Botnet Detection Based On Network Traffic Flow Statistical Features and Model Based Clustering

G. Kirubavathi and S.Nalini
{g.kiruba@gmail.com, nalini.kamalini@gmail.com}

Department of Mathematics and Computational Sciences, PSG College of Technology, Coimbatore, India

Abstract. Botnet is one of the most notorious threats to cybersecurity and cyberspace, providing a distributed platform for multiple illegal activities, such as DDoS, spamming, phishing, click fraud, identity theft, etc. Regardless of numerous methods have been proposed to detect botnets, botnet detection is still a challenging issue, as botmaster’s are continuously improving bots to write them stealthier. Existing botnet detection mechanisms are not cope-up with the modern botnets. In this paper, we propose a novel approach to detect botnet based on network traffic flow behavior analysis using model based clustering called Gaussian Mixture Model (GMM). We have analyzed the botnet traffic flow statistical behaviors in a managed environment. The proposed model effectively detects the bot irrespective of their structural properties. Our experimental evaluation based on real-world data shows that the proposed model can achieve high detection accuracy with a low false positive rate using traffic flow behaviors. We have compared the proposed model with traditional clustering techniques such as K-Means and X-Means clustering. Our model achieves the improved detection rate compared to the K-Means and X-Means clustering. Also we have compared our proposed model with existing botnet detection methods. Our model achieves the better detection rate with minimum number of features than the prevailing methods.

Keywords: botnet detection; network flows; statistical features; model based clustering.

1 Introduction

A botnet is a collection of compromised hosts, i.e. Zombies or bots remotely controlled by an attacker called a botmaster through a command and control (C&C) channel. Due to their enormous size (tens of thousands of systems can be connected at the same time), they pose a serious threat to cybersecurity. B. Sending spam, launching distributed denial of service (DDoS) attacks, identity theft, click fraud, etc. Two of the most important attacks that botnets represent on the Internet are spam [1] and DDoS attacks [2, 3]. Some of the largest spam botnets send literally billions of messages a day, as shown in Figure 1. Cybercriminals use a variety of bots to carry out DDoS attacks on Internet servers. One of the most popular bots is called Black Energy. Figure 2 shows the Blackenergy botnet attacks on targets. Botnets threaten cyberspace with thousands of infected computers.
According to [4], the scope of the botnet breakout is becoming more critical day by day, as shown in fig 3. Botnet has attempted to control zombies remotely and instruct them using botmaster commands via the C&C channel. The C&C channel is an integral part of a botnet. Different botnets can organize their C&C channel in different ways. Botnets can be centralized, decentralized and hybrid according to their C&C channels and communication protocols (HTTP, P2P, IRC, IM, etc). According to the Microsoft intelligent report [5], the use of various C&C mechanisms is shown in fig.4. The centralized IRC-based C&C structure is the most widely used botnet structure. All the bots/zombies in a botnet are connected to a single C&C channel to receive the commands form the botmaster. The botmaster uses a central server to issue his commands to the zombies / bots on the network. All zombies on the network are visible on the C&C channel.

Centralized IRC-based botnet structures [6] are easy to build, easy to administer, and faster to respond to commands. But it is also easier to fight centralized botnets, as the entire zombie network will be neutralized if the C&C channel is blocked. Recently, botmaster have started using HTTP to manage their web-based centralized botnets. HTTP is a prominent communication protocol as it comprises of majority of the internet traffic [7]. These web-based C&C bots try to blend into normal HTTP traffic, which consequently makes them more difficult to be identified as HTTP is used as network communication protocol in many applications. The
Zeus botnet [8] is the most popular botnet using HTTP as the communication protocol. The Zeus botnets are estimated to include millions of compromised computers around the world. The Zeus botnet is mainly used to steal critical information including login credentials to email accounts, financial services etc. Nowadays, the botmaster started to build botnets with more resilient C&C structures using peer-to-peer (P2P) protocol [9, 10, 11, 12]. Bots belonging to P2P botnet form an overlay network in which any of the bots can be used by the botmaster to distribute commands to the others bots. In P2P botnet Commands are transferred from bot to bot, each bot has a list of several neighbors and any command received by a bot from one of its neighbors will be sent to the others, further distributing it across the zombie network. It offers higher resiliency, since even if a significant portion of P2P botnet is taken down the remaining bots may still be able to communicate with each other and with botmaster.

