International Journal of Computer and Electrical Engineering, Vol.2, No.6, December, 2010
1793-8163

New Approach for Detection of IRC and P2P Botnets

Hossein Rouhani Zeidanloo, *Member, IACSIT,* Farhoud Hosseinpour and Farhood Farid Etemad

***Abstract*— Botnet is most widespread and occurs commonly in today's cyber attacks, resulting in serious threats to our network assets and organization's properties. Botnets are collections of compromised computers (Bots) which are remotely controlled by its originator (BotMaster) under a common Commond-and-Control (C&C) infrastructure. In this paper, we proposed a new general detection framework which currently focuses on P2P based and IRC based Botnets. Since Artificial Immune System (AIS) is a new bio-inspired model which is applied for solving various problems in the field of information security, we used this concept in our proposed framework to make it more efficient. Our framework in P2P part is based on definition of Botnets. Botnet has been defined as a group of bots that perform similar communication and malicious activity patterns within the same Botnet. We utilized AIS to effectively detect malicious activities in P2P part. Our framework in IRC part is based on calculating Delay Time (Td) which is a time frame between sending IRC NICK command and IRC JOIN command. The point that distinguishes our proposed detection framework from many other similar works is that there is no need for prior knowledge of Botnets such as Botnet signature.**

***Index Terms*—AIS; Botnet; Bot; IRC; P2P**

I. Introduction

Nowadays, the most serious manifestation of advanced malware is Botnet. To make distinction between Botnet and other kinds of malware, we have to comprehend the concept of Botnet. For a better understanding of Botnet, we have to know two terms first, Bot and BotMaster and then we can properly define Botnet. *Bot* – Bot is a new type of malware installed into a compromised computer which can be controlled remotely by BotMaster for executing some orders through the received commands. After the Bot code has been installed into the compromised computers, the computer becomes a Bot. Contrary to existing malware such as virus and worm which their main activities focus on attacking the infecting host, bots can receive commands from BotMaster and are used in distributed attack platform. *BotMaster* – BotMaster is also known as BotHerder, is a person or a group of person which control remote Bots. *Botnets*- Botnets are networks consisting of large number of Bots [5].

Bots usually distribute themselves across the Internet by looking for vulnerable and unprotected computers to infect. When they find an unprotected computer, they infect it and then send a report to the BotMaster. The Bot stay hidden until they are informed by their BotMaster to perform an attack or task. Other ways in which attackers use to infect a computer in the Internet with Bot include sending email and using malicious websites. Based on our understanding, we could say that the activities associated with Botnet can be classified into three parts: (1) *Searching* – searching for vulnerable and unprotected computers. (2) *Distribution* – the Bot code is distributed to the computers (targets), so the targets become Bots. (3) *sign-on* – the Bots connect to BotMaster and become ready to receive command and control traffic. [5]

The difference that this framework has with previous works that Hossein *et al.* proposed in [1, 21] is the using of Artificial Immune System (AIS) in one of the components in P2P part and also proposing a novel approach for detection of IRC bots which is based on calculating Delay time(Td) which is a time frame between sending IRC NICK command and IRC JOIN command. Artificial Immune System (AIS) is a new bio-inspired model which is applied for solving various problems in the field of information security. AIS is defined as [3] "Adaptive systems, inspired by theoretical immunology and observed immune functions, principles and models, which are applied to problem solving." akin to other bio-inspired model such as genetic algorithms, neural networks, evolutionary algorithms and swarm intelligence [20]. According to Gomez [24], AIS has two main characteristics:

1. Immune Memory (Self adapting): Biological immune system [24] acclimatize to pattern of some known protein structure go with non-self, and remember it for the upcoming communications. The ability of memorizing assists the immune system to identify the pathogens in future interactions to quickly take essential action to eradicate the pathogen. Thus the biological immune system acts swiftly and proficiently against previously identified pathogens. This technique of detection is referred to as memory based detection. In the case that immune system comes across pathogens which have same structure as the memory cells, the secondary response is commenced which in contrast with primary response is more effective and efficient [25].

2. Self/non-Self Discrimination: The artificial immune system has the capability to differentiate between the self space (the cells which are owned by the system) and non-self space (foreign entities to the system) which is obtained by T-cells. Negative Selection algorithm is proposed by Forrest [6] which presents a framework to discriminate between self and non-self entities. In this algorithm, at first a set of detectors are produced and then they are compared with a set of normal sets (self), to make sure that non of detectors are

Manuscript received Jun 30, 2010.
Hossein Rouhani Zeidanloo, Farhoud Hosseinpour and Farhood Farid Etemad, Faculty of Computer Science and Information System, University of Technology Malaysia(UTM), Kuala Lumpur, Malaysia. (Email: H_Rouhani@hotmail.com, F.Hosseinpour@gmail.com, Fa.Faridetemad@gmail.com .

not reactive to self data. If any of detectors are matched with any self entity the system will eliminate them and the remaining will be kept.

As a substitute to self/non-self discrimination, Danger model is proposed by Matzinger [28, 26]. According to this hypothesis the main cause of an immune response is that a photogene harms the system and it is thus dangerous and not because of it is unknown to the system. The Danger Model works on the premise that the main director of the immune system is the body's tissues and not the immune cells. The chemical danger signals are released by the distressed tissues to rouse the immune response whereas the calming or self signals are released by healthy tissues which provide the tolerance for the immune system [23].

AIS is inspired from human immune system (HIS) which is a system of structures in human body that recognize the foreign pathogens and cells from human body cells and protect the body against those disease [2].HIS is very complicated and tries to detect and remove any disease. According to Forrest et al. [7] "HIS is diverse, distributed, error tolerant, dynamic, self-monitoring (or self-aware) and adaptable." Like HIS which protects the human body against the foreign pathogens, the AIS suggest a multilayered protection structure [4] for protecting the computer networks against the attacks. Consequently it has been focused by network security researchers to utilize and optimize it in IDS. In this work we have utilized AIS techniques to effectively detect the malicious activities such as spamming and scanning in bot infected networks.

The rest of the paper is organized as follows. In Section 2, we analyze different Botnet communication topologies. In Section 3, we review the related work. In Section 4, we describe our proposed detection framework and all its components and finally conclude in section 5.

II. Botnet Communication Topologies

According to the Command-and-Control(C&C) channel, we categorized Botnet topologies into two different models, the Centralized model and the Decentralized model.

A. *Centralized model*

In this model, one central point is in charge for exchanging commands and data between the BotMaster and Bots. In this model, BotMaster chooses a host (usually high bandwidth computer) to be the central point (Command-and-Control) server of all the Bots. The C&C server runs certain network services such as IRC or HTTP. The main advantage of this model is small message latency which cause BotMaster easily arranges Botnet and launch attacks. Since all connections happen through the C&C server, therefore, the C&C is a critical point in this model. In other words, C&C server is the weak point in this model. If somebody manages to discover and eliminates the C&C server, the entire Botnet will be useless and ineffective. Thus, it becomes the main negative aspect of this model. IRC and HTTP are two common protocols that C&C server uses for communication in this model. Figure 1 shows the basic communication architecture for a Centralized model. There are two central points that forward commands and data between the BotMaster and his Bots.

