Botnets Detection Using Message Sniffing

Walid Al-Ahmad and Ayat Al-Ahmad*
Gulf University for Science & Technology, Department of Computer Science
Mishref, Kuwait
Alahmed.w@gust.edu.kw
*The Hashemite University, Department of Software Engineering
Zarqa, Jordan
ayatn@hu.edu.jo

ABSTRACT

A botnet is a large number of compromised computers which are used to create and send spam, viruses or a flood of network messages as a distributed denial of service attack for different purposes such as curiosity or identity theft among others. The growing popularity of botnets compels to find proper countermeasures, while existing defense mechanisms can hardly keep pace with the progress of botnet technologies. The aim of this paper is to develop a botnet detection technique by implementing a botnet application, based on the TCP protocol, to capture and filter a packet header in a local network. This sniffer application also provides a statistical chart that shows malicious bots featured messages.

KEYWORDS

Botnets; security countermeasures; message sniffing; security attacks; network protocols.

1 INTRODUCTION

As the information technology progress has penetrated almost into all the domains of life, the Internet usage has become nearly indivertible everywhere in the world. Daily activities, such as online shopping, Internet banking, business, etc. in a wide range have created an opportunity that this technology be exploited. Among the various forms of threats, botnets are emerging as the most serious threat against the Internet. Bots are software applications running as an automated task over the Internet. A botnet is a network of compromised end hosts (bots) which is under the remote command of a botmaster. Botnets, presently, comprise an increasing portion of Internet malware which harm networks' security. The number of infected computers is steadily growing day by day. According to Symantec [1], about a million bot computers are present at any given moment. Hackers attempt to compromise a computer or a network to perform their illegal actions such as launching distributed denial of service attacks against critical targets, malware dissemination, phishing, and click fraud just to mention a few. A group of hosts in different locations that are controlled by a person who initiates the malicious attempt in order to perform an attack are called botmaster. The victims who are controlled by one or more botmaster are called Zombies or bots. The bots are managed via so called command and control (C&C) channels. Via these channels, a botnet botmaster /operator issues commands to the bots, e.g. to organize attacks. Choosing a C&C structure and protocol is one of the most important decisions an attacker needs to take before building a botnet

There is a mounting interest among researchers to understand the botnets phenomenon and implement techniques to detect and prevent them. One of such research work attempts to shed light on the question of botnet membership. The authors focus on IRC botnets because of their continuing prominence in the Internet today. More specifically, they survey a number of techniques for determining botnet membership and examine the different views they generate. In addition, they show that the issue of botnet membership extends beyond single botnet considerations in that potential cross-botnet relationships add another challenge to estimating membership. They discover a number of mechanisms to estimate the

size of botnet and highlight the challenges associated with each. They show results gained from recent mechanisms. They also examine potential hidden structures among botnet they tracked and highlighted their implication on determining botnet membership [2].
Another research work has implemented an algorithm for detecting a botnet. The authors mention features of botnet DNS traffic that is distinguishable from legitimate DNS traffic. They defined the key feature of DNS traffic called group activity, as they studied and grasped botnets behavior. They developed an algorithm that differentiates a botnet DNS query by using group activity feature. In addition they analyzed the algorithm to prove feasibility of their mechanism [3].

Yet another research work has proposed an approach that uses network-based anomaly detection to identify botnet C&C channels in a local area network without any prior knowledge of signatures or C&C server addresses. This detection approach can identify both the C&C servers and infected hosts in the network. The authors evaluated their tool (BotSniffer) using many real-world network traces. According to the authors, the results showed that BotSniffer could detect real-world botnets with high accuracy and had a very low false positive rate [4].

The approach for detecting botnets in [5] is to examine flow characteristics such as bandwidth, duration, and packet timing looking for evidence of botnet command and control activity. The results show that botnet evidence can be extracted from a traffic trace containing almost 9 million flows.

