Analysis of a Botnet Takeover

This article describes an effort to take control of a particularly sophisticated and insidious botnet and study its operations for a period of 10 days. It summarizes what the authors learned and reports on what has happened to that botnet since.

Brett Stone-Gross, Marco Cova, Bob Gilbert, Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna
University of California, Santa Barbara

Botnets, networks of malware-infected machines (bots) controlled by an adversary, are the root cause of a large number of Internet security problems. They're the primary way cybercriminals carry out their nefarious tasks, such as sending spam, launching denial-of-service attacks, or stealing personal data. A particularly sophisticated and insidious variety is called Torpig, malware designed to harvest sensitive information such as bank account and credit-card data from its victims.

To learn more about how botnets operate and what information they collect—particularly, in the case of centralized IRC- and HTTP-based botnets—you can attempt to hijack the entire botnet. In this article, we describe our experience with actively seizing control of the Torpig (also called Sinowal or Anserin) botnet and studying its operations for 10 days. During this time, we observed more than 180,000 infections and recorded almost 70 Gbytes of data. Although Torpig typically targets bank account and credit-card data, we found that it also steals a variety of other personal information. Ultimately, we were able to determine the botnet's size precisely and compare our results to alternative ways of counting botnet populations. The data provides a vivid demonstration of the threat that botnets in general and Torpig in particular present to today's Internet. We also report on what has happened in the time that has passed since we lost control of the Torpig botnet and discuss some of the ethical and legal considerations of this type of research.

Torpig Life Cycle

On the surface, Torpig is one of many Trojan horses infesting today's Internet. 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 malware. A review of the Torpig life cycle, shown in Figure 1, illustrates the nature of the threat.

Torpig's victims acquire the malware as part of the Mebroot rootkit, which takes control of a machine by replacing the system's Master Boot Record (MBR). Attackers modify legitimate but vulnerable webpages (step 1 in Figure 1) with the inclusion of HTML tags that cause the victim's browser to request JavaScript code (step 2) from a webpage under the attackers' control (step 3)—a so-called drive-by download attack[1]. The JavaScript launches exploits against the browser or some of its components, such as ActiveX controls and plugins. If any exploit is successful, the victim's machine downloads and executes a program from the drive-by download server. The victim's machine then becomes a bot (step 4).

Mebroot has no malicious capability on its own. Instead, it provides a generic platform that installs, uninstalls, and activates other modules (such as DLLs). Mebroot initially contacts the Mebroot command-and-control (C&C) server to obtain malicious modules (step 5). It places these modules, in encrypted form, in the `system32` directory so that if the user

 1540-7993/11/$26.00 © 2011 IEEE

reboots the machine, it can reuse them immediately without having to contact the C&C server. It also timestamps the modules and names them after existing files in the same directory (with 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 (that is, the type and version number of the currently installed modules) and to receive any updates. All communication with the C&C server occurs via HTTP requests and responses using a sophisticated, custom encryption algorithm.

In the case of the Torpig botnet, the Mebroot C&C server distributes the Torpig malware modules, and Mebroot injects them into some number of applications. These might include the Service Control Manager (`services.exe`), the file manager, Web browsers (for example, Microsoft Internet Explorer, Firefox, and Opera), FTP clients (such as CuteFTP and LeechFTP), email clients (such as Thunderbird, Outlook, and Eudora), instant messengers (for example, Skype and ICQ), and system programs (such as the command line interpreter `cmd.exe`). After the injection, Torpig can inspect all the data handled by the infected programs and identify and store interesting pieces of information, such as credentials for online accounts and stored passwords.

Every 20 minutes Torpig contacts the Torpig C&C server to upload stolen data (step 6). This communication with the server also occurs over HTTP, protected by a simple obfuscation mechanism based on XORing the clear text with an 8-byte key and base64 encoding the result. (Security researchers broke this scheme at the end of 2008, and tools are now available to automate the decryption, such as Don Jackson's Untorpig, available from http://www.secureworks.com/research/tools/untorpig/.) The C&C server can reply to a bot in one of several ways. The server can simply acknowledge the data in what we call an `okn` response, because of the string contained in the server's reply. The C&C server can also send a configuration file to the bot (we call this an `okc` response), obfuscated by a simple XOR-11 encoding. This file 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.

