused their credentials for accessing 368,501 web sites. While this percentage is slightly lower than the results reported in the poll conducted by Sophos, it is close enough to confirm and validate it.

In addition to checking for credential reuse, we also conducted an experiment to assess the strength of the 173,686 unique passwords discovered in the experiment above. To this end, we created a UNIX-like password file to feed John the Ripper, a popular password cracker tool [31]. The results are presented in Figure 13.

About 56,000 passwords were recovered in less than 65 minutes by using permutation, substitution, and other simple replacement rules used by the password cracker (the "single" mode). Another 14,000 passwords were recovered in the next 10 minutes when the password cracker switched modes to use a large wordlist. Thus, in less than 75 minutes, more than 40% of the passwords were recovered. 30,000 additional passwords were recovered in the next 24 hours by brute force (the "incremental" mode).

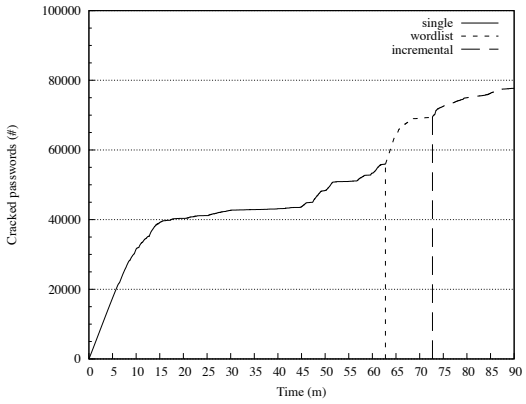

Figure 13: Number of passwords cracked in 90 minutes by the John the Ripper password cracker tool. Vertical lines indicate when John switches cracking mode. The first vertical line represents the switching from simple transformation techniques ("single" mode) to wordlist cracking, the second from wordlist to brute-force ("incremental").

7. RELATED WORK

Other analyses of both Mebroot and Torpig have been done [9, 12, 25]. These primarily focus on the Master Boot Record (MBR) overwriting rootkit technique employed by Mebroot. We complement this work, since the focus of our analysis has been on the Torpig botnet.

Torpig utilizes a relatively new strategy for locating its C&C servers, which we refer to as *domain flux*. Analyses of other bot families like Kraken/Bobax [1], Srizbi [46], and more recently, Conficker [32], have revealed that the use of the domain flux technique for bot coordination is on the rise. We present domain flux in detail and discuss its strengths and weaknesses, and we propose several remediation strategies.

Botnet takeover as an analysis and defense strategy have been considered elsewhere. Kanich *et al.* infiltrated the Storm botnet by impersonating proxy peers in the overlay network. They demonstrated their control by rewriting URLs in the spam sent by the bots [22]. Recent efforts to disrupt the Conficker botnet have focused on sinkholing future rendezvous domains in order to disable the botmaster's ability to update the infected machines [19]. Our takeover of Torpig is closest in spirit to the latter effort, as we also took advantage of the shortcomings of using domain flux for C&C.

Determining the size of a botnet is difficult. Many studies have used the number of unique IP addresses to estimate the number of compromised hosts [34]. Recently, Conficker has been reported to have infected between one and ten million machines using this heuristic [32]. The Storm botnet's size was approximated by crawling the Overnet distributed hash table (DHT) and counting DHT identifiers and IP address pairs [18]. We believe many of these studies overestimate the actual bot population size for the reasons we detailed previously. On the contrary, we have provided a detailed discussion of how we determine the size of the Torpig botnet.

There has been work focused on understanding the information harvested by malware. For example, Holz *et al.* analyzed data

from 70 dropzone servers containing information extracted from keyloggers [16]. Also, a Torpig server was seized in 2008, resulting in the recovery of 250,000 stolen credit and debit cards and 300,000 online bank account login credentials [38]. Furthermore, Franklin *et al.* classified and assessed the value of compromised credentials for financial and other personal information that is bought and sold in the underground Internet economy [10]. Unlike these studies, we analyzed live data that was sent directly to us by bots. This allows us to gain further insights, such as the timing relationships between events.

8. CONCLUSIONS

In this paper, we present a comprehensive analysis of the operations of the Torpig botnet. Controlling hundreds of thousands of hosts that were volunteering Gigabytes of sensitive information provided us with the unique opportunity to understand both the characteristics of the botnet victims and the potential for profit and malicious activity of the botnet creators.