In fact, botnet detection has been recently an interesting research topic and many approaches have been proposed. This paper addresses the question whether a sniffer can function as security mechanism in local networks. The goal of this work is to take advantage of sniffing software, which originally has a negative usage to harm network security, utilizing it as a security tool in detecting botnets, thereby protecting local area networks from malicious codes produced by bots. A sniffing program is developed for that purpose. The program analyzes the sniffed packets based on their contents; IP header, protocol (TCP) and data to detect the possible existence of bots on a local area network.
The rest of the article is organized is follows: section 2 outlines the system architecture of the proposed botnet detection system. Section 3 then presents several aspects of the design and implementation of the system. Finally, section 4 concludes the article.

2 SYSTEM REQUIREMENTS

In this section, the architecture and system requirements are specified. The requirements for the sniffing are:

- The system is implemented on a Linux operating system for its suitability for network operations such as packet filtering by utilizing raw sockets (see figure 1) for its sniffer program [6, 7]. This choice was made because it provides a way to take packets from a NIC network driver and move up to the application layer directly without passing through the other layers (network, transport and presentation layers). The system needs botnet environment (client/server) applications separately, which are written in the C language.
- The key component is the sniffer program, which functions as a monitoring factor and displaying the output. This program is also written in the C language because this language is more integrated with the kernel level to have access to raw socket libraries. This system works only in a local area network.
- A computer with Linux operating system that carries the core work. Its broadcast mode of the interface LAN must be enabled so as to receive all the packets in the same network (simply by typing ifconfig Ethernet_ID promisc).

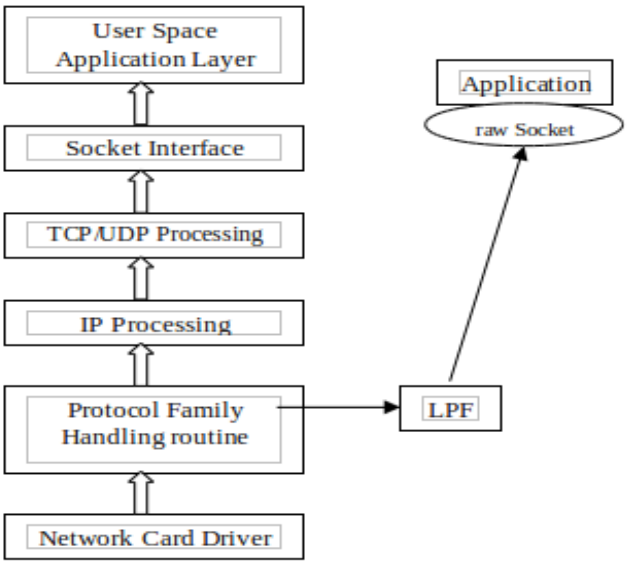

Figure 1. Raw socket function.

3 SYSTEM DESIGN and IMPLEMENTATION

The main purpose of this research work is to develop a program to detect a botnet based on a sniffer mechanism in a LAN. The program works by monitoring a group of activities and checks the content of the messages exchanged between bots and botmasters. The monitoring process is carried out by a sniffer program.

The introduced botnet detection mechanism is based on the TCP protocol and works according to the following phases: First, discovering several features of a botnet system and how it works. Second, monitoring traffic and capturing the IP headers of the TCP packets. Third, showing a statistical chart to make distinction between legal and illegal activities, storing them in black and white lists.

3.1 Developing a TCP Botnet

For developing botnet based TCP message exchange, a real situation of a botnet environment based on client/server architecture has to be created that represents a logical network of the botnet. The botnet application consists of two different parts. One part is the client (bot) application which is mostly infiltrate into computer clients. This application software could be combined with .jpg, .doc, .zip etc. files, or it could come through scamming websites, or through USB memory sticks, and gets transferred to the victim's computer.

After the application has been executed, it copies and pastes itself into sensitive parts of the operating system such as “C:\windows\system32” that contains thousands of windows files and services. The infiltrated file, then, could not be easily found. The second part to be implemented by the bot system is creating a path where all startup paths are located so as to start functioning once the infected computer starts working. Once the server establishes connection with its clients, they are ready to receive commands from the server.