B. *Decentralized Model*

Due to main disadvantage of Centralized model attackers started to build alternative Botnet communication system that is much harder to discover and to destroy. Hence, they decided to find a model in which the communication system does not completely depending on only some selected servers and even discovering and destroying a number of Bots. As a result, attackers exploit the idea of Peer-to-Peer (P2P) communication as a Command-and-Control(C&C) pattern which is more resilient to failure in the network. The P2P based C&C model will be used significantly in Botnets in the near future, and definitely Botnets that use P2P based C&C model impose much bigger challenge for defense of networks. Since P2P based communication is more robust than Centralized C&C communication, more Botnets will move to use P2P protocol for their communication.

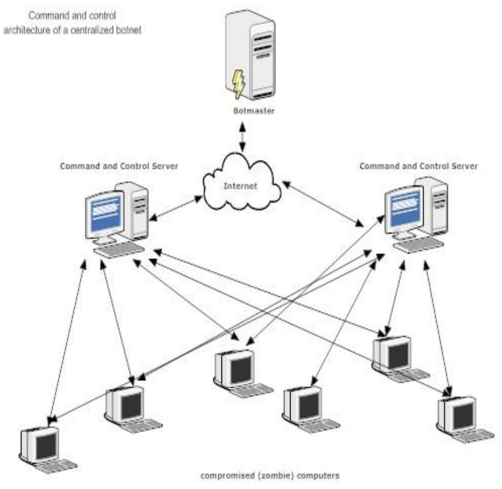

Command and control
architecture of a centralized botnet
Botmaster
Command and Control Server
Internet
Command and Control Server
compromised (zombie) computers

Figure 1. Command and control architecture of a Centralized model

In the P2P model, as shown in Figure 2, there is no Centralized point for communication. Each Bot keeps some connections to the other Bots of the Botnet. Bots act as both Clients and servers. A new Bot must know some addresses of the Botnet to connect there. If Bots in the Botnet are taken offline, the Botnet can still continue to operate under the control of BotMaster. P2P Botnets aim at removing or hiding the central point of failure which is the main weakness and vulnerability of Centralized model.

Some P2P Botnets operate to a certain extent decentralized and some completely decentralized. Those Botnets that are completely decentralized allow a BotMaster to inject a command into any Bots, and have it either be broadcasted to a specified node. Since P2P Botnets usually allow commands to be injected at any node in the network, the authentication of commands become essential to prevent other nodes from injecting incorrect commands.

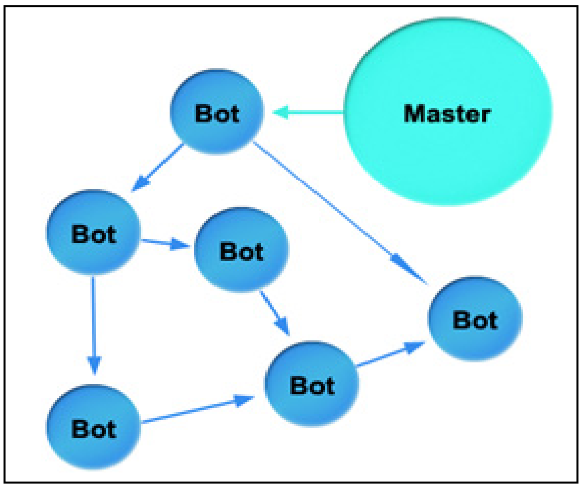


Figure 2. Example of Peer-to-peer Botnet Architecture

III. Related work

Different approaches have been proposed for detection of Botnet. There are essentially two approaches for botnt detection. One approach is based on locating honeynets in the network. And another approach is Intrusion Detection Systems (IDS).[22]

A. Honeypots

Honeypot in security community refers to a computer system that is used as a trap to draw the attention of attackers to attack this computer system. All Honeypots have a unique concept. They are computer systems that don't have any production value. According to this concept, all interactions between honeypot and other systems are suspicious and must be investigated. For example, you may put a honeypot web server in the DMZ of your network. Because it is a honeypot and does not have any productive activities, any interaction with this web server might be a malicious activity or an unauthorized access.

This technique is very effective for collecting and tracking Botnets which provides a full analyzing and examination of bot behaviors as well as bot binary for researchers. The useful information that honeypot can gather are (i) signature of bots for content-based detection (ii) information of Botnet C&C mechanism/servers (iii) unknown security holes that enable bots to penetrate the network (iv) tools and techniques that attacker use (v) the motivation of attacker.

There are many papers discussed how to apply honeynets for Botnet detection [29,30,31]. For example, Nepenthe [32] is a low-interaction honeypot that simulate some vulnerability and provides some features for collection of malware binaries. Freiling et al. [33], used honeypot to track Botnets in the network and generate an early report for understanding the consequences of Botnets.

B. Intrusion Detection System(IDS)

We can categorize this part into two groups which are signature-based Botnet detection and anomaly-based Botnet detection and then consider the current and most well-known Botnet detection techniques in each group

1) Signature-based Botnet detection: Signature-based Botnet detection technique uses the signatures of current Botnets for its detection. For instance, Snort [34] is capable to monitor network traffic to find signature of existing bots. This method has several advantages, such as immediate detection and impossibility of false positives. But Signature-based detection approach is only capable to be used for detection of well-known Botnets. Consequently, this solution is not functional for unknown bots. It means that zero-day bots attacks cannot be detected. More important, very similar bots with slightly different signature may be missed of detection.

One of the well-known signature-based Botnet detection techniques is Rishi [35] that matches known nick-name patterns of IRC bots. Rishi is primarily based on passive traffic monitoring for suspicious IRC nicknames, IRC servers, and uncommon server ports. They use n-gram analysis and a scoring system to detect bots that use uncommon communication channels, which are commonly not detected by classical intrusion detection systems [35]. The disadvantages of this method are that it cannot detect encrypted communication as well as non-IRC Botnets. Moreover, this method is unable to detect bots without using known nickname patterns.

2) Anomaly-based Botnet detection: Anomaly-based detection approaches try to detect Botnets based on a number of network traffic anomalies such as high network latency, high volumes of traffic, traffic on unusual ports, and unusual system behavior that could show existence of bots in the network [36]. Nevertheless this technique meets the problem of detecting unknown Botnets.

In 2005, Dagon [37] proposed a method to discover Botnet C&C servers by detecting domain names with unusually high or temporally intense DDNS query rates. This method is similar to the approach proposed by Kristoff [38] in 2004.