In both centralized and decentralized P2P structures, a bot bursts small packets across the network while actively searching for susceptible hosts. Based on the behavior of this bot, we analyze the statistical characteristics of the network traffic flow. These statistical characteristics of traffic flow, such as the size and number of packets, can be used to identify bots in both centralized and decentralized structures with model-based clustering called the Gaussian Mixture Model (GMM) [13]. The GMM is commonly used of unsupervised learning because it can dig out the data patterns and cluster those sharing similar data behaviors together [14]. The GMM mixture modeling apriori specifies a model and attempts to estimate the parameters of the model using Expectation-Maximization (EM) algorithm [15].

Detecting botnets has several benefits by analyzing the behavior of network traffic flow. First, detection is not limited to the launch or attack phase, but detects bots at each stage of their life cycle. The second benefit is that bot detection is more cost effective compared to other approaches that implement deep payload analysis. In this research, we compare the proposed model with two other traditional clustering techniques, namely K-means and X-means clustering. Our model achieves improved identification accuracy compared to others.

The rest of this article is structured as follows. Section 2 summarizes and discusses the work related to botnet detection. Section 3 presents our proposed detection system. Section 4 describes data collection and analysis. Section 5 shows the experimental results and the evaluation. Section 6 concludes the work.

2 Related work

Botnets are an existing and growing threat to the global cyber community. Detecting botnet is challenging since botnets use a wide variety of protocols such as IRC, HTTP, P2P, IM, etc to communicate with their Command & Control (C&C) server and moreover, they constantly keep changing the location of the C&C server. Newer botnets have started to use protocols based on HTTP, P2P, IM, and DNS, making it even more difficult to distinguish their communication patterns.

In recent years, network security researchers have become concerned with detecting and tracking botnets as it is a major research topic in the cybersecurity world. There is a large collection of literature on botnet detection. Furthermore, botnet detection approaches using flow
analysis techniques have only emerged in recent years [16, 17]. Botnet detection techniques can be divided into signature-based detection, anomaly-based detection, DNS-based detection, and mining-based detection [18]. Our approach is based on mining-based analysis of network traffic flow behavior due to its popularity. Anomaly-based techniques are generally based on anomalies in network behavior, such as high network latency, activities on unused ports. On the other hand, C&C traffic does not usually show abnormal behavior. It is mostly hard to differentiate C&C traffic from usual traffic behavior. At this point of view, machine learning based data mining techniques are very useful to extract unexpected network patterns.

Livadas et al. [19] proposed a flow-based approach to detect C&C traffic from IRC-based botnets. They used three different classifiers to group the flow behavior. The approach consists of two stages. The first stage uses mining algorithms to classify traffic flows into chat or non-chat flows, while the second stage uses mining algorithms to classify IRC flows as malicious or non-malicious. The use of a Bayesian network classifier showed potential for accurate classification of IRC botnet flows, with a relatively high false negative rate of 1020% and a false positive rate of 3040%. The presented approach does not depend on the traffic payload that the encrypted C&C channel detection allows. They demonstrated that it is possible to separate the streams into malicious and non-malicious using mining algorithms.

Strayer et al. [20] developed a method to detect botnet C&C traffic through passive analysis of network flow information. Its approach is based on flow properties like duration, bytes per packet, bits for the second, TCP flags, etc. The proposed network-based approach to detecting botnet traffic uses two-step processes that include first separating IRC flows and then detecting botnet C&C traffic from normal flows. This technique is specific to IRC-based botnets.

In my previous work [41], worked with same dataset with classification algorithms such as Adaptive boosting, naïve Bayesian and support vector machine. Among them naïve Bayesian classifier outperforms all other algorithms.

Gu et.al [21] proposed a novel mining-based system called BotMiner. The system takes advantage of the underlying uniformity behavior of botnets and detects them by attempting to observe and group traffic flow behavior to identify hosts with normal and malicious communication patterns and activities. The intuition is behind the system is that, bots belonging to the same botnet are likely to behave similarly in terms of communication patterns. The system has many desirable features but it needs long monitoring time and unforged large scale data to detect malicious activities; however real botnets communicates silently with large number of small packets, and forges their information.

