What's Clicking What? Techniques and Innovations of Today's Clickbots*

Brad Miller[1], Paul Pearce[1], Chris Grier[1,2], Christian Kreibich[2], and Vern Paxson[1,2]

[1] Computer Science Division
University of California Berkeley
{bmiller1,pearce,grier}@cs.berkeley.edu
[2] International Computer Science Institute
{christian,vern}@icir.org

Abstract. With the widespread adoption of Internet advertising, fraud has become a systemic problem. While the existence of *clickbots*—malware specialized for conducting *click-fraud*—has been known for a number of years, the actual functioning of these programs has seen little study. We examine the operation and underlying economic models of two families of modern clickbots, "Fiesta" and "7cy." By operating the malware specimens in a controlled environment we reverse-engineered the protocols used to direct the clickbots in their activities. We then devised a *milker* program that mimics clickbots requesting instructions, enabling us to extract over 360,000 click-fraud directives from the clickbots' control servers. We report on the functioning of the clickbots, the steps they employ to evade detection, variations in how their masters operate them depending on their geographic locality, and the differing economic models underlying their activity.

1 Introduction

Online advertising forms a vital part of the modern Internet economy. Millions of websites profit from an ecosystem of advertising networks and syndication chains. This widespread adoption of internet advertising has given rise to systematic fraud. The percentage of fraudulent ad clicks, called *click-fraud*, has steadily increased over recent years. Recent estimates suggest the fraud-rate is as high as 22% [8].

In the predominant form of click-fraud, a malicious party sets up a website filled with ads and deceives an advertising network into registering ad clicks, earning revenue for each click[1]. *Clickbots*, malware which automatically click on ads, can produce this fraudulent traffic. A challenge for clickbot operators is producing traffic in such a way that advertising agencies do not detect it as non-human or fraudulent.

In this study, we present an analysis of clickbot techniques and the associated infrastructure that supports click-fraud. We obtained samples of two clickbot families, which we named "Fiesta" and "7cy," in order to study the operation of clickbots. We executed the binaries in a controlled environment to prevent harmful side effects, such as actually participating in click-fraud. By monitoring the controlled execution of the bots, we

* Student co-leads listed alphabetically.

[1] In a second form of click-fraud, a malicious party deliberately focuses clicks on a competitors advertisements in an attempt to exhaust that party's advertising budget [18].

T. Holz and H. Bos. (Eds.): DMIVA 2011, LNCS 6739, pp. 164–183, 2011.

reverse-engineered their command and control (C&C) protocols to determine how the bots respond to the commands they receive. This analysis enabled us to develop C&C servers and websites for the bots to interact with, allowing greater exploration of bot behaviors. We then devised a *milker* program that mimics a clickbot requesting instructions, enabling us to extract 366,945 click-fraud directives from the clickbots' control servers.

Throughout our analysis, we compare both families of clickbots to Clickbot.A [9] in order to illuminate how clickbots have evolved in recent years. We find two major innovations. The first regards the underlying economic model used by the Fiesta family. In this model a middleman has emerged, acting as a layer of abstraction between ad syndicates and the clickbots that generate traffic. This middleman provides an intermediary between the traffic originator (bots and users) and the ad syndicates. Fiesta clickbots generate traffic that flows towards this middleman and is then laundered through a series of ad syndicates in an effort to hinder fraud detection.

The second innovation concerns complex behavior that attempts to emulate how humans browse the web. 7cy mimics a human browsing the web by both randomizing the click targets as well as introducing human-scale jitter to the time between clicks. Through the use of our milker we also discover that 7cy control servers distribute fraud directives that vary by the geographic region of the bot. Thus, while Fiesta generally uses indirection to accomplish click-fraud, 7cy uses random clicks, timing, and location-specific behavior to ostensibly present more realistic browsing behavior.

In § 2 we survey the related work in the field. § 3 describes our methodology for executing bots in a safe environment. § 4 discusses the Fiesta clickbot in depth, and § 5 looks at 7cy. We discuss potential defenses and then conclude in § 6.

2 Related Work

While there is a well-developed body of literature describing both botnets and click-fraud, there has been little work directly studying clickbots themselves. We discuss existing work dealing with clickbots, as well as tangential work describing the various aspects of click-fraud and botnets.

Clickbots: The only academic work analyzing the functionality of a botnet performing click-fraud focused on a bot named Clickbot.A [9]. Clickbot.A conducted low-noise click-fraud through the use of syndicated search engines. Daswani et al. employed a combination of execution, reverse-engineering, and server source code analysis to determine how Clickbot.A performed fraud. The clickbot used compromised web servers for HTTP-based C&C, and restricted the number of clicks performed for each bot, presumably to limit exposure to the ad agency. In addition to describing the botnet behavior, the investigators estimate the total fraud against Google using economic aspects of syndicate search and click pricing. Our work analyzes multiple clickbot specimens to understand the changes in both the economic model and bot operation in two modern clickbots. We expect that criminals are constantly improving bot technology in order to remain competitive against ad agencies that improve their fraud detection. Throughout this paper we use Clickbot.A as a reference for comparison.

Detecting Automated Search: Researchers have dedicated considerable effort to methods for differentiating search queries from automated and human sources. Yu et al. observe details of bot behavior in aggregate, using the characteristics of the queries to identify bots [23]. Buehrer et al. focus on bot-generated traffic and click-through designed to influence page-rank [3]. Kang et al. propose a learning approach to identify automated searches [15]. These efforts do not examine bot binaries or C&C structure, focusing instead on techniques for the search engine to identify automated traffic.

Defending Against Click-Fraud: Both academia and industry have explored click-fraud defenses. Tuzhilin studied Google's click-fraud countermeasures in response to a lawsuit filed against Google over potentially fraudulent clicks, and concluded that the countermeasures are reasonable [20]. Separately, academia has proposed purely technical solutions. Not-A-Bot (NAB) [11] combats bot activity through detection mechanisms at the client. In the NAB system the client machine has a trusted component that monitors keyboard and mouse input to attest to the legitimacy of individual requests to remote parties. In addition to NAB, Juels et al. likewise propose dealing with click-fraud by certifying some clicks as "premium" or "legitimate" using an attester instead of attempting to filter fraudulent clicks [14]. Kintana et al. created a system designed to penetrate click-fraud filters in order to discover detection vulnerabilities [16]. Our work complements click-fraud defenses by exploring the techniques clickbots use and has the potential to improve click-fraud detection.

Botnets: There is extensive work examining botnets and reverse-engineering bots and their C&C protocols [1, 5–7, 12, 13, 19]. Dispatcher is a technique that automatically reverse-engineers botnet C&C messages and was used to uncover details in the MegaD spamming botnet C&C protocol [5]. In a later work, Cho et al. used Dispatcher to conduct an extensive infiltration of the MegaD botnet, developing milkers to determine the C&C structure and mine data about the botnet's internal operations [7]. We employ a similar milking technique to explore the C&C structure of the clickbots under study.

