International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2015): 6.391

J-Botnet Detector: A Java Based Tool for HTTP Botnet Detection

Thejiya V[1], N Radhika[2], B Thanudhas[3]

[1]MTech Computer Science, Amrita Vishwa Vidyapeetham, Amrita Nagar PO, Coimbatore, India- 641112

[2]Associate Professor, Amrita Vishwa Vidyapeetham, Amrita Nagar PO, Coimbatore, India- 641112

[3]Group Director, Scientist/Engineer, VSSC/ISRO, Trivandrum

Abstract: ***Botnets are collections of compromised computers (Bots) which are remotely controlled by its originator (Bot master) under a common Command and Control framework. In this paper, we are concentrating on HTTP bots. It is more widespread and more difficult to detect compared to all other bots. In the HTTP bots, bot masters use HTTP protocol to hide their activities among the normal web flows and can easily avoid current detection methods like firewalls. Therefore the chance of HTTP attack is increasing day by day. We are proposing a new technique for HTTP botnet detection. As our system is designed in java, it can potentially run on various platforms like MS-Windows and UNIX systems or on any other handheld devices. It also includes a web based monitoring interface which can give alerts about attacks and it can help network administrators to better understand the network traffic and possible attacks.***

Keywords: Distributed Denial of Service, Command and Control server, Distributed hash tables, Deep Packet Inspection, Java Native Interface, Time To Live.

1. Introduction

A botnet is a network of compromised computers (bots) that have been infected with malicious code, and can be remotely-controlled through commands sent via the Internet by a bot master, without the knowledge of the owners of the computers [13]. The word botnet comes from robot and network [11]. More and more new types of botnet attacks have been discovered. Botnet detection has been a research topic in recent years. The architecture diagram of botnet is shown in Figure1.

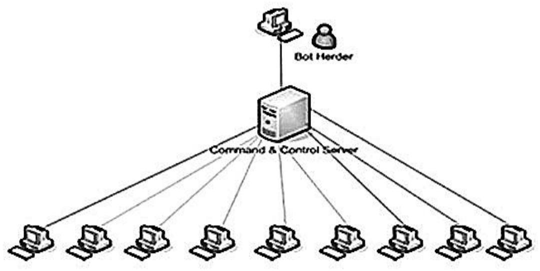


Figure 1: Architecture diagram of Botnet

The main difference between botnet and other malware is the Command and Control (C and C) channel [12].This channel allows bots to receive commands and perform malicious activities. The three important bots are Internet Relay Chat (IRC) bots, Peer to Peer (P2P) bots and HTTP bots. Both IRC and HTTP bots can be used to control a number of infected systems [1]. But there are some fundamental differences in their architecture. In an IRC botnet, the infected hosts join an IRC channel and waits commands from a bot master that has also joined the channel [2]. In this way, commands can be pushed to infected clients at any time. The HTTP mechanism requires the infected client to send an HTTP GET request to a C and C website [3]. Hosts that are infected within this type of infrastructure use a pull mechanism and can only receive commands after they have sent a request to the C and C server. HTTP is the protocol most commonly used for the delivery of data over the Internet. This includes human readable content like websites and images and also binary data transported in uploads and downloads. Because of all these features, HTTP is available in nearly every network connected to the Internet and is rarely filtered. This protocol is used to publish the commands on web servers [3]. Instead of remaining in connected mode, HTTP bot periodically visits certain web servers to get updates or new commands. This model is called PULL style and continues at a regular interval that is defined by the bot master [10]. Using HTTP protocol, Bots hide their communication flows among the normal HTTP flows [4] .This paper describes a method for detecting potential HTTP bot activity based on repeated HTTP connections to a C and C server. Rest of the paper is structured as follows. Section II describes the overviews of Botnets. Section III presents existing methods for detecting Botnets. Section IV describes the proposed methodology. Section V refers to overall architecture and section VI discusses about experimental results. The paper is concluded in section V and the references to this paper are mentioned at the end.

2. Overview of Botnets

Botnet is a common term referring to a collection of automated software robots that run without human intervention [9]. They are mostly malicious in nature; however they can also be associated with a network of distributed computers. An attacker usually gains control by infecting computers with a virus or other malicious code that gives the attacker access [4]. Bot master is a server node or attacker node. This node will send instruction to any other node. It monitors other nodes and maintains details about the bot. A bot master issues a special command called a report to the botnet. It will instruct every bot to send its information to a specified machine that is controlled by the bot master.

2.1 Classification of Botnets

Depending on the C and C mechanism, a bot can communicate with the bot master or other bots, which makes botnets different from traditional Trojans, worms and other malware. The bot master can issue commands to thousands of bots to launch an attack [8]. Depending on the type of communication of bots to C and C, there are different kinds of botnets. They are described below.

Centralized botnets: A centralized botnet usually contains a C and C server, which is used by its bot master to send commands to the bots [18]. All the commands are sent by the bot master, there is no command transmitted among the bots. Bots ask for new commands from the C and C server periodically or the C and C server sends commands periodically. As the C and C server is the core of a centralized botnet, a botnet will be disabled as soon as it is found by a detector [15]. However, because the commands are transmitted fast and the latency is low, it is still being widely used in the wild. IRC-based botnets and HTTP-based botnets are both centralized botnets.

