International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 3 (2015) pp. 6987-7007

http://www.ripublication.com

Examining Zeus Botnet by Adopting Key Extraction and Malicious Traffic Detection Framework using DNS

S. Nagendra prabhu[1] and D. Shanthi[2]

[1]Dept of Computer Science, R.V.S School of Engineering & Technology
[2]Dept of Computer Science, P.S.N.A College of Engineering & Technology
Dindigul, India spnage123@gmail.com

Abstract

Malware is one of the bearing dangers to the Internet security as a rule, and to business transactions specifically. Malware recognition devices, approach & calculation still call for successful and productive results. The Botnet have inclined a sententious part of the Internet malware attacks. One of the most prominent botnet is Zeus, which its main equitable is to defraud banking accounts for financial profit, sending spam, launching denial-of-service attacks. This paper centralizes on Zeus botnet implementation and rectification which is one of the distinct, terrible, and widely diffused financial malware. The contributions of this paper are manifold: First, Complete defilement process of Zeus botnet is realized on computer networks and internet. Second, brief explanation of Botnet Command & Control. Further it is destined to proposed adopting key extraction and malicious traffic detection method using DNS that specifically targets Zeus malware. The proposed approach simultaneously monitors and analyzes the unconstitutional access and has the ability to detect Zeus bot in user's computers and identify the bot master address with improved efficiency than existing algorithm.

Keyword- Malware, botnet, Zeus, Key extraction, Domain name Server, Traffic detection

I. INTRODUCTION

The Internet is an articulation infrastructure that interconnects the universal community of end users and content servers. In the last few years, internet malware concurrence have enlarged into better-organized and most commonly sustain and easily accessible botnet. A botnet, or the army of bots (zombies), is comprised of more than thousands or tens of thousands of compromised computers. Although statistics show that the number of botnets is increasing [1], most Internet users are still unaware of what is going on and how serious the problem is. Many of these users'

computers are easily compromised by bot malware and then become members of botnets. Since bot malware usually does not affect regular uses of compromised computers, bot masters or bot herders can control these compromised computers remotely and ask them to carry out malicious activities, such as sending SPAMs, launching distributed denial of service (DDoS) attacks, and stealing personal private information.

It is used for a fabulously broad range of crimes – ranging from banking credential more profit center enterprise. E-mail spam, lunching denial-of-service attack, adware, and click fraud are few examples of these emerging trends. Botnet are the fundamental cause of this problem. The term bot is short for robot. Criminals distribute malicious software (also known as malware) that can turn computer into a bot also Known as a zombie. When this appears, the victims' computer can perform automated tasks over the Internet, without knowing of the computer user. Criminals typically use bots to infect large numbers of computers. These computers form a network, or a botnet. Criminals use botnet to send out spam email messages, spread viruses, attack computers and servers, and commit other kinds of crime and fraud. If a computer becomes part of a botnet, the victim's computer might slow down and might recklessly be helping criminals.

In June 2009 that Zeus was major threat that deserves a reverse engineering effort. In fact, this foresight was confirmed in July 2009 when a security publication from Damballa positioned Zeus as the number 1 botnet threat with 3.6 million infections in the US alone (about 19% of the installed base of PCs in the US [2]). It was also predicted that Zeus is guilty in 44% of the banking malware infections. Recently, Symantec Corporation referred to this crimeware toolkit as the "King of the Underground Crimeware Toolkits" [3]. The Zeus-based botnet [2] led the Top 10 magisterial 19% of botnet infections for the year. For the first quarter of 2010, the Zeus-based botnet endured, which attained top position in 2010. Top 10 largest botnet and still until now the top ten of botnet list. To tell about Zeus, the answer found in Damballa which is a US company that protects, detects and removes botnet for enterprise businesses. They have published the ten top botnet that they have encountered in 2009 [4].

Zeus functions

The main purpose of Zeus is to steal online credentials as specified by the hacker. Zeus has integrated an authoritative remote-control function into the botnet so that the assaulter can now "take complete domination of the person's PC. It performs the following actions: acquisition of user information, piracy of protected storage information, FTP passwords, robbing online credential information as specified by a Downloaded configuration file and processes it, Zeus Lower browser security settings by altering IE registry entries, do additional task as communicated by command & control server [4].

Related work

A. Detecting Botnets with Encrypted Command and Control Channels

Bot masters increasingly encrypt command-and-control (C&C) communication [5] to

evade existing intrusion detection systems. In C&C traffic analysis shows that at least ten prevalent malware families avoid well-known C&C carrier protocols, such as IRC and HTTP. Six of these families e.g., Zeus P2P, Pramro, Virut, and Sality do not exhibit any characteristic n-gram that could serve as payload-based signature in an IDS.

Given knowledge of the C&C encryption algorithms, the algorithm detect these evasive C&C protocols by decrypting any packet captured on the network. In order to test if the decryption results in messages that stem from malware, PROVEX, a system that automatically derives probabilistic vectorized signatures. PROVEX learns characteristic values for fields in the C&C protocol by evaluating byte probabilities in C&C input traces used for training. This way to identify the syntax of C&C messages without the need to manually specify C&C protocol semantics, purely based on network traffic. The evaluation shows that PROVEX can detect all studied malware families, most of which are not detectable with traditional means.

Despite its naive approach to decrypt all traffic, we show that PROVEX scales up to multiple Gbit/s line speed networks.

B. Detection of Zeus Botnet in Computers Networks and Internet

Huge spread use of internet with wide scale spread of E-commerce processes becomes a great motivation for the attackers to move their goals from fun to finical profits. Attackers tend to use botnets which is a group of computers managed by botmaster to perform malicious activities which are criminal jobs. One of the most popular botnet is Zeus, which its main objective is to steal banking accounts for finical profit, so it is called king of botnet[6].

C. Detection of Spam Hosts and Spam Bots Using Network Flow Traffic Modeling

An approach for detecting e-mail spam originating hosts, spam bots and their respective controllers based on network flow data and DNS metadata. Establishing SMTP traffic models of legitimate vs. spammer SMTP clients and then classifying unknown SMTP clients with respect to their current SMTP traffic distance from these models. An entropy-based traffic component extraction algorithm is then applied to traffic flows of hosts identified as e-mail spammers to determine whether their traffic profiles indicate that they are engaged in other exploits [8].

Spam hosts that are determined to be compromised are processed further to determine their command-and-control using a two-stage approach that involves the calculation of several flow-based metrics, such as distance to common control traffic models, periodicity, and recurrent behavior. DNS passive replication metadata are analyzed to provide additional evidence of abnormal use of DNS to access suspected controllers. For examples of detected controllers in large HTTP(S) botnets such as Cutwail, Ozdok and Zeus, using flow data collected from backbone network.