3 Methodology

In this section we outline our environment and procedures for executing clickbots without risk of malicious side effects. We studied two "families" of clickbots, Fiesta and 7cy, within our experimental framework[2]. Since we obtained samples that did not have useful or consistent anti-virus labels we took the names Fiesta and 7cy from domain names the bots visited while performing click-fraud.

We obtained the Fiesta and 7cy samples by infiltrating several malware Pay-Per-Install (PPI) services as part of a larger study on malware distribution [4]. PPI services use varied means to compromise machines and then distribute malware to the compromised hosts in exchange for payment on the underground market [21]. We used behavioral characteristics to group multiple harvested binaries representing different

[2] The MD5 hashes of the Fiesta specimens are c9ad0880ad1db1eead7b9b08923471d6 and 5bae55ed0eb72a01d0f3a31901ff3b24. The hashes of the 7cy specimens are 7a1846f88c3fba1a2b2a8794f2fac047, b25d0683a10a5fb684398ef09ad5553d, 36ca7b37bb6423acc446d0bf07224696, and 782538deca0acd550aac8bc97ee28a79.

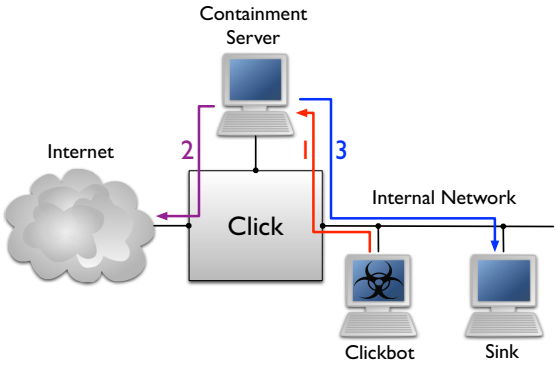


Fig. 1. Our basic containment architecture, showing how a containment server can interact with an infected VM and a "sink" server. The clickbot's communication is routed through the containment server (1), which can allow the traffic (perhaps modified) out to the Internet (2), or redirect it back into the farm to the sink (3).

versions of a given family of malware. We selected Fiesta and 7cy for further analysis because their connection behavior and content was the most interesting. A third potential clickbot family remains unanalyzed.

We executed the clickbots in virtual machines hosted on VMware ESX servers. A central gateway, implemented using Click [17], moderates network access for the VMs. The gateway routes each outbound connection to a "containment server" that decides on a per-flow basis whether traffic is forwarded, dropped, rewritten, or reflected back into the contained network. The containment server makes these decisions based on packet header information as well as packet content. Figure 1 shows a simplified view of this approach.

Given this architecture, we implemented containment policies that allowed us to execute the clickbot specimens safely. These policies required understanding the basic behavioral patterns of the bots and the other parties involved. This was an iterative process in which we repeatedly examined the bot behavior connection by connection, starting from a default-deny policy. Each step in this iterative process involved manually examining connections and packet payloads by hand to verify the nature of the communication. In some cases, this meant examining data logs, and in other cases it involved manually visiting websites. We white-listed connections deemed safe, and then restarted the bot in order to identify the next communication.

We needed the capability to replay pre-recorded communication and interact with the bot entirely within the farm in order to explore each clickbot's behavior and C&C protocol. Therefore, we built a special HTTP "sink" server that impersonated the destinations of outbound clickbot flows. This server allowed us to respond to network communication using a server under our control rather than releasing the traffic from the farm or dropping the flow. The sink server accepted all incoming connections and returned predefined responses as a function of the HTTP header data. Since the bot used HTTP for C&C as well as web browsing, we used the same HTTP server to simulate both C&C and web traffic. Initially, we replayed previously seen C&C responses. Then, we manually explored and perturbed the plain-text C&C protocol and fed these modified responses

back into the bots within the farm. Using this technique we reverse-engineered much of the protocol used for both bot families. As we understood more of the C&C protocols, we modified the responses to change the behavior of the bot. Using our capability to replay previously observed communications and explore new communication variants, we accelerated our analysis and developed a broader understanding of the clickbots' behavior.

4 The Fiesta Clickbot

We selected the Fiesta clickbot as the first specimen for in-depth analysis. The primary innovation we discovered during this evaluation is the monetization opportunity created by separating traffic generation (bots) from ad network revenue. In this model intermediate pay-per-click (PPC) services "launder" clicks generated by bots and then deliver them to ad networks. The intermediate PPC service abstracts the botmaster from advertising networks and is a new development since the investigation into Clickbot.A. We have not found any record of the security community studying the Fiesta clickbot[3]. In this section we describe Fiesta's behavior and structure, then discuss an economic model for click-fraud based on our observations. We conclude with a discussion of the bot's prevalence.

4.1 C&C Structure

There are two key players in the operation of Fiesta: a botmaster running a C&C server, and the self-described "Fiesta Pay-Per-Click (PPC) Profitable Traffic Solution." Fiesta PPC operates a store-front complete with signup and forum services. Although we named this clickbot Fiesta after the Fiesta PPC service, we believe the PPC service and the botmaster to be separate entities with different economic incentives. This relationship is discussed further in § 4.2.

Immediately upon infection the Fiesta bot establishes an HTTP connection with the bot's C&C server. The server's IP address is statically coded into the binary and remains constant for the lifetime of the bot. Using this server, the bot performs a one-time connection that informs the C&C server that a new infection has occurred. After this initial connection the bot settles into a constant cycle of click-fraud. Figure 2 gives a high-level overview of Fiesta's behavior for a single click. One click constitutes one act of click-fraud and involves multiple queries to the C&C server and multiple interactions with web pages. We observed our Fiesta bot performing about three such clicks per minute.

A fraudulent click begins with Fiesta requesting a list of search query terms from its C&C server, shown as step 1 in Figure 2. In response the bot receives several hundred varied terms; Table 1 shows some samples. We observed these terms changing frequently, appearing arbitrary in nature, and containing typographical errors. Once the bot receives this query list, it randomly selects one term that it will use for the remainder of this click.

[3] One variant observed was classified by an Anti-Virus vendor as the parasitic Murofet trojan [2]. We believe this to be a case of mistaken identity resulting from a standard Fiesta binary becoming infected and playing host to an instance of Murofet.