1) IRC-based botnets: IRC-based botnets play an important role in the development of botnets. It is the originator of all kinds of botnets. The first botnet was based on IRC. At the beginning, people developed IRC-based bots to help people when they were chatting [2]. But, the idea was used by some malicious developers, and then the earliest IRC-based botnet emerged. It could be used by an attacker to control the hosts in the network and to carry out malicious activities to some victim targets. It is the foundation of today's botnets. Because the communication characteristics were easily detected and the data was not encrypted, it is relatively easier to detect IRC based botnets compared to other kinds of botnets [11]. However, it is also easier to construct IRC-based botnets, which makes them popular among attackers. For the detection of IRC-based botnets, researchers already did a great deal of theoretical studies of them and also a large number of experiments. So, there are some effective approaches to detect IRC-based botnets.

2)HTTP-based botnets: When hackers started to seek new C and C mechanisms for botnets, HTTP-based botnets came into being. Different from IRC-based botnets, the traffic of HTTP-based botnets can be hidden in normal HTTP traffic [4]. So, it can evade the interdiction of firewalls so as to achieve the purpose of avoiding detection. The spam model of the Rustock rootkit used encrypted HTTP in C and C communications to make it difficult to be detected. Black Energy is another typical HTTP-based botnet. Detecting HTTP-based botnets is harder than IRC-based botnets, because HTTP-based botnets can hide their traffic better.

3) P2P-basedbotnets: Detecting or mitigating a P2P botnet is very challenging. Different from centralized botnets, C and C servers are not used in P2P botnets. All the communication relies on the P2P protocols used by P2P botnets. Attackers can send a command through any bot in a botnet and this bot will forward the command until all the other bots receive the command [15]. A bot in a P2P botnet acts as a server as well as a client, which means it can both send commands and receive commands. There is a large portion of P2P traffic in the Internet, so the traffic coming from P2P botnets is hidden in the benign P2P traffic just as HTTP-based botnets [14]. Even if we find some bots in a P2P botnet, it still can be a problem to mitigate an entire botnet, because a P2P botnet can work well when some bots are removed from it. Although P2P botnets are more complicated than centralized botnets, they are more robust.

2.2 Botnet Infection Methods

Botnets use a wide range of approaches to infect computers [13]. The infection component of a botnet is often referred to as a worm because it replicates itself to other computers. There are four main methods of infection:

Email – Emails sent may contain infected attachments and if they are opened, the computer will be compromised [13]. In the past, these were easier to spot because they were from unknown or suspicious email addresses. But these emails are now using social engineering pretending to be from companies such as banks or from people that are known to the user. Often when a machine is infected the user's address book will be read and emails were sent to all contacts to increase infection. It is becoming increasingly difficult to spot a genuine email from a phishing email.

Link Spam – Link spam is a URL that takes the user to malicious content [13]. They can be delivered via a variety of methods such as email, social networking sites and instant messaging. All these use social engineering to try and convince the user that is genuine. These messages can often appear to be from friends or trusted organizations but are spoofed; spoofed as in pretending to be from someone that it is not.

Website – Websites are often looks like a well-known site such as YouTube or an existing trusted website is hacked and malicious codes inserted [13]. There are two types of attacks: client-side attacks and exploit downloads. In a client-side attack a user visits the website; the malicious code will run and attempt to use several vulnerabilities on the web browser in order to gain access to the user's machine. If vulnerability is used successfully, the user's machine will be infected. The website will use several vulnerabilities because success will depend on how up-to-date the software is and whether third party plug-ins are on the computer. An exploit download is where the user is prompted to download a file. If the user accepts the download, the computer will execute the file and infect the machine. The website will then use social engineering to get the user to be accepted.

Exploits – This is similar to client-side attacks however the vulnerabilities are not limited to the web browser [1]. Infected machines will scan the network for other computers and will use a list of exploits or weak password attacks against any machines found.

Spam and Phishing emails – There were an estimated 107 billion spam messages sent every day in 2009. 89.5 billion (83%) of those spam emails are sent by Botnets [7]. Spam is a big problem not just because it is a nuisance to the user but because phishing also misleads them into giving confidential information such as usernames, passwords and credit card information. Even though most spam mail is blocked or ignored by users, 0.0001% of users respond to the email that is meant to have more than $100 profit per day.

DDoS – Distributed Denial of Service (DDoS) attacks by botnets is an important and widespread problem because they are very hard to defend against [1]. DoS attacks occur when a server is flooded with traffic which uses up all the bandwidth, meaning genuine traffic cannot reach the server. Traditional attacks are easy to defend against using firewalls which block the attackers IP. However with click fraud, the traffic comes from too many sources. They are usually spread all across the Internet which makes it virtually impossible to tell whether it is malicious or genuine traffic.
Key logging – Botnets also gain a lot of information from any infected computers in a network. Every piece of information and everything typed on an infected machine could be compromised using a key logger. A key logger is a piece of software that records everything that is typed [10]. Passwords for websites such as PayPal and banking websites would be very profitable.

Click Fraud – Click fraud is where bots visit a website pretending to be genuine users and click on sponsored commercials claiming to make a profit for the website owner [1]. This fraud takes money from online advertising companies who pay a small amount per click. It is hard to detect this fraud because each click has a different IP address because they are distributed so broadly over the Internet.

Malware – Botnets spread malwares to increase the number of infected machines [13]. There are many methods used and the most popular one is email. However more recently, social networks and Instant Messengers such as Facebook, Twitter, etc. send messages to users pretending to be friends. These messages contain URLs to malicious web pages that contain exploits. If the user visits the site, their machine could be compromised. The major botnets involved in this type of behavior is Koobface. It targets Microsoft Windows computers and interacts with social networks including Facebook, MySpace, Twitter and Bebo in order to spread infection. When a computer is infected it reads the user's cookies and then downloads the appropriate modules used to gain access to the social networks. It sends out messages to friends with URL links to some of the video website. The video looks like YouTube but asks the users to install a plugin to be able to view it. This is a Trojan and infects the user machine.