Torpig uses phishing attacks to actively elicit additional, sensitive information from its victims beyond that which it might acquire 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 webpage), Torpig issues a request to an injection server. The server's response identifies a *trigger page* on the target domain to instigate the attack (typically the site's login page), a URL on the injection server that contains the phishing content (the injection URL), and several parameters for fine-tuning the attack (for example, specifying whether the attack is active and the maximum number of times to launch it). The second step occurs when the user visits the trigger page: Torpig requests the injection URL from the injection server and puts the returned content into the user's browser (step 7). This content typically consists of an HTML form that asks the user for sensitive information, such as credit-card and social security numbers.

Even attentive users find these phishing attacks difficult to detect. The injected content carefully reproduces the target webpage's style and "look and feel," and the injection mechanism defies all phishing indicators included in modern browsers. For example, the SSL configuration appears correct, as does the URL displayed in the address bar, as shown in Figure 2, a screenshot of a Torpig phishing page for Chase Bank.

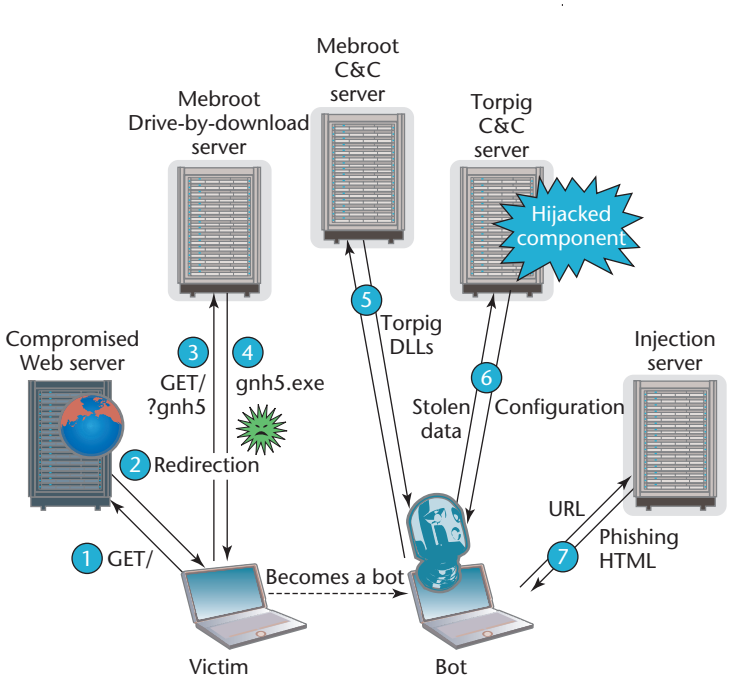

Figure 1. The Torpig network infrastructure. Shaded in gray are the components owned by the criminals. The Torpig command-and-control server is the component that we "hijacked." Step 1: Attackers modify vulnerable webpages. Step 2: Modified page redirects victim's browser to drive-by download server. Step 3: Vulnerable browser requests JavaScript. Step 4: Victim downloads and executes Mebroot to become a bot. Step 5: Bot obtains Torpig modules. Step 6: Bot uploads data stolen from victim's computer. Step 7: When browsing a targeted site, victim is redirected to HTML injection server for man-in-the-browser attack.

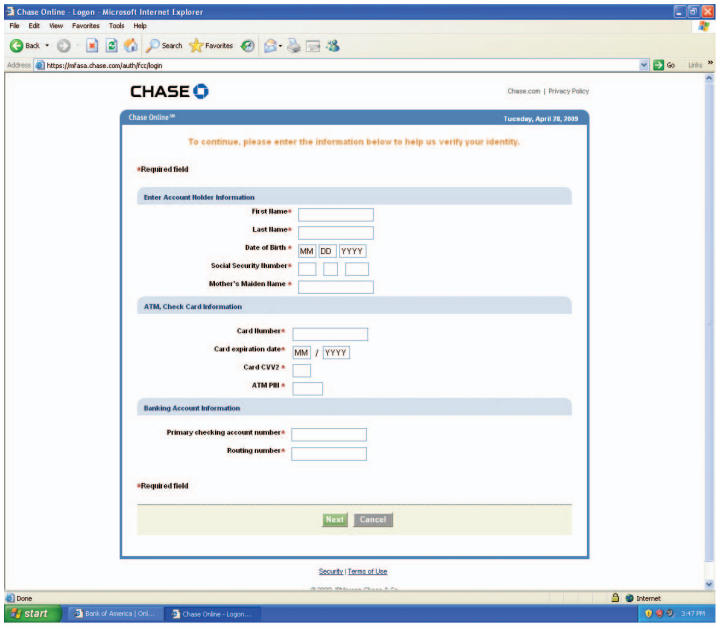

Figure 2. A man-in-the-browser phishing attack. Not only does the page have the same style as the original webpage, but the URL correctly points to the login page, the SSL certificate appears to be valid, and the status bar displays a padlock.