Our proposed model addresses some of the disadvantages of previous bot detection methods. Our model is based on the idea with the aim of identifying the bot regardless of its structural properties. To do this, we first observe the network activity of a bot in a controlled environment. We then analyze the essential network behaviors of the bot based on logged traffic. Our model uses mining methods to group the behavior of the network traffic flow in order to identify and group the botnet and the normal communication that is shared with others [22-24]. However, in addition to the previous work, our model has a number of outstanding features. First of all, our model does not rely on prior knowledge of botnet structures. Second, it is resistant to the occurrence of encrypted communications traffic because it does not verify the contents of the packet.
3 Proposed Detection System:

The proposed detection system utilizes Gaussian Mixture Model with Expectation-Maximization Algorithm. We extracted the TCP and UDP-based statistical characteristics of the network traffic flow for our proposed work. Since then, botnets have mainly used TCP and UDP-based connections to communicate with the C&C server and carry out malicious activities. The building block of the proposed system is specified in Fig. 5. The proposed model consists of training and recognition phases. In the training phase, we collected numerous background network traffic flow traces, HTTP botnet traffic flow traces and P2P botnet traffic flows, and normal flow traces from the ISOT dataset [25]. We extract the statistical characteristics of the network flow from the collected traffic flows. The extracted statistical feature vectors are transformed into a GMM model using the Expectation-Maximization training algorithm. During the detection phase, compute the mixing propositions of each statistical feature vector instances and assign the instances to the corresponding mixing proposition cluster component.

Fig 5. Building Blocks of Proposed Detection System

3.1. Statistical Features Extraction

We observe the network behavior of a botnet at the level of the TCP/UDP flow with centralized and decentralized botnet structures. In centralized botnet structure, the C&C channel runs through IRC or HTTP protocol. The IRC based centralized botnet mainly focusing on TCP
and UDP port for their resource sharing. The HTTP based centralized botnet do not maintain a
connection with C&C server, but they periodically download the instructions using web requests
from the web server through TCP connections. In decentralized P2P botnet traffic flows are
primarily focusing on TCP/UDP flows. In P2P network each peer is using UDP to search and
TCP to fetch the information. During the bot communication there is significant changes in
TCP/UDP flow irrespective of their structures. Also, bots within a botnet behave similar
communication pattern. Since, bots are non-human driven, its pre-programmed. When the
normal traffic, these TCP/UDP flow statistical features are arbitrariness. A deep analysis on
TCP/UDP flows from our traffic traces collected through experiments and ISOT dataset [25].
The statistical results show that remarkable difference between normal traffic and botnet traffic
with TCP/UDP flow. We have mined the TCP/UDP flow based statistical features for our
proposed system based on the network behavior of bots. The statistical feature set can be
defined as \(<pack\_TCP, pack\_UDP, byte\_TCP, byte\_UDP, Duration>\). The statistical features
are listed in table 1.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pack_TCP</td>
<td>No. of packets per TCP flow</td>
</tr>
<tr>
<td>pack_UDP</td>
<td>No. of packets per UDP flow</td>
</tr>
<tr>
<td>byte_TCP</td>
<td>No. of Bytes per packet in a TCP flow</td>
</tr>
<tr>
<td>byte_UDP</td>
<td>No. of Bytes per packet in a UDP flow</td>
</tr>
<tr>
<td>Duration</td>
<td>Flow duration</td>
</tr>
</tbody>
</table>