2.3 Bot's Life Cycle

The general life cycle of a botnet contains four phases: Initial Infection and Propagation, accepting instruction and malicious activities.

Initial Infection and Propagation – Bots can spread and propagate only if there are existing vulnerable hosts [4]. The attacker appends the bot by attaining access to the victim's host. The victim's host is infected with a bot by exploiting some operating through malicious websites by opening emails that contain malicious data or downloading malicious payload from P2P network. The malicious payload is transparent to the user. The user has no clue that his computer is being used for malicious purposes. Once a bot is installed on a victim's machine, it changes the system configuration to start each time when the system boots. The bot might have the functionality to spread itself by sending out more emails or scanning more computers, thus creating a botnet. If an attack is detected, it can only be traced back to the source not to a bot master.

Command and Control server (C and C) -Remote Control Channel –Once the bot is installed, it will get instruction to connect to command and control (C and C) server to receive commands [4]. Command and control allows control of compromised machines from one centralized system, usually through IRC channel. The bot connects infected machines to the IRC server with a randomly generated nickname. Then the bot joins the attacker's channel with a predefined password and remains inactive waiting for the command from the bot master. The bot master validates the identity of the bots using a password, ensuring that the bots cannot be controlled by others. Once authenticated, the bot master issues commands in the channel and all bots react and respond to the commands.

Accepting Instructions – The botmaster logs into the IRC server and starts distributing commands [4]. Commands are sent to infected machines via the controllers. These commands can include instructions such as downloading additional payload to bots. Addition to that the bot master can transfer files to or from bots and can perform DoS attack, or use them for other malicious activities. Spread to additional machines – Botmaster uses C and C to achieve new vulnerabilities of other systems which allow bots to spread to other machines through the local networks, often circumventing firewall and IDS.

2.4 Examples of Bots

During the past few years, thousands of different bots have been developed. The most notable bots include Eggdropbots, GTbots, Sdbots, Agobots, Spybots, Phatbots and other Peer-to-Peer bots. A brief description of each type of bot is presented below.

Eggdrop Bot– Eggdrop bot is one of the most eminent and widely used IRC bot [21]. It was written in 1993 by Robby Pointer to monitor a single channel. Eggdrop bot was written in C language. The user can add functionality to the bot since it allows execution of user added Tool Command Language (TCL) scripts. The Eggdropbot allowed secure assignment of privileges between bots and sharing the user/banlists to control floods. These features of Eggdropbot allowed IRC operators to connect many bots together, thus making it a powerful weapon.

GTBots (Global Threat Bots) – GTBots appeared in late 2000 [6]. GTBot masters used tools such as Hide Window with their bots to hide the presence of their bots on the infected machines [21]. Some GTBots tried to spread on to a local network with the help of PsExec. GTBots are sometimes hooked on one of the system start up files or launched by a service. Once they start to work, they join the bot master IRC channel. The master starts to issue the commands and the bots respond to these commands. An example of GTBots is Backdoor IRC Aladinz. Most of the GTBots were used to target individual users. They could be also used to attack other IRC networks. The GTBots master can order his/her bots to flood the channels with a vast amount of garbage text, which consumes the server bandwidth and consequently, crashes the server.

SdBots – In the beginning of 2002, a new significant bot development was made by hosting the Sdbot [21]. Sdbot was a well-distributed bot and written in C++ which makes it a very powerful weapon for attackers. Additionally, Sdbot has attached its own IRC client within its executable. The early version of Sdbot installs itself onto a registry key so that it can be loaded on start-up. Sdbot has added an extra feature to existing bots by using an efficient port tunneling, file download and execution and finally a flexible C++ source base language.

AgoBots – Agobot has appeared in late 2002. Agobot incorporated most features and functionalities of Sdbot, but performed in a more sophisticated and vigorous manner than Sdbot [21]. In comparison to Sdbot, Agobot uses other capabilities such as exploits for network propagation, encrypting connections, and polymorphism. As more versions of Agobot were launched, the authors of Sdbot decided to upgrade their bot functionalities as well. In October 2003, a new version of Agobot was released. The new version of Agobot source included new functionalities which make it function similar to a worm but it requires a botmaster command and does not run automatically.

Spybots – Spybot is written in C and it affects MS-Windows system and has similar functionalities as Sdbot.Spybot functionalities include local file manipulation, key logging, process or system manipulation and remote command execution [21]. In addition, Spybot has the ability to perform different types of scanning such as horizontal scan and vertical scan, attacking NetBIOS open shares and establishing DDoS attacks. The previous bots targeted MS-Windows operating system. Other bots were implemented to target other operating systems. For example, Q8Bot and Kaiten are written for Unix/Linux OS. Similar to Windows bots, they implement all common bot features such as DDoS attack and executing arbitrary commands. Perl-based bot is also used on Unix-based system. Perl-based bot is simple to implement and contains few hundred lines of source code and mainly used to perform DDoS attack.

Peer-to-Peer bots – Many peer-to-peer bots have been implemented in the past. For example, Phatbot uses most of the functionalities implemented in Agobot but relatively communicates using WASTE technology instead of IRC protocols [10]. WASTE is an encrypted open-source peer-to-peer protocol for small networks. The author of Phatbot removes the encryption from the WASTE code to delete the process of distribution of public keys. In order to find other peers, Phatbot uses a Gnutella cache server. The Phatbot records itself with a list of URLs using GNUT, a Gnutella client. In order to communicate with other infected hosts, each host attaches to these cache servers to retrieve the list of Gnutella clients, but on a different port. In order to connect to the Phatbot WASTE network, a WASTE client is needed and connects to a peer found on the cache servers. One problem when using WASTE client is that the WASTE protocol is designed for small networks, therefore, the scalability issue is raised. Other peer-to-peer bots include Sinit and Nugache. The Sinit bot finds other infected hosts by sending special discovery packets on port53 using random IP address. Once the infected host receives this message, a connection between the two hosts is established and the bots start to interchange their peers. The Nugache malicious bot is the first bot to build a malicious P2P network. The communication between peers is encrypted. One of the most dangerous peer-to-peer bots is the Storm Bot. Peacomm uses a well-known peer-to-peer protocol called Overnet on different ports in order to communicate with other peers. The infected host sends and receives large number of UDP packets. The bot creates a list of other peers to communicate with it. Some of these peers are legitimate hosts which use Overnet clients.