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 that protect the communication in the Torpig botnet (except those used by the Mebroot C&C) are insufficient to guarantee basic security (confidentiality, integrity, and authenticity). This was a weakness that enabled us to seize control of the botnet.

Taking Control of the Botnet

A fundamental aspect of any botnet is its coordination—that is, how the individual bots identify and communicate with their C&C servers. We set out to take control of Torpig through its C&C channel.

Traditionally, bots located their C&C hosts through their IP addresses, DNS names, or node identifiers in peer-to-peer overlays. Recently, botnet authors have found ways to make these schemes more flexible and robust against take-down actions, for example, by using IP *fast-flux* techniques.[2] With fast-flux, bots query a certain domain that corresponds to a set of IP addresses that change frequently. While this makes it more difficult to take down or block a specific C&C server, the use of only one domain name constitutes a single point of failure.

Several recent botnets, including Torpig, use *domain flux* instead, in which each bot independently uses a domain generation algorithm (DGA) to compute a list of domain names. The bot attempts to contact them in order until one successfully resolves to an IP address and the corresponding server provides a valid response. The bot then treats that host as genuine until the next round of domain generation. Domain flux is increasingly popular among botnet authors; the Kraken/Bobax and Srizbi bots and, more recently, the Conficker worm used similar mechanisms. By reverse engineering the domain generation algorithm, it's possible to predict the domains the bots will attempt to contact.

Torpig's DGA relies on the current date and a numerical parameter. The algorithm first computes a "weekly" domain name—call it *dw*—that depends on the current week and year but is independent of the current day. (In other words, it remains constant for the entire week.) The bot then appends TLDs to the weekly domain name, generating domains such as *dw*.com, *dw*.net, and *dw*.biz. If attempts to reach its C&C server at those domains fail, Torpig computes a "daily" domain, say, *dd*, which also depends on the current day. (In other words, it generates a new domain each day.) Again, the bot tries *dd*.com first, with fallbacks to *dd*.net and *dd*.biz. If these domains also fail, Torpig attempts to contact the domains hard-coded in its configuration file. The DGA used in Torpig is completely deterministic: given a specific current date, all bots generate the same list of domains in the same order.

In practice, the Torpig controllers registered the weekly .com domain and, in a few cases, the corresponding .net domain for backup purposes. However, we observed that the botmasters tended not to register many of the future Torpig C&C domains in advance, which was a critical factor in enabling our hijacking. We were able to register the .com and .net domains that the botnet would be using for three consecutive weeks, from 25 January 2009 to 15 February 2009, and set up our own server to appear as a C&C host—a process known as *sinkholing*.

We set up an Apache Web server on our machines to receive and log bot requests, and we recorded all network traffic. We then automated the process of downloading the data from our hosting providers. Once a data file was downloaded, we removed it from the hosting provider's server—thus, if our servers were compromised, an attacker would not have access to any historical data.

Mebroot domains allow botmasters to upgrade, remove, and install new malware components at any time, and criminals control them tightly. On 4 February 2009, the Mebroot controllers distributed a new Torpig binary that updated the domain generation algorithm, ending our control prematurely. But during the 10 days that

we controlled the botnet, we collected over 8.7 Gbytes of Apache log files and 69 Gbytes of raw network traffic. We encrypted all collected traffic and postprocessed the data using a 256-bit AES key known by only those working on the project and stored offline. After our experiment was completed, we copied the encrypted data to an external drive, removed the data from our machines, and placed the drive in a safe.

Botnet Analysis

As mentioned previously, we collected almost 70 Gbytes of data over the 10 days that we controlled Torpig. Here, we review our data analysis and important insights into the size of botnets and their victims.

Data Collection and Format

All bots communicate with the Torpig C&C through HTTP POST requests, using a URL that 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 bot encrypts the submission header and the body using the Torpig encryption algorithm, and it uses the bot identifier (a token based on the infected machine's hardware and software characteristics) as the symmetric key, sending it in the clear.

After decryption, the submission header consists of several key value pairs that provide basic information about the bot. More precisely, the header contains the timestamp for the last update of the configuration file (`ts`), the IP address of the bot or a list of IP addresses for a multihomed 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`).