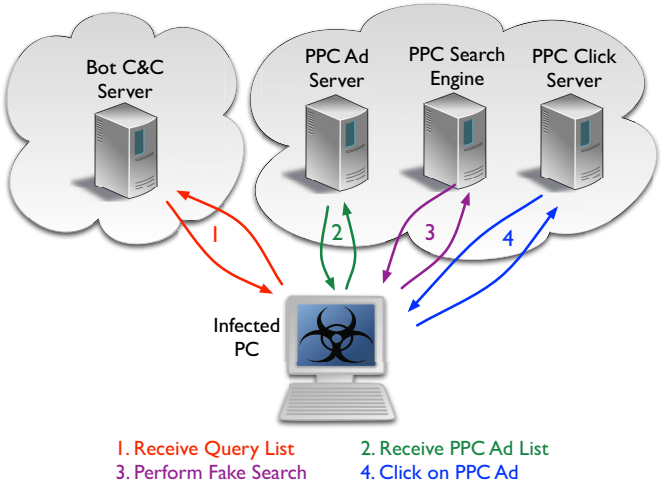


Fig. 2. The basic behavioral architecture of the Fiesta clickbot. Communication occurs in the order specified by the numeric labels. This pattern repeats indefinitely.

Table 1. A sample of search query terms returned to the bot by the Bot C&C Server during a single exchange

nerdy shirts	cruise special	fifa world cup qualifiers
potato canon	among the hidden solution	kitchen aid dishwashers
ftv plus	scooby doo online games	yahoo real estate
online video	card cheap credit machine	oakland newspaper
cheap insurance life uk	camera disposable pentax	tapes on self help
celtic names	debt and consolidation	bozeman schools. mt
justin om	dallas nursing institute	anniversary gifts by year
vxb bearings	discount hotel booking	station nightclub fire video

After the bot selects a search query, the bot begins communicating with the Fiesta PPC service, labeled step 2 in Figure 2. Initially the bot performs a request to receive ads that correspond to the selected search query. In response, the PPC Ad Server returns approximately 25 ads in XML format. Figure 3 shows an example of the PPC Ad Server XML response. Information contained in each record includes an ad URL, title, keywords, and "bid." The PPC Ad Server returns ads that vary greatly. Some ads directly relate to the search, while others appear random. The bot selects one of the ads at random from the PPC Ad Server response, biasing its selection towards high bid values. After selecting an ad, the bot performs a search for the original search query at a search engine operated by the PPC service. The bot then informs the search engine which ad it is about to click via a separate HTTP request (step 3 in Figure 2). Lastly, the bot will contact the PPC Click Server and actually perform the ad click (step 4 in Figure 2).

The PPC Ad Server returns ad URLs that point to the PPC Click Server. Each ad URL contains a unique 190-character identifier that is used when the bot issues an HTTP request to the PPC Click Server to signal a click. The PPC Click Server responds

```
<?xml version="1.0" encoding="UTF-8"?>
<records>
 <query>u2 tour</query>
 ...
 <record>
  <title>Looking for u2 tour?</title>
  <description>Find u2 tour here!</description>
  <url>http://u2-tour.com</url>
  <clickurl>http://CLICK_SERVER/click.php?c=UNIQUE_ID</clickurl>
  <bid>0.0004</bid>
  <fi>52</fi>
 </record>
   ...
 <record>
   <title>Style Fashion Show Review</title>
   <description>Chanel</description>
   <url>http://www.style.com</url>
   <clickurl>http://CLICK_SERVER/click.php?c=UNIQUE_ID</clickurl>
   <bid>0.0023</bid>
   <fi>39</fi>
 </record>
 ...
</records>
```

Fig. 3. Sample Fiesta ad C&C returned in response for a fake search for "u2 tour." The C&C syntax is abbreviated as necessary for space.

to all clicks with HTTP `302` redirect responses, beginning the fraudulent click. Figure 4 shows the process that occurs once the bot issues a request to the PPC Click Server, with arrows representing HTTP redirects. A single click will cause the bot to receive redirects to three or four different locations, eventually settling on a legitimate website such as `style.com` or `accuweather.com`. The resolution of this redirection chain completes a single act of click-fraud.

We believe the sites directly linked to the PPC Click Server along the redirection chains in Figure 4 are advertising sub-syndicates (i.e., entities that show ads generated by other ad networks in exchange for some portion of generated revenue) that have entered into syndication agreements with legitimate advertising services. The legitimate advertising services include *BidSystems* and *AdOn Network*. We believe some of the ad sub-syndicates are illegitimate based on other services hosted on the same IP addresses, as well as the frequency with which the sub-syndicate's IP addresses appear in malware reports.

Interestingly, the Fiesta bot issues requests to the Fiesta Ad Server with HTTP referrers from Fiesta search engines, yet performs searches *after* issuing ad requests. This implies that the PPC service could detect clicks occurring by this bot given the improper request ordering.

4.2 Fiesta Economic Model

Based on our observations of the Fiesta clickbot and our investigation of the Fiesta PPC service, we believe that we have identified the economic model of both the Fiesta PPC service and the Fiesta clickbot. This model introduces the notion of a click-fraud middleman whose job is to abstract ad revenue from the generation of fraudulent traffic.

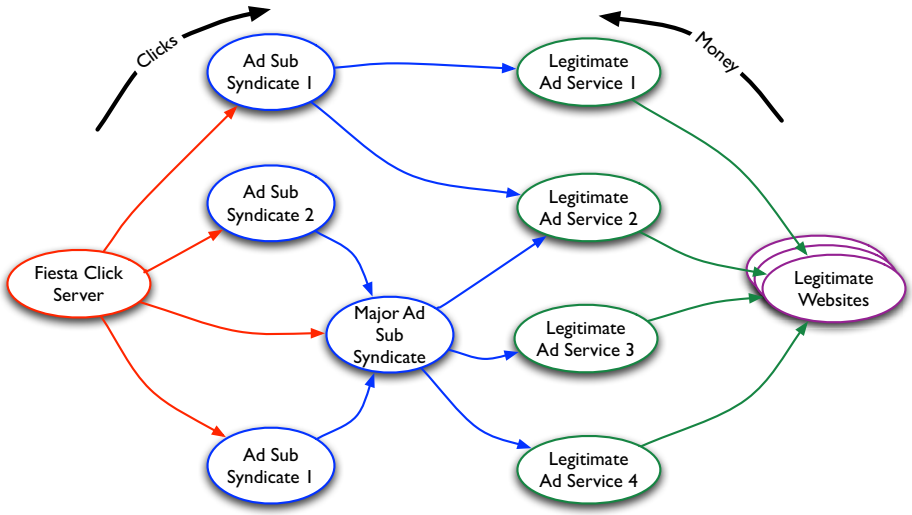


Fig. 4. The expanded Fiesta click redirection chain. This graph represents the possible redirection paths beginning with a click on the Fiesta click-server and ending at a legitimate website. One click will take one path through the graph. Redirections flow from left to right, and money flows from right to left.

This is a significant change in the economic structure of the click-fraud ecosystem that was not present for Clickbot.A.

We suspect that Fiesta PPC has entered into revenue sharing agreements with several advertising sub-syndicates. As part of this agreement, Fiesta PPC operates a search engine that displays the ad sub-syndicate's ads. Each of these ads is routed through the Fiesta Click Server. When an ad click occurs, the revenue is divided between the parties. The Fiesta PPC service then distributes the search engine traffic amongst their ad sub-syndicates through the use of the Fiesta PPC Click Server.