3. Existing Methods For Detecting Botnets

Botnets can be identified through their connections to the Command and Control channel or their activity when given commands. The large amount of traffic involved in sending spam or performing a distributed denial of service attack makes their detection very easy. However, delicate behaviors such as key logging or click fraud are easier to mask as normal traffic. As detection methods are developed, botnets have changed their behavior. Many of them now use encryption to avoid having the content of their traffic from being recognized. When detectors started using the size of packets to identify bot traffic, they began adding random amount of padding to their packets. Even simple changes to a bot can make it very difficult to detect. The Nugache bot changed from using TCP port 8 for its C and C to randomly numbering the port on each host [20]. The botnet quickly became very difficult to find. Once a botnet is found, even simple analysis such as determining its size can be non-trivial. Changing DHCP addresses can make it seem larger, while NAT can make it seem smaller. This constant battle between those running botnets and those trying to destroy them has led to a variety of methods of detection. They include starting from simple methods such as honeypots or inspecting IRC and DNS traffic to complex methods of monitoring all traffic on a network. Once a botnet has been found, knowledge of its workings can allow researchers to collect information from within.

3.1 Honeypots

[17] Introduced a honeypot based botnet detection approach. A honeypot is a trap system designed to capture and analyze bots. The performance of the system and its traffic are watched to gain information about any bots that attempt to infect it. Comprehensive information on the contents, behavior, and protocols of a botnet can be found using honeypots. The details such as the location of the C and C server, structure of the botnet, and authentication details can also be gained. These honeypots can be very simple consisting of as little as a vulnerable computer and it is set behind a proxy to monitor its traffic. Merely connecting a system like this to the Internet can result in infection within

12 hours [16]. Honeypots are one of the initial methods of finding botnets. But, it still serves as a valuable early detection method for new bots. Honeypots that appear to be predominantly useful by having a stable IP address, being universally accessible, or having more computing power can become central nodes of a P2P botnet. In this way, many infected hosts may connect to the honeypot and even more data can be collected on them.

3.2 IRC tracking

Bots began on IRC channels, and it is still a common method of C and C [12]. This makes inspecting IRC traffic, one of the simplest and oldest methods of detecting bots. Either simple content comparison can be done to watch for known bots or more complex analysis such as comparing the total traffic from a host to its IRC traffic, or the speed of its responses can be done. If an IRC channel is known to be used for a botnet, researchers can connect to it themselves. If they understand the protocol for long enough for their client to appear to be a bot, large amounts of information can be gathered.

3.3 DNS tracking

Botnets often use DNS in exclusive ways [18]. Most ordinary users will perform DNS requests at irregular intervals and continuously over their sessions. Bots often perform requests in spouts, requesting large numbers of DNS entries at once and then demanding no more. The types of DNS entries used by C and C servers often differ from those of normal hosts. Once a particular infected host is found, more can be found by looking for similar DNS activity. And once specific domain names are identified as part of a botnet, explorations can be done to see where those names have been cached. On a survey of DNS servers, it was found that 11% of them contained entries for known botnet servers [18]. The complex update of DNS used by bots can actually be a drawback. If a botnet uses fast-flux, the TTL (Time To Live) information will need to be set very low. Researchers can use this indicator to find suspicious entries and then begin to continually recover the information. Often, many bootstrapping hosts can be found this way [19]. Because this behavior is expected for new bots, the researcher’s behavior will not notify the bot master. Botnets that frequently use new domain names can be a disadvantage as well. If the algorithm used to generate the names is cracked, then researchers can register new names before the bot master can. This allows them to capture most of the traffic envisioned for the bot master. Even if the protocols used are unfamiliar, information can still be gathered on the number and location of infected hosts and the original owner of the network are incapable of distributing new commands.

3.4 Traffic classification

If the traffic for an entire network can be surveyed, analysis can be performed to classify the traffic and search for botnets. Because this normally involves large amounts of data and is useful for other applications. The methods to do this are a significant research topic on their own. First, the amount of traffic to be examined is condensed. Simple analysis methods can separate out traffic unlikely to be bots. This can involve simple categorization based on port numbers, accumulated statistics, or even machine learning techniques [5]. Then with this reduced data set, more complex algorithms can be used. The simplest method is to perform fingerprinting, examining the contents of the traffic to that of known bots. More complex methods, such as fuzzy logic can also be applied in its place. A fuzzy logic system would extract features from the data to find characteristics common among botnets [5]. For example, if the C and C server is offline, then bots will generate a large number of failed DNS queries at common intervals. When bots search for their C and C using a list of IPs, they will similarly create a number of unsuccessful network connections. Finally, bots that receive the same command will completely receive traffic of the same size. Ones these features are extracted from the data; they can be grouped together and compared to known botnet traffic. Normally this method of detection results in possibilities, not definite yes or no answers, but can find previously unknown bots.