The request body consists of *data items* of different types, depending on the stolen information. Table 1 shows the different data types that we observed during our monitoring, in order of frequency. *Form data* items contain the contents of HTML forms submitted via POST requests by the victim's browser. Torpig collects the URL of the form's host, the URL for the form's submission, and the name, value, and type of all form fields. These data items frequently contain the usernames and passwords required to authenticate with websites. Note that credentials transmitted over HTTPS aren't safe from Torpig, since Torpig accesses them before encryption through the SSL layer.

Email items consist of email addresses, presumably useful for spam purposes. The *Windows password* data type is used to transmit Windows passwords and other uncategorized data elements. Torpig obtains this information from email clients, such as Outlook, Thunderbird, and Eudora. *POP account*, *HTTP account*, *FTP account*, and *SMTP account* data types contain the credentials used to access these accounts at their respective servers. 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, *mailbox account* items contain the configuration information for email accounts—that is, the email address associated with the mailbox and the credentials required to access the mailbox and to send emails from it.

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

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

Botnet Size

In order to better understand the scale of the threat posed by Torpig, we needed to determine the botnet's size. We refer to two definitions as introduced by Rajab et al.[3]: its *footprint*, which indicates the total number of machines that have been compromised over time, and its *live population*, which denotes the number of compromised hosts simultaneously communicating with the C&C server.

The Torpig architecture provides an advantageous perspective for measuring 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, we had a complete view of the machines in the botnet. In addition, our collection methodology was entirely passive and thus avoided active probing that might have otherwise polluted the measured network. Fortunately, Torpig generates and transmits unique and persistent IDs that make for good identifiers of infected machines.

Counting bots using submission header fields. As a starting point to estimate the botnet's footprint, we analyzed the nid field that Torpig sends in the submission header. By reverse engineering the Torpig binary, we were able to reconstruct the algorithm used to compute this 8-byte value. The algorithm

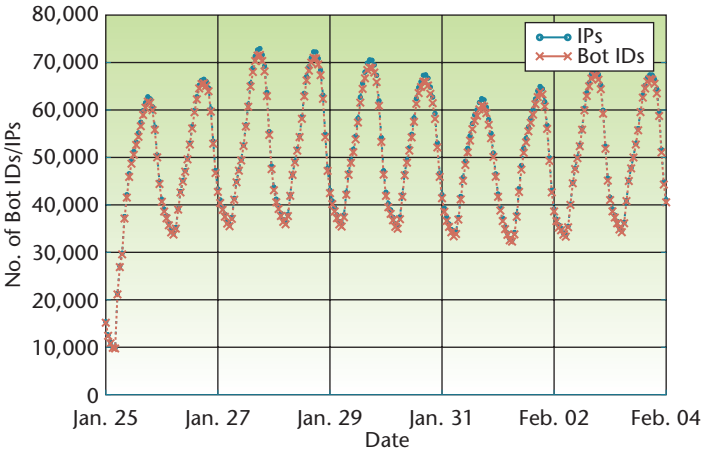

Figure 3. Unique bot IDs and IP addresses per hour. The graph illustrates that the number of unique IP addresses per hour provides a good estimation of Torpig's live population.

first queries the infected machine's 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 then tries to extract the same information from the primary physical hard disk drive (IDE or SATA). It then uses the disk information as input to a hashing function that produces the final nid value. If the attempts to retrieve hardware information fail, Torpig obtains the nid value by concatenating a hard-coded value with the Windows volume serial number.

We attempted to validate whether the `nid` is unique for each bot by correlating this value with the other information provided in the submission header and with bot connection patterns to our server. In particular, we expected that all submissions with a specific `nid` would also report the same values for the `os`, `cn`, `bld`, and `ver` fields. Instead, we found 2,079 cases for which this assumption did not hold. Therefore, we conclude that counting unique `nids` underestimates the botnet's footprint. As a reference point, between 25 January 2009 and 4 February 2009, we observed 180,835 `nid` values.

To more accurately identify the infected machines, we used the `nid`, `os`, `cn`, `bld`, and `ver` values from the submission header. Although the `nid` value is mostly unique among bots, the other fields help distinguish different machines that have the same `nid`. In particular, Torpig determines the `os` (OS version number) and `cn` (locale information) fields with the system calls `GetVersionEx` and `GetLocaleInfo`, respectively, which don't change unless the user modifies the locale information or changes the OS. The Torpig binary contains hard-coded values for the `bld` and `ver` fields. By counting unique tuples from the Torpig headers consisting of (`nid`, `os`, `cn`, `bld`, `ver`), we estimated that the botnet's footprint for the 10 days of our monitoring consisted of more than 182,000 machines.