Investigation of the Fiesta PPC website supports these theories. The site contains links to traffic bid information based on region, web forums (which are currently not working), and an affiliate application form.

Inserting a middleman into the click-fraud path is an innovative structural development. A PPC service allows botmasters to generate revenue without regard to the specifics or defenses of advertising networks, while simultaneously allowing the middleman service to focus on ad revenue without engaging in traffic generation.

Concurrent with our own work, a separate study discovered a business relationship between the Fiesta PPC service and the operators of the Koobface botnet [22]. This study detailed the economics of Koobface, revealing that the botnet generated some of its revenue by interacting with various affiliate services in exchange for payment. Koobface utilized Fiesta PPC as an affiliate in addition to other PPC services. We believe the Fiesta bot operator has established a similar business relationship with the Fiesta PPC service.

4.3 Prevalence

We observed two different PPI services distributing Fiesta bots across a three month timespan. In one instance, a PPI service served the Fiesta bot binary for over 12 hours. Through creative use of the Google search engine combined with our own samples, we were able to locate four different C&C server IP addresses across three different domain names. Since the C&C server used by our bot was hard-coded into the binary and varied between dropper services, we suspect that Fiesta bots released via different mechanisms use different C&C servers. Using the same Google techniques, we have also identified 22 domain names that act as search engines for the Fiesta service, spread across three IP addresses.

5 The 7cy Clickbot

Clickbot.A, 7cy and Fiesta differ significantly in their behavior. While Fiesta and Clickbot.A use levels of indirection between organizations to launder clicks, 7cy attempts to evade detection by simulating human web-browsing behavior. The 7cy C&C language controls the bot's click behavior by specifying an initial site to "surf," a series of page content patterns for identifying desirable links to click on, and an inter-click delay time. The bot then leverages timing by introducing a random amount of jitter into the delay between clicks. Separately, the C&C directs the bot to generate more browsing traffic during popular times such as the evening and the workday. Compared to Fiesta, 7cy requires a substantially different C&C language and surrounding botmaster infrastructure, which we detail next.

5.1 C&C Structure

A 7cy bot locates the C&C server by resolving the domain name `in.7cy.net`, and then communicates with that server using HTTP. We show a sample C&C request in Figure 5. Line 1 includes the bot's network MAC address, presumably as a unique identifier. Line 3 presents a user-agent specific to the bot family, as well as a version number.

After receiving a request, the C&C server will respond with one of three messages: (*i*) an instruction to wait for a specified time period, (*ii*) an HTTP `302` response redirecting the bot to another C&C server, or (*iii*) specific click-fraud instructions. We refer to the latter as an instruction "batch." Each batch is comprised of "jobs," and each job is comprised of "clicks." Within a given batch, each job specifies a different site to target for click-fraud, and each click corresponds to an HTML link to visit. Jobs specify

```
1 GET /p6.asp?MAC=00-0C-29-24-29-12&Publicer=bigbuy HTTP/1.1
2 Host: in.7cy.net
3 User-Agent: ClickAdsByIE 0.7.4.3
4 Accept-Language: zh-cn,zh;q=0.5
5 Referer: http://in.7cy.net/p6.asp
6 Content-Type: application/x-www-form-urlencoded
7 Connection: Close
```

Fig. 5. Initial 7cy C&C request. The MAC-based bot ID and user-agent are shown in bold.

```
1 http://housetitleinsurance.com
2 http://www.google.com/url?sa=t&source=web&ct=res&cd=8&url=http://housetitleins...
3 90
4 15
5 CLICK
6 /search/{|||}epl={|||}yt={|||}qs
7 RND
8 5
9 NOSCRIPT
```

Fig. 6. Excerpt of response from C&C server. Note that whitespace between lines is removed.

web-surfing sessions: their clicks are applied to sites as they result from previous clicks, rather than to the initial site. On average there are 18 jobs per batch and 9 clicks per job.

Figure 6 shows an excerpt of one batch. Lines 1 through 4 constitute a header for a single job. Line 1 specifies the website at which the bot should begin the browsing session. Line 2 specifies the referrer to use in the initial request to the target site, although an actual request to the referring site is never made. Line 3 specifies a time limit after which new instructions will be requested if the clicks have not been completed yet. Line 4 specifies the number of clicks in the job. The structure seen in lines 5 through 9 describes a particular click. This structure (with possibly different values) repeats the number of times denoted on line 4 to complete the job. Line 5 specifies the action to perform. Several values other than `CLICK` have been observed, but seem to be ignored by our specimens. Given the rarity of these other commands (less than 0.01% of total commands), we suspect they are erroneous, or a new feature still in testing. Line 6 specifies token patterns to search for on the current page when selecting the next click. Tokens are delimited by the five characters "{|||}". Line 8 specifies a time delay for the bot to wait before performing the next click. Once all clicks in the job are specified, a new job header occurs or the C&C transmission ends.

After receiving a batch of instructions from the C&C server, a bot will begin traversing web pages as described in the instructions. Many of the sites targeted by the bot are hosted at parked domains. Examples include `housetitleinsurance.com`, `quickacting.com`, and `soprts.com`. We call these websites *publishers*. These sites mainly consist of links and ads within the page body, and keywords in the HTML `meta` tag which relate to the theme suggested by the domain name. They may also provide a link to contact the owner and purchase the domain name.

Although the domains and advertising networks vary across jobs, the traffic patterns follow progressions that we can aggregate into a graph, shown in Figure 7. Edges correspond to HTTP requests made by the bot. The origin of an edge corresponds to the domain of the referrer used in the request, and destination of an edge corresponds to the host to which the request is made. The first HTTP request a bot makes in a job is always for the URL of the publisher's website, using a referrer header mimicking a previous Google search (not actually performed) which could plausibly lead the user to the publisher's website (step 1). Next, the bot loads the publisher's website as a browser would, fetching all supporting objects (pictures, scripts, etc.) as dictated by the initial request (steps 2, 3). Once the bot has downloaded the publisher's webpage, it selects an in-page ad matching the search pattern specified via C&C for clicking. If multiple

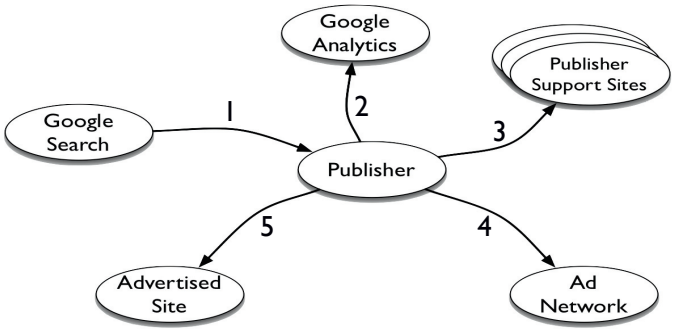


Fig. 7. General progression of a 7cy "job." Arrows represent HTTP requests and go from the domain of the refer to the domain of the host. Note that the publisher is often a parking page.

links on the page match the pattern, the bot makes a random selection. Each link on the webpage points to a "trampoline" page at the publisher's site, resulting in HTTP `302` redirects to the ad network and on to the actual advertised site. This behavior allows the publisher and the ad network to detect when the bot clicks on an ad. The bot follows this redirection chain (step 4) and loads the advertised site (step 5). A job often includes several clicks designed to simulate link-clicking behavior on the advertised site.