Within the bot (client) application, there is an infinite loop to receive command messages from the server. As a message is received by the client, it is analyzed and separated into an indicator and array of strings. The indicator is used to detect which procedure should be called. Each procedure accounts for performing one type of attack such as syn flood, taking parameters in order to deliver the message command. For instance, the procedure responsible for "syn flood attack" takes two parameters; the first is for target attack and the second is for the interval of how many times syn packets are sent to the specified target.

The server (botmaster) is characterized by being only command sender, not to receive any, while the client only implements commands from the server. There is also an infinite loop in the server like in the client application which consists of listening and binding function for each client (bot); it attempts to join the logical network of botnets. After the binding process is completed, the server creates a range of IP addresses from the local hosts to send them messages. In the same loop there is a user prompt that asks the botmaster to enter a command such as (syn_1.2.3.4_200000, meaning do syn attack to the destination 1.2.3.4 address 200000 times). Just by pressing enter the

message will be sent to the defined range of IP addresses at the same time.

3.2 Botnet Features on LAN

The following features (mentioned below) appear in a botnet when it is built and it is only based on local area network. Most of the connections are established almost simultaneously. For instance, employees in a company are supposed to turn on their computers at the same time or between 8:30 – 9:00. Then the application will be run with startup and the first thing it tries is to establish the connection. Since the server is always sending commands to its clients and they just receive commands, the botnet could be characterized by having one-to-many relation (server-to-multi clients). In fact, this is centralized command and control (C&C) architecture, the most commonly used botnet architecture. In such architecture, botmasters use central servers to issue their commands to the selected sets of bots. Bots act as clients where the botnet is built using the client-server scheme [8].

3.3 Sniffer Program

In their efforts to defend against botnets threats, researchers developed detection schemes of C&C communication. Basically all approaches share the idea of detecting and assessing patterns in the communication flow [9]. The detection scheme developed in this current work follows the same footsteps. It uses a sniffer program to capture and analyze communication between a botnetmaster and its clients (bots).

A sniffer is a program that monitors data traveling over a network. Originally it is used to perform malicious attempts in networks; however it has been used in this current work as a detector of malicious attempts, especially botnets based on TCP. The basic idea is to match various protocol fields, and the payload of a packet, against pre-defined patterns of abnormal or suspicious content. The sniffer is written in the C language and works on a Linux operating system, through utilizing raw socket. Figure 2 below figure shows an output from the sniffer program.

When the application is run, the sniffer takes three parameters; the first parameter is file name; it takes the parameter of the file name, then searches for the name to ensure whether or not it exists; if it does not, it creates one. The file is vital for the sniffer because it carries the received information from the packet such as source and destination IP addresses, source and destination Mac addresses, and data. The second parameter is the name of the network interface device that is defined on the system, and acts as an interface to the network it is connected to in order to receive all packets. After that the sniffer creates a raw socket then binds it with the network interface device. The third parameter is used to select the number of the packets to be captured. A flowchart of the sniffer program is shown in figure 3.

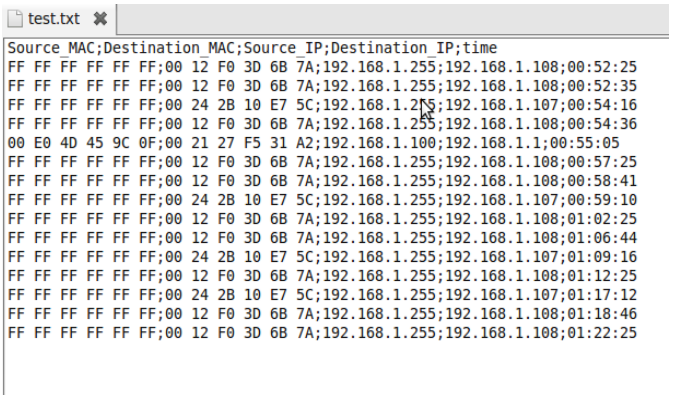

Figure 2. Output from Sniffer Program.