There are a number of lessons learned from the analysis of the data we collected, as well as from the process of obtaining (and losing) the botnet. First, we found that a naïve evaluation of botnet size based on the count of distinct IPs yields grossly overestimated results (a finding that confirms previous, similar results). Second, the victims of botnets are often users with poorly maintained machines that choose easily guessable passwords to protect access to sensitive sites. This is evidence that the malware problem is fundamentally a *cultural* problem. Even though people are educated and understand well concepts such as the physical security and the necessary maintenance of a car, they do not understand the consequences of irresponsible behavior when using a computer. Therefore, in addition to novel tools and techniques to combat botnets and other forms of malware, it is necessary to better educate the Internet citizens so that the number of potential victims is reduced. Third, we learned that interacting with registrars, hosting facilities, victim institutions, and law enforcement is a rather complicated process. In some cases, simply identifying the point of contact for one of the registrars involved required several days of frustrating attempts. We are sure that we have not been the first to experience this type of confusion and lack of coordination among the many pieces of the botnet puzzle. However, in this case, we believe that simple rules of behavior imposed by the US government would go a long way toward preventing obviously-malicious behavior.

Acknowledgments

The research was supported by the National Science Foundation, under grant CNS-0831408.

We would like to acknowledge the following people and groups for their help during this project: David Dagon, MELANI/GovCERT.ch[3], and the Malware Domain List community[4].

9. REFERENCES

[1] P. Amini. Kraken Botnet Infiltration. http://dvlabs.tippingpoint.com/blog/2008/04/28/kraken-botnet-infiltration, 2008.

[2] S. Burnette. Notice of Termination of ICANN Registrar Accreditation Agreement. http://www.icann.org/correspondence/burnette-to-tsastsin-28oct08-en.pdf, 2008.

[3] A. Burstein. Conducting Cybersecurity Research Legally and Ethically. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2008.

[4] E. Cooke, F. Jahanian, and D. McPherson. The zombie roundup: Understanding, detecting, and disrupting botnets. In *Usenix Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, 2006.

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

[6] D. Dagon, C. Zou, and W. Lee. Modeling Botnet Propagation Using Time Zones. In *Symposium on Network and Distributed System Security*, 2006.

[7] Finjan. How a cybergang operates a network of 1.9 million infected computers. http://www.finjan.com/MCRCblog.aspx?EntryId=2237, 2009.

[8] J. Fink. FBI Agents Raid Dallas Computer Business. http://cbs11tv.com/local/Core.IP.Networks.2.974706.html, 2009.

[9] E. Florio and K. Kasslin. Your computer is now stoned (...again!). *Virus Bulletin*, April 2008.

[10] J. Franklin, V. Paxson, A. Perrig, and S. Savage. An Inquiry into the Nature and Causes of the Wealth of Internet Miscreants. In *ACM Conference on Computer and Communications Security*, 2007.

[11] F. Freiling, T. Holz, and G. Wicherski. Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks. In *European Symposium on Research in Computer Security (ESORICS)*, 2005.

[12] GMER Team. Stealth MBR rootkit. http://www2.gmer.net/mbr/, 2008.

[13] D. Goodin. Superworm seizes 9m pcs, 'stunned' researchers say. http://www.theregister.co.uk/2009/01/16/9m_downadup_infections/, 2009.

[14] P. Guehring. Concepts against Man-in-the-Browser Attacks. http://www2.futureware.at/svn/sourcerer/CAcert/SecureClient.pdf, 2006.

[15] P. Gutmann. The Commercial Malware Industry. In *DEFCON conference*, 2007.

[16] T. Holz, M. Engelberth, and F. Freiling. Learning More About the Underground Economy: A Case-Study of Keyloggers and Dropzones. Reihe Informatik TR-2008-006, University of Mannheim, 2008.

[17] T. Holz, C. Gorecki, K. Rieck, and F. Freiling. Measuring and Detecting Fast-Flux Service Networks. In *Symposium on Network and Distributed System Security*, 2008.

[18] T. Holz, M. Steiner, F. Dahl, E. Biersack, and F. Freiling. Measurements and Mitigation of Peer-to-Peer-based Botnets: A Case Study on Storm Worm. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2008.

[19] J. Hruska. Cracking down on Conficker: Kaspersky, OpenDNS join forces. http://arstechnica.com/business/news/2009/02/cracking-down-on-conficker-kaspersky-opendns-join-forces, February 2009.

[20] D. Jackson. Untorpig. http://www.secureworks.com/research/tools/untorpig/, 2008.

[21] B. Kang, E. Chan-Tin, C. Lee, J. Tyra, H. Kang, C. Nunnery, Z. Wadler, G. Sinclair, N. Hopper, D. Dagon, and Y. Kim. Towards complete node enumeration in a peer-to-peer

[3]Email: cert@melani.admin.ch, URL:http://www.melani.admin.ch/index.html?lang=en

[4]http://www.malwaredomainlist.com/

botnet. In *ACM Symposium on Information, Computer & Communication Security (ASIACCS 2009)*, 2009.