5.2 Specific Fraud Example

In order to demonstrate several details of the traffic produced by a 7cy bot, we now present excerpts of traffic from an actual job. In this job, the publisher is `housetitleinsurance.com`, the ad network is `msn.com`, and the advertised site is `insureme.com`. Figure 8 shows the bot's initial request to the publisher's website and the corresponding response. Note that as the bot is now issuing requests to published websites, the `User-Agent` presented in line 3 of the request has been changed to `Mozilla/4.0` rather than `ClickAdsByIE`.

In this instance, after the bot had loaded the publisher's site the bot clicked on a link to the publisher's own domain. This caused the bot to send another series of requests to the publisher as the corresponding site was loaded. Throughout this exchange, we observed that the publisher changed a portion of the cookie originally set on the bot in Figure 8 and began including a value similar to the cookie in links included on the page. This is seen in Figures 8-10 as the bold portion of the cookie changes.

The use of cookies and referrers represents an increase in complexity over the techniques used by Clickbot.A [9]. Clickbot.A strips the referrer from requests in order to obscure the site at which the traffic originated. While there are plausible reasons for removing the referrer in normal use cases, removing the referrer does introduce a notable characteristic into the traffic. Likewise, cookies help the traffic produced by 7cy to appear more natural and could potentially be used to construct the illusion of a user or browsing session for an ad network.

The bot will ultimately browse away from the publisher's website when a request is issued to the ad network in order to obtain a redirect to the actual website being advertised. This request is shown in Figure 9. Note that this request uses a `Referer`

```
GET / HTTP/1.0
Referer: http://www.google.com/url?sa=t&source=web&ct=res&cd=8&url?sa=t&source...
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: housetitleinsurance.com
```

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 13776
X-AspNet-Version: 4.0.30319
Set-Cookie: SessionID=6492595d-c592-419a-bf16-0cad97eef767; path=/
Set-Cookie: VisitorID=5f68a43f-6cf3-4a2f-831c-127ce007b646&Exp=11/29/2013 8:38...
Set-Cookie: yahooToken=qs=06oENya4ZG1YS6...HO8xG7uL E1uBAe 5qKwGUov0xhAWIvfCJZ1E...
```

Fig. 8. Selected headers from request (top) and response (bottom) of a publisher's webpage

```
GET /?ld=4vnjCbJ-GAvwzaNZFHBC2hWDhbZSs2HbnQAVmreNgXqjJdT0CGnrnZiVXS01aPdMH1DdL...
Referer: http://housetitleinsurance.com/online/find/home/owner/find/home/owner...
         ...yt=qs%3d06oENya4ZG1YS6...HO8xG7uLGV-ZMa5qKwGUov0xhAWIvfCJZ1EtVWLOl...
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: 948677.r.msn.com
```

```
HTTP/1.1 302 Object Moved
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Location: http://www.insureme.com/landing.aspx?Refby=614204&Type=home
Set-Cookie: MSConv=4vfcf6a1f935caa89943fce63a4bbf1574fc5c1f28c000945ebcd99d208...
Set-Cookie: MSAnalytics=4v76de0ef30bff74b972b5855ec3be14bc0c26342d22158a9ceaa6...
Content-Length: 202
```

Fig. 9. Selected request (top) and response (bottom) headers for an advertised site's URL

which includes a value similar to the `yahooToken` value previously set as a cookie on the bot by the publisher. The changed portion is shown in bold.

Lastly, a request is made to load the actual site being advertised. In this particular case the parking page has been moved within the same domain so a `301` redirect is issued. This is unrelated to the click-fraud infrastructure. As shown in Figure 10 this request also includes a referrer header which includes the `yahooToken` value used in other requests.

```
GET /landing.aspx?Refby=614204&Type=home HTTP/1.0
Referer: http://housetitleinsurance.com/online/find/home/owner/find/home/owner...
         ...yt=qs%3d06oENya4ZG1YS6...HO8xG7uLGV-ZMa5qKwGUov0xhAWIvfCJZ1EtVWLOl...
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.insureme.com
```