3.5 Anomaly Analysis

Anomaly analysis is parallel to traffic classification, but rather than trying to classify all traffic, distinctions from the rule are searched for; [7] introduced this approach. Usually bots act very differently than humans. Humans will have a habit to use a set of common sites, while bots tend to be spread at random. Humans have a tendency to do a small number of things at once, while bots will connect to a large variety of peers. These differences can be misused to separate out bots from humans. Rather than analyze all the traffic, samples are taken. These samples are grouped into traffic flows based on source and destination hosts and ports. This simple method will determine similar traffic patterns such as IRC, port scanning, and P2P activity. When behavior over time is considered, we can identify hosts that act persistently which is a key behavior of botnets. This approach is highly scalable because it searches for trends in traffic, not specific elements of traffic. Further analysis could be done on traffic found by this method to gain more specific information or to reduce the number of false positives.

3.6 Infiltration-Based Peer-Monitoring

Peer-to-peer botnets rely on the bots to perform routing and indexing of data [13]. When bots use known P2P protocols such as Distributed Hash Tables (DHT), monitoring nodes can be inserted into the network. With enough nodes spread throughout the key space, large portions of the botnet and its behavior can be determined by logging the incoming searches and data. With additional knowledge of the operations of the botnet, information about the monitoring nodes can be inserted into the network to increase the amount other nodes depend on them. However, this requires far more knowledge of the operation of the botnet and may expose the presence of the monitoring.

4. Proposed Methodology

These network-based detection systems can be used to detect botnets in either the infection phase or the control phase with favorable detection performance. However, as both botnets and the Internet are vigorously growing, these detection systems are facing two major challenges: the stealthiness of botnets' attacks and the huge volume of network traffic. First, in reaction to the intensive growth of law enforcement on punishing cybercrimes, bot masters instruct their botnets to perform attacks in an increasingly silent manner for evading detection. Such stealthy attacks stimulate little anomaly and thus become extremely hard to be observed in the network traffic. Recent study has confirmed that botnets have started to influence popular email services to send spams, instead of directly sending spams from bot-infected computers. To be specific, a bot master instructs his or her bots to sign in popular email service using stolen accounts and then send spams through the email service. Such malicious purposes, including signing in and then sending email to popular email service are very hard to be observed in the monitored networks. Subsequently, existing botnet detection systems may fail. As the population of networked computers and devices is huge and keeps blowing up, the volume of network traffic is high and grows fast. Such huge volume of traffic demands great scalability for network-based detection systems, which means that the detection systems need to process a large volume of network traffic efficiently. However, as most of existing botnet detection systems rely on Deep Packet Inspection (DPI) to analyze packet content, which is computationally expensive, their scalability is significantly controlled. As a consequence, when these detection systems are deployed in high-speed or high-volume networks, they may not be able to afford to perform detailed analysis for all network traffic related to bot-infected hosts and thus fail to detect bots. To summarize, existing network-based botnet detection systems may fail to detect botnets, which represent one of the most serious threats against Internet security. New network-based botnet detection systems are therefore needed to address these challenges. In our research work, we are concentrating on HTTP bots for the purpose of experimentation and observations. Generally HTTP Bot Traffic consists of:

- Automated Polling Mechanism
- Initiates with a GET request from the infected computer to its C and C server.
- The infected computer asks for a configuration file (config.bin) which will have commands for the bot. The command can be to sleep and check back after some time or to attack a particular victim or to post data to the C and C, etc.
- The C and C server replies with a HTTP/1.1 200 ok message along with the config file.
- The next traffic will depend upon the commands received by the bot. Bots find the C and C proxy by looking up the domain name instead of using static IP addresses.
- If there are multiple C and C, we can't find botnet by analyzing the IP address during their periodic communication. So the proposed method includes the following steps:

Step1: Capture HTTP packets from the network. Even though the address of the C and C server may vary, the packet sizes of bot traffic and the intended URI from the header field remains more or less similar in http botnet traffic packets.

Step 2: Analyze the timings of the arrival of the packets in each recognizable pattern and decide whether the packets are malicious or not. After analyzing the traffic data, we added rules into the database. Filtering is done based on Port number, Number of requests, IP address, MAC address and packet size. Advantages of this method are:

- No pre-defined signature is required.
- Payload analysis is not required.
- Clustering of packets does not depend upon the address of the C and C.
- DNS traffic analysis is not necessary.
- For each cluster of captured packets, the techniques discussed can be applied with successful results.
- It can be used to identify new http botnets.

5. Overall Architecture

In this section we will discuss the overall architecture of our J-Botnet detector program, and describe in detail the following components in J-Honey pot: packet capturing, packet injection, database, tcpdump, and log modules. Our J-Botnet detector tool is not a simple java program. It has the following features.

1) Incoming packets are recorded into SQL relational data base to allow faster searching and data persistence.
2) Record the incoming packets in to a tcp dump format file.
3) Has web based monitoring.

Fig 2 shows the overall architecture of our J-Botnet detector.

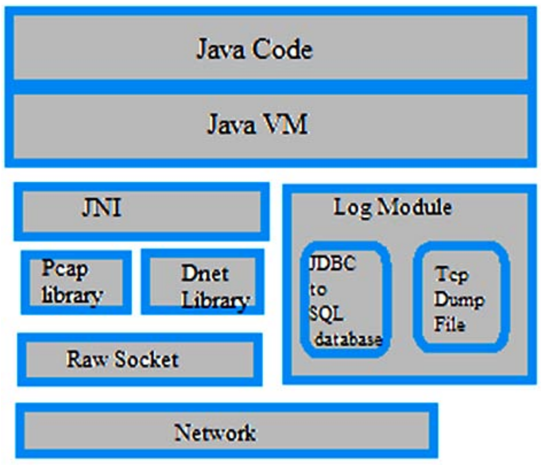


Figure 2: Overall architecture of J-Botnet detector

However, there are some features that are not included in our J-Botnet detector, for example like the C event library. This is because java does not offer any API that allows any asynchronous event notifications. For our purpose, we used JDK timer class to simulate timeout events.