Botnet size vs. IP count. It's well known that counting the number of infected bots by counting the unique IP addresses that connect to the botnet's C&C server is problematic, due to network effects such as DHCP churn and NAT.

During 10 ten days of monitoring, we observed 182,914 bots. In contrast, 1,247,642 unique IP addresses contacted our server during the same period. Therefore, taking the IP count as the botnet's footprint would overestimate the actual size by an order of magnitude.

While the aggregate number of unique IP addresses distorts the size of the botnet's footprint, counting IP addresses can help determine a close approximation of the botnet's size using other metrics. The median and average sizes of Torpig's live population were 49,272 and 48,532, respectively. The live population fluctuates, with peaks corresponding to 9:00 a.m. Pacific Standard Time (PST), when the most computers are simultaneously online in the US and Europe. The smallest live population occurs around 9:00 p.m. PST, when more people in the US and Europe are offline. The observed number of unique bot IDs and unique IP addresses per hour are virtually identical, as shown in Figure 3—on average, the bot IDs were only 1.3 percent fewer than the number of IP addresses. Thus, the number of unique IP addresses per hour provides a good estimation of the botnet's live population.

DHCP and NAT effects account for the difference between IP count and the actual bot count. Networks using DHCP or connecting through dial-up lines allocate clients (machines on the network) an address from a pool of available IP addresses. The allocation is often dynamic, meaning that a client doesn't always get the same IP address, which 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 some ISPs that rotated IP addresses so frequently that almost every time an infected host connected to us, it had a new IP address. In one instance, a single host changed IP addresses 694 times in just 10 days. In other cases, a host was associated with different IP addresses on the same autonomous systems but in different class B/16 subnets. Overall, we observed 706 different machines with more than 100 unique IP addresses each.

Threats and Data Analysis

In our research, we found that Torpig creates a considerable potential for damage, due not only to the sheer volume of data it collects but also to the amount of computing resources the botnet makes available.

Financial Data Stealing

Torpig specifically goes after information that's easy to monetize in the underground market, particularly financial information such as bank accounts and credit-card numbers. The typical Torpig configuration file lists roughly 300 domains belonging to banks and other financial institutions that will be the target of the phishing attacks described earlier.

In 10 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). At the other end of the spectrum, a large number of companies had only a handful of compromised accounts; for example, 310 had 10 or fewer. We noticed that Torpig obtained 38 percent of its stolen credentials from the browser's password manager, 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.)

Torpig also targets 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. Based on IP address geolocation, we surmise that 49 percent of the card numbers came from victims in the US, 12 percent from Italy, and 8 percent 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).

Quantifying the value of the financial information stolen by Torpig is an uncertain process because of the nature of the underground markets. The 2008 "Symantec Report on The Underground Economy" (http://www.symantec.com/content/en/us/about/media/pdfs/Underground_Econ_Report.pdf.) indicated (loose) ranges of prices for common goods: it priced credit cards between US$0.10 and $25, and bank accounts from $10 to $1,000. If these figures are accurate, in 10 days of activity, the Torpig controllers could have earned anywhere between $83,000 and $8.3 million.

We also wanted to determine the rate at which the botnet produces new financial information for its controllers—a botnet that generates all of its value in a few days and only recycles stale information afterward is less valuable than one that steadily produces fresh data. Although we observed the botnet for only 10 days and therefore can't draw unequivocal conclusions, the Torpig bots did continuously steal and report new bank accounts and credit-card numbers during that period.

Password Analysis

A poll conducted by Sophos in March 2009 (http://www.sophos.com/pressoffice/news/articles/2009/03/password-security.html) found that a third of 676 Internet users surveyed neglect the importance of using strong passwords and admitted that they reused online authentication credentials across different Web services. It's reasonable to trust the results of a poll, but it's also important to cross-validate the results, as people might not always report accurately.

Our takeover of the Torpig botnet offered us the rare opportunity to obtain the necessary evidence to validate Sophos's results. The benefits of the credential analysis we performed are twofold. First, it makes it possible to rely on real collected data and not on user-provided information. Second, the amount of data we collected from the Torpig-infected machines was two orders of magnitude bigger than that used in the Sophos poll.