In 2007, Choi *et al.* [39] suggested anomaly mechanism by monitoring group activities in DNS traffics. They defined some special features of DNS traffics to differentiate valid DNS queries from Botnet DNS queries. This method is more efficient than the prior approaches and can detect Botnet despite the type of bot by looking at their group activities in DNS traffic [39].

Strayer *et al.* [40] proposed a network-based approach for detecting Botnet traffic which used two step processes including separation of IRC flows at first, and then discover Botnet C&C traffic from normal IRC flows [40]. This technique is specific to IRC based Botnets.

Masud *et al.* [41] proposed effective flow-based Botnet traffic detection by mining multiple log files. They proposed several log correlation for C&C traffic detection. They categorize an entire flow to identify Botnet C&C traffic. This method can detect non-IRC Botnets[41].

Botminer [42] is the most recent approach which applies data mining techniques for Botnet C&C traffic detection. Botminer is an improvement of Botsniffer. It clusters similar communication traffic and similar malicious traffic. Then, it performs cross cluster correlation to identify the hosts that share both similar communication patterns and similar malicious activity patterns. Botminer is an advanced Botnet detection tool which is independent of Botnet protocol and structure. Botminer can detect real-world Botnets including IRC-based, HTTP-based, and P2P Botnets with a very low false positive rate [42].

In BotHunter [43], bot infection pattern are modeled to use for recognizing the whole process of infection of Botnet in the network. All behavior that occur the bot infection such as target scanning, C&C establishment, binary downloading and outbound propagation have to model by this method. This method gathers an evidence-trail of connected infection process for each internal machine and then tries to look for a threshold combination of sequences that will convince the condition for bot infection. The BotHunter uses snort with adding two anomaly-detection components to it that are SLADE (Statistical payLoad Anomaly Detection Engine) and SCADE (Statistical scan Anomaly Detection Engine).

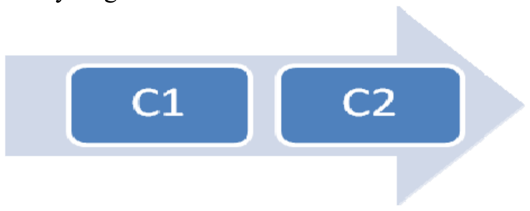
IV. Proposed Botnet Detection Framework and Components

Our proposed framework is based on passively monitoring network traffics which can be categorized as IDS approach. Our framework in P2P part is based on definition of Botnets. Botnet has been defined as a group of bots that perform similar communication and malicious activity patterns within the same Botnet. Our framework in IRC part is based on calculating Delay time(Td) which is a time frame between sending IRC NICK command and IRC JOIN command. We share similar idea for definition of botnet as proposed by Gu *et al.* in Botminer[8].

Figure 3 shows the architecture of our proposed botnet detection system, which consists of 5 main components: Filtering, Application Classifier, Traffic Monitoring, Malicious Activity Detector Using AIS, Delay Time Analyzer and Analyzer. Filtering is responsible to filter out unrelated traffic flows. The main benefit of this stage is reducing the traffic workload and makes application classifier process more efficient. Application classifier is responsible for separating IRC and HTTP traffics from the rest of traffics. Malicious activity detector Using AIS is responsible to analyze the traffics carefully and try to detect malicious activities that internal host may perform and separate those hosts and send to next stage. Traffic Monitoring is responsible to detect the group of hosts that have similar behavior and communication pattern by inspecting network traffics. Delay Time analyzer is responsible to detect IRC bots inside the network. Analyzer is responsible for comparing the results of previous parts (Traffic Monitoring and Malicious Activity Detector) and finding hosts that are common on the results of both lists.

A. Filtering

The main objective of Filtering is to reduce the traffic workload and makes the rest of the system perform more efficiently. Figure 4 shows the architecture of the filtering.



In C1, we filter out those traffics which targets (destination IP address) are recognized servers and will unlikely host Botnet C&C servers. For this purpose we used the top 500 websites on the web (Http://www.alexa.com/topsites), which the top 3 are google.com, facebook.com and yahoo.com.

In C2, we filter out handshaking processes (connection establishments) that are not completely established. Handshaking is an automated process of negotiation that dynamically sets parameters of a communications channel established between two entities before normal communication over the channel begins. It follows the physical establishment of the channel and precedes normal information transfer [44]. To establish a connection, TCP uses a three-way handshake; in this case we filter out the traffics that TCP handshaking have not completed. Like a host sends SYN packets without completing the TCP handshake. Based on our experience these flows are mostly caused by scanning activities.

B. Application classifier

Application Classifier is responsible to separate IRC and HTTP traffics from the rest of traffics and send them to Monitoring & Clustering and HTTP component. For detecting IRC traffics we can inspect the contents of each packet and try to match the data against a set of user defined strings. For this purpose we use payload inspection that only inspects the first few bytes of the payload and looking for specific strings. These IRC specific strings are NICK for the client's nickname, PASS for a password, USER for the username, JOIN for joining a channel, OPER that says a regular user wants to become a channel operator and PRIVMSG that says the message is a private message [45]. Using this strategy for detecting IRC traffic is almost simple for most network intrusion detection software like Snort. In some cases botmasters are using encryption for securing their communication that make using packet content analysis strategy useless. This issue actually is not our target here.

In next step, we also have to separate Http traffics and send to Centralized part. For this purpose we also can inspect the first few bytes of Http request and if it has certain pattern or strings, separate it and send it to centralized part. For detecting Http traffics we focus on concept of Http protocol. Like most network protocols, HTTP uses the client-server model: An HTTP client opens a connection and sends a *request message* to an HTTP server (e.g. "Get me the file 'home.html'"); the server then returns a *response message*, usually containing the resource that was requested("Here's the file", followed by the file itself). After delivering the response, the server closes the connection (making HTTP a *stateless* protocol, i.e. not maintaining any connection information between transactions).[46]

In the format of Http request message, we are focusing on Http methods. Three common Http methods are "GET", "HEAD", or "POST": [47]

- A GET is the most common Http method; it says "give me this resource"
- A HEAD request is similar to GET request, except it asks the server to return the response headers only, and not the actual resource (i.e. no message body). This is helpful to consider characteristics of resources without downloading it which can help in

saving bandwidth. We use HEAD when no need for a file's contents.

- A POST request is used to send data to the server to be processed in some way, like by a CGI script. A POST request is different from a GET request in the following ways:
    - There's a block of data sent with the request, in the message body. There are usually extra headers to describe this message body.
    - The *request URI* is not a resource to retrieve; it's usually a program to handle the data you're sending.
    - The HTTP response is normally program output, not a static file.