```
HTTP/1.1 301 Moved Permanently
Cache-Control: no-cache, no-store
Pragma: no-cache
Content-Length: 44447
Location: http://www.insureme.com/home-insurance-quotes.html
Set-Cookie: ASP.NET_SessionId=4u4vxv45vupmetvi2f4ghoqu; path=/; HttpOnly
```

Fig. 10. Selected request (top) and response (bottom) headers for a request for an advertised site

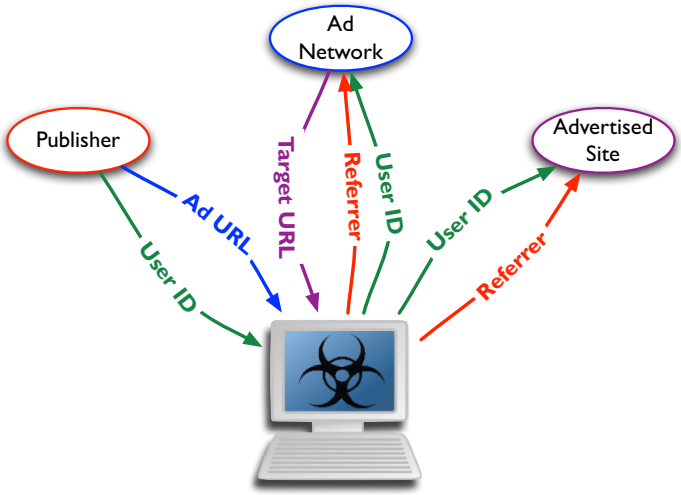


Fig. 11. Flow of critical information in 7cy click-fraud

We summarize the pattern of traffic described above in Figure 11. Each line corresponds to a piece of information and each bubble corresponds to a distinct server. The publisher's domain name is included in the referrer to both the ad network and the advertised site. The ad URL is supplied by the publisher and contains the domain of the ad network. The target URL is supplied by the ad network and contains the advertised site domain name. Lastly, the `yahooToken` likely containing a user ID is set by the publisher and given to the ad network and the advertised site.

5.3 7cy Economic Model

While the economic structure of 7cy is relatively clear, the parties involved and their roles are not. The pattern of traffic suggests that the advertised site (e.g., `insureme.com`) is paying the ad network (e.g., `msn.com`), which is paying the publisher (e.g., `housetitleinsurance.com`). Additionally, the domain names appear to be registered to multiple distinct parties. Unfortunately, it is unclear whether the publisher is paying the botmaster, or the publisher itself is the botmaster. If the publisher is paying the botmaster then it is unclear exactly how many distinct parties are acting as publishers.

5.4 Timing and Location Specific Behaviors

Timing Variance: In order to appear more human, the bot introduces jitter into the delays specified by the C&C language. The results labeled "Experiment" in Figure 12 show the differences we observed between the time delay specified in the C&C and the bot's actions in our contained environment. We conducted these measurements by feeding the bot artificial C&C with specific wait times while reflecting all HTTP traffic internally to a sink server within our farm. We then measured the inter-arrival time of requests at that sink. In order to confirm that the jitter observed was the result of the

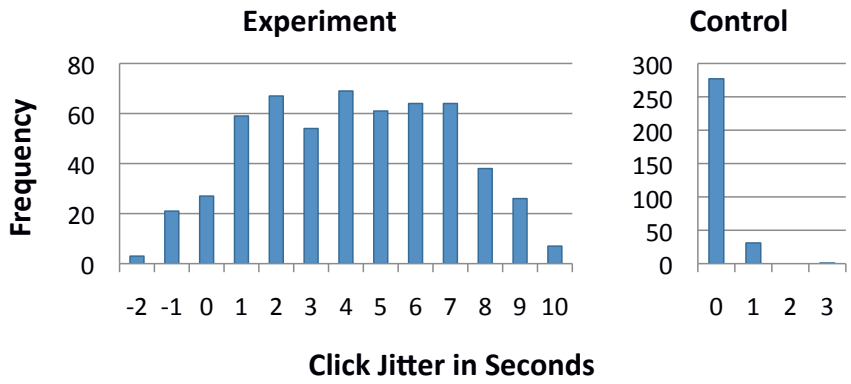


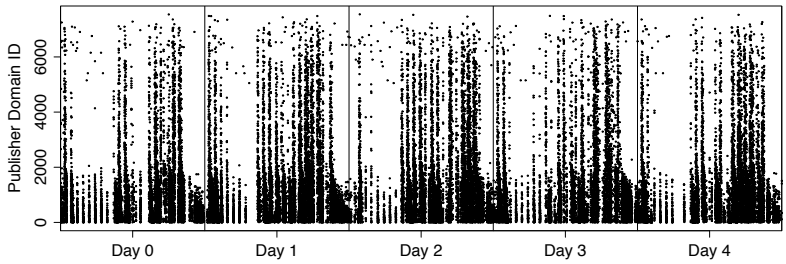
Fig. 12. Measurement of the amount of jitter introduced into inter-click delay by the clickbot binary and surrounding infrastructure compared to jitter introduced by infrastructure alone

bot's own behavior and not our honeyfarm environment, we also performed a control experiment in which we made HTTP requests at a constant rate from within an inmate VM, and then measured the variance in the same way as with the actual bot. The results labeled "Control" in Figure 12 indicate that the jitter introduced by the honeyfarm is infrequent, small, and always positive. Combined, these results show that the 7cy clickbot is introducing both positive and negative variance, on the order of seconds, into the inter-click delay.

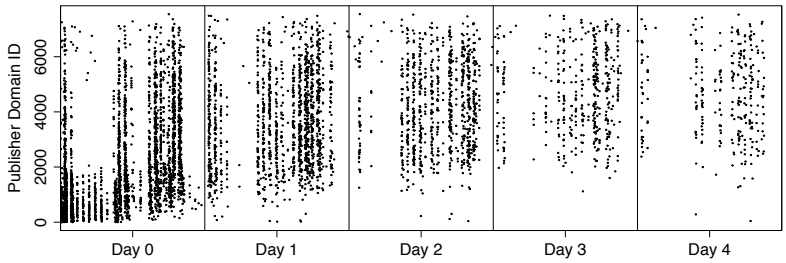
Instructions Over Time: In order to gather data about the characteristics of the C&C over time and the influence of bot location on the behavior of control servers, we also conducted a C&C milking study of the 7cy infrastructure. As part of our study we build a milker which connected to 7cy C&C servers via Tor [10] exit nodes in 9 countries throughout North America, Europe and Asia. These countries were Canada (CA), Spain (ES), France (FR), Hong Kong (HK), Japan (JP), South Korea (KR), Russia (RU), Singapore (SG), and the United States (US).

Our milker mimics the behavior of a 7cy bot by contacting the 7cy C&C server and requesting work. We record all network traffic in the exchange, but do not carry out any of the fraud specified by the C&C server. Our milker is implemented entirely in Python and is approximately 370 lines of code. Our study milked the C&C server continuously for five days starting Thursday, January 13 2011, at 5am GMT.

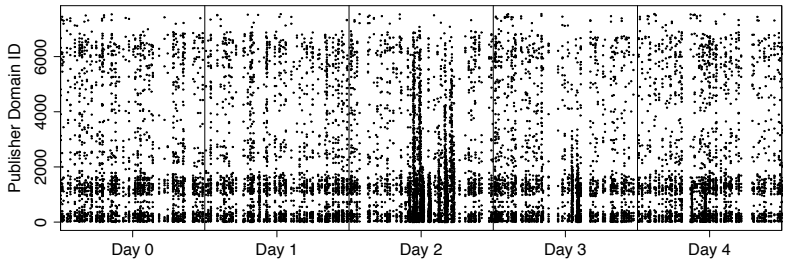
All initial C&C requests, regardless of the Tor exit node, were sent to `in.7cy.net`. This mimicked the behavior we observed in our actual specimens. Recall from Section 5.1 that the C&C server's responses can be classified as "Wait" (delay for a fixed time period), "Moved" (a `302` redirect to another server), or "Batch" (instructions for click-fraud). On occasion the C&C server returned a `400`-level error code, empty response, or no response, all of which we classify as "Other." When connecting from Japan, the C&C server occasionally returned a web page which seemed to be under development. We likewise classify this as "Other."



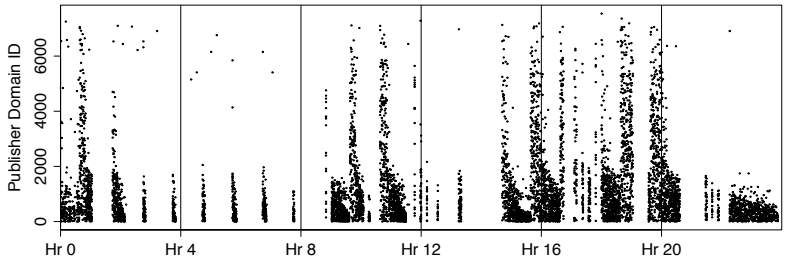
(a) Publisher domains seen from milking via the US over all 5 days of the study.



(b) Publisher domains seen from milking via the US over all 5 days of the study, with domains plotted only the first time they occur.



(c) Publisher domains seen from milking via Japan over all 5 days of the study.



(d) Publisher domains seen from milking via the US during the 1st 24 hours.

Fig. 13. The plots above show the domains the milker was directed to for click-fraud as a function of time. The vertical axis represents all domains observed, ordered from most frequently seen (bottom) to least frequently seen (top). Frequency was defined with respect to the entire study. Note that Day 0 begins at 5am GMT.

We observe that the C&C servers target some sites for click-fraud more often than others. The C&C samples obtained by our milker contained 366,945 jobs directing traffic towards 7,547 unique domains. An analysis of the traffic reveals that although 7,547 unique domains were seen across all countries, 75% of jobs targeted only 1,614 of the domains.