Then the program continues to enter into a finite loop determined by the number of packets parameter. Inside the loop the sniffer tries to create sockets, if any error occurred, the program will end. Otherwise, the program will continue and tries to bind the created socket with the Ethernet defined in the system. Again, if any error occurs the program will terminate. Otherwise, it keeps performing its tasks. Because the sniffer application works in a local area network it should only deal with the packets which travel within the network. These packets belong to private IP addresses and ignore all packets outside the LAN (Internet). Therefore, a filter needs to be placed to filter the IP header since source and destination IP

addresses are stored there, and if they meet the private IP addresses (class C addresses) the process will go smoothly and pass the next step of the program, otherwise it drops the packets off the Internet (public IP).

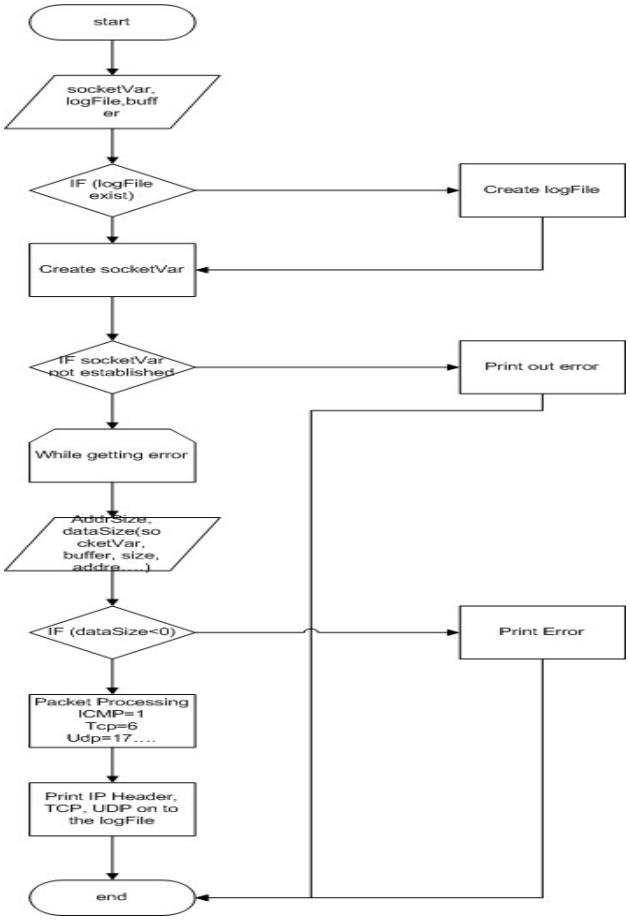

Figure 3. Sniffer Program Flowchart.

The next filter forces the received packet from the IP filter which is called protocol filter because the sniffer application is designed to detect botnet based TCP and to ignore all other protocol types. There is a structure in Unix System, namely the raw socket which contains an integer field called protocol used to identify protocol type. Its value for TCP protocol is 16; this field should be used so that TCP protocol could be easily separated from other types of protocol such as UDP, ICNP, STP, etc.

After the packet satisfied both filters, the print procedure will follow. This procedure analyses the packet based on destination MAC address, distention IP address, computer name, data and date and time. These values will be stored in the file identified by first parameter in a way they could be imported from the application data and required queries apply on them. By comparing the released packets from the host, it can be statistically analysed using a MS application such as access or excel according to time and to follow up to a period of time. Additionally, their contents cannot be checked from strange and anomalous messages such as: syn_1.2.3.4_2000000; For example, when the server sends this message to its clients; it means to implement syn flood attack for IP 1.2.3.4. Any of these strange messages with the previous conducted statistics lead us to identify botnet in the LAN.