Therefore we inspect the traffics and if the first few bytes of an Http request contain "GET", "POST" or "HEAP", it's the indication of Http protocol and will separate those flows and send them to Centralized part. After filtering out Http and IRC traffics, the remaining traffics that have the probability of containing P2P traffics are send to Traffic Monitoring part and Malicious Activity Detector. However in parallel we can use other approaches for identifying P2P traffics. We have to take into consideration that P2P traffic is one of the most challenging application types. Identifying P2P traffic is difficult both because of the large number of proprietary p2p protocols, and also due to the deliberate use of random port numbers for communication. Payload-based classification approaches tailored to p2p traffic have been presented in [49, 48], while identification of p2p traffic through transport layer characteristics is proposed in [47]. Our suggestion for using specific application or tools for identifying P2P traffics other than sending remaining traffics is use of BLINC [50] that can identify general P2P traffics. In contrast to previous methods, BLINC is based on observing and identifying patterns of host behavior at the transport layer. BLINC investigates these patterns at three levels of increasing detail (i) the social, (ii) the functional and (iii) the application level. This approach has two important features. First, it operates in the dark, having (a) no access to packet payload, (b) no knowledge of port numbers and (c) no additional information other than what current flow collectors provide.[50]

C. *Delay Time Analyzer*

Since IRC bots were the early sample of bots that have emerged and drew the attention of computer security specialists to the threat of botnet phenomenon, many detection techniques have been developed and installed to stop and filter IRC malicious traffic in different networks. Due to this fact, attackers started to craft their codes to complicated ones which are hard to detect. To meet their requirements they started to use IRC encrypted traffic and also manipulate IRC protocol to bypass signature based intrusion detection approaches. In this part we propose a novel approaches for detection of hosts that have been joined to IRC bots. We put our approach in a component called Delay Time analyzer.

This step includes analyzing traffic to detect flows that most probably belong to botnet communication. To reach our target, we need to capture some specifications of each flow and store them in a database for further processing. We are using Audit Record Generation and Utilization System (ARGUS) to gather information regarding network flows. The following parameters is needed to obtain from each flow: Source IP(SIP) address, Destination IP(DIP) address, Source Port(SPORT), Destination Port(DPORT) and Duration. We store all of the mentioned parameters in a database similar to Table 1.

We are managing to exclude normal IRC traffics from possible IRC botnet traffics in this part. First step of our solution is based on separating those flows in our database that have same destination IP addresses which we believe are the potential IRC bots.

IRC bots try to connect to IRC server and join a channel in the first step of their life cycle to get further commands from the botmaster. For each communication flow we define a *delay time* (Td) parameter. Delay time(Td) is a time frame between sending IRC NICK command and JOIN command which indicates the time which a host delays between authenticating itself to the IRC server and joining a channel. For detecting NICK and JOIN commands in traffics, we can inspect the contents of each packet and try to match the data against JOIN and NICK commands. For this purpose we use payload inspection that only inspects the first few bytes of the payload and looking for NICK and JOIN commands. We can use a simple timer which can record the time that NICK and JOIN command has been detected for each flow and can be calculated based on this formula:

$$Td=T(nick) - T(join)$$

Since bots are automated programs that are immediately joining channel after authentication; therefore, Td is small for bots that connecting to their botmaster. We have to take into consideration that human users of IRC manually are joining to the channels using command line of graphical user interface which leads to takes much time than automated joining.

Based on experiments in our lab, we found out that normal IRC communication between users and IRC servers have almost four times more delay time (Td) in comparison to IRC bots. However, it could be various depend on characteristics of network (e.g. network speed, equipment). Therefore, we are using an average for delay time (Td) of flows based on this formula:

$$Td(av)= \Sigma Tdi\ (i=1 \text{ to } N)/N$$

Consequently, if delay time(Td) for each network flows is bigger than four times of average delay time(Td(av)) we can claim that it is an indication of normal IRC communication and omit those flows from our database.

$$Tdi(i=1\ldots n) > 4 * Td(av)$$

Thus, the remaining flows in database is belong to IRC bots communication which referring to specific hosts inside our network.

D. *Traffic monitoring*

Traffic Monitoring is responsible to detect a group of host that has similar behavior and communication pattern by inspecting network traffics. Consequently, we are capturing network flows and record some special information in each flow. We are using Audit Record Generation and Utilization System (ARGUS) which is an open source tool [17] for monitoring flows and record information that we need in this part. Each flow record has following information: Source IP(SIP) address, Destination IP(DIP) address, Source Port(SPORT), Destination Port(DPORT), Duration, Protocol, Number of packets (*np*) and Number of bytes (*nb*) transferred in both directions. Then, we insert this information in a data base as shown in Table 1, in which $\{fi\}i = 1 \dots n$ are network flows. After this stage, we specify the period of time which is 6 hours and during each 6 hours, all n flows that have same Source IP, Destination IP, Destination port and same protocol (TCP or UDP) are marked and then for each network flow $\{fi\}$ (row) we calculate Average number of bytes per second and Average number of bytes per packet:

- Average number of bytes per second(*nbps*) = Number of bytes/ Duration
- Average number of bytes per packet(*nbpp*) = Number of Bytes/ Number of Packets

Then, we insert these two new values (*nbps* and *nbpp*) including SIP and DIP of the flows that have been marked into another database, similar to Table 2. Therefore, during the specified period of time (6 hours), we might have a set of database, $\{di\}i = 1 \dots m$ in which each of these databases has the same SIP, DIP, DPORT and protocol (TCP/UDP). We are focusing just at TCP and UDP protocols in this part.

As we mentioned earlier, the bots belonging to the same botnet have the same characteristics. They have similar