While we controlled the botnet, Torpig bots stole 297,962 unique credentials (username and password pairs), sent by 52,540 different Torpig-infected machines. For each infected host we retrieved all the unique username and password pairs it submitted and calculated the number of distinct Web services each credential went with. Our analysis found that almost 28 percent of the victims reused their credentials, for accessing 368,501 webpages. While this percentage is slightly lower than the Sophos results, it's 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 we discovered. To this end, we created a Unix-like password file to feed John the Ripper (http://www.openwall.com/john/), a popular password cracker tool. The cracker recovered approximately 56,000 passwords in less than 65 minutes by using permutation, substitution, and other simple replacement rules (the "single" mode). It recovered another 14,000 passwords in the next 10 minutes when it switched modes to use a large word list, for a total of more than 40 percent recovered in less than 75 minutes. Over the next 24 hours, it recovered an additional 30,000 passwords by brute force (the "incremental" mode).

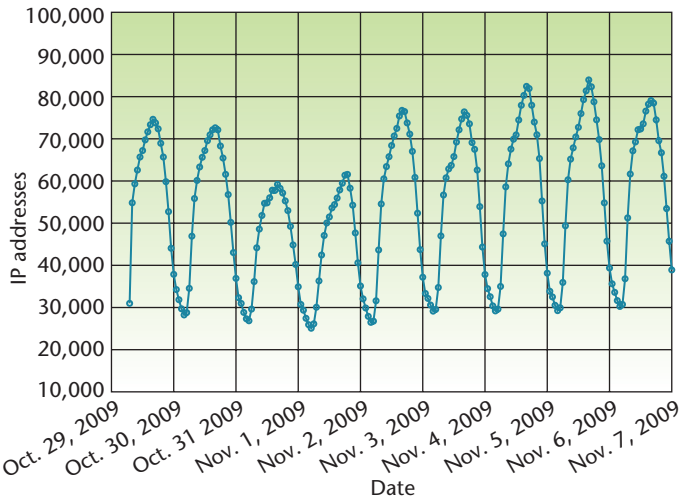

Figure 4. Unique IP addresses per hour 30 October 2009 through 7 November 2009. The Torpig botnet still poses a significant threat even after more than 4 years of scrutiny.

Aftermath

It's been more than a year since we took over the Torpig botnet.

Current Status

Over the past year, the Torpig botnet has changed in interesting ways, particularly as regards its data theft capabilities and the programs that it targets. Torpig has improved its man-in-the-browser functionality and can now dynamically rewrite the HTML content of specific webpages, using a modified version of the Sizzle JavaScript library, to hide fraudulent transactions when a victim logs into a compromised account. This enables a criminal to transfer money out of a bank account and extend the amount of time before the victim is likely to notice. It also increases the chances that a criminal can clear funds from an account before the financial institution can revert the transaction.

Current versions of Torpig also enable the Windows HelpAssistant account that's accessible via the Remote Desktop (RDP) protocol. This lets criminals control the machine remotely through the Windows graphical user interface, as if they were sitting in front of it. We suspect that Torpig has this functionality so that criminals can log in and manually dig through files and documents on a victim's machine to steal sensitive data. The miscreants appear to be interested in espionage in addition to financial theft.

Another significant modification is in the communication protocol, which occurred several months after our experiment. The new protocol is identical to that used by the Mebroot rootkit, and its encryption is similar to the Data Encryption Standard (DES). Therefore, it's harder for researchers and law enforcement to mimic a Torpig C&C server and to decrypt and repatriate stolen data. It's likewise difficult to determine if the operators of Torpig switched the encryption algorithm as a direct response to our work, or if it was simply part of their periodic software updates and would have occurred anyway. (Mebroot had already implemented this encryption before our work with Torpig,)

The Torpig botnet's size has been relatively stable since our sinkhole closed in February 2009. Figure 4 shows the number of IP addresses connecting to the C&C server domain names for 10 days in late October and early November of that same year, demonstrating that despite the attention that the Torpig botnet received from our work, it has remained virtually unscathed. Roughly 50,000 infected machines were online concurrently in November 2009, with a peak of 80,000 infected machines online at the same time.

Ethical and Legal Considerations

Our experiment in sinkholing the Torpig botnet is similar to other experiments that recently tried to shed light on the characteristics of the underground economy.[4] These types of studies are tremendously important for understanding and combating computer crime, but they have serious ethical implications, as they involve the computers of unsuspecting users and sometimes leverage and piggyback on infrastructures created by criminals. Because of the delicate nature of this research, researchers must adhere to strict ethical criteria. The computer security research community has seen a lively debate about these issues recently; for example, the NDSS symposium in February 2010 featured a panel titled "Ethics in Networking and Security Research."