3.2. Model based Clustering

In recent years, model-based clustering approach is widely applied in statistical based
network security domain [26, 27]. In this work, we proposed model based cluster using GMM
with EM algorithm. This approach gives much better performance than existing methods. Since,
the GMM is a probabilistic model that assumes all the statistical features are generated from a
mixture of Gaussian distributions. Mixture models are more general than partitioning and fuzzy
clustering. In GMM, clusters can be characterized by a less number of parameters. Also, it can
find the characteristics descriptions for each cluster component. Whereas, the conventional
clustering algorithms are largely heuristic and formal inference is not possible. But, model based
clustering is an alternative. The EM algorithm is a broadly applicable statistical approach for
maximizing complex likelihoods and handling the incomplete data [28]. It is an efficient
algorithm to estimate the parameters of the mixture model. The EM algorithm starts with some
initial random parameters and then repeatedly applies the E-step and M-step to generate better
parameter estimation. The parameters of the model are chosen by maximizing the log-likelihood
of the training data with respect to the model. The random variable \(x\) is a statistical feature vector
set \(<pack\_TCP, pack\_UDP, byte\_TCP, byte\_UDP, Duration>\) were extracted from network
flow traffic. Each statistical feature instance represents the traffic behavior corresponding to a
single flow. The network flow statistical features can be modeled using GMM. Each statistical
feature instance \(x\) is normally distributed random variable is viewed as coming from a mixture

\[
p(x) = \sum_{c=1}^{k} w_c f_c(x; \mu_c, \Sigma_c)
\]

of density
Where x- network flow statistical features such as pack_TCP, pack_UDP, byte_TCP, byte_UDP, Duration. c=1, 2… k number of mixture cluster component densities; 1 ≤ c ≤ k. 
W_c ~ (1 ≤ c ≤ k) are mixture weights or mixing propositions which satisfy \( w_c ≥ 0 \) and \( \sum_{c=1}^{k} w_c = 1 \). The mixing probabilities are used to group the statistical features from the training dataset to the corresponding cluster components such as normal, botnet and outliers.

\[ f_c(x; \mu_c, \Sigma_c) = \frac{1}{(2\pi)^{N/2}|\Sigma_c|^{1/2}} \exp \left\{ -\frac{1}{2}(x - \mu_c)^T \Sigma_c^{-1} (x - \mu_c) \right\} \]

Each cluster component is modeled using the gaussian distribution with mean \( \mu_c \) and covariance matrix \( \Sigma_c \). The mean \( \mu_c \) is calculated for each statistical features such as pack_TCP, pack_UDP, byte_TCP, byte_UDP, Duration for each mixing probabilities. For example there are three mixing probabilities \( W_c \sim \{c=1,2,3\} \) in a training dataset. The mean \( \mu_c \) is calculated as follows

<table>
<thead>
<tr>
<th>C=1</th>
<th>C=2</th>
<th>C=3</th>
</tr>
</thead>
<tbody>
<tr>
<td>pack_TCP</td>
<td>( \mu_{c1} )</td>
<td>( \mu_{c2} )</td>
</tr>
<tr>
<td>pack_UDP</td>
<td>( \mu_{c4} )</td>
<td>( \mu_{c5} )</td>
</tr>
<tr>
<td>byte_TCP</td>
<td>( \mu_{c7} )</td>
<td>( \mu_{c8} )</td>
</tr>
<tr>
<td>byte_UDP</td>
<td>( \mu_{c10} )</td>
<td>( \mu_{c11} )</td>
</tr>
<tr>
<td>Duration</td>
<td>( \mu_{c13} )</td>
<td>( \mu_{c14} )</td>
</tr>
</tbody>
</table>

\( \mu_{c1}, \mu_{c2}, ..., \mu_{c3} \) are the numerical values for the mean of each features. The covariance matrix \( \Sigma_c \) is the N-by-N matrix. As the statistical features are autonomous, the covariance matrix decreases to a diagonal matrix. The diagonal covariance matrix is computationally efficient. The covariance matrices are calculated for the statistical features in the training data of each mixing probabilities. Calculation of the covariance matrix \( \Sigma_c \) for three mixing probabilities is given as follows.

The covariance matrix for the first mixing probabilities / mixture weight for the training dataset

<table>
<thead>
<tr>
<th>pack_TCP</th>
<th>pack_UDP</th>
<th>byte_TCP</th>
<th>byte_UDP</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Sigma_{c1} )</td>
<td>( \Sigma_{c2} )</td>
<td>( \Sigma_{c3} )</td>
<td>( \Sigma_{c4} )</td>
<td>( \Sigma_{c5} )</td>
</tr>
</tbody>
</table>

\( \Sigma_{c1}, \Sigma_{c2}, ..., \Sigma_{c5} \) are the numerical values of the covariance matrix. Where i=1,2,3.