D. A Survey of Botnet and Botnet Detection

Among the various forms of malware, botnets are emerging as the most serious threat against cyber-security as they provide a distributed platform for several illegal

activities such as launching distributed denial of service attacks against critical targets, malware dissemination, phishing, and click fraud. The defining characteristic of botnets is the use of command and control channels through which they can be updated and directed [9].

Recently, botnet detection has been an interesting research topic related to cyber-threat and cyber-crime prevention. This paper is a survey of botnet and botnet detection. The survey clarifies botnet phenomenon and discusses botnet detection techniques.

This survey classifies botnet detection techniques into four classes: signature-based, anomaly-based, DNS-based, and mining-base. It summarizes botnet detection techniques in each class and provides a brief comparison of botnet detection techniques.

E. A fuzzy pattern-based filtering algorithm for botnet detection

Botnet has become a popular technique for deploying Internet crimes. Although signature-based bot detection techniques are accurate, they could be useless when bot variants are encountered. Therefore, behavior-based detection techniques become attractive due to their ability to detect bot variants and even unknown bots. The behavior is based botnet detection system based on fuzzy pattern recognition techniques.To identify bot-relevant domain names and IP addresses by inspecting network traces. If domain names and IP addresses used by botnets can be identified, the information can be further used to prevent protected hosts from becoming one member of a botnet[10].

To work with fuzzy pattern recognition techniques, the design several membership functions based on frequently observed bots' behavior including:

- Generate failed DNS queries
- Have similar DNS query intervals
- Generate failed network connections
- Have similar payload sizes for network connections.

Membership functions can be easily altered, removed, or added to enhance the capability of the proposed system. In addition, to improve the overall system performance, develop a traffic reduction algorithm to reduce the amount of network traffic required to be inspected by the proposed system. Performance evaluation results based on real traces show that the proposed system can reduce more than 70% input raw packet traces and achieve a high detection rate (about 95%) and a low false positive rates (0–3.08%). Furthermore, the proposed FPRF algorithm is resource-efficient and can identify inactive botnets to indicate potential vulnerable hosts [6].

F. On the Analysis of the Zeus Botnet Crimeware Toolkit

In presents reverse engineering results for the Zeus crimeware toolkit which is one of the recent and powerful crimeware tools that emerged in the Internet underground community to control botnets. Zeus has reportedly infected over 3.6 million computers in the United States. On analysis aims at uncovering the various obfuscation levels and shedding the light on the resulting code. Accordingly, explain

the bot building and installation/infection processes. In addition, detail a method to extract the encryption key from the malware binary and use that to decrypt the network communications and the botnet configuration information [11].

The reverse engineering insights, together with network traffic analysis, allow for a better understanding of the technologies and behaviors of such modern HTTP botnet crimeware toolkits and opens an opportunity to inject falsified information into the botnet communications which can be used to defame this crimeware toolkit.

Bot detection systems can be classified into two categories, i.e., signature-based and behavior-based systems. Although a signature-based system is accurate, it has the following drawbacks. First, signature-based systems is not possible to detect unknown bots. Second, a string signature is for a specific bot. When a bot has a variant, even it behaves similar, string signatures cannot work for it. Hence, the false negative rates may increase when new bots are developed. Third, as the number of bot variants increases, the false positive rates may increase as well. This is because an extremely large database containing all identified bots' signatures may accidentally match benign software. Finally, it is possible for a bot to bypass signature- based checks by using code obfuscation techniques. On the contrast, behavior-based systems try to identify bot activities by using observed particular bot behavior. If well tuned, behavior-based systems are able to perform similar to signature-based systems in terms of detection rates. In addition, a behavior-based system does not need to maintain a signature database to detect bots. Such a system can be much more lightweight than a signature-based system.

In this paper, propose a behavior-based system to detect malicious domain names and IP addresses used by Zeus botnets. The contribution of the paper is threefold. First, we propose an effective traffic reduction algorithm to reduce the amount of traffic that is required to be checked by a bot detection system[10].Second, propose a generic framework to detect zeus botnets based on Adopting key extraction method and malicious traffic detection using DNS. Evaluation results show that the proposed bot detection system has good detection rates.

Roadmap

This paper is composed as accompanies in Section 2 portray about the complete implementation of Zeus botnet, its components and configuration of Zeus bot. In Section 3 provided a detailed about botnet command & control, control panel installation and web page infusion. In Section 4, formally defines the main problem and sub-problems the proposed system is going to resolve and explains the core of the proposed system, introduce the methodology used for Zeus exposure and its elimination, providing an algorithm name called adopting key extraction and malicious traffic detection methods. While in Section 5, give some finishing up comments and future enhancement.

II. IMPLEMENTATION OF ZEUS BOTNET

A. Component of Zeus

The Zeus wrongdoing ware tool compartment is a situated of systems which have

been depicted to setup a botnet over a high-scaled arranged framework. Harshly, the Zeus botnet plans to make machines act as spying executors with the plan of getting fiscal success. The Zeus malware can log inputs that are entered by the client and additionally to catch and change information that are laid out into website pages. Stolen information can hold message addresses, passwords, web saving money accounts, charge card numbers, and transaction validation numbers.

The overall structure of the Zeus crime ware toolkit consists of five segments [6]:

1) A control panel which obliges a set of PHP scripts that is utilized to reviewer the botnet and gather the stolen data into MySQL database and afterward show it to the botmaster. It likewise permits the botmaster to screen, control, and oversee bots that are enrolled inside the botnet.
2) Configuration files that are used to contrive the botnet parameters. It incorporate two files: the configuration file (config.txt) that lists the basic information and the web injects file (webinjects.txt) that identifies the targeted websites and defines the content injection rules.
3) A generated encrypted configuration files config.bin, which dominance an encrypted version of the configuration parameters of the botnet.
4) A generated malware binary file bot.exe, which is considered as the bot binary file that infects the victims' machines.
5) A builder program that engender two files: the encrypted configuration file (config.bin) and the malware (actual bot) binary file bot.exe.

B. ***Configuration and Botnet Creation***

Xampplite program is used which empower us to make our computer as a server that is Zeus botnet used client-Server approach, and Xampplite provide an environment to run and execute (.php) files where it is the extension of the server of Zeus botnet. The first step in constructing a bot executable is to edit the configuration file. The configuration instructs the bot how to connect to the botnet, and it also contains information on what user data to gather and how to do so. The configuration file is in two parts, Static & Dynamic Configuration.