We established two main ethical criteria for our underground economy research: no user should be worse off as a result of our activities, and our activities should be beneficial for the society at large. The first criterion is important because it puts the well-being of the cybercrime victims above researchers' desire to test a hypothesis or perform an experiment. It was why we decided not to send the victims of the Torpig botnet a command that would uninstall the malware: while appealing, this action might have had unpredictable repercussions on the user environment.

The second criterion states that the research should improve the public's security. Our experiment had three societal benefits. First, the financial institutions involved were able to identify and secure compro-

Related Work in Understanding the Botnet Phenomenon

Given the importance of the botnet problem, researchers have invested significant effort into gaining a better understanding of the phenomenon. One approach to studying botnets is to perform *passive analysis* of secondary effects of the activity of compromised machines. For example, researchers have collected spam mails that were likely sent by bots,[1] or they focus on DNS queries[2,3] or DNS blacklist queries[4] performed by bot-infected machines. A more active approach to studying botnets is via *infiltration,* whereby researchers join a botnet to perform analysis from the inside using an actual malware sample or a client simulating a bot. To achieve this, they use honeypots, honey clients, or spam traps to obtain a copy of a malware sample. They then execute the sample in a controlled environment, which makes it possible to observe the traffic exchanged between the bot and its command-and-control (C&C) servers.

References

1. L. Zhuang et al., "Characterizing Botnets from Email Spam Records," *Proc. 1st Usenix Workshop Large-Scale Exploits and Emergent Threats*, Usenix Assoc., 2008, http://www.usenix.org/events/leet08/tech/full_papers/zhuang/zhuang.pdf.
2. M.A. Rajab et al., "A Multifaceted Approach to Understanding the Botnet Phenomenon," *Proc. ACM Internet Measurement Conf.*, ACM Press, 2006, pp. 41–52.
3. M.A. Rajab et al., "My Botnet is Bigger than Yours (Maybe, Better than Yours): Why Size Estimates Remain Challenging," *Proc. 1st Usenix Workshop Hot Topics in Understanding Botnets*, Usenix Assoc., 2007, http://www.usenix.org/event/hotbots07/tech/full_papers/rajab/rajab.pdf.
4. A. Ramachandran, N. Feamster, and D. Dagon, "Revealing Botnet Membership Using DNSBL Counter-Intelligence," *Proc. 2nd Workshop on Steps to Reducing Unwanted Traffic on the Internet*, Usenix Assoc., 2006, pp. 49–54.

mised accounts. Second, we provided useful information to law enforcement about the inner workings of the Torpig/Mebroot infrastructure, which might help in eradicating this threat. Third, by sharing our experience with the research community, we believe that we improved the understanding of this type of malware infrastructure.

Like most computer scientists, we're neither ethics experts nor lawyers. However, our experience in carrying out these experiments, as well as discussions that we had with colleagues after reporting our results, made us aware of several issues that need attention. One is that of oversight: some organization, committee, or other body should be responsible for providing feedback on the ethical soundness of these experiments. One possible approach is for researchers to seek Institutional Review Board (IRB) approval from their home institutions before starting any experiments. We didn't get prior approval because we weren't planning to use human subjects in our experiments, but we did obtain the necessary approval after realizing that we were collecting personally identifiable information. One limitation of IRB approval agencies is that the members of the review board might not have the domain-specific knowledge necessary to fully understand the actual implications or side effects of an experiment.

Another oversight approach is to have the program committees (PCs) of computer security conferences determine what the ethical boundaries are. By accepting or rejecting papers and providing feedback to the authors, these committees can help shape what are "acceptable practices" in the security research community. Unfortunately, the use of PCs to establish and enforce ethical criteria also has disadvantages. First, the reviewing process almost never allows for an appeal in which scientists can explain in detail the ethical reasoning behind their choices. Second, PCs usually change substantially across conferences and even in different years for the same conference. Consequently, it's difficult to provide consistent feedback to paper authors.

As security research on the underground economy expands its focus and intertwines itself with the world of organized cybercrime, ethical issues will become even more relevant. We believe that the next few years will be key in shaping a new ethical stance for the computer security community, and we hope that studies like ours might be the starting point for a constructive discussion of ethical issues.