5.1 Packet capture Module

Packet capturing function is accomplished via JPcap API. It provides a Java API to C packet library called pcap. But JPcap is not a complete conversion of popular C Pcap library. It provides the basic functionality called packet capturing. JPcap is currently under development and contains many bugs which we have debugged and fixed.

5.2 Packet Injection Module

The packet injection module is a Java Native Interface (JNI).This is necessary for injecting crafted IP and Ethernet packets which is essential to the deception part of our program. Since java does not provide such mechanism directly, JNI becomes the most logical mechanism.

5.3 Database Module

The module called SQL data base is added mainly for two reasons. First, it is easier to search for a particular packet or range of packets using SQL and only thing to do is to construct the correct SQL syntax. Second, the SQL data base allows different representation of data to be generated such as web GUI. The data base records all the packets that are received by J-Botnet detector program.

5.4 Tcpdump Module

The existing methods did not have a function that dumps the packets it receives into a tcpdump file. This is relatively easy to do and actually necessary for botnet detection. The tcpdump data contains real attack attempts which provide a valuable resource for us in designing a botnet detection engine. We observed that the tcpdump data collected through J-Botnet detector program only contain limited information about network connections. So that we could able to design an efficient system that can catch potential attacks with only limited amount of information available.

5.5 Log Module

The log module is based on the original event logging function in existing methods. It is expanded to include exception logging (SQL and Java run time), data base logging and tcpdump logging. The exception logs allow us to record any Java run-time exception messages, which are very helpful for debugging purposes.

5.6 Analysis of Data

The tcpdump data collected from J-Botnet detector were analyzed for possible attacks, scans and viruses. Based on this tcpdump data collected from traffic data, new rules were manually added into the database in order to capture such attacks.

5.7 Monitoring Interface

The existing botnet detection methods lack a graphical interface that allows user to visualize the events that occurred in the network. It would be essential to have a GUI interface that can display current network condition so that the network administrator can better control and understand their network. The interfaces provide easy accessibility for users.

5.8 Web Interface

The web interface can run independently without much user configuration. The independence came from data base module that we described earlier. Since past events were recorded in SQL data base, the web GUI can analyze events without having to interfere with normal operations of the program. The web GUI is built using java technology.

The GUI has the following features:

1) Ability to display packet information from the SQL database.
2) Ability to display real time network traffic from data base data, as well as historical traffic statistics.
3) It provides a real time configuration interface
4) Configuration of alerts message notification.

6. Experimental Results

We set up traffic monitors to work at the network of the Indian Space Research Organization(ISRO).Once the behavior of a botnet was understood, our goal is to generate a rule based detection method on the traffic data. We used Wireshark to inspect data from live network traffic which is shown in Fig.3.

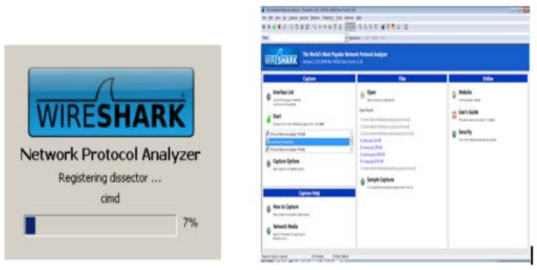
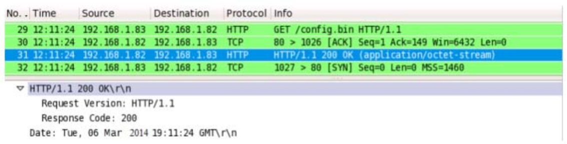


Figure 3: Wireshark welcome Screen and Protocol Analyzer Screen Shot

It has the ability to track real time network traffic and its characteristics. Alternatively we could download pcap files and open them using wireshark to analyze it. After performing the data analysis we can get an idea about various protocols, botnet topologies botnet behaviors and life cycle. After some general observation, we identified some unique characteristics of the traffic data in the infected host after infection and based on that we generated rules into the data base. It can verify with these rules on future traffic data and can easily give alert message on bad traffic. Fig 4 shows the bot traffic data detected using wire shark.

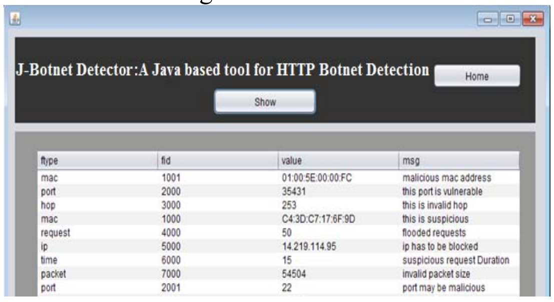


```
No. . Time      Source         Destination    Protocol  Info
   29 12:11:24  192.168.1.83   192.168.1.82   HTTP      GET /config.bin HTTP/1.1
   30 12:11:24  192.168.1.82   192.168.1.83   TCP       80 > 1026 [ACK] Seq=1 Ack=149 Win=6432 Len=0
   31 12:11:24  192.168.1.82   192.168.1.83   HTTP      HTTP/1.1 200 OK (application/octet-stream)
   32 12:11:24  192.168.1.83   192.168.1.82   TCP       1027 > 80 [SYN] Seq=0 Len=0 MSS=1460

 ▽ HTTP/1.1 200 OK\r\n
     Request Version: HTTP/1.1
     Response Code: 200
   Date: Tue, 06 Mar  2014 19:11:24 GMT\r\n
```