III. **BOTNET COMMAND AND CONTROL**

Approximately 5% of PCs around the world efficacy be infected right now by a malicious code, so at least, 50 million. This significant rate can be explained by many factors like the lack of application and system updates, the lack of users' cautiousness or the rising number of attacks' vectors (social network, mobile applications, spam, compromised websites. Those machines are a set of infected hosts which communicate with one or several Command and Control servers, also name C&C servers. There are many advantages of owning a botnet for cybercriminals, such as: Stealing banking information, sending spam massively, Performing Denial of Service attacks [4], [14].

A. ***Control Panel Installation***

The Control Panel is an open source PHP application that can be run on an IIS or

Apache web server. Some additional software, most of which is described in the documentation, is also required. A MySQL user with relevant permissions must also be set up. When the system is ready the Control Panel code can be transcribed into the web server directory. The install page can then be get hold of from a browser. If any errors are made when filling in this form, the user is given a helpful message. Once this form is completed the leftover of the setup is done automatically.

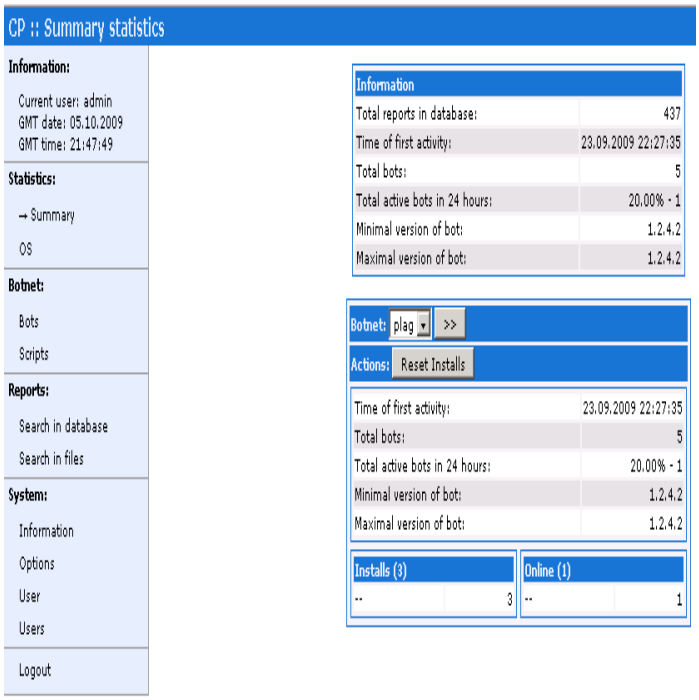

Fig. 1. Xampplite Control panel

The following figure shows total number of bots and provision to look their summary with the help of control panel [6],[9] .

B. ***Botnet Communication***

Information sent through the Zeus botnet is scrambled with Rc4 encryption. In this operation a key stream is produced from the botnet secret word, and is XOR with the information. The same watchword is utilized to encode all information that is passed through the botnet. Changing the botnet secret key obliges that the sum of the bot executables be redesigned to a raise that consolidates the new secret key. The dynamic config index additionally must be remodelled and the server secret word transformed from the Control Panel. The point when a computer is contaminated with the Zeus bot, its first correspondence with the server is a solicitation for the dynamic config file [14], [16].

Stolen data is regularly sent to the botnet's dropzone through two different communication channels. The first one, referred in the Zeus configuration file as log, consists in a small keep-alive message containing all the main status information about the zombie, i.e. botID, botnetID, IPaddress, bot OS, etc. Notably, this packet is

sent to the dropzone every 2 min by default, and consists in the most frequent communication type between the zombie and its C&C. Thus, its periodic emission could be leveraged by an anomaly based IDS to identify an infected computer inside a network. The second communication flow used by

Zeus, referred as report, occurs less frequently than the log one, i.e. by default every 10 min. As well as the zombie status information included in the log packet, it contains also all the data that has been stolen in the system. Hence, this message is usually bigger than the previous one, and packet fragmentation according to the network MTU size is often required by the OS for sending the entire TCP segment.

C. ***Web Page Injection***

One extensive feature of the Zeus bot is its ability to progressively inject dynamic into web pages viewed from an infected computer. This is done on-the-fly, as data passes from the server to the client browser. A snippet of the configuration data for this is shown below. It does a fairly straight forward examination and inserts operation [7], [9].

Sample coding for web page injection:

```
1:  set_url http://www.bank.com/login.html GP
2:  date_before
3:  name=”password”*</tr>
4:  data_end
5:  date_inject
6:  <tr><td>PIN;</td><td><input type =”text” name=”pinnumber” id=”pinnumber”/></td></tr>
7:  date_end
8:  date_after
9:  date_end
```

Instantly, Zeus targets these URLs to steal information and to modify the content of specific web pages before they get advertised on the user‘s screen[17]. The attacker can define rules that are used to yield a web form data. When a victim visits a targeted site, the bot steals the credentials that are entered by the victim. Thereafter, it posts the encrypted information to a drop location that is meant to store the bot update reports. This server decrypts the stolen information and stores it into a database [5].

IV. **ZEUS EXPOSURE**

The working scenario of a Zeus botnet can be classified into two phases. One is the infection phase and another is the attack phase. In the infection phase, a bot herder tries to expand the size of its army of bots. The bot herder commands existing bots to compromise more users’ computers. There are many techniques to compromise a computer such as exploiting software vulnerabilities and social engineering. Once a targeted host is compromised, remote controllable software, which is downloaded from a binary-download server, is installed and launched so that the information about

the compromised computer is reported to the bot herder. In the attack phase, a bot herder sends commands to compromised hosts, i.e., the bots. On receipt of the commands, each bot launches various tasks based on the instructions embedded in the commands. A bot herder is therefore able to ask bots to collect valuable information, report botnet status, and launch attacks to target hosts [18].

There are three stages in the algorithm in figure 2: traffic reduction, Adopting Key Extraction and Malicious Traffic Detection. First, input traffic is passed to the traffic reduction stage. Then, filtered packets are passed to the feature extraction stage. Finally, the fuzzy pattern recognition stage is used to detect malicious domain names and IP addresses based on extracted features.