The legal aspects of performing network monitoring research are a complicated issue. Often, the various parties (criminals, victims, and researchers) reside in countries all over the world, and what might be legal in one country might be illegal in another. For our research, we followed several guidelines set by US federal law, including the Wiretap Act, the Pen Register and Trap and Trace Act, and the Patriot Act. One of the main points of all these laws governing packet sniffing (wiretaps) is the intent of those conducting the monitoring. In addition, 18 U.S.C. §2511 includes a provider protection clause stating that a system administrator can monitor a network to protect the service and its users. Courts have generally extended this protection to include network security researchers who have some active protection goal in mind.[5]

We also spoke with a lawyer, whose opinion was that our collection of information might not be considered a wiretap because users' systems volunteered the data, and we didn't intercept the information in transit to a different destination (that is, our server was the bot software's intended destination). The lawyer also thought that because our intent was to help the victims recover from the compromise of their computer, the collection of the information was legal.

Several lessons emerged from the analysis of the data we collected, as well as from the process of obtaining (and losing) the botnet. 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). In addition, 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. Finally, we learned that interacting with registrars, hosting facilities, victim institutions, and law enforcement is a rather complicated process. □

References

1. N. Provos and P. Mavrommatis, "All Your iFRAMEs Point to Us," *Proc. 17th Usenix Security Symp.*, Usenix Assoc., 2008, pp. 1–15.
2. T. Holz et al., "Measuring and Detecting Fast-Flux Service Networks," *Proc. 16th Network and Distributed System Security Symp.*, The Internet Soc., 2008; http://www.isoc.org/isoc/conferences/ndss/08/papers/16_measuring_and_detecting.pdf.
3. M.A. Rajab et al., "My Botnet is Bigger than Yours (Maybe, Better than Yours): Why Size Estimates Remain Challenging," *Proc. 1st Usenix Workshop on Hot Topics in Understanding Botnets*, Usenix Assoc., 2007; http://www.usenix.org/event/hotbots07/tech/full_papers/rajab/rajab.pdf.
4. C. Kanich et al., "Spamalytics: An Empirical Analysis of Spam Marketing Conversion," *Proc. 15th ACM Conf. Computer and Communications Security*, ACM Press, 2008, pp. 3-14.
5. P. Ohm, D. Sicker, and D. Grunwald, "Legal Issues Surrounding Monitoring During Network Research (Invited Paper)," *Proc. ACM Internet Measurement Conf.*, ACM Press, 2007, pp. 141-148.

Brett Stone-Gross *is a PhD candidate in computer science at the University of California, Santa Barbara, where he's a member of the Computer Security Lab. His research interests involve many facets of the underground Internet economy including botnets, click fraud, and fake antivirus campaigns. Stone-Gross is a member of the IEEE. Contact him at bstone@cs.ucsb.edu.*

Marco Cova *is a lecturer in the School of Computer Science at the University of Birmingham, UK. His research interests include Web security, vulnerability analysis, electronic voting security, and intrusion detection. Cova has a PhD in computer science from the University of California, Santa Barbara. He is a member of the IEEE and the IEEE Computer Society. Contact him at m.cova@cs.bham.ac.uk.*

Bob Gilbert *is a PhD candidate in computer science at the University of California, Santa Barbara, where he's a member of the Computer Security Lab. His research interests include malware analysis and malware defense. Gilbert is a member of the IEEE. Contact him at rgilbert@cs.ucsb.edu.*

Richard A. Kemmerer *is the Computer Science Leadership Professor and a past department chair of the Department of Computer Science at the University of California, Santa Barbara. His research interests include formal specification and verification of systems, computer system security and reliability, programming and specification language design, and software engineering. Kemmerer has a PhD in computer science from the University of California, Los Angeles. He is a fellow of the IEEE Computer Society and of the ACM, and is a member of the IFIP Working Group 11.3 on Database Security and of the International Association for Cryptologic Research. Contact him at kemm@cs.ucsb.edu.*

Christopher Kruegel *is an associate professor and holder of the Eugene Aas Chair in Computer Science at the University of California, Santa Barbara. He's also involved in the International Secure Systems Lab. His research interests include most aspects of computer security, with an emphasis on malware analysis, Web security, network security, and vulnerability analysis. Contact him at chris@cs.ucsb.edu.*

Giovanni Vigna *is a professor in the Department of Computer Science at the University of California, Santa Barbara. His current research interests include malware analysis, Web security, vulnerability assessment, and intrusion detection. He has been the program chair of the International Symposium on Recent Advances in Intrusion Detection (RAID 2003), of the ISOC Symposium on Network and Distributed Systems Security (NDSS 2009), and of the IEEE Symposium on Security and Privacy (S&P 2010 and 2011). Vigna has a PhD from Politecnico di Milano, Italy. He is a member of the IEEE and the ACM. Contact him at vigna@cs.ucsb.edu.*