BOTCLOUDS
The Future of Cloud-based Botnets?

Kassidy Clark, Martijn Warnier and Frances M. T. Brazier
Faculty of Technology, Policy and Management, Delft University of Technology, Delft, The Netherlands

Keywords: Botnet, Click fraud, Cloud, Cloud services, DDoS, Extrusion detection, Intrusion detection, Security.

Abstract: Many Cloud Service Providers (CSP) offer access to scalable, reliable computing resources following a pay-as-you-go model. Research into security of the Cloud focusses mainly on protecting legitimate users of Cloud services from attacks by external, malicious users. Little attention is given to prohibit malicious users from using the Cloud to launch attacks, such as those currently done by botnets. These attacks include launching a DDoS attack, sending spam and perpetrating click fraud. This paper discusses the threat of Cloud-based botnets, or *botclouds* and the need for new techniques to detect them. Two experiments show how simple and cheaply these attacks can be launched from botclouds.

1 INTRODUCTION

Cloud services refer to the provisioning of hardware and software resources across the Internet (Armbrust et al., 2010). Cloud Service Providers (CSP) typically offer both refined software services, such as databases and raw compute resources, such as storage or processing power. Customers often use these services following a pay-as-you-go model. Using Cloud services, companies can choose to, in effect, rent computer resources, rather than to invest in them outright, also providing elasticity of computing resources. For instance, if a cloud service customer discovers that he has over estimated his needs and has thus over-provisioned cloud services, he can scale down the size of his cloud. The reverse is also true, if a customer finds that he has under-provisioned cloud services, he can scale up the size of his cloud with little effort.

There are a growing number of CSPs including Microsoft, Google and Amazon Web Services (AWS). AWS is currently the largest[1]. AWS offers a web interface for human access, as well a scriptable, Java-based Application Programming Interface (API) for automated access.

Research into the security of the Cloud focuses primarily on protecting legitimate users of Cloud services from attacks by external, malicious users (Anthes, 2010; Maggi and Zanero, 2010; Chen et al., 2010). Little attention has been given to the possibility of malicious users using the Cloud to launch attacks, such as those currently performed by botnets. There is evidence that botnets are moving into the Cloud[2]. Due to their ease of use, high reliability and scalability, Cloud services provide malicious users a new platform with which to launch attacks, literally with the click of a button.

This paper discusses the possibility of attacks originating from Cloud-based botnets, or *botclouds*. The simplicity of these attacks is demonstrated in two experiments. Section 2 gives an overview of botnets, including attacks, structure and detection techniques. The concept of botclouds is presented in Section 3. Two botcloud attacks are demonstrated in Section 4. Finally, the implications of botclouds and botcloud detection are discussed in Section 5.

2 BOTNETS

A *bot* is a partially autonomous piece of software that can be controlled remotely. The person controlling a bot is referred to as a *botmaster*. A group of bots under control of a botmaster is referred to as a *botnet*. A botnet is created by first infecting a computer without the knowledge or consent of its owner by, for example, sending a virus as an email attachment (Ianelli

[1] Techno-Pulse. http://www.techno-pulse.com/2009/12/top-cloud-computing-service-providers.html. Accessed: November 2010.

[2] SecurityFocus. http://www.securityfocus.com/brief/1046. Accessed: November 2010.

Clark K., Warnier M. and M. T. Brazier F..
BOTCLOUDS - The Future of Cloud-based Botnets?.
DOI: 10.5220/0003381305970603
In *Proceedings of the 1st International Conference on Cloud Computing and Services Science* (CLOSER-2011), pages 597-603
ISBN: 978-989-8425-52-2

and Hackworth, 2005). Once a computer is infected with bot software, it contacts the botmaster. At this moment, the bot becomes part of the botnet. The botmaster can then send orders to the bot to carry out (malicious) tasks. Botnets can comprise thousands or even millions of bots, such as the BredoLab botnet estimated to have 30 million bots[3].

2.1 Botnet Attacks

Common attacks launched by botnets include (1) launching Distributed Denial of Service (DDoS) attacks, (2) sending spam email and spreading malware, (3) stealing private information and (4) performing click fraud (Jing et al., 2009; Ianelli and Hackworth, 2005).

DDoS attacks are used to overload a target's servers so that legitimate traffic can no longer access them (Mirkovic and Reiher, 2004). This is achieved by simultaneously flooding a target domain with requests until the response time to load a webpage is longer than a legitimate user is willing to wait. Thus, the system appears to have crashed. In some cases, just the threat of a DDoS attack is often enough for criminals to extort money from businesses.