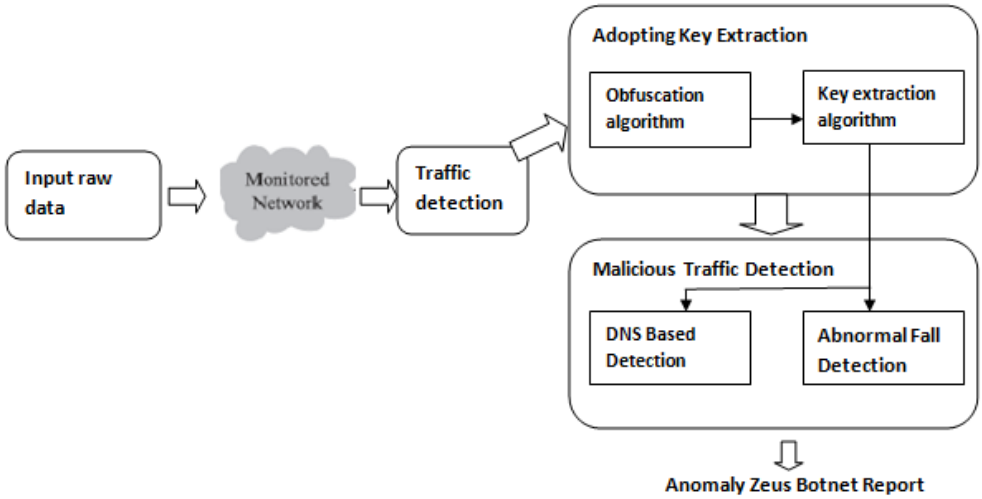

Fig. 2. Functional architecture of the ZEUS Botnet detection system.

A. Existing solution to Traffic Reduction

It is common that the traffic generated by bots are mixed with regular network traffic. To improve botnet detection efficiency, existing researches often reduce the amount of input traffic by filtering out bot-irrelevant traffic. Hence, a bot detection algorithm is able to concentrate only on bot traffic. A good traffic reduction algorithm may improve the overall system performance. However, if not well designed, it could increase false negative rates and / or false positive rates. Some common criteria used to filter out input traces are listed below [10]:

- ***Eliminate all port-scan activities***

Although port-scan is an essential step to compromise a remote host, it is not used when bots are communicating with each other. Therefore, it is possible to filter out port-scan activities without degrading the bot detection performance. Some port-scan packets have specific patterns. For example, a TCP port-scan packet with both the SYN flag and the RST flag are set.

- ***Ignore peer-to-peer traffic:***

If a detection system focuses only on IRC botnets, it often filters out P2P traffic and hence gets a significant traffic reduction rate. However, such a system cannot detect P2P bots. There are already a lot of systems to identify peer-to-peer traffic.

- ***Skip short lived flows:***

Filter out flows containing only a few packets or lasting only for a few seconds. These flows do not correspond to bots that are standing by at the ready [12].

- ***Filter out based on black lists and white lists:***

If the source or the destination address of a packet is well-known, it is often not necessary to check it. Hence, the packet can be safely ignored.

In Traffic reduction method It is common that input raw packet traces contain many different types of packets. Since most of them are not relevant to botnet detection, they should be filtered out. With an accurate and efficient traffic reduction algorithm, it enables a botnet detection system to run in a more efficient way. It is true that a good traffic reduction filter can reduce the data needed to be processed and hence increases the overall system performance. However, if a filter eliminates data improperly, bot detection rates could decrease. Therefore, criteria for traffic reduction must be carefully considered. In the proposed solution, we use only one intrinsic traffic reduction filter, as shown in Fig. 3. To prevent botnets from being detected, it is common for bots to dynamically retrieve the IP addresses of C&C servers. A bot herder is able to register several domain names and asks the bots to look up the IP addresses of these domain names. As a result, bots need to send DNS queries frequently to get the IP addresses currently being used by C&C servers. Since bots' activities often start with DNS queries, this characteristic can be used to filter out bot-irrelevant traffic. Based on this feature, we check DNS query and response packets and put returned IP addresses from the DNS into an IP address list. A packet is sent to the feature extraction stage if and only if its source or destination address is listed in the IP address list [10].

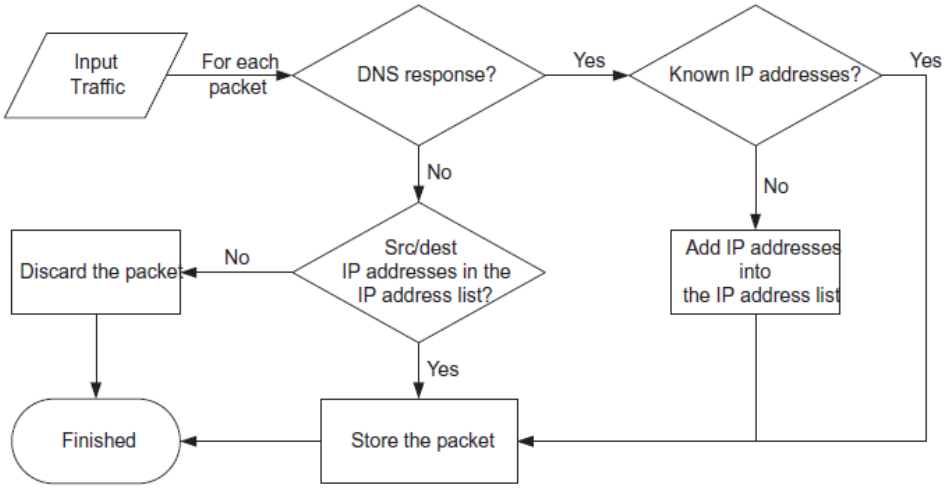

Fig. 3. The procedure of traffic reduction.

B. Adopting Key Extraction Methodology

One of the aims of this research was to obtain enough parts of γ needed to decipher the Zeus configuration file, which is usually around 69 KB [15]. By executing a Zbot sample in our sandbox, it was possible to build a packet containing chosen-plaintext

by inflating a modified cookie, i.e. here in after contrast-cookie, with ordinary strings. The contrast-cookie takes a relevant role because it allows us to enrich our chosen-plaintext to obtain a bigger known plain text, which implies being able to consecutively retrieve the desired size of γ. On top of that, we were able to extract the last m bytes of the derivate key γ and to decipher the configuration file without knowing neither the RC4 keystream K nor its seed. In addition, the proposed approach would also save the researchers from carrying out the static malware analysis to search either the keystream or the seed in the system volatile memory[4].

According to Obfuscation key Theorem [4],[19] the derivate key γ can be applied to any string Z obtained when ciphering a string Q formed by a set of n unknown values and m known ones with key K; however, in order to obtain γ, it is necessary to establish a correspondence between P_i and P'_i, which requires to know n in advance. A Zeus report contains two unknown substrings: the Zeus header and the Zeus trojan item 10002, i.e. the botnet-id. Although the length of the Zeus header is known, this is not the case for the Zeus botnet-id. Given that the mentioned field has a limited length, we tackle with this issue by performing a brute-force attack on the Zeus botnet-id item body length. According to the Zeus packet structure explained before, the length of every Zeus item body is declared in 4 bytes of its header. Thus, the maximum length of the field is 2^{32} bytes long. However, the Zeus Builder toolkit does not allow to create malwares containing a botnet-id value bigger than 20 characters. Therefore, we just need to make an attempt over 20 different positions as for where to insert our known environmental item values. Algorithm 1 is in charge of creating the set of all the possible obfuscated texts O′, depending on their different lengths. Note that in line 5 the insert function is used for adding the character ‘‘A’’ at the fixed position[3] x, and shifting all the next values by one byte in the string. The algorithm takes the chosen-plaintext as input, and returns a multimodal-array with all the related obfuscated texts[4].