Figure 13 shows the domains targeted by the 7cy bots with respect to both country and time. The horizontal axis represents the time in seconds since the start of the milking study. The vertical axis is the sorted target domain ID, where domain 0 is the most targeted domain, and domain 7,546 is targeted least. Figure 13(a) plots the domains sent to our US Tor exit node milker over the 5 day period. The distinctive gaps in the data are the result of successive wait commands given to us by the C&C server. Figure 13(d) is an expanded view of the first day of Figure 13(a). We see that the server seems to work on an hourly cycle, dispensing a minimum amount of click-fraud instructions to the most trafficked sites each hour. The US exit node received more click-fraud instructions during peak US Internet use times, such as the work day and evening. In non-peak hours, the US exit node received more wait instructions to generate a lower volume of traffic. Interestingly, all exit nodes except Japan and Korea showed timing patterns in sync with the US despite time zone differences.

Japan and Korea, however, display a pattern similar to each other and distinct from other countries examined. Figure 13(c) shows domains served to the Japanese Tor exit nodes with respect to time, over the entire 5-day milking period. This figure shows the same distinctive bands, however the distances between bands and widths vary. While these countries do appear to have a strong, periodic schedule and do visit some sites considerably more than others, traffic appears to be distributed relatively uniformly throughout the day.

Figure 13(b) shows the same data as Figure 13(a) except all duplicate domains have been removed. This means once a specific domain has been observed at time t, it is no longer plotted for time greater than t. This figure illustrates that although there is a clear periodic pattern to the traffic, the domain names involved vary throughout the observed time. The other countries surveyed have similar behavior.

Beyond differences in timing, the distinct behavior seen in Japan and Korea is also characterized by differences in the C&C instructions received. Table 2 depicts the requests and responses made by the milker. Japan and Korea are redirected to `3.95622.com` relatively frequently, although no other countries are directed there. Similarly, Russia, Spain, Germany, Hong Kong, Canada and the United States are redirected to `1.95622.com`, although Japan and Korea are rarely directed to that domain. Only Singapore received no redirects.

In addition to differences in traffic handling and timing, milked C&C revealed a correlation in which domains were served to which countries. In order to determine the degree of correlation between two countries, we calculate for each country the percentage of overlap of its domains to the two countries' combined set of domains. In order to develop a standard of correlation, Table 3 shows the result of correlating a randomly selected half of a country's traffic with the remaining half. This provides a standard for determining that two countries are similar. Pairwise analysis of countries revealed that all countries other than Japan and Korea are strongly correlated. These results are

Table 2. Types of C&C responses received at geographically diverse locations

Country	Host	Total Req.	Wait	Batch	Moved to 1.95622.com	Moved to 3.95622.com	Other
CA	1.95622.com	193	24	167	0	0	2
	in.7cy.net	2,947	338	2,360	193	0	56
ES	1.95622.com	217	35	178	0	0	4
	in.7cy.net	2,935	327	2,333	216	0	59
FR	1.95622.com	215	18	192	0	0	5
	in.7cy.net	2,883	336	2,252	215	0	80
HK	1.95622.com	323	26	290	0	0	7
	in.7cy.net	2,253	438	1,465	323	0	27
JP	1.95622.com	10	0	10	0	0	0
	3.95622.com	777	378	396	0	0	3
	in.7cy.net	1,656	176	292	10	778	400
KR	1.95622.com	1	1	0	0	0	0
	3.95622.com	1,191	598	590	0	0	3
	in.7cy.net	1,286	22	37	1	1,193	33
RU	1.95622.com	139	14	121	0	0	4
	in.7cy.net	3,520	259	3,048	139	0	74
SG	in.7cy.net	4,238	160	4,000	0	0	78
US	1.95622.com	225	29	194	0	0	2
	in.7cy.net	3,022	322	2,425	225	0	50

presented in more detail in Table 4, where we see that Japan and Korea are somewhat correlated with each other and relatively uncorrelated with the rest of the world.

Although there is a strong correlation between domains served and country, the cause for this correlation is not clear as the domains served to Japan and Korea appear similar to domains served to all other countries. The domains which are more common in Japan and Korea do not appear to host content in either Japanese or Korean nor contain ads specific to Japan or Korea. The correlation between Japanese and Korean IP addresses and domains served may be related to the ad network being targeted by the bot, rather than the target audience for the content on the domain. We speculate that perhaps some ad networks are more tolerant of or prefer traffic from one country as opposed to others, and so it is more profitable to direct traffic from those countries to those ad networks. As the format of the ad URL is determined by the ad network, this viewpoint is supported by the fact that a similar correlation was seen in tokens to search for in URLs to click on.

The location and time-specific behaviors displayed by 7cy represent a notable departure from the methods of Clickbot.A [9]. 7cy displays time-sensitive behavior both in the randomness introduced to inter-click timings as well as the variations of traffic load

Table 3. Correlation within each country if visits are partitioned randomly in half. Correlation is measured as the percent of domains seen in both halves which were seen in either half.

	CA	ES	FR	HK	JP	KR	RU	SG	US
Internal Correlation	63.7	61.7	59.8	55.6	43.7	65.2	68.0	74.4	63.0

Table 4. Correlation of domain names served to various countries. Note that all countries except Japan and Korea are strongly correlated to each other as evidenced by their correlation to the US.

	CA	ES	FR	HK	JP	KR	RU	SG	US
US	79.5	79.6	78.0	72.5	32.0	12.1	81.0	83.3	100.0
JP	32.2	32.2	31.8	32.7	100.0	42.6	31.5	31.4	32.0
KR	12.4	12.3	12.1	12.3	42.6	100.0	12.0	12.1	12.1

with respect to time-of-day. The evidence of location-specific behavior also represents an added degree of complexity over Clickbot.A. These behaviors make bot-generated traffic appear more realistic and represent an advance in emulating human behavior.

6 Discussion and Conclusion

We have presented an in-depth analysis of two distinct families of clickbots: Fiesta and 7cy. From our analysis we have derived extensive behavioral information about these families. This allowed us to establish a profile of the capabilities of the bots as well as the economic motives and incentives of the parties involved. Utilizing these insights into bot behavior and the structure of click-fraud systems, we are now able to discuss potential techniques for defenses and safeguards against bot-generated traffic.