Another common use of botnets is the sending of *spam and malware*. It is estimated that more than 97% of all email is unsolicited, bulk email (also known as spam) and the majority is generated by botnets (Anselmi et al., 2010). E-mail containing spam messages or malicious attachments can be sent from an infected machine using either a user's personal account or their Internet Service Provider's (ISP) e-mail server. If each single machine sends only 10 such messages per day, the massive size of most botnets can thus send millions of spam messages each day.

Private information is stolen from bot infected computers using keyloggers or making periodic screenshots (Ianelli and Hackworth, 2005). Keyloggers capture passwords and usernames at the moment that these are entered into a protected website, such as a personal email site. In addition, a bot can identify and copy sensitive financial data, such as credit card information.

Click fraud can be used to attack different targets for different reasons. A typical use of click fraud is to intentionally click on advertisements from pay-per-click providers, such as Google or Adsense (Kshetri, 2010). There are two general motivations for this kind of fraud: inflationary and competitive (Wilbur and Zhu, 2009). Inflationary click fraud is when attackers can earn money by clicking on advertisements they themselves are hosting. The pay-per-click providers then charge the targeted company for these clicks and pass this money on to those who actually host the advertisements, in this case the attackers.

Competitive click fraud is when attackers click the advertisements of a company with the goal of driving up that company's advertising costs. In this case, the attackers click on any advertisement of that company, regardless of where they are hosted and regardless of who receives money for the click.

Another use of click fraud is to artificially influence online polls. Online polls allow users to vote on various topics. Examples include news sites that poll how their readers feel about a certain news story or software developers that poll their users for feature requests. Often there is no monetary incentive to influence the results, but rather only a psychic benefit such as enjoyment.

2.2 Botnet Classification

Detecting and combating botnets remains a difficult task, but some progress has been made. Different botnet detection techniques correspond to different network structures (Dagon et al., 2007) and communication protocols (Jing et al., 2009). Based on these two attributes, two general classifications are recognized: Internet Relay Chat (IRC) based botnets and (newer) Peer to Peer (P2P) based botnets.

IRC is a text-based messaging protocol based on the traditional client/server model. An infected computer acts as an IRC client and contacts the IRC server to join the network. The botmaster then uses the IRC server to send instructions and data to the client for execution. This structure is highly centralized around the IRC server(s). If a server is taken offline, all bots that depend on it will no longer be able to receive or execute further instructions.

P2P based botnets use a more decentralized, distributed model without a centralized server. Each bot has a list of known peers comprising only a small part of the entire botnet. A botmaster sends data and instructions to one or more peers, that in turn pass these on to all other peers they know. These messages then traverse the entire network. As there is no central server, there is also no central point of failure, thus making this class of botnet more robust against attack.

Botnet creators have found that truly decentralized networks are relatively slow to organize and react to new commands, so most P2P based botnets employ a hybrid structure. Hybrid structures use a layer of super-peers that maintain long lists of known peers. When a new peer joins the network, it first contacts a

[3] InfoSecurity. http://www.infosecurity-magazine.com/view/13620/bredolab-downed-botnet-linked-with-spamitcom/. Accessed: November 2010.

super-peer and downloads this list. This download process and the role of these super-peers makes the hybrid structure more prone to detection by system users and administrators (Schoof and Koning, 2007).

2.3 Botnet Detection Techniques

The main methods for traditional botnet detection are honeypots and intrusion detection (Zeidanloo et al., 2010). *Honeypots* are unprotected computers that are intentionally (allowed to be) infected by botnets. Researchers observe the bots to learn about the rest of the network, including its behavior, structure and identities of its members. With this information, they can take countermeasures against the rest of the network.

Intrusion detection is the process of monitoring a host or network and analyzing the incoming network traffic. Traffic can be analyzed for known botnet activity or for general suspicious anomalies.

DNS tracking is a type of intrusion detection that analyzes DNS queries between bots and their server (Jing et al., 2009). As most bots must first contact a server (or super-peer) to join the network, researchers look for these messages to learn which computers are infected. For instance, if a particular IRC server is known, researchers can analyze network traffic for DNS queries to this server. This reveals which computers are infected. This also works in the opposite direction, if a computer is known to be infected (e.g. using a honeypot), researchers can watch for DNS queries to the server, thus learning where the server is located. Once learned, the address of the server can be added to public DNS blacklists (DNSBL). Queries to blacklisted addresses are blocked, thus preventing new from joining the botnet.