Algorithm 1. Create Obfuscation Set algorithm

```
1.  function CREATEOBFSET (text) ⇨ The chosen plain text
2.    T  ←  text
3.    x  ←  112
4.    for i ← 0, 20 do
5.        insert(Tx, ‘‘A’’)
6.    Oi′ ←  visualEncrypt(T)
7.    i  ←  i ++
8.  end for
9.  return O′ ⇨  The obfuscated set
10. end function
```

Once we generate our set of O′, we can use it for retrieving the derivate key as explained before. The key extraction pseudo code is expressed in Algorithm 2.

Algorithm 2. Key extraction algorithm

```
1.   function EXTRACTKS(payload1, payload2, knownPlain)
2.    P′  ←   knownPlain
3.    C   ←   payload1
4.    D   ←   payload2
5.    O′  ←   createObfSet (P′ )
6.    for i ←  0, | O′ | do
7.    for t ←   0, | O′ i| do
8.    Si, t ←   O′ i,t ⊕ Ct
9.    i     ←   t++
10.   end for
11.   i     ←   i++
12.   end for
13.   for i ←  0, |S| do
14.   for t ←  0, |Si| do
15.   O′ ′ i,t ←   Si,t ⊕ Dt
16.   i     ←   t++
17.   end for
18.   Ei    ←   calcEntropy(O′ ′ i)
19.   i     ←   i++
20.   end for
21.   v     ←   calcMin(E)
22.   γ     ←   S[v]
23.   return γ
24.   end function
```

In algorithm 2: Lines 1–4. The function EXTRACTKS() is called with three arguments: the intercepted payload C, the chosen-plain text P′ previously generated, as well as a second payload D, corresponding to another infected computer belonging to the same botnet. Line 5. The set of obfuscated texts O′ is created in order to test all possible lengths of Zeus' botnet-id item. Lines 6–12. A derivate keys matrix S is generated as explained before, where every element S_i corresponds to one possible Zeus botnet-id item length. Lines 13–20. In order to discover the appropriate derivate key, every key S_i is tested against the second payload D. As a result, a matrix O′ ′ containing all the possible decryptions of D is obtained. Moreover, the entropy of each possible decryption is computed in line 18. Lines 21–24. The appropriate derivate key $S_v = \gamma$ is identified due to the lower entropy Ei resulting of its application to the payload D.

In this research further demonstrates how the entropy of these obfuscated messages O remains nearly constant, allowing us to automatically identify malicious traffic even if it is not fully deobfuscated. This propriety derives from the zero padding strings used by Zeus for composing its messages. As can be seen in Fig. 4, when these strings are obfuscated by the Chronus algorithm[4]. This algorithm routine, the resulting output contains several text parts with repeated characters . As a result, the calculated entropy drastically decreases in comparison with the one

calculated for the cipher text.

C. Malicious traffic identication

The last phase consists of detecting certain Zeus traffic on a production network, having a set of valid Zeus keystreams □={ γ_0 γ_1 …γ_i}, obtained as detailed in previous subsection. To perform this test, set up five different virtual machines V = { v_0,… v_4 } in an isolated network, which constantly generate common network traffic towards a fixed hosts range S = {s0, s2, . . . ,s_i}, e.g. web browsing and POP mail consulting. Next, we infected

one of them with a previously created Zeus malware sample mw1, and recorded all network traffic generated by

this machine during Δt_e. As discussed above, botmasters tend to migrate C&C servers, to avoid remediation, and frustrate takedown efforts. To maximize the number of victims that migrate to a new C&C server, botmasters tend to favor shorter TTL periods for domains. This of course is not universally the case, but botnets that use lengthy TTL periods must keep C&C servers up for at least that length of time, or suffer a loss in victim population with each server migration.

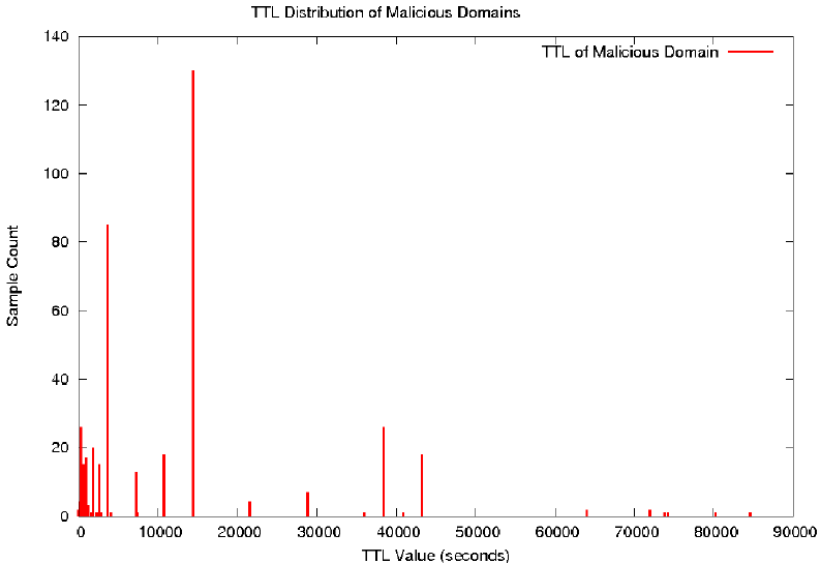

Fig no 4: A histogram of 20 sampled botnet C&C TTLs

In the above figure it shows a distribution of botnet C&C TTLs. The sampling started with 20 “active botnets”. It typically had a TTL > 86400. Many of the domains have very short TTL periods for the life of the botnet. In general, a CDF shows what fraction of an overall distribution falls below a particular threshold. In Figure number, its shown that 50% of the population had a TTL below 2 hours, and 85% of the population used less than 3 hours. With some exceptions, TTL periods for legitimate sites are often set for days. Short TTL periods are therefore an indication (but not proof) that a domain is suspicious. At the very least, they help one rank domains, so that an analyst is more productive in a manual review. Caution should be used when

focusing exclusively on short-TTL values. First, this parameter is easily manipulated by botmasters. (For example, most Dynamic DNS services provide a simple interface to let domain owners adjust TTL values). As such, botnet detection based solely on TTL values is extremely brittle. Second, it is quite common for legitimate domain owners to shorten TTL values in advance of IP renumberings or server migration. That is, many legitimate domains with long TTL values (e.g., TTL = 86400) will shorten TTLs in advance of network maintenance that results in IP changes, all to minimize disrupting clients. Network administrators observing DNS traffic at their network edge can use short TTLs to prioritize suspicious domains, and assist investigations. Similarly, researchers who encounter domains gathered from honeypots and binary analysis should note short TTL periods, since they are a hallmark of suspicious activities.