The model

\[ c = 1, 2, ..., K \] Gaussian cluster component densities (i.e. the number of cluster components such as normal, botnet, outlier). The dataset consists of \( X \sim \{x_n | n=1, 2, ..., 5\} \). Estimate the model parameters using EM algorithm. Such that \( \lambda = <w_c, \mu_c, \Sigma_c> \) by maximizing the log likelihood function \( l(x | \lambda) = p(x_1, x_2, ..., x_n | \lambda) \). Presume \( \lambda^* \) is the estimation value which can maximize the \( l(x | \lambda) \), then we have \( \lambda^* = \max l(x | \lambda) \). The EM algorithm starts with some initial random parameters \( \lambda^0 = <w_c^0, \mu_c^0, \Sigma_c^0> \) to estimate the posterior probability for every \( n \) and \( c \). Using this posterior probability to re-estimate the parameters through E-step and M-step by maximizing the likelihood function.
3.3. GMM training algorithm

1. Initialize the mixture weights / mixing probabilities $w_c^0$ randomly such that their sum is equal to 1, i.e., $\sum_{c=1}^{k} w_c = 1$.

2. Set the mean $\mu_c^0$ of every mixture weights / mixing probabilities by choosing the instance arbitrarily, in such a way no two mixture weights have the identical mean.

3. Set the covariance matrix $\Sigma_c^0$ of every mixture weights to the N-by-N matrix. So, the parameter initialization: $\lambda^0 = \langle w_c^0, \mu_c^0, \Sigma_c^0 \rangle$

4. Until the mean and covariance matrix of mixture weights are converge
   a. For each instance in the given dataset, calculate
      i. E-step: posterior probability $p(c|x_n)$ is calculated for each and every data instance $X \sim \{x_n \mid n=1, 2, ..., N\}$ and each and every mixture component $c$.

$$p(c \mid x_n) = \frac{w_c f_c(x_n, \mu_c, \Sigma_c)}{\sum_{c=1}^{K} w_c f_c(x_n, \mu_c, \Sigma_c)}$$

5. Re-estimate the model parameters according to the posterior probabilities $p(c|x_n)$:
   i. Recompute the probability of each mixture weights

$$w_c = \frac{1}{N} \sum_{n=1}^{N} p(c|x_n)$$

   ii. Recompute the mean of each mixture weights

$$\bar{x}_c = \frac{\sum_{n=1}^{N} p(c|x_n) x_n}{\sum_{n=1}^{N} p(c|x_n)}$$

   iii. Recompute the covariance matrix of each mixture weights

$$\Sigma_c = \frac{\sum_{n=1}^{N} p(c|x_n) (x_n - \bar{x}_c)^2}{\sum_{n=1}^{N} p(c|x_n)}$$

3.4. Testing

During the testing stage, it utilizes the mixing probabilities, means then variances of different cluster component mixtures obtained from the training phase. The probability that the $n^{th}$ instance, $x_n$ belongs to the cluster component $c$ is found using $p(c|x_n)$. Where $c$ is the number of cluster component in the statistical features dataset. While applying model-based clustering technique to botnet detection, we originate two basic assumptions such as the input statistical features are composed of three clusters, particularly botnet, normal and outliers. The size of the
botnet cluster is always smaller than the size of the normal cluster. Therefore, we can easily label the botnet cluster according to the size of each cluster. The botnet detection algorithm is based on the posterior probability produced by EM algorithm. The posterior probabilities exemplify the likelihood that the instance approximates to a specified Gaussian component. The greater the value of posterior probability for each instance belonging to a specified Gaussian component, the higher the approximation is. As an effect, instances are assigned to the corresponding Gaussian components according to their posterior probabilities. Through the empirical experiments, the posterior probability of the botnet data instance is stuck between 0.2 to 0.4. Apply the value of the posterior probabilities as a threshold $t = \{0.2$ to 0.4$\}$ to the botnet cluster component.