General traffic analysis is another type of intrusion detection that can be applied to both IRC and P2P based botnets. This technique analyzes messages being sent to and from a machine and compares the type, content, protocol or messaging pattern to a list of known botnets. For instance, certain botnets are known to either use certain ports, or send messages of a certain size, or at a certain interval, or to a certain type of host (Chandrashekar, 2009; Noh et al., 2009). To detect previously unknown botnets, researchers can look for anomalies or unusual network activity. For instance, if ‘typical’ network usage is known, then suspicious anomalies that differ from this safe baseline can be investigated.

3 BOTCLOUDS

Rather than use a network of infected machines, Botmasters can use Cloud services to build botnets. Botmasters purchase a large group of machines from a CSP and install a bot on each machine to form a botnet. Cloud-based botnets, or *botclouds*, have several advantages over traditional botnets. A traditional botnet requires substantial time to build, whereas a botcloud can be online in minutes. In addition, while a botnet is unreliable due to the constant threat of infected computers being switched off by their owners, a botcloud is always online and ready. Finally, a botnet cannot fully utilize the processor or bandwidth resources due to the constant threat of detection or computer use by the owner; however, a botcloud can be fully utilized with no fear of interruption.

3.1 Botcloud Attacks

A DDoS attack can be launched by a botcloud and can, at least temporarily, have the same effect as a DDoS attack launched by a traditional botnet. This attack can, in theory, be detected and neutralized by a CSP. This, however, will only work if they are monitoring for this type of activity. For instance, if a CSP is actively monitoring outgoing traffic for surges of HTTP requests that match a known pattern of DDoS activity, the host sending these packets can be identified and, as a consequence, disconnected or shut down. Therefore, a botcloud perpetrating this attack can be successful in shutting down the target domain, but perhaps only for a short period of time.

Sending spam from a botcloud can be (and has been[4]) accomplished. A common defense against spam is to blacklist the range of IP addresses from which the spam is being sent. However, blacklisting a large range of IP addresses of a CSP might block access to many legitimate services, such as customers that are hosting their email servers in the Cloud. This very problem was encountered when the spam blacklist service Spamhaus.org detected spammers in the Amazon Cloud and thus blocked a large block of IP addresses used by Amazon Cloud customers[5]4. This attack could also have been detected by a CSP, but only if it is actively monitoring for it. Therefore, a botcloud perpetrating this attack may be able to send massive amounts of spam, but only for a short period of time.

Click fraud can also be carried out by a botcloud. This attack is more difficult to detect than the first

[4] Washington Post. http://blog.washingtonpost.com/securityfix/2008/07/amazon_hey_spammers_get_off_my.html. Accessed: November 2010.

two. Most detection techniques work from the side of the pay-per-click provider to differentiate between clicks generated by humans and those generated by machines (Haddadi, 2010; Zhang and Guan, 2008). If a fraudulent click is detected, it is removed from the official tally, but no further action is taken. Unfortunately, pay-per-click providers do not have a strong incentive to stop click fraud. In fact, they have a strong economic incentive to allow it to occur as they have financial benefit from each click, regardless of its validity (Kshetri, 2010). Therefore, a botcloud perpetrating click fraud could remain in the Cloud for a long period of time without detection. Without sophisticated traffic analysis on the part of the CSP, this attack might never be detected or permanently neutralized.

3.2 Botcloud Detection

As discussed above, the main methods of botnet detection are honeypots and intrusion detection. Porting these methods to the Cloud is not a straightforward process. Deploying honeypots in the Cloud requires that a CSP monitors all activity on all or a subset of the machines used by CSP customers. As customers have paid for these machines, there may be (legal) objections to this privacy breach (Ruiter and Warnier, 2011). Furthermore, even if all system activity is monitored, it is non-trivial to differentiate legitimate from illegitimate activity.

Deploying intrusion detection software on each machine of the Cloud individually is similarly complex. Most intrusion detection algorithms are trained using a safe baseline of ‘normal’ incoming network activity. This baseline is compared to new patterns of network activity to determine if abnormal, and thus suspicious, activity is occurring. In the Cloud, this would require working closely with each individual customer to develop this safe baseline; an impractical task.