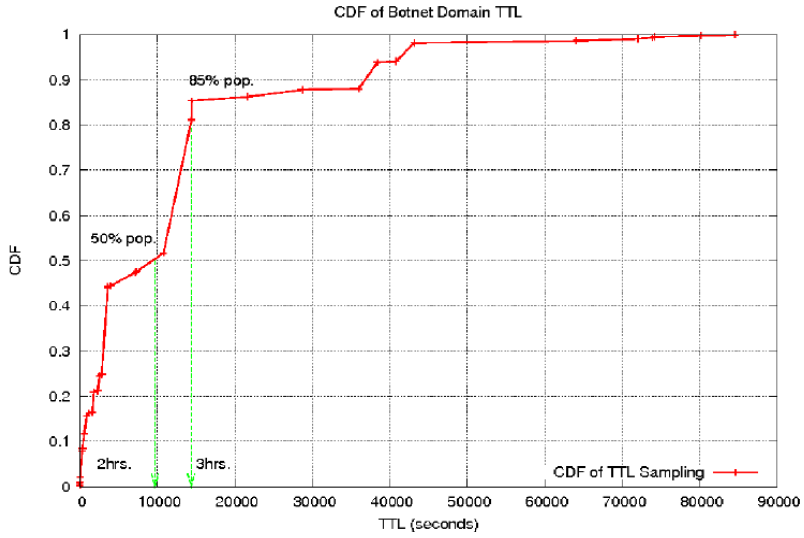

Fig no 5: A CDF of Botnet C&C TTL. Here the majority of the population is under a few hours.

V. EVALUATION

To generate real botnet traces, Zeus file is created using Zeus crime ware toolkit and then run executable bots in an unpatched Windows XP SP3 operating system installed in VirtualBox virtual machines. All input and output network traces of the virtual machines are captured and stored in a MySQL database. Each bot is run for 24 h. Both packet headers and complete packet payloads are stored for further analysis. In addition to collect traffic generated by bots, we also collect real network traces from a campus network.

In IP phae for each destination IP address identified in network flows, where,

- Suppose the maximum time interval between a request and its response is less than n seconds. We define α as a fixed size set that contains n counters, i.e., α = {a1,a2, . . . ,an}. Each counter in α has an initial value of zero.
- β is the total number of network requests.

- Suppose the maximum payload size is less than b bytes. We define γ as a fixed size set that contains b + 1 counters, i.e., γ = {r0, r1, r2, . . . , rb}. Each counter in c has an initial value of zero.

In IP phase, defining the following three states and their corresponding membership functions:

a) ***Inactive malicious IP address***: Assume that if an IP address receives many requests but does not respond, it is highly probable that the destination IP address is an inactive malicious IP address. We define a membership function X1 to calculate the probability of being an inactive malicious IP address:

$$X1(x) = \begin{cases} 1, & \sum \propto = 0 \; and \; \beta \geq \sigma \\ 0, & Otherwise. \end{cases} \quad \text{----------} \quad (1)$$

In the equation, $\propto$ is a threshold for the number of retries. When a destination IP address has been reconnected for more than $\propto$ times, the destination IP address is treated as malicious.

b) Malicious IP address: Since the computer with malicious IP address, e.g C&C server, often provide the same commands to bots, it can be observed that connections to these malicious IP addresses would have similar payload sizes. Without counting a payload size of zero, we define a membership function X2 to calculate the probability of being a malicious IP address

$$X2(x) = \begin{cases} \frac{\max(\gamma)}{\beta - r0}, & \beta - ro \geq \rho \\ 0, & Otherwise. \end{cases} \quad \text{----------} \quad (2)$$

Here bots always try to reach malicious IP addresses as possible as they could. If the number of network flows established with the destination IP address is less than a threshold ρ, the IP addresses is treated as benign and thus X2 has a value of zero.

c) Normal IP address: Defining a membership function X3 to calculate the probability of being a normal IP address. If a destination IP address has no failed network flows and the payload sizes are diverse, it would be a benign address. Therefore, the function X3 is defined as

$$X2(x) = 1 - \max\{X1(x), X2(x)\} \quad \text{----------} \quad (3)$$

In DNS phase, for each identified domain name, we define a feature vector x = (α,β,γ) for the domain name, where

- Suppose the maximum time interval between two successive DNS queries is less than n seconds. Define α as a fixed size set that contains n counters, i.e., α = {a1,a2, . . . ,an}. Each counter in α has an initial value of zero.
- β is the total number of DNS responses.
- γ is the number of failed DNS responses.

In this phase, we define the following three states and each state has its own membership function:

a) Inactive malicious DNS query, assume that a DNS query about an inactive malicious domain name usually gets a failed DNS response. Therefore, more failed DNS responses should lead to a higher membership value. Based on the observation, we define a membership function X1 which is used to calculate the probability of being an inactive malicious DNS query. The function X1 is

defined as

$$X1(x) = 1 - \frac{\beta - \gamma}{\beta} \quad \text{----------} \quad (4)$$

b) Malicious DNS query Since malicious DNS queries usually have similar time intervals. If most DNS queries for an identified domain name have similar time intervals, it could be a malicious domain name. We define a membership function X2 to calculate the probability of contacting a malicious domain name. The function X2 is defined as

$$X2(x) = \begin{cases} \frac{\max(\alpha)}{\sum \alpha} & , \sum \alpha \geq \rho \\ 0 \end{cases} \quad \text{----------} \quad (5)$$

In the equation, we define a threshold ρ. If the number of observed DNS queries is less than ρ, we believe that the identified domain name is benign and thus X2 has a value of zero.

c) Normal DNS query We define a membership function X3 to calculate the probability of being a normal DNS query. If an identified domain name has no failed DNS response, low query frequency, and diverse time intervals, it would be a benign domain name. Therefore, the function X3 is defined as

X3(x) =1-max {X1(x), X2(x)} ------- (3)

Numeric results

In this result it is tried for different thresholds of ρ for botnet traces to check the false negative rates (FNR) of malicious domain names and IP addresses. As ρ increases, FNR also increases. This is because some bots generate very few packets and the involved domain names or malicious IP addresses would not be detected if ρ is too large.