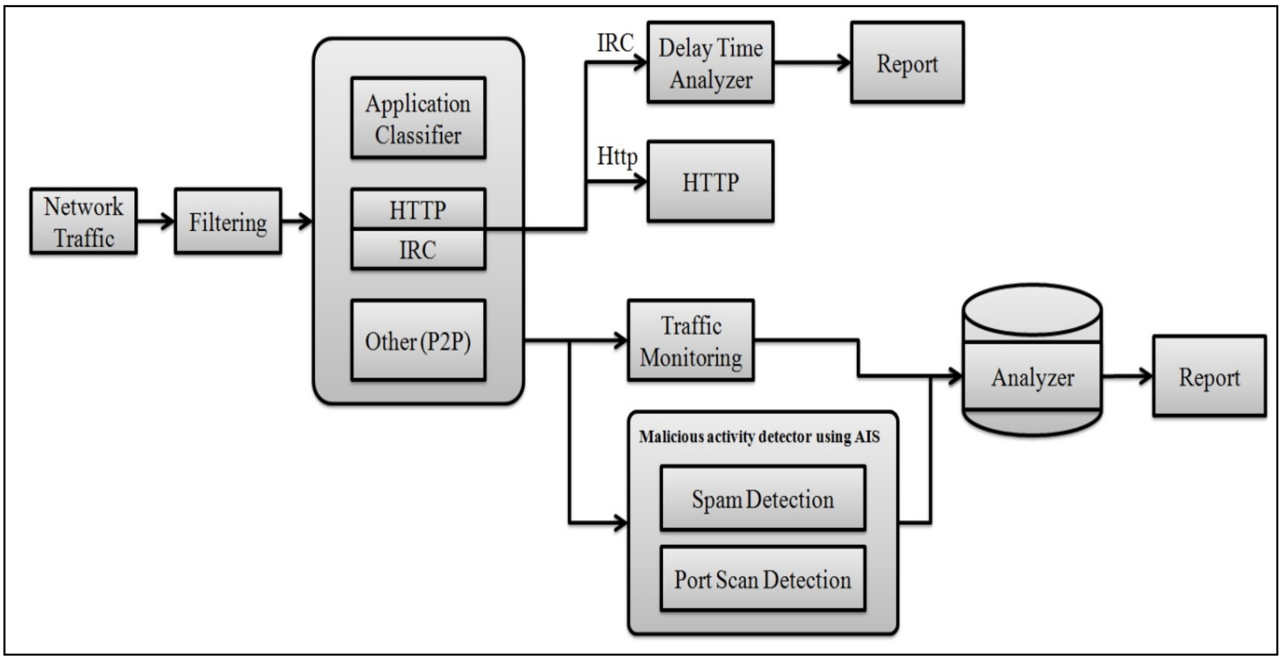


Figure 3: Architecture overview of our proposed detection framework

behavior and communication pattern, especially when they want to update their commands from botmasters or aim to attack a target; their similar behaviors became more obvious.

Therefore, next step is looking for group of databases that are similar to each other.

We proposed a simple solution for finding similarities among group of databases. For each database, we can draw a graph in x-y axis, in which x-axis is the Average Number of Bytes per Packet (*nbpp*) and y-axis is Average Number of Byte Per Second (*nbps*). (X, Y)= (bpp, bps). For example, in database (di), for each row we have *nbpp* that specify x-coordinate and have *nbps* that determine y-coordinate. Both x-coordinate and y-coordinate determine a point (x,y) on the x-y axis graph. We do this procedure for all rows (network flows) of each database. At the end for each database, we have a number of points in the graph; that by connecting those points to each other; we will come up with a curvy graph.

Next step is comparing different x-y axis graphs and during that period of time (each 6 hours), those graphs that are similar to each other are clustered in the same category. The results will show some x-y axis graphs that are similar to each other. Each of these graphs is referring to their corresponding databases in the previous step. We have to take a record of SIP addresses of those hosts and send the list to the next step for analyzing the data.

TABLE 2: DATABASE FOR ANALOGOUS FLOWS

Source Port	Destination Port	*nbps*	*nbpp*

TABLE 1: RECORDED INFORMATION OF NETWORK FLOWS USING ARGUS

fi	SIP	DIP	SPORT	DPORT	Protocol	*np*	*nb*	duration
f1								

.								
.								
fn								

A. *Malicious activity detector using AIS*

In this part we have to analyze the outbound traffic from the network and try to detect the possible malicious activities that the internal machines are performing. Each host may perform different kind of malicious activity but Scanning , Spamming, Binary downloading and exploit attempts are the most common and efficient malicious activities a botmaster may command their bots to perform [18, 19]. In this paper, we just focus on scanning and spam-related activities. In our framework we have utilized the AIS technique to detect these activities. Actually the idea of using AIS in this framework has been proposed before by Hossein *et. al.* [51] .The outputs of this part are the list of hosts which perform malicious activities.

The artificial immune system provides a multilayered protection mechanism to discriminate between the malicious and safe activities. To preserve the generality AIS divides all activities to self and non-self. According to this assumption the self activities are defined as the system's known and normal activities and the non-self are defined as the system's abnormal and harmful activities. However there might be some activities which are unknown to system but harmless. So they are considered as self activities in AIS. By defining this categorization AIS is used to discriminate between self and non-self activities. The detection procedure in AIS is done in two main steps [6]:

1. Generating and training the detectors: The AIS employs learning techniques using a training data set which contain malicious patterns and system's normal activities to train the detectors against the malicious activities [6,7,9]. The training procedure is done using negative selection algorithm [7].

2. Monitoring the network activities using the detector sets: After training the detectors it is time to face with real system's traffic. If any detector is stimulated with any data, the malicious activity will be reported and the detailed information about the activity is stored.

The training data for the first step can be obtained from standard data sets in general applications. In this framework in order to obtain an ultimate and accurate result we use data capturing technique for a period of time T_C^* to capture normal and abnormal system activities and store them to be used for training the detectors.

Obviously the data which is used for spam detection is a set of legitimate and spam message which is delivered in data capturing time.

1) Spam-related Activities: E-mail spam, known as Unsolicited Bulk Email (UBE), junk mail, is the practice of sending unwanted email messages, in large quantities to an indiscriminate set of recipients. A number of famous Botnets which have been used specially for sending spam are Storm Worm which is P2P Botnet and Bobax that used Http as its C&C.

There are some similarities [10] between SPAM messages and pathogenic microorganisms such as evolution of spam messages by changing the keyword to alternative spelling (e.g. low instead of law) to evade from usual SPAM filters. As well, the similarity between pattern matching techniques used to detect the SPAMs and matching algorithms which are utilized in natural immune system and AIS.

Up to now some immune inspired model had been introduced for SPAM detection [10]. A work which is done in 2003 by Oda and white [11] uses a regular pattern matching with assigning a weight to each detectors and incrementing and decrementing the weight when it binds with SPAM and legitimate messages respectively. In 2006 Bezerra et al. [12] put forward a new model based on a competitive antibody network by presenting the messages as binary vectors and using a technique for dimensionality reduction [10]. Finally, Guzella et al. [10] in 2008 presented a new effective model with ability to detect more than 98% of SPAM messages. This work which we have utilized in our framework is named as innate and adaptive immune system (IA-AIS). In this model negative selection and clonal selection algorithms are combined to train the detectors and detect the SPAM messages.

All training data are represented as microorganisms in that the message body and subject are corresponding as antigens and the sender's email address is corresponding as molecular patterns. This structure of the microorganism analyzes the email in to two parts which facilitate the detection of the SPAM. According to training dataset some sender's email addresses are marked as SPAM so they will be easily distinguished at first step, while the subject and body of the message as antigens are recognized by T-Cells and B-Cells. Figure 5 shows the construction of the microorganisms from the email.

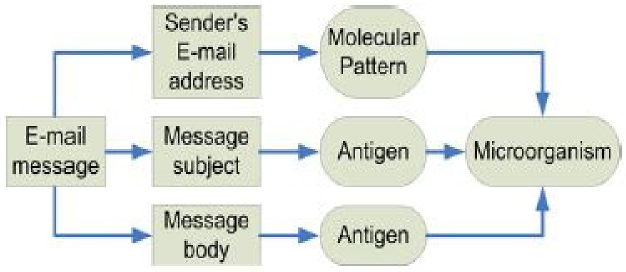


Figure 5 : microorganism generation from email [10].