In addition to monitoring network activity entering the Cloud, network activity exiting the cloud can also be monitored for suspicious activity. This is referred to as outbound intrusion detection or *Extrusion Detection* and has been applied to the domain of Internet Service Providers detecting outgoing spam (Clayton, 2004). Proactively monitoring for suspicious outbound activity, such as DDoS or spam, will alert a CSP to the presence of malicious users and prompt further action, such as termination of those users. Click fraud remains the exception and requires close collaboration between pay-per-click service providers and CSPs. Without these minimal security measures, CSPs leave it to the victims of botcloud attacks to report the attack. This approach, while costing less time and money for the CSP will possibly lead to a bad reputation of that CSP.

4 EXPERIMENTATION

This section discusses two experiments to demonstrate the possibility of creating a botcloud and launching attacks. For both experiments, a leading CSP was used to lauch the attacks and a web server, owned by the authors of this paper, was used as the target machine.

4.1 DDoS Attack

The first experiment builds a botcloud to perform a DDoS attack on a low-end, personal web server. The server is a Dell PowerEdge 830 with a 2.6GHz processor and 2GB RAM. The server runs Debian Linux and the Apache Webserver with PHP and MySQL capability. The server hosts a test site consisting of a small ($<$ 5K), static HTML page and a dynamically generated PHP page that queries a MySQL database when executed.

A separate workstation is used to monitor the responsiveness of the server. This is done using a standard web browser and refreshing the test site periodically. Before the attack, the page refreshes in less than one second. In addition, the server is contacted every second using the *ping* command. Before the attack, the server consistently responds in less than 100 milliseconds.

A fully automated script creates a botcloud of 20 bots. Each bot is commanded to send 1000 simultaneous requests to the web server for a period of 60 seconds. When the script is executed, it creates a botcloud that sends 20,000 simultaneous page requests per second for a duration of one minute.

The essence of this script is shown in Figure 1. The script works as follows. First, a machine instance is initiated using the *run-instances* CSP command. This command can specify the disk image, number of machines and machine type (e.g. amount of memory or disk). An asymmetric key (cloudkey) is used to automate the login process. After the machine has started, the unique identifier (IDENT) is used to lookup the IP address (IPADDRESS) by executing the *describe-instances* CSP command. Once the IP address is known, a new process is spawned that logs into the newly created machine using the cloudkey. This process then executes the *httperf* script to flood the target with HTTP requests. Finally, the machine

```
# begin loop of 20 bots
for i in {1..20}
do
  # initiate a bot instance
  IDENT=‘run-instances disk_image1 -n 1 -g open -k cloudkey1 -t type1 | grep INSTANCE | cut -f 2‘

  # get ip address of running bot
  IPADDRESS=‘describe-instances $IDENT | grep INSTANCE | cut -f 4‘

  # spawn new process to lauch 1000 requests per second for 60 seconds
  ssh -o StrictHostKeyChecking=no -i ~/bin/cloudkey1.pem root@$IPADDRESS \
  'httperf --hog --www.example.com --uri /index.html --rate 1000 --num-conn 60000 --timeout 1' &

  # terminate bot
  terminate-instances $IDENT
done
```

Figure 1: Summary of DDoS botcloud script.

is shutdown using the *terminate-instances* CSP command.

This attack overloaded the server. After only 10 seconds, the web browser could no longer refresh the page. Server response to ping requests slowed down until approximately only one of 10 requests were answered while the others timed-out (after one second). This particular script targeted a plain HTML page, but could be modified to request a page that requires more server processing, such as a PHP page that queries a local database. This would require fewer requests to overload the server.

4.2 Click Fraud

The second experiment builds a botcloud to perform click fraud on an online poll hosted by a local server. The server hardware and software is identical to the description in the first experiment. To simulate standard click fraud detection techniques, the server executes tests before accepting a vote as legitimate. First, the web server checks for a unique IP address. Therefore, no single machine can vote twice from the same IP address. Secondly, to prevent the same machine from voting from multiple IP addresses (e.g. by switching networks), the server saves a cookie on the machine casting the vote. Before accepting a vote, the server tests for the presence of this cookie. If present, the vote is rejected.

The final fraud detection technique records a unique identifier for each vote cast. This unique identifier is created by adding a timestamp to each vote. These timestamps are logged for later analysis.