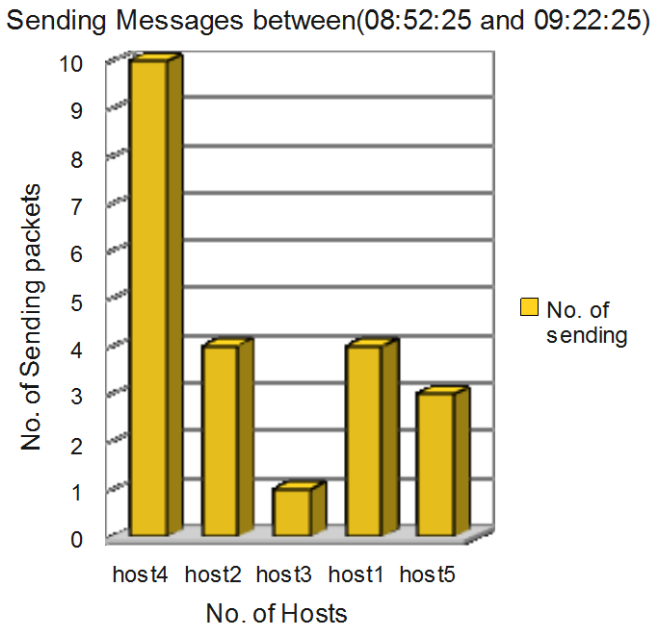

Figure 4. Statistics Chart for Sending Messages

The above figure illustrates how the server (botmaster) could be spotted through analysing the sent messages by the hosts, showing five hosts regularly send messages. The data has been taken between 08:52:25 and 09:22:25 for a number of days. It indicates that Host number 4 is a server(botmaster) as it has been sending more messages to its clients; hence, the other hosts almost had sent the same rate of messages and far

less than host 4; so by monitoring a period of time we can estimate that (host1, host2, host3, host5) became bots.

4 CONCLUSIONS

Botnets is emerging as a major threat to the Internet. Botnet detection is a challenging and an important countermeasure to defend against this threat. This paper has presented a botnet sniffer program, a LAN anomaly-based botnet detection system that explores the message exchange as part of the command and control activities between a botmaster and its bots. The basic idea behind the program was to use a sniffer as a defending tool for bots actions within the network. As originally was common, a sniffer application was functioning as a malicious program. The sniffer application filters and captures the header packets in a LAN and detects those bot featured messages, putting them under two categories (legal and illegal), which constituted the output of the program.

5 REFERENCES

[1] D. Turner, ed.(2004), "Symantec Internet security report trends for January-June 2004, Symantec, Vol. VI, 2004, pp. 1-55.

[2] M. A. Rajab, J. Zarfoss, F. Monroe, and A. Terzis, "A Multifaceted approach to understanding the botnet phenomenon," ACM SIGCOMM/USENIX Internet Measurement Conference, 2006.

[3] C. Hyunsang, L. Hanwoo L. Heejo, and K. Hyogon, "Botnet detection by monitoring group activities in DNS traffic," 7th IEEE International Conference on Computer and Information Technology, 2007.

[4] G. Gu, J. Zhang, and W. Lee, "BotSniffer: detecting botnet command and control channels in network traffic," 16th Annual Network & Distributed System Security Symposium, 2008.

[5] W. Strayer, R. Walsh, C. Livadas, and D. Lapsley, "Detecting botnets with tight command and control," 31st IEEE Conference on Local Computer Networks, 2006.

[6] M. J. Donahoo and L. Kenneth, TCPIP Sockets in C, 2nd Edition, Calvert, 2009.

[7] B. Hall, Beej's Guide to Network Programming, Beej Jorgensen Hall, 2009

[8] E. Y. UCE, "A Literature survey about recent botnet trends," retrieved from www.geant.net/Media_Centre/Media_Library/.../botnet_trends_M2.pd

[9] M. Feily, A. Shahrestani, and S. Ramadass," A Survey of botnet and botnet detection," Third International Conference on Emerging Security Information, Systems and Technologies, 2009