The various cluster component probability for each instance is equal to the posterior probability of the corresponding instance of the dataset, which is defined as

$$
\begin{align*}
\text{If } p_j(c|x_n) = t \text{ then } c = \text{botnet} \\
\text{Else} \\
\text{If } p_j(c|x_n) > t \text{ then } c = \text{normal} \\
\text{Else} \\
c = \text{outlier}
\end{align*}
$$

Where, $x_n$ is statistical features in the dataset; $c$ is number of cluster component and $p_j(c| x_n)$ is the conditional/posterior probability of $x_n$ belonging to particular cluster component $c$.

Algorithm 1 represents a complete GMM based botnet detection.

**Algorithm 1: Pseudo code of the proposed GMM based botnet detection**

Function: GMM_Botnet_Detection (dataset X~$\{x_n | n= 1, 2, 3, \ldots , N\}$) returns clusters and posterior probability $p(c|x_n)$

**Initialization:**
- Statistical features dataset $= \phi$; $j \leftarrow 0$
- Initial parameters $\{w^j_c, \mu^j_c, \Sigma^j_c\}$, $1 \leq c \leq k$, are arbitrarily created;
- Compute the initial log-likelihood $L_j$;

**Repeat:**
- For $1 \leq c \leq k$, $1 \leq n \leq N$
- Compute posterior probability $p_j(c|x_n)$
  - $\leftarrow j+1$
  - Re-estimate $\{w^j_c, \mu^j_c, \Sigma^j_c\}$ by using current posterior probability $p_j(c|x_n)$, $1 \leq c \leq k$, $1 \leq n \leq N$;

**Calculate the current log likelihood $L_j$;**

**Until:** ($p_j(c|x_n) = \max (p_j(c|x_n))$),

Assign $x_n$ to $c$

Return $c$, $c = 1, 2, 3, \ldots , k$ number of clusters
5 Dataset Collection and Analysis

To collect botnet traffic from the Internet, we created a botnet configuration with seven systems in our lab that includes a C&C interface and zombie machines. The C&C interface is hosted on the http://botsample.6te.net website. The typical architecture of our botnet configuration is fig. 6 and 7 illustrate an example screenshot of a collection of traces. The Zeus and Spyeye botnets are installed using the drive-by download mechanism. Once Zeus and Spyeye are installed, the antivirus and security software on the victim's (zombie) computer will be disabled to avoid detection. Zeus injects itself into the address space of Windows Explorer. After successfully installing the bot binary, the victim's computer will turn into a zombie. The zombie then communicates with the C&C servers which are encoded in the bot's binary. During bot communication, network traffic traces were collected for each botnet 5 hours a day, 6 days a week. Similarly, normal traffic was collected by the National Knowledge Network with a bandwidth of 100 Mbps. Table 1 shows the collected botnet traffic traces.

<table>
<thead>
<tr>
<th>Bot Family</th>
<th>Trace Size</th>
<th>Packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spyeye</td>
<td>Trace1</td>
<td>14.63 GB</td>
</tr>
<tr>
<td></td>
<td>Trace2</td>
<td>15.65 GB</td>
</tr>
<tr>
<td>Zeus</td>
<td>Trace1</td>
<td>16.24 GB</td>
</tr>
<tr>
<td></td>
<td>Trace2</td>
<td>11.6 GB</td>
</tr>
</tbody>
</table>

Table 1. Botnet Traces
By analyzing the botnet traffic that has been collected, it has a large number of small packets and pursues constant communication. Because web-based botnets do not maintain the connection to the web server. However, they often communicate with the web server to download commands and update the bot's code. These send a large number of small packets in bot communication. Additionally, Zombie drops small packets across the network when it actively searches for vulnerable hosts on the network. Fig. 8 shows the botnet traffic flows. Charts are drawn on different time scales during bot communication. Normal traffic follows randomness in packet size and inconsistency in communication packets. Figure 9 shows the normal flow of traffic. Diagrams are recorded on different time scales during normal communication.