Through our study of the Fiesta clickbot we have described a click-fraud model in which a service provider acts as a middleman for fraudulent traffic. The middleman pays money for generated traffic, and generates his own revenue through agreements with ad sub-syndicates. This previously undescribed approach could allow advances in traffic generation to be abstracted away from advances in hiding fraudulent clicks, potentially driving click-fraud innovation.

Studying the 7cy clickbot allowed us to observe the specific mechanisms employed by a clickbot attempting to mimic the behavior of a human. The C&C protocol allows the botmaster to dictate or sell traffic at a fine granularity. We observed the attempt to simulate human-like behaviors, including random browsing of the advertised site and randomized inter-click delays. By milking 366,945 click-fraud instructions from the C&C servers via IP addresses in 9 countries we were able to study this botnet's click-fraud in detail and discover region-specific behavior.

Having described the behavior of both the Fiesta and 7cy clickbots, we would like to offer a brief discussion of potential techniques for detecting bot-generated traffic. One approach sites could employ is to develop a set of features characteristic of legitimate traffic and flag browsing sessions which appear abnormal according to these features. Features which would address the bots seen in this paper include the user's mouse movements on the page and the depth of browsing through the site, as these bots differ significantly in these regards from a typical human. Advertised sites may further this technique by correlating atypical features with the domain name of the referrer, and in doing so build a list of publisher domains in use by botmasters. Another conceivable detector is the invisible embedding of HTML links into a site's pages. Any visitor clicking such "honeylinks" is likely to be a bot. In addition to the above suggestions for potential detection techniques, we have pursued collaboration with industry to enhance bot detection.

Building upon this work, we plan to further develop evasion techniques, our understanding of clickbot size and population diversity, and build a more complete taxonomy of modern clickbots.

Acknowledgements

We would like to thank Chia Yuan Cho for his assistance with our efforts to locate bot C&C via Google searches, Eric Brewer for his direction in project development, and the anonymous reviewers of this paper for their helpful guidance.

This work was supported by the Office of Naval Research (ONR) under MURI Grant No. N000140911081, the National Science Foundation (NSF) under Grant No. 0433702 and 0905631, and the Team for Research in Ubiquitous Secure Technology (TRUST). TRUST receives support from NSF Grant No. 0424422. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of ONR, NSF, or TRUST.

References

1. Abu Rajab, M., Zarfoss, J., Monrose, F., Terzis, A.: A Multifaceted Approach to Understanding the Botnet Phenomenon. In: Proc. of SIGCOMM (2006)
2. Bodmer, S., Vandegrift, M.: Looking Back at Murofet, a ZeuSbot Variants Active History (November 2010) `http://blog.damballa.com/?p=1008`
3. Buehrer, G., Stokes, J.W., Chellapilla, K.: A Large-scale Study of Automated Web Search Traffic. In: Proc. of Workshop on Adversarial Information Retrieval on the Web (2008)
4. Caballero, J., Grier, C., Kreibich, C., Paxson, V.: Measuring Pay-per-Install: The Commoditization of Malware Distribution. In: Proc. of the USENIX Security (2011)
5. Caballero, J., Poosankam, P., Kreibich, C., Song, D.: Dispatcher: Enabling Active Botnet Infiltration using Automatic Protocol Reverse-Engineering. In: Proc. of ACM CCS (2009)
6. Chiang, K., Lloyd, L.: A Case Study of the Rustock Rootkit and Spam Bot. In: Proc. of the 1st Workshop on Hot Topics in Understanding Botnets, USENIX Association (2007)
7. Cho, C.Y., Caballero, J., Grier, C., Paxson, V., Song, D.: Insights from the Inside: A View of Botnet Management from Infiltration. In: Proc. of LEET (2010)
8. Click Fraud Rate Rises to 22.3 Percent in Q3 2010 (October 2010), `http://www.clickforensics.com/newsroom/press-releases/170-click-fraud-rate-rises-to-223-percent-in-q3-2010.html`
9. Daswani, N., Stoppelman, M.: The Anatomy of Clickbot.A. In: Proc. of the Workshop on Hot Topics in Understanding Botnets (2007)
10. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The Second-Generation Onion Router. In: Proc. of USENIX Security (2004)
11. Gummadi, R., Balakrishnan, H., Maniatis, P., Ratnasamy, S.: Not-a-Bot: Improving Service Availability in the Face of Botnet Attacks. In: Proc. of the 6th USENIX Symposium on Networked Systems Design and Implementation, pp. 307–320 (2009)
12. Holz, T., Steiner, M., Dahl, F., Biersack, E., Freiling, F.: Measurements and Mitigation of Peer-to-Peer-based Botnets: A Case Study on Storm Worm. In: Proc. of the LEET (2008)
13. John, J.P., Moshchuk, A., Gribble, S.D., Krishnamurthy, A.: Studying Spamming Botnets Using Botlab. In: Proc. of the USENIX NSDI (2009)
14. Juels, A., Stamm, S., Jakobsson, M.: Combating Click Fraud Via Premium Clicks. In: Proc. of the USENIX Security (2007)

15. Kang, H., Wang, K., Soukal, D., Behr, F., Zheng, Z.: Large-scale Bot Detection for Search Engines. In: Proc. of WWW (2010)
16. Kintana, C., Turner, D., Pan, J.Y., Metwally, A., Daswani, N., Chin, E., Bortz, A.: The Goals and Challenges of Click Fraud Penetration Testing Systems. In: Proc. of the Intl. Symposium on Software Reliability Engineering (2009)
17. Kohler, E., Morris, R., Chen, B., Jannotti, J., Kaashoek, M.F.: The Click Modular Router. ACM Transactions Computer Systems 18, 263–297 (2000), http://doi.acm.org/10.1145/354871.354874
18. Kshetri, N.: The Economics of Click Fraud. IEEE Security Privacy 8, 45–53 (2010)
19. Polychronakis, M., Mavrommatis, P., Provos, N.: Ghost turns Zombie: Exploring the Life Cycle of Web-based Malware. In: Proc. of LEET (2008)
20. Tuzhilin, A.: The Lane's Gift vs. Google Report (2006) ,http://googleblog.blogspot.com/pdf/Tuzhilin_Report.pdf
21. The Underground Economy of the Pay-Per-Install (PPI) Business (September 2009), http://www.secureworks.com/research/threats/ppi
22. Villeneuve, N.: Koobface: Inside a Crimeware Network (November 2010), http://www.infowar-monitor.net/reports/iwm-koobface.pdf
23. Yu, F., Xie, Y., Ke, Q.: SBotMiner: Large Scale Search Bot Detection. In: Proc. of the Intl. Conference on Web Search and Data Mining (2010)