In generation of microorganisms each characters are encoded to a custom 6-bit encoding in order to produce a set of visually similar characters (e.g. 0 and O , 3 and E) to produce different sequences of words which differ in only 1 or 2 bits. By using this technique the detectors with receptor "test" can bind with an antigen "t3st".

All detectors are trained using legitimate and SPAM message patterns which are collected during data capturing period. The legitimate and SPAM messages are represented as self and Non-Self data respectively. The process of training is shown in Figure 6. Self and Non-self molecular patterns are used for generating the macrophages. Regularity T-cells are generated by randomly selecting from self antigens and B-cells and T-cells selected using same procedure then negative algorithm is applied to eliminate the self reactive detectors. The output of this step is a trained detector set which contain the Macrophages for detection of the SPAM based on the sender's email address and B-Cells and Helper T-Cells for analyzing the message body an subject and the T-Cells for generation of Co-stimulation signals to confirm the detection of the B-Cells.

After training the detectors, system is ready for detection. Before investigation of the emails, they must be converted to a microorganism based on the structure that is given in previous section. At fist the microorganism is presented to macrophages in order to analyze its molecular patterns. If any macrophage is activated, the email will be marked as SPAM. Otherwise the antigen of microorganism will be analyzed by B-Cells. If no B-Cell is stimulated, then the message will be marked as legitimate message. Otherwise if any B-Cell stimulated, T-Cells will be used for final investigation and the result will determine whether the message is SPAM or legitimate.

In order to fit this technique in our framework for detecting the Spamming activities the detector sets are sent to each host in the network. Once any Spamming activity is reported by each host it will be sent to analyzer sender to be investigated.

1) Scanning: Scanning activities may be used for malware propagation and DOS attacks. Most scan detection has been based on detecting N events within a time interval of T seconds. This approach has the problem that once the window size is known, the attackers can easily evade detection by increasing their scanning interval. There are number of solutions for detecting the port scanning. Dendritic Cell Algorithm (DCA) based on artificial immune system is currently applied [16] for port scanning detection.

DCA presented by Greensmith et al. [14] is another human inspired algorithm which is currently added to artificial immune systems [13]. DCA is based on the new immunological theory called Danger Theory (DT) proposed by Aikelin et al. [15] which is an alternative technique within the AIS. Dendritic cells are a kind of antigen presenting cells within the immune system that are sensitive to changes in signals derived from their surrounding tissues [13, 14]. According to study of Greensmith DCA is an effective approach for detecting a large scale port scans during an extended time. Hence we have utilized DCA to detect the port scanning activities performed by bot infected hosts.

In DCA, dendritic cells like a natural data fusion cells process the input signals collected from surrounding environment and produce two kinds of output signals namely IL-12 and IL-10 as response. Each DC is able to collect and process the data and stimulate a cumulative output. The algorithm uses two compartments, one for collecting and processing of data called tissue and another for analyzing of data called lymph node. Population of antigens with their signal matrix is stored in a vector. Different signals of each antigen can be stored in signal matrix. All antigens with their corresponding signals are queue in the tissue and each time a set of 10 sampling populations are selected from antigen vector together with their corresponding signals. All sampling data proceed in the system and cumulative output value is updated per any round. Figure 7 illustrates the DCA framework.

Each DC is reactive to changes in signal matrix. Input signals are categorized in to four categories:

1. PAMPs: signatures of abnormal activities. For example number of errors per second. Presence of this signal usually signifies an anomaly.

2. Danger Signals: is a measure of abnormality which increases by value. For example number of network packages per second. Higher value of this signal indicates more probability of abnormality.

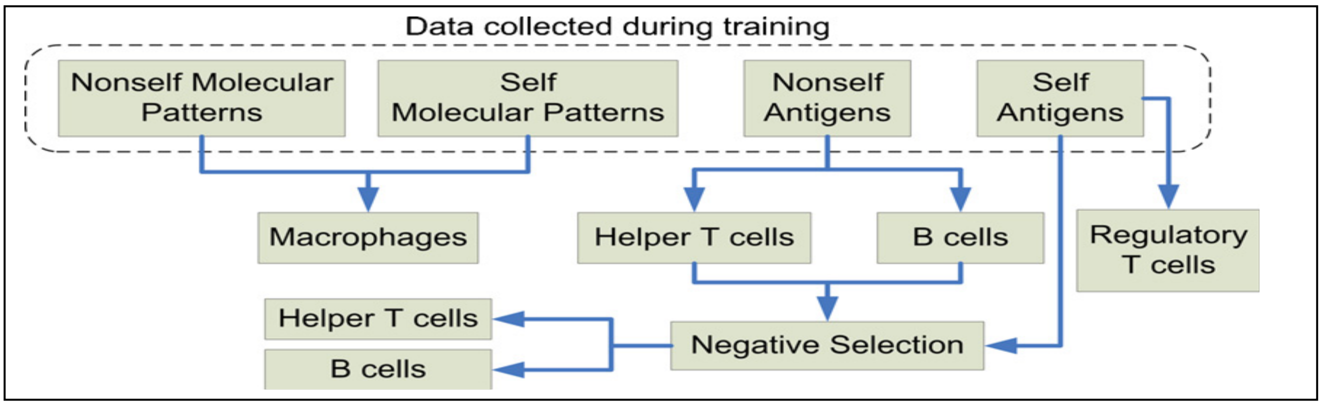


Figure 6: Training Procedure [10]

3. Safe Signals: is a measure of normal behavior which increases by value. For example low tolerance of packet sending amount.

4. Inflammation: a common system's signal which is inadequate to determine the systems status without presence of other signals. This signal used to magnify the effects of supplementary signals.

The process of detection is done using 7 signals from surrounding environment: two danger signals, two safe signals, two PAMP signals and one inflammatory [13]:

"The danger signals derived from:

1. Number of sent packets per second.
2. Ratio of TCP packets to all packets.

The safe signals are derived form:

1. The rate of changes in sending of network packets
2. Average network packet size

The PAMP signals are derived from:

1. The number of destination host unreachable errors per second

2. The number of sent and received TCP packets
The inflammatory signal is based on remote toot log-in"

The input signals are proceed in the system according to a simple weighted sum. The combination of the input signals lead to either IL-12 or IL-10 signals. The higher value of danger and PAMP signals, the higher mature IL-12 output signals. Reciprocally the higher safe signals, the higher semi mature IL-10 output signals. A schematic diagram of processing the signals in DCA is shown in Figure 8.

A. Analyzer

Analyzer which is the last part of our proposed framework for detection of Botnets, is responsible for finding common hosts that appeared in the results of previous parts (Traffic Monitoring and Malicious Activity Detector).