Figure 4: Bot traffic detected using Wire shark

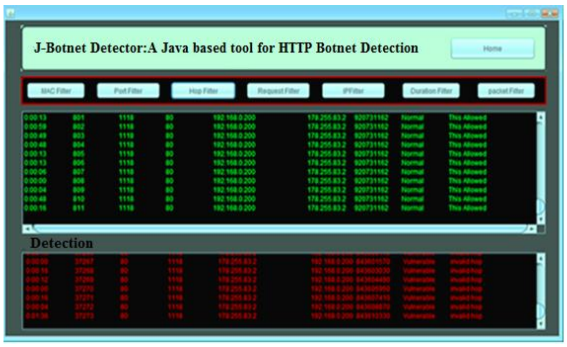
We filter the traffic data by adding rules into the database. That is filtering is done based on MAC address, Port number, Number of requests, IP address and packet size. Since rules can be updated manually according to our needs so that our botnet detection tool can detect new upcoming botnet attacks also. The new rules along with the existing rules in the data base can handle most of the cases of Botnet detection. Some of the rules that we have added to our database is shown in Fig 5.


J-Botnet Detector:A Java based tool for HTTP Botnet Detection
Home
Show

ftype	fid	value	msg
mac	1001	01:00:5E:00:00:FC	malicious mac address
port	2000	35431	this port is vulnerable
hop	3000	253	this is invalid hop
mac	1000	C4:3D:C7:17:6F:9D	this is suspicious
request	4000	50	flooded requests
ip	5000	14.219.114.95	ip has to be blocked
time	6000	15	suspicious request Duration
packet	7000	54504	invalid packet size
port	2001	22	port may be malicious

Figure 5: Rules hired inside the Database for detecting Attacks

When the tool runs according to the rule, there are more than 50 bots detected within one week. The alerts generated with better user interface contains attacks detected based on ports being scanned, malicious MAC address, invalid packet size and hop, flooded number of requests, etc. Fig 6 shows the alerts generated by our tool.



```
0:00:13  801  1118  80  192.168.0.200  178.255.83.2  920731162  Normal  This Allowed
0:00:59  802  1118  80  192.168.0.200  178.255.83.2  920731162  Normal  This Allowed
0:00:49  803  1118  80  192.168.0.200  178.255.83.2  920731162  Normal  This Allowed
0:00:48  804  1118  80  192.168.0.200  178.255.83.2  920731162  Normal  This Allowed
0:00:13  805  1118  80  192.168.0.200  178.255.83.2  920731162  Normal  This Allowed
0:00:13  806  1118  80  192.168.0.200  178.255.83.2  920731162  Normal  This Allowed
0:00:06  807  1118  80  192.168.0.200  178.255.83.2  920731162  Normal  This Allowed
0:00:00  808  1118  80  192.168.0.200  178.255.83.2  920731162  Normal  This Allowed
0:00:04  809  1118  80  192.168.0.200  178.255.83.2  920731162  Normal  This Allowed
0:00:48  810  1118  80  192.168.0.200  178.255.83.2  920731162  Normal  This Allowed
0:00:16  811  1118  80  192.168.0.200  178.255.83.2  920731162  Normal  This Allowed
```

```
0:00:00  37267  80  1118  178.255.83.2  192.168.0.200  843601570    Vulnerable  invalid hop
0:00:16  37268  80  1118  178.255.83.2  192.168.0.200  [illegible]  Vulnerable  invalid hop
0:00:12  37269  80  1118  178.255.83.2  192.168.0.200  [illegible]  Vulnerable  invalid hop
0:00:06  37270  80  1118  178.255.83.2  192.168.0.200  [illegible]  Vulnerable  invalid hop
0:00:16  37271  80  1118  178.255.83.2  192.168.0.200  [illegible]  Vulnerable  invalid hop
0:00:04  37272  80  1118  178.255.83.2  192.168.0.200  [illegible]  Vulnerable  invalid hop
0:01:36  37273  80  1118  178.255.83.2  192.168.0.200  [illegible]  Vulnerable  invalid hop
```

Figure 6: Alerts generated by J-Botnet Detector

7. Conclusion

Botnets are a form of malware that gives third parties control of infected computers generally for malicious purposes. They are different from other malware because of their use of a command and control channel to receive orders. Because of their variety and desire to avoid detection, botnets are difficult to detect. In this paper, we are proposing a new tool called J-Botnet detector, a Java based network tool with web-based monitoring and rule-based intrusion detection capability. We have interfaced it with SQL database and developed a rich set of logging functionalities and provided a convenient GUI for users to visualize the results. The results or alerts generated using our tool showed significant performance enhancement as compared with the existing engines. We implemented this tool on various networks. It has the ability to detect botnet attacks in user's computers who use the Internet and it exhibited remarkable performance. So far the prevailing Botnet detection engines only works well for existing vulnerabilities but not for the future vulnerabilities. We are planning to enhance this J-Botnet Detector engine, so that rules can be updated automatically as well as periodically.

8. Acknowledgement

We offer our humble gratitude at the lotus feet of Sri Sat guru Mata Amritanandamayi Amma, our spiritual mother who is the guiding light in our lives. We make use of this opportunity to express our sincere gratitude to all those who have helped me to complete this work successfully. We would like to express our sincere gratitude and thanks to the entire NISD team at ISRO, Thiruvananthapuram, for their continuous encouragement and supervision for the completion of this work. We also would like to extend our thanks to everyone in Department of Computer Science and Engineering, Amrita Vishwa Vidyapeetham, and other colleagues for their insightful comments and constructive suggestions to improve the quality of this work. Last but not the least; we would like to express our sincere thanks to our friends for their valuable suggestions and helpful discussions.