An automated script creates a botcloud of 1000 bots. Each bot is commanded to cast a single, fraudulent vote. When the script is executed, it creates a botcloud that casts 1000 fraudulent votes, each with a unique IP address. The essence of this script is shown in Figure 2.

This script works as follows. First, a bot instance is initiated using the *run-instances* CSP command. After the machine starts, the unique identifier (IDENT) is used to lookup the IP address (IPADDRESS) of the machine. Once the address is known, a new process is spawned to login to the bot using the cloudkey. This process then executes the *wget* command to execute the fraudulent vote. Finally, the bot is terminated using the *terminate-instances* CSP command.

To avoid timestamp anomaly analysis done by the web server, randomness is added to the script in two places. First, between loops, the script rests for a random length of time (e.g. 2 seconds, 5 minutes, etc.). This prevents the web server from detecting inexplicable surges of votes. Second, random padding is added to the timestamp of each vote request. This is done by choosing a random, three digit number and appending this to the vote request.

The attack completed successfully. The web server registered 1000 votes from unique machines with unique IP addresses. Furthermore, visual analysis of the voting log and its timestamps did not reveal any obvious anomalies, such as clusters of votes.

5 DISCUSSION

While these experiments may not accurately represent the effects of a large scale attack, they demonstrate the possibility of constructing such an attack. They illustrate the straightforward manner with which such attacks can be launched. Both of these attacks were constructed and executed in less than one day and for approximately 100 euros and both were successful in their respective goal. Furthermore, neither attack was detected or shutdown by the CSP.

The price of renting a botnet is difficult to estimate, as these activities take place on the obscured online black markets. However, some researchers at

```
# begin loop of 1000 bots
for i in {1..1000}
do
  # rest for random length of time (to avoid detection)
  REST=$RANDOM
  let ”REST %= $MAXREST”
  sleep $REST

  # initiate a bot instance
  IDENT=‘run-instances disk_image1 -n 1 -g open -k cloudkey1 -t type1 | grep INSTANCE | cut -f 2‘

  # get ip address of running bot
  IPADDRESS=‘describe-instances $IDENT | grep INSTANCE | cut -f 4‘

  # random padding (to avoid detection)
  PADDING=0
  while [ ”$PADDING” -le 100 ]
  do
    PADDING=$RANDOM
    let ”PADDING %= 1000”
  done
  DATE=‘date +%s‘$PADDING

  # spawn new process to cast fraudulent vote
  ssh -o StrictHostKeyChecking=no -i ~/bin/cloudkey1.pem root@$IPADDRESS \
  ’wget http://www.example.com/frontmodules/vote.class.php?id=786359&_=’$DATE &

  # terminate bot
  terminate-instances $IDENT
done
```

Figure 2: Summary of click fraud botcloud script.

VeriSign estimate this price between $9 an hour and $67 a day[6]. While this is cheaper than the equivalent botcloud, botnets are also less reliable than Cloud services, as discussed above. To make matters worse, criminals willing to launch botnet attacks are most likely also willing to commit identity theft. When creating an account for Cloud services, a false name and stolen credit card information is used[7], thus making the cost of the service a non-issue. With a dozen stolen credit cards, a criminal could launch a series of a dozen botclouds, possibly on different CSPs. When one Cloud is finally detected and shut down, the next is launched, and so on, resulting in an ongoing, massive attack.

The experiments presented above used a relatively small number of machines to launch the attacks, however the very nature of Cloud services allows these attacks to scale. The same scripts could be used to launch an attack using thousands or tens of thousands of machines across multiple CSPs.

As botnet detection techniques continue to improve and the price of Cloud services continues to drop, botmasters will move their activities to the Cloud. However, most current botnet detection methods will not easily port to the Cloud. The CSP is in the best position to detect botclouds and should be responsible for maintaining intrusion detection mechanisms to detect not only incoming attacks (as they do now) but also outgoing attacks. The CSP has complete control of the Cloud and thus complete control of any botclouds they detect. In general, the CSP is in a better position to detect attacks better than other actors. For instance, an extrusion detection system designed to identify click fraud could be deployed across all nodes under the control of a single CSP. To achieve the same coverage outside of the Cloud would require multiple Internet Service Providers (ISP) to implement and coordinate the same system.

As with the pay-per-click providers mentioned earlier, CSPs do not currently have a strong incentive to monitor all customers from the time they start using Cloud services. As long as the customer pays, they can use the service. Moreover, analyzing all outgoing traffic costs time and resources. As long as CPSs continue to take action to shut down malicious users only after the fact (e.g. after the attack has been carried out), Cloud based attacks will increase.