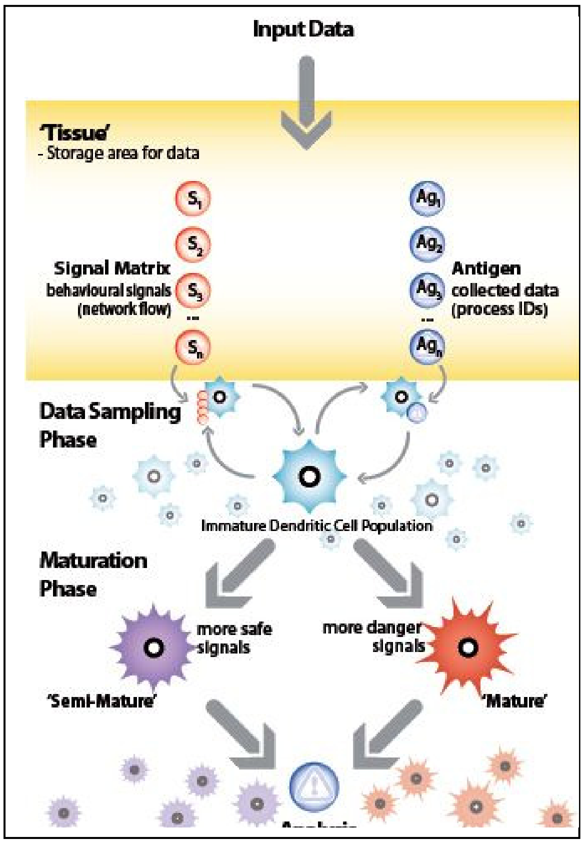


Figure 7: DCA framework [13].

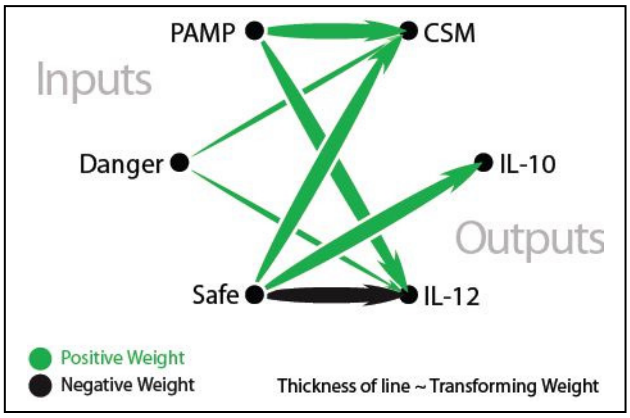


Figure 8: schematic diagram of processing the signals in DCA [13].

V. CONCLUSION

In comparison to other kind of malware, botnets are harder to monitor and shutdown and detection of them become a challenging problem. Artificial Immune System are new bio-inspired models which currently been applied for security problems in computer science. In this paper, we have utilized the AIS for malicious activity detection in bot infected hosts. In our proposed detection framework, we monitor the group of hosts that show similar communication pattern in one step and also performing malicious activities in another step and try to find common hosts in them. The point that distinguishes our proposed detection framework from many other similar works is that there is no need for prior knowledge of botnets such as botnet signature. In addition, we plan to further improve the efficiency of our proposed detection framework with adding unique detection method in HTTP part and make it as one general system for detection of Botnet and try to implement it in near future.

REFERENCES

[1] Hossein Rouhani Zeidanloo, Azizah Abdul Manaf, "Botnet Detection by Monitoring Similar Communication Patterns". *International Journal of Computer Science and Information Security*, Vol. 7, No. 3, Pages 36-45, March 2010, ISSN 1947-5500.

[2] K.W. Yeom, J.H. Park: An Immune System Inspired Approach of Collaborative Intrusion Detection System Using Mobile Agents in Wireless Ad Hoc Networks. CIS (2) 2005: 204-211 [2005]

[3] L.N. de Castro, J. Timmis. "Artificial Immune Systems: A New Computational Intelligence Approach" Springer, 2002.

[4] Fu, H., Yuan, X. and Hu, L. Design of a four-layer model based on danger theory and AIS for IDS. *International Conference on Wireless Communications, Networking and Mobile Computing*. IEEE. 2007.

[5] Zeidanloo, H.R; Manaf, A.A. "Botnet Command and Control Mechanisms". Second International Conference on Computer and Electrical Engineering,2009. ICCEE. Page(s):564-568

[6] S. Forrest, A.S. Perelson, L. Allen, R. Cherukuri, Self–nonself discrimination in a computer, in: Proc. IEEE Symposium on Research Security and Privacy, 1994, pp. 202–212.

[7] S. Hofmeyr, S. Forrest, Architecture for an artificial immune system, Evolutionary Computation. 2000. 7 (1) 1289–1296.

[8] Gu, G., Perdisci, R., Zhang, J., and Lee, W. (2008). BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. Proceedings of the 17th USENIX Security Symposium (Security'08).

[9] J.E. Hunt and D.E. Cooke, "Learning Using an Artificial Immune System", *Journal of Network and Computer Application,* 19, pp. 189-212, 1996.

[10. T.S. Guzellaa, T.A. Mota-Santosb, J.Q. Uchôac, and W.M. Caminhasa. "Identification of SPAM messages using an approach inspired on the immune system" science direct. 2008. 92(3). 215-225.

[11] Oda, T.,White, T. Increasing the accuracy of a SPAM-detecting artificial immune system. In: Proceedings of the IEEE CEC, 2003, vol. 1, pp. 390 396.

[12] Bezerra, G.B., Barra, T.V., Ferreira, H.M., Knidel, H., de Castro, L.N., Zuben, F.J.V., 2006. An immunological filter for SPAM. Lect. Notes Comput. Sci. 4163, 446–458.

[13] Greensmith, J., Aikelin, U. "Dendritic cells for SYN scan detection". Genetic And Evolutionary Computation Conference. 2007.

[14] J. Greensmith, U. Aickelin, and S. Cayzer. Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection. In ICARIS-05, LNCS 3627, pages 153–167, 2005.

[15] U Aickelin, P Bentley, S Cayzer, J Kim, and J McLeod. Danger theory: The link between ais and ids. In Proc. of the Second Internation Conference on Artificial Immune Systems (ICARIS-03), pages 147–155, 2003.

[16] J. Greensmith, U. Aickelin, and J. Twycross. Articulation and clarification of the dendritic cell algorithm. In ICARIS-06, LNCS 4163, pages 404–417, 2006.

[17] Argus (Audit Record Generation and Utilization System, http://www.qosient.com/argus)

[18] COLLINS, M., SHIMEALL, T., FABER, S., JANIES, J., WEAVER, R., SHON, M. D., and KADANE, J., "Using uncleanliness to predict future botnet addresses,," in Proceedingsof ACM/USENIX Internet Measurement Conference (IMC'07), 2007

[19] ZHUGE, J., HOLZ, T., HAN, X., GUO, J., and ZOU, W., "Characterizing the ircbased botnet phenomenon." Peking University& University ofMannheim Technical Report, 2007.