References

[1] Ahamadkarim, Rosli bin salleh, Muhammad shiraz, Syed Adeel Ali shah, Irfanawan, NorBadrulAnur, "Botnet detection techniques:review, future trends and issues, " JournalofZhejiangUniversitySCIENCE(Computer and Electronics), January 23, 2014.
[2] RoshnaR.S, VinodhEwards, "Botnet Detection Using Adaptive Neuro Fuzzy Inference System, " International Journal of EngineeringResearchandApplications(IJERA)Vol. 3, Issue 2, March -April 2013, pp.1440-1445.
[3] Dr.Laheeb M.Ibrahim, Karam Hatim Thanoon, "Detection of Zeus Botnet in Computers Networks and Internet, " International Journal of Information Technology and Business Management, 29th October 2012, Vol.6 No.1
[4] Ravi Kishore Sharma, Gajendra Singh Chandel, " Botnet detection and resolution challenges: a survey paper, " International Journal of Computer, Information Technology & Bioinformatics (IJCITB), June 2012 ISSN:, Volume-1, Issue-1
[5] S.S Garasia, D.P.Rana, R.G.Mehta, "Http Botnet detection using frequent pattern set mining, "International Journal of Engineering Science and Advanced Technology, Volume-2, Issue-3, 619-624, May-June 2012.
[6] Bhumika, Viveksharma, "Design and implementation of Honeyd to simulate virtual Honeypots, " IOSR Journal of Computer Engineering, Volume 3, Issue 1, July-Aug.2012, PP 28-34.
[7] Haritha.S.Nair, VinodhEwards S E, "A Study on Botnet Detection Techniques, " International Journal of Scientific and Research Publications, Volume 2, Issue 4, April 2012 1 ISSN 2250-3153.
[8] SajjadArshad, MaghsoudAbbaspour, Mehdi Kharrazi, HoomanSanatkar, " An Anomaly-based Botnet Detection Approach for Identifying Stealthy Botnetsm, " International Conference on Computer Applications and Industrial Electronics (ICCAIE 2011), IEEE, 2011.
[9] SajjadArshad, MaghsoudAbbaspour, Mehdi Kharrazi, HoomanSanatkar, "An Anomaly based Botnet

Detection approach for identifying stealthy Botnts, " IEEE, 2011.

[10] Chia-Mei Chen, Ya-HuiOu, Yu-Chou Tsai, " Web Botnet Detection based on Flow Information", IEEE, 2010

[11] Hossein Rouhani Zeidanloo, AzizahBt Abdul Manaf, "Botnet Detection by Monitoring Similar Communication Patterns, " (IJCSIS) International Journal of Computer Science and Information Security, Vol. 7, No. 3, 2010.

[12] Hossein Rouhani Zeidanloo, Azizah Bt Manaf, Payam Vahdani, FarzanehTabatabaei, Mazdak Zamani, "Botnet detection based on traffic monitoring" , Centre for Advanced Software Engineering University of Technology Malaysia in International Conference on Networking and Information Technology, 2010.

[13] Zhenqi Wang, Li Fu, "The research of detecting IRC botnet based on k-means algorithms, "Second International Conference on Communication Systems, Networks and Applications, 2010.

[14] Narn-YihLee, Hung-Jen Chiang , " The research of botnet detection and prevention, " IEEE, 2010.

[15] Miroslaw Szymczyk, " Detecting Botnets in Computer Networks Using Multi-AgentTechnology, "2009Fourth International Conference on Dependability of Computer System.

[16] Yu Yao, Jun-Wei Lv, Fu-xiang Gao, GeYu, Qing-xuDeng, "Detectingand defending against worm attacks using Bot-honeynet, " Second International Symposium on Electronic Commerce and Security, 2009.

[17] Yu Yao, Jun-weiLv, Fu-xiangGao, Ge Yu, Qing-xu Deng, "Detecting and Defending against Worm Attacks Using Bot-honeynet, " 2009 Second International Symposium on Electronic Commerce and Security, IEEE, 2009.

[18] Chao-His Yeh, Chung-Huang Yang, "Design and implementation of Honeypot systems based on Open-source software, " IEEE, 2008

[19] Hyunsang Choi, Hanwoo Lee, Heejo Lee, Hyogon Kim, "Botnet detection by monitoring group activities in DNS traffic, " SeventhInternationalconferenceonComputerandInformation Technology, IEEE, 2007

[20] CliffiC. Zou Ryan Cunningham, "Honeypot-Aware Advanced Botnet Construction and Maintenance, " 2006 International Conference on Dependable Systems and Networks (DSN'06), IEEE, 2006.

[21] John Canavan, "The Evolution of Malicious IRC Bots, " White paper: Symantec Security, Proceedings of the VB2005 Conference

Author Profile

Thejiya V received the B.Tech degree in Computer Science & Engineering from Amrita School of Engineering, India under Amrita University in 2012 and M.Tech degree in Computer Science & Engineering from the same University and College. Her current research interest is in Network Security.

N Radhika has done her Post Doctorate in the Department of Computer Science and Engineering, Indian Institute of Technology, Madras, in the domain of Wireless Sensor Networks (Smart grids), and her Doctorate from Government College of Technology, Coimbatore in the area of Mobile Adhoc Network. She is currently working as Professor in the Department of Computer Science and Engineering, Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore and also holds the position of Academic Co-ordinator at the School of Engineering, Coimbatore Campus.

B Thanudhas is currently working as a Group Director at VSSc/ISRO. He is currently doing research in network security and wireless networks.