[6] VeriSign. http://www.verisign.co.uk/press/page_201004.html. Accessed: December 2010

[7] O’Reilly Community. http://broadcast.oreilly.com/2009/03/blame-the-credit-card-franchis.html. Accessed: November 2010

6 CONCLUSIONS

This paper discusses the ability to launch attacks from within the Cloud against external targets. Two experiments demonstrate the simplicity and low cost of launching such attacks. Porting traditional botnet detection techniques to the Cloud is not straightforward, thus new techniques are required. One possible technique is extrusion detection. This would require CSPs to monitor outbound traffic to detect and respond to suspicious activity. Current policy is to wait until the victims of attacks contact the responsible CSP at which point action is taken to disable the attack. Until CPSs implement a comprehensive botcloud detection and removal policy, botmasters will continue to move their malicious activities into the Cloud and botclouds will continue to grow.

Possible areas of future work include research into Cloud deployment of extrusion detection systems and designing incentives for CSPs to proactively monitor for botclouds.

ACKNOWLEDGEMENTS

This work is a result of support provided by the NLnet Foundation (http://www.nlnet.nl).

REFERENCES

Anselmi, D., Boscovich, R., et al. (2010). Security intelligence report. Technical Report Volume 9, Microsoft.

Anthes, G. (2010). Security in the cloud. *Communications of the ACM*, 53(11):16–18.

Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al. (2010). A view of cloud computing. *Communications of the ACM*, 53(4):50–58.

Chandrashekar, J. (2009). The Dark Cloud: Understanding and Defending Against Botnets and Stealthy Malware. *Intel® Technology Journal*, 13(2).

Chen, Y., Paxson, V., and Katz, R. (2010). What's New About Cloud Computing Security. Technical Report Report No. UCB/EECS-2010-5, University of California, Berkeley.

Clayton, R. (2004). Stopping spam by extrusion detection. In *First Conference on Email and Anti-Spam*.

Dagon, D., Gu, G., Lee, C., and Lee, W. (2007). A taxonomy of botnet structures. In *acsac*, pages 325–339. IEEE Computer Society.

Haddadi, H. (2010). Fighting online click-fraud using bluff ads. *ACM SIGCOMM Computer Communication Review*, 40(2):21–25.

Ianelli, N. and Hackworth, A. (2005). Botnets as a vehicle for online crime. *CERT Coordination Center*, pages 1–28.

Jing, L., Yang, X., Kaveh, G., Hongmei, D., and Jingyuan, Z. (2009). Botnet: Classification, attacks, detection, tracing, and preventive measures. *EURASIP journal on wireless communications and networking*.

Kshetri, N. (2010). The economics of click fraud. *IEEE Security and Privacy*, pages 45–53.

Maggi, F. and Zanero, S. (2010). Rethinking security in a cloudy world. Technical report, Dipartimento di Elettronica e Informazione, Politecnico di Milano.

Mirkovic, J. and Reiher, P. (2004). A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2):39–53.

Noh, S., Oh, J., Lee, J., Noh, B., and Jeong, H. (2009). Detecting P2P botnets using a multi-phased flow model. In *Third IEEE International Conference on Digital Society*, pages 247–253.

Ruiter, J. and Warnier, M. (2011). Privacy regulations for cloud computing, compliance and implementation in theory and practice. In Gutwirth, S., Poullet, Y., de Hert, P., and Leenes, R., editors, *Computers, Privacy and Data Protection: an Element of Choice*, chapter 17, pages 293–314. Springer.

Schoof, R. and Koning, R. (2007). Detecting peer-to-peer botnets. *University of Amsterdam*, http://www.science.uva.nl/ delaat/sne-2006-2007/p17/report.pdf.

Wilbur, K. and Zhu, Y. (2009). Click fraud. *Marketing Science*, 28(2):293–308.

Zeidanloo, H., Shooshtari, M., Amoli, P., Safari, M., and Zamani, M. (2010). A taxonomy of Botnet detection techniques. In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, volume 2, pages 158–162. IEEE.

Zhang, L. and Guan, Y. (2008). Detecting click fraud in pay-per-click streams of online advertising networks. In *The 28th International Conference on Distributed Computing Systems*, pages 77–84. IEEE.