[20] J. Timmisa, A. Honec, T. Stibord and E. Clarka, "Theoretical advances in artificial immune systems". In: Theoretical Computer Science. science direct. 2008. 403(1): 11-32.

[21] Hossein Rouhani Zeidanloo and Azizah Abdul Manaf, "Botnet Detection Based on Passive Network Traffic Monitoring". International Conference on Computer Communication and Network, Orlando, FL, USA, July 2010. (In Print)

[22] Hossein Rouhani Zeidanloo, Mohammad Jorjor Zadeh shooshtari, Payam Vahdani Amoli, M. Safari and Mazdak Zamani, "A Taxonomy of Botnet Detection Techniques". *International Conference on the 3nd IEEE International Conference on Computer Science and Information Technology*. Chengdu, China, July 2010.(In Print)

[23] Fanelli, R. L. A Hybrid Model for Immune Inspired Network Intrusion Detection. *In: Artificial Immune Systems*. Phuket, Thailand. 107-119. 2009.

[24] Gomez, J. *Soft computing techniques for intrusion detection system.* Ph.D. Thesis. University of Memphis. 2004.

[25] Golait, N. R. *Artificial Immune System: Theories, Design and Dangers.* M.S. Thesis. University of New York. 2008.

[26] Matzinger, P. The Danger Model in Its Historical Context. *In: Scandanavian Journal of Immunology.* 2001. 54: 4–9.

[27] Forrest, S., Perelson, A. S., Allen, L. and Cherukuri, R. Self-nonself discrimination in a computer. In Proceedings of the 1994 IEEE Symposium on Security and Privacy, page 202. IEEE Computer Society, 1994.

[28] Matzinger, P.: The Danger Model: A Renewed Sense of Self. *Science.* 2002. 296:301-305

[29] A. Ramachandran and N. Feamster, "Understanding the network-level behavior of spammers," in *Proc. ACM SIGCOMM*, 2006.

[30] K. K. R. Choo, "Zombies and Botnets," Trends and issues in crime and criminal justice, no. 333, Australian Institute of Criminology, Canberra,March 2007.

[31] M. Vrable, J. Ma, J. Chen, D. Moore, E.Vandekieft, A. C. Snoeren, G.M. Voelker, and S.Savage," Scalability, Fidelity and Containment in the Potemkin Virtual Honeyfarm," in *Proc. ACM SIGOPS OperatingSystem Review*, vol. 39(5), pp. 148–162, 2005.

[32] P. Baecher, M. Koetter, T. Holz, M. Dornseif., and F. Freiling, "The nepenthes platform: An efficient approach to collect malware," in *Proceedings of International Symposium on Recent Advances in Intrusion Detection (RAID'06)*, (Hamburg), September 2006.

[33] F. Freiling, T. Holz, and G. W i c h e r s k i , "Botnet Tracking: Exploring a Root-cause Methodology to Prevent Denial of Service Attacks," in *Proceedings of* 10th European Symposium on Research in Computer Security (ESORICS'05), 2005.

[34] Snort IDS web page. http://www.snort.org, March 2006.

[35] J. Goebel and T. Holz. Rishi: Identify bot contaminated hosts by irc nickname evaluation. In *Proceedings of USENIX HotBots'07*, 2007.

[36] B. Saha and A, Gairola, "Botnet: An overview," *CERT-In White PaperCIWP-2005-05*, 2005.

[37] D. Dagon, "Botnet Detection and Response, The Network is the Infection," in *OARC Workshop*, 2005.

[38] J. Kristoff, "Botnets," in 32nd Meeting of the North American Network Operators Group, 2004.

[39] H. Choi, H. Lee, H. Lee, and H. Kim, "Botnet Detection by Monitoring Group Activities in DNS Traffic," in *Proc. 7th IEEE International* Conference on Computer and Information Technology (CIT 2007), 2007,pp.715-720.

[40] W. Strayer, D. Lapsley, B. Walsh, and C. Livadas, *Botnet Detection Based on Network Behavior*, ser. Advances in Information Security.Springer, 2008, PP. 1-24.

[41] M. M. Masud, T. Al-khateeb, L. Khan, B. Thuraisingham, K. W.Hamlen, " Flow-based identification of Botnet traffic by mining multiple in *Proc. International Conference on Distributed Framework& Application,Penang,Malaysia.2008*

[42] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: Clustering analysis of network traffic for protocol- and structure independent Botnet detection," in Proc. 17th USENIX Security Symposium, 2008

[43] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee. BotHunter: Detecting malware infection through ids-driven dialog correlation. In *Proceedings of the 16th USENIX Security Symposium (Security'07)*, 2007.

[44] http://en.wikipedia.org/wiki/Handshaking

[45] J. Rayome. " IRC on your dime? What you really need about Internet Relay Chat, CIAC/LLNL. May 22, 1998

[46] HTTP Made Really Easy, http://www.jmarshall.com/easy/http

[47] T. Karagiannis, A.Broido, M. Faloutsos, and kc claffy. Transport layer identification of P2P traffic. In ACM/SIGCOMM IMC, 2004.

[48] T. Karagiannis, A.Broido, N.Brownlee, kc claffy, and M.Faloutsos. Is P2P dying or just hiding? In IEEE Globecom 2004, GI.

[49] S. Sen, O. Spatscheck, and D. Wang. Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures. In WWW, 2004

[50] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC:multilevel traffic classification in the dark," In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 229-240, Philadelphia, Pennsylvania, 2005

[51] Hossein Rouhani Zeidanloo, Farhoud Hosseinpour and Parnian Najafi Borazjani. "Botnet Detection Based on Common Network Behaviors by Utilizing Artificial Immune System(AIS). ". 2nd International Conference on Software Technology and Engineering. San Juan, Puerto Rico, USA. , 2010.(In Print)

Hossein Rouhani Zeidanloo received his B.Sc. in Software Engineering from Meybod University, Iran. He has completed his Master's degree in Information Security at the Universiti Teknologi Malaysia (UTM) in 2010. He has published five papers in few international journals and also published many papers for international conferences around the world. His area of interest is Malware Detection, Wireless Security and Ethical Hacking.

Farhood Hosseinpour received his B.Sc in Computer Science from comprehensive University of Tabriz, Iran in 2007. He received his M.Sc in Information Security from University Technology Malaysia (UTM) in 2010. His research interests include network security, computer forensic, intrusion detection systems, artificial immune system, and biologically inspired approaches to computer security.

Farhood Farid Etemad holds B.Sc. in Computer Hardware Engineering from Ferdowsi University of Mashhad, Iran and received his Master's degree in Information Security from Universiti Teknologi Malaysia (UTM). He is currently contributing in research areas of Malware detection and botnet behavior analysis.