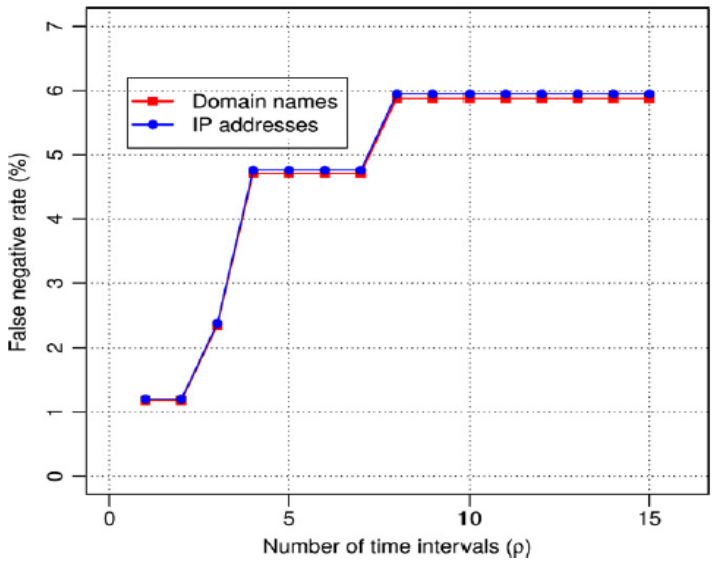

Fig no 6: Malicious domain names and IP addresses: false negative rate vs. ρ

DNS Request Rates

To help identify suspicious domains, we can rank and prioritize domains based on their associated request rates. The theory is that malicious domains (with large numbers of victims) should tend to have a larger volume of recursive and SOA refreshes. The problem of course is that some legitimate domains also have very high request rates. To address this problem, we can look at patterns of resolutions associated with different levels of a domain. We can classify DNS requests as either second-level domain (SLD) requests, such as example.com, or third-level subdomain requests (3LD), such as foo.example.com. To avoid increased costs and additional risks, botmasters tend create botnets within 3LDs, all under a common SLD. For example, a botmaster may purchase the string example.com from a registrar, and then also arrange for DNS service for the 3LDs botnet1.example.com, botnet2..., and so on. The botmasters use subdomains in order to avoid creating a new domain, different SLD for each new botnet, e.g., example1.com, example2.com.

Each transaction to create such a domain involves risk. The seller may be recording the originating IP for the transaction, requiring the bot master to use numerous stepping stones or proxies. Some registrars are careful about screening and validating the whois contact information provided by the domain purchaser. Some dynamic registrars require phone numbers and other identification. If the purchase is performed with stolen user accounts, there is a further risk of being caught. Since many DNS providers offer subdomain packages (e.g., a few free subdomains with DNS service) this allows the botmaster to reuse their purchased domain, and minimize both their costs and risk. Botmasters see another advantage in using subdomains. Even if service to a 3LD is suspended, service to other 3LDs within the same SLD is usually not disrupted. So, if botnet1.example.com is blocked, traffic to normaluser.example.com and botnet2.example.com is not disrupted. This lets botmasters create multiple, redundant DDNS services for their networks, all using the same SLD.

By comparison, most normal users usually do not employ subdomains when adding subcategories to an existing site. For example, if a legitimate company owns example.com, and wants to add subcategories of pages on their web site, they are more likely to expand the URL (e.g., example.com/products) instead of using a 3LD subdomain (e.g., products.example.com). This lets novice web developers create new content cheaply and quickly, without the need to perform complicated DNS updates (and implement virtual hosts checking in the web server) following each change to a web site. This is, of course, essentially a sociological observation about how botmasters and normal users behave when creating subdomains and domain content. There will be exceptions, and the behavior of both groups can also change. But the motivating factors (risk, cost, and convenience) should persist. We therefore assume that, in the large, this observation may hold for a class of botnets (but certainly not all). This fact helps us design a simple detection system. We can score domains based on the number of sibling and child domain lookups that occur. Thus, we can "penalize" the ranking of domains using the traffic volumes set to sister domains. For example, if one observes large amounts of legitimate traffic to google.com, and large volumes of botnet traffic to botnet1.example.com and botnet2.example.com, we can

sift out the botnets by scoring the parent zone, example.com, based on the traffic directed at its children. One can think of this as ranking families of domains, based on the amount of traffic sent to the parent zone's subtree. Similarly, it mirrors some of the analysis provided by Fig no 6, when run in trace mode, as discussed above. Logically, this must start at the SLD-level.

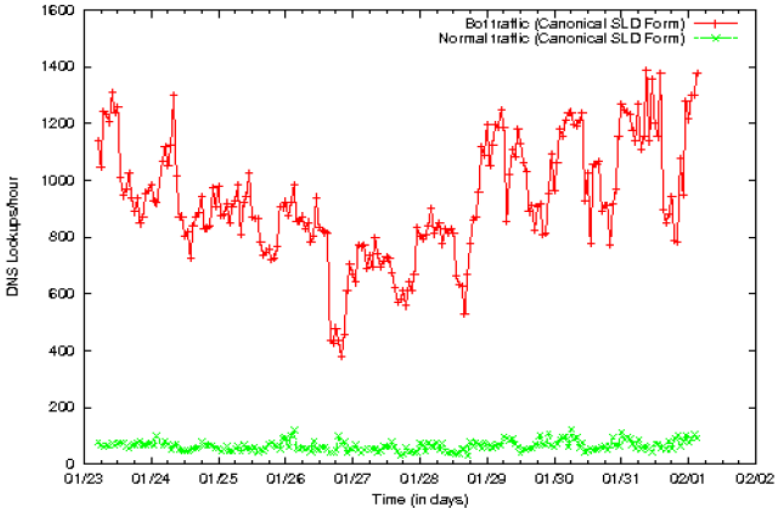

Fig no 6: Comparison of Canonical DNS Request Rates

When SLD domain traffic is placed into a canonical form (based on the volume of traffic directly to subdomains), it becomes much easier to distinguish the normal and bot traffic. Since the botnet traffic tends to favor a family of related domains (e.g., botnet1.example.com, botnet2.example.com), ranking the domains based on the traffic to a particular subtree helps separate the signal (bot traffic) from the noise (normal traffic).

VI. CONCLUSION AND FUTURE WORK

This paper has discussed key properties of traffic reduction algorithm to reduce the amount of traffic and how botnets are identified by adopting key extraction and malicious traffic detection methods using DNS. Based on common bot host behavior, the algorithm is divided into two stages: (1) Key extraction: identify the input raw data and processing of the Bots specific activities; and (2) malicious Traffic Detection using DNS. Botnets generate large waves of DNS traffic. This is dampened by the impact of caching, both at the host application/stub level, and at the recursive level. The detection and remediation of botnets is assisted by DNS sensors as well. By logging *all* addresse associated with a domain, passive DNS lets administrators expand investigations, and implement more complete remediations. To increase the network agility of a botnet, botmasters favor short TTL periods, or the time a domain is cached by a host or caching DNS server. Since botnets often have many victims, large volumes of DNS traffic are associated with botnets, particular at authority DNS

servers. Many legitimate domains also experience large volumes of traffic. These detection techniques are merely heuristics. Botmasters are human adversaries, and can respond to detection strategies.