![Fig 8. Botnet traffic flows](image1)

![Fig 9. Normal traffic flows](image2)

We also used publicly available ISOT datasets [25] for our approach. The ISOT data set is a combination of various malicious and non-malicious data sets accessible to the public. Malicious traffic in this registry contains Storm and Waledac botnets. Waledac is the most widespread P2P botnet and is widely regarded as the successor to the Storm botnet with an additional decentralized communication protocol. The Storm botnet uses Overnet as a communication channel, Waledac only uses HTTP communication and a DNS network based on Fastflux. P2P botnets perform processes like search, post, etc. using the UDP protocol and file transformation using the TCP protocol. This process creates a large number of small packets in the botnet's communication traffic. Since then, botnet traffic flows have been lower than normal traffic flows. To show the non-malicious, they integrated two different data sets, one from Ericsson Research's Traffic Lab in Hungary [28] and the other from Lawrence Berkeley National Lab (LNBL) [29]. When analyzing the ISOT dataset, some interesting facts are observed. The traffic pattern displayed by the bot is consistent, as it regularly updates queries while communicating with other zombies on the botnet. Also, botnet C&C commands generally only generate small packets.

### 6 Experimental Results and Evaluation

We use Java to run a statistical feature extraction component that analyzes the traffic of network flows and extracts the statistical feature vectors. Each instance of statistical characteristic represents the traffic behavior that corresponds to a single flow. In addition to the feature extraction component, we used two machine learning packages, Weka [31] and JavaML [32], to create the recognition model. In order to evaluate the efficiency of our model, we execute a series of experiments with respect to collected botnet traces and ISOT dataset. Our
model consists of training and testing phases. For model training, the dataset composed of 71,661 instances which include 35,096 normal instances, 12,460 ISOT dataset instances, 10,198 Zeus instances and 10,907 Spyeye instances. This training data are clustered into normal and botnet with GMM model based clustering. The testing dataset consists of 6,709 ISOT dataset instances, Zeus 5,491 instances and spyeye 5,873 instances. The experimental result shows the normal instance clusters are always higher than the botnet instance clusters. The cluster mixing propositions for botnet clusters are lies between 0.2 to 0.4. The normal cluster proposition is higher than the botnet cluster propositions. Below the botnet cluster propositions are called as outliers. The outlier cluster dose not disrupts the clustering process. Table 2 shows the results of clustered propositions of ISOT, Zeus and spyeye datasets.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Botnet Clusters</th>
<th>Normal Clusters</th>
<th>Outliers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cluster-1</td>
<td>Cluster-2</td>
<td></td>
</tr>
<tr>
<td>ISOT</td>
<td>0.30160162</td>
<td>0.2370760</td>
<td>0.43552354</td>
</tr>
<tr>
<td>Spyeye</td>
<td>0.3396960</td>
<td>0.5120811</td>
<td>0.1482229</td>
</tr>
<tr>
<td>Zeus</td>
<td>0.33261712</td>
<td>0.62547514</td>
<td>0.04190774</td>
</tr>
</tbody>
</table>

Table 2. Mixing propositions for different clusters

The performance of our proposed model has been evaluated with different traditional clustering techniques such as X-means [33, 34] and K-means [35, 36] for same data set. We have used three metrics to evaluate performance of our proposed model, namely, Detection Rate (DR), False Positive Rate (FPR), and Receiver Operating Characteristic (ROC). Table 3. Shows the results of performance estimation of detection rate and false positive rate with traditional clustering techniques and proposed model. Through the performance experiments, our model undoubtedly outperforms the state-of-art solution for botnet detection with great detection rate and low false positive rate compared with others.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Datasets</th>
<th>Detection Rate</th>
<th>False Positive Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-Means</td>
<td>ISOT</td>
<td>93.12</td>
<td>0.899</td>
</tr>
<tr>
<td></td>
<td>Spyeye</td>
<td>93.20</td>
<td>0.951</td>
</tr>
<tr>
<td></td>
<td>Zeus</td>
<td>93.46</td>
<td>0.922</td>
</tr>
<tr>
<td>X-Means</td>
<td>ISOT</td>
<td>94.27</td>
<td>0.946</td>
</tr>
<tr>
<td></td>
<td>Spyeye</td>
<td>94.26</td>
<td>0.866</td>
</tr>
<tr>
<td></td>
<td>Zeus</td>
<td>94.21</td>
<td>0.913</td>
</tr>
<tr>
<td>Proposed Model</td>
<td>ISOT</td>
<td>99.17</td>
<td>0.074</td>
</tr>
<tr>
<td></td>
<td>Spyeye</td>
<td>99.25</td>
<td>0.312</td>
</tr>
<tr>
<td></td>
<td>Zeus</td>
<td>99.26</td>
<td>0.267</td>
</tr>
</tbody>
</table>

Table 3. Performance estimation of K-Means, X-Means and Proposed model

Table [4] shows a comparison between our model and some of the existing botnet detection techniques to measure the performance of our model. The result shows that the proposed model achieves better detection than existing methods.