[22] C. Kanich, C. Kreibich, K. Levchenko, B. Enright, G. Voelker, V. Paxson, and S. Savage. Spamalytics: An Empirical Analysis of Spam Marketing Conversion. In *ACM Conference on Computer and Communications Security*, 2008.

[23] C. Kanich, K. Levchenko, B. Enright, G. Voelker, and S. Savage. The Heisenbot Uncertainty Problem: Challenges in Separating Bots from Chaff. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2008.

[24] A. Karasaridis, B. Rexroad, and D. Hoeflin. Wide-scale botnet detection and characterization. In *USENIX Workshop on Hot Topics in Understanding Botnet*, 2007.

[25] P. Kleissner. Analysis of Sinowal. `http://web17.webbpro.de/index.php?page=analysis-of-sinowal`, 2008.

[26] J. Leyden. Conficker botnet growth slows at 10m infections. `http://www.theregister.co.uk/2009/01/26/conficker_botnet/`, 2009.

[27] J. Leyden. Conficker zombie botnet drops to 3.5 million. `http://www.theregister.co.uk/2009/04/03/conficker_zombie_count/`, 2009.

[28] R. McMillan. Conficker group says worm 4.6 million strong. `http://www.cw.com.hk/content/conficker-group-says-worm-46-million-strong`, 2009.

[29] D. Moore, G. Voelker, and S. Savage. Inferring Internet Denial of Service Activity. In *Usenix Security Symposium*, 2001.

[30] G. Ollmann. Caution Over Counting Numbers in C&C Portals. `http://blog.damballa.com/?p=157`, 2009.

[31] Openwall Project. John the Ripper password cracker. `http://www.openwall.com/john/`.

[32] P. Porras, H. Saidi, and V. Yegneswaran. A Foray into Conficker's Logic and Rendezvous Points. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2009.

[33] N. Provos and P. Mavrommatis. All Your iFRAMEs Point to Us. In *USENIX Security Symposium*, 2008.

[34] M. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. My Botnet is Bigger than Yours (Maybe, Better than Yours) : Why Size Estimates Remain Challenging. In *USENIX Workshop on Hot Topics in Understanding Botnet*, 2007.

[35] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A Multifaceted Approach to Understanding the Botnet Phenomenon. In *ACM Internet Measurement Conference (IMC)*, 2006.

[36] A. Ramachandran and N. Feamster. Understanding the Network-level Behavior of Spammers. In *ACM SIGCOMM*, 2006.

[37] A. Ramachandran, N. Feamster, and D. Dagon. Revealing Botnet Membership Using DNSBL Counter-Intelligence. In *Conference on Steps to Reducing Unwanted Traffic on the Internet*, 2006.

[38] RSA FraudAction Lab. One Sinowal Trojan + One Gang = Hundreds of Thousands of Compromised Accounts. `http://www.rsa.com/blog/blog_entry.aspx?id=1378`, October 2008.

[39] S. Saroiu, S. Gribble, and H. Levy. Measurement and Analysis of Spyware in a University Environment. In *Networked Systems Design and Implementation (NSDI)*, 2004.

[40] M. Shields. Trojan virus steals banking info. `http://news.bbc.co.uk/2/hi/technology/7701227.stm`, 2008.

[41] Sophos. Security at risk as one third of surfers admit they use the same password for all websites, Sophos reports. `http://www.sophos.com/pressoffice/news/articles/2009/03/password-security.html`, March 2009.

[42] SpeedMatters.org. 2008 Report on Internet Speeds in All 50 States. `http://www.speedmatters.org/document-library/sourcematerials/cwa_report_on_internet_speeds_2008.pdf`, August 2008.

[43] Symantec. Report on the underground economy. `http://www.symantec.com/content/en/us/about/media/pdfs/Underground_Econ_Report.pdf`, 2008.

[44] The Spamhaus Project. ZEN. `http://www.spamhaus.org/zen/`.

[45] VeriSign iDefense Intelligence Operations Team. The Russian Business Network: Rise and Fall of a Criminal ISP. `blog.wired.com/defense/files/iDefense_RBNUpdated_20080303.doc`, 2008.

[46] J. Wolf. Technical details of Srizbi's domain generation algorithm. `http://blog.fireeye.com/research/2008/11/technical-details-of-srizbis-domain-generation-algorithm.html`, 2008.

[47] L. Zhuang, J. Dunagan, D. Simon, H. Wang, I. Osipkov, G. Hulten, and J. Tygar. Characterizing botnets from email spam records. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2008.