In the proposed paper, Although traffic reduction brings benefits, the above criteria, it also raise some concerns. First, ignoring P2P traffic eliminates the possibility of identifying P2P bots. Second, skipping short lived flows may cause failures in detecting inactive botnet traffic. An inactive bot is a bot that is not able to connect to its command and control (C&C) server either temporarily or persistently. An inactive bot should also be detected so that the network can be protected if the bot becomes active again. Third, when black lists and white lists are used to reduce input traffic, the lists must be well managed and make sure that the lists are up-todate. This is because a benign host may be compromised and then turned into a malicious one at any time. If a list is not updated accordingly, the malicious traffic involved with a newly compromised host may be incorrectly ignored. The highlighed future work will be rectified soon.

ACKNOWLEDGMENT

AUTHORS THANK THE MANAGEMENT OF R.V.S SCHOOL OF ENGINEERING & P.S.N.A COLLEGE OF ENGINEERING FOR RENDERING THEIR CONTINUOUS SUPPORT, ENCOURAGEMENT AND COOPERATION THROUGHOUT THE PROGRESS OF THE WORK.

REFERENCES

[1] B. McCarty, Botnets: big and bigger, IEEE Security and Privacy 1 (4) (2003) 87–90.

[2] Meisam Eslahi, Rosli Salleh, Nor Badrul Anuar "Bots and Botnets: An Overview of Characteristics, Detection and Challenges" 2012 IEEE International Conference on Control System, Computing and Engineering, 23 - 25 Nov. 2012, Penang, Malaysia.

[3] S. Nagendra Prabhu, D. Shanthi , "A Survey on Anomaly Detection of Botnet in Network" , International Journal of Advance Research in Computer Science and Management Studies, Volume 2, issue 1, January 2014

[4] Marco Riccardi, Roberto Di Pietro, Marta Palanques, Jorge Aguilà Vila, "Titans' revenge: Detecting Zeus via its own flaws" ELSEVIER - Computer Networks, Volume 57, issue 2, February 2013

[5] H.Gu, J. Zhang, and W. Lee, "Botsniffer: Detecting botnet command and control channels in network traffic," in Proc. 15th Annual Network and distributed System Security Symposium (NDSS'08), 2008.

[6] Dr. Laheeb M.Ibrahim, karam Hatim Thanoon "Detection of Zeus botnet in computer network and Internet" 29th October 2012. Vol 6 No.1, JITBM & ARF

[7] L. Wenke, W. Cliff, and D. David, Eds., Botnet Detection: Countering the Largest Security Threat, ser. Advances in Information Security. Springer-Verlag New York, 2008, vol. 36.

[8] L. Wenke, W. Cliff, and D. David, Eds., Botnet Detection: Countering the Largest Security Threat, ser. Advances in Information Security. Springer-Verlag New York, 2008, vol. 36.

[9] Amit kumar Tyagi, G.Aghila, "A wide scale survey on Botnet" in International Journal of Computer Applications (0975-8887), Volume 34-No.9, November 2011.

[10] Kuochen Wang, Chun-Ying Huang , Shang-Jyh Lin , Ying-Dar Lin, "A fuzzy pattern-based filtering algorithm for botnet detection" Proceeding of Computer Networks, ELSEVIER, 2011

[11] Binsalleeh H., Ormerod T.,‖ On the Analysis of the Zeus Botnet Crimeware Toolkit‖ , 2010,IEEE, Computer Security Laboratory, Concordia University Montreal, Quebec, Canada.

[12] Banking C. Livadas, R. Walsh, D. Lapsley, W.T. Strayer, Usilng machine learning technliques to identify botnet traffic, in: Proceedings of the 31st IEEE Conference on Local Computer Networks, IEEE, 2006, pp. 967–974.

[13] Dr. Laheeb M.Ibrahim, karam Hatim Thanoon "Detection of Zeus botnet in computer network and Internet" 29th October 2012. Vol 6 No.1, JITBM & ARF

[14] 13. Wyke J.,‖ What is Zeus?‖ , 2011, Threat Researcher, SophosLabs UK

[15] Abuse.ch, The Swiss Security Blog. <https://zeustracker.abuse.ch/ statistic.php> (cited 19.12.11).

[16] Amit kumar Tyagi, G.Aghila, "A wide scale survey on Botnet" in International Journal of Computer Applications (0975-8887), Volume 34-No.9, November 2011.

[17] Configuration setting of Zeus botnet and web page injection is available: http://resources.infosecinstitute.com/botnets-unearthed-the-Zeus-bot/

[18] Yazan Boshmaf ⇑, Ildar Muslukhov, Konstantin Beznosov, Matei Ripeanu, "Design and analysis of a social botnet", ELSEVIER - Computer Networks, Volume 57, issue 2, 2013

[19] M. Sharif, A. Lanzi, J. Giffin, W. Lee, Impeding malware analysis using conditional code obfuscation, in: Network and Distributed System Security (NDSS), Citeseer, 2008.

AUTHOR PROFILE

S. Nagendra Prabhu is currently working as Assistant Professor in Department of Computer Science & Engineering, R.V.S Educational Trust's Group of Institution and pursuing his PhD in Information and Communication Engineering from Anna University, Chennai, India. He completed his Master of Engineering in Network Engineering from Anna University, Coimbatore and completed his Bachelor of Engineering in Computer Science and Engineering from K.C.G College of Technology, Chennai. His research interest includes Cloud computing, Botnet attack, Web based network Security. Currently the author is working on security issues in Cloud Computing.

Dr.D.Shanthi Saravanan received her B.E. Computer Science and Engineering in 1992 from Thiagarajar College of Engineering, Madurai, Tamil Nadu and M.E. Computer science and Engineering from Manonmaniam Sundaranar University, Tamil Nadu and PhD in Soft Computing from Birla Institute of Technology, Ranchi. She is currently working as a Professor & Head in Department of Computer Science & Engineering, PSNA College of Engineering and Technology. She has more than 21 years of Teaching and Programming Experience. She is member of various professional societies like IEEE, CSTA, IAENG and IACSIT. Her research interests include Genetic Algorithms, Neural Networks, Intelligent Systems, Image Processing, Embedded System, M-Learning and Green Computing. She has published more than 25 papers in International Conferences and Journals and also published 5 books in computing and applications. She is the reviewer of various international journals.