<table>
<thead>
<tr>
<th>Detection Methods</th>
<th>Botnet Data</th>
<th>Number of features</th>
<th>No. of bot samples</th>
<th>C&amp;C Structure independent</th>
<th>Detection Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authors</td>
<td>Method</td>
<td>Environment</td>
<td>IRC</td>
<td>P2P</td>
<td>HTTP</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------</td>
<td>------------------------</td>
<td>-------</td>
<td>-------</td>
<td>--------</td>
</tr>
<tr>
<td>Livadas et al. [18]</td>
<td>Botnet traffic is generated within controlled environment</td>
<td>10</td>
<td>1</td>
<td>IRC</td>
<td>92.00%</td>
</tr>
<tr>
<td>Saad et al. [24]</td>
<td>Botnet traffic is captured using honeypots.</td>
<td>11</td>
<td>2</td>
<td>P2P</td>
<td>89.00%</td>
</tr>
<tr>
<td>W.Lu et al. [25]</td>
<td>Botnet traffic is generated within controlled environment</td>
<td>256</td>
<td>2</td>
<td>IRC</td>
<td>95.00%</td>
</tr>
<tr>
<td>Masud et al. [37]</td>
<td>Botnet traffic is generated within a controlled environment</td>
<td>20</td>
<td>2</td>
<td>IRC</td>
<td>95.20%</td>
</tr>
<tr>
<td>Nogueira et al. [38]</td>
<td>Botnet traffic is generated within a controlled environment</td>
<td>8 - 16</td>
<td>1</td>
<td>YES</td>
<td>87.56%</td>
</tr>
<tr>
<td>Liao et al. [39]</td>
<td>Botnet traffic is generated within controlled environment</td>
<td>12</td>
<td>3</td>
<td>P2P</td>
<td>92.00%</td>
</tr>
<tr>
<td>Kirubavathi et al. [40]</td>
<td>Botnet traffic is generated within controlled environment</td>
<td>6</td>
<td>2</td>
<td>HTTP</td>
<td>99.025%</td>
</tr>
<tr>
<td>Proposed Model</td>
<td>Botnet traffic is generated within controlled environment</td>
<td>5</td>
<td>4</td>
<td>YES</td>
<td>99.22%</td>
</tr>
</tbody>
</table>

Table 4. Performance Comparison with existing methods

Another interesting performance comparison measure is ROC. Through ROC we can compare the proposed model with K-Means and X-Means clustering for the same dataset. The accuracy of
Fig 12. ROC curve for Spyeye Dataset

Fig 13. ROC curve for Zeus Dataset

Fig 14. ROC curve for ISOT Dataset
the K-Means, X-Means and proposed model is visualized by the Receiver Operating Characteristics (ROC) curves shown in Fig. 12, Fig. 13 and Fig. 14. The curves show the impact of the Detection Rate (DR) and the False Positive Rate (FPR) for K-Means, X-Means and proposed model. As shown in figures, the ROC performance of our proposed model is the best among the other clustering methods.

**Conclusion:**

In this paper, we proposed a novel botnet detection model based on network traffic flow statistical behavior analysis using model based clustering called Gaussian Mixture Model. Observed the bot activities in a controlled environment, we notice that botnet network traffic flows features are similar in statistical behaviors. Our model extracts the statistical behaviors and groups the similar behaviors into cluster. In GMM, clusters are represented as probabilistic models. The proposed model does not rely on payload information, so it can detect the encrypted bot communication traffics. The evaluation shows that our proposed model can detect the bot effectively irrespective of their structural properties with a very low false positive rate.

**References**


