I. Introduction
As an infrastructure for performing malicious activities, botnet can be used for distributed denial-of-service (DDoS) attacks, spamming, click fraud, identity theft, etc. Early botnets exhibited a centralized topology [2, 3], whereas more recent botnets [4, 8] started a topology shift into a peer-to-peer architecture. The main reason of this shift is a single point of failure in centralized C&C server architecture [5].

Recent advances in cloud computing and introduction of MapReduce [1] paradigm have been used in many data intensive computations. Certain advantage of Hadoop framework, an open-source version of MapReduce, is ability to execute tasks in distributed manner in a Hadoop Distributed File System (HDFS) [12] on commodity hardware. Moreover, it has its own recovery and fault-tolerance mechanisms. Our proposed method utilizes the advantages of Hadoop as well as behavioral flow analysis.

II. Related Work
Early botnet detection systems have been utilizing a numerous signature-based approaches [6]. A scalable signature-based approach was presented in Kargus [7] by accelerating signature matching in GPU.

Flow-based approach proposed in Gu et al. relies on the botnet lifecycle activities. Further, BotMiner [8] was proposed with the idea of correlating similar malicious activities with similar flows.

BotGraph [10] is one of the first applications having utilized the MapReduce
paradigm in spamming botnet detection. BotCloud [11] is another detection method utilizing large graph processing capabilities of Hadoop. They have adapted the PageRank algorithm in the context of botnet detection and correlated the page rank of node with its probability of being bot.

III. Detection technique and methodology

3.1 Approach overview

The overview of the system architecture is shown in Fig. 1. The Module 1 is used to parse pcap files in parallel in the HDFS. This library is adopted from the work of Lee et al. [13].

```
Module 1:
  Parsing library
Module 2:
  P2P host detection
Module 3:
  Botnet detection
```

Fig. 1. Overview of the system architecture.

3.2 P2P host detection and implementation

The main purpose of Module 2 in Fig. 1 is to detect hosts with any kind of P2P activity. In order to differentiate P2P applications from normal user behavior (e.g. browsing, file downloads), we consider a number of features listed below.

Failed Connections. Normally, P2P applications expose higher number of failed connections due to the peer churn [14] phenomenon. We consider as failed any TCP or UDP flow with outgoing packet but no response packet, and a TCP flow with a reset packet.

Unresolved connections. DNS utilization behavior of P2P applications is different from the one of normal traffic [15]. Hosts running P2P applications resolve the IP list from the peers as opposed to DNS query. Thus we consider the number of DNS queries (answers) sent (received) as well as whether the flow have been previously resolved from DNS answer.

Destination subnet diversity. Another distinction of P2P traffic from normal Internet traffic is the diversity of destination hosts. Usually those hosts are scattered around numerous subnets separated geographically. Thus, we extracted the following two features: number of distinct IPs contacted by the host, and the number of different /16 prefix subnets connected by the host. Fig. 2 represents detailed design of Module 2 (P2P host detection) in Hadoop framework.

```
1. Compute #1 of DNS packets
2. If #1 > Θ1 then return
3. Compute #2/#3 of distinct destination subnets/Ps
4. If #2 < #2 or #3 < #3 then return
5. Return (flow stats)
```

Fig. 2 Design of Module 2 in Hadoop framework.

3.3 Fine-grained P2P botnet detection and implementation

Only traffic from the hosts running P2P applications is passed further to Module 3 in Fig. 1. The duty of Module 3 is to differentiate between hosts running legal P2P applications (e.g. Skype, eMule) and botnet infected ones. Here we utilize the following observations.

First of all, we use the observation that
P2P botnets have persistent flows compared to normal P2P applications [14].

Second observation is that the set of hosts connected by bots have more common hosts compared to legal P2P application.

ISOT dataset created by Information Security and Object Technology (ISOT) research lab at the University of Victoria was used for the benchmark [16]. Additionally we used a dataset consisting of legal P2P applications from the research group at Georgia Tech [9].

IV. Results and Discussion

Table 1 shows the detection results for Modules 2, and 3 of the detection system.

<table>
<thead>
<tr>
<th>Type of P2P host</th>
<th>Module 2 (#detected/#total)</th>
<th>Module 3 (#botnet/ #P2P host)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skype</td>
<td>6/7</td>
<td>1/6</td>
</tr>
<tr>
<td>eMule</td>
<td>2/2</td>
<td>0/2</td>
</tr>
<tr>
<td>Vuze</td>
<td>2/2</td>
<td>0/2</td>
</tr>
<tr>
<td>FrostWire</td>
<td>2/2</td>
<td>0/2</td>
</tr>
<tr>
<td>uTorrent</td>
<td>2/2</td>
<td>0/2</td>
</tr>
<tr>
<td>Storm botnet</td>
<td>13/13</td>
<td>12/13</td>
</tr>
<tr>
<td>Waledac botnet</td>
<td>3/3</td>
<td>3/3</td>
</tr>
<tr>
<td>Total</td>
<td>30/31</td>
<td>P2P bot: 15/16, Legal P2P: 1/14</td>
</tr>
</tbody>
</table>

In Module 2, we detect all kinds of P2P hosts. Detection includes legitimate as well as malicious P2P hosts. The results of this stage have only one host running Skype not detected as P2P (false negative). Other P2P hosts are detected with 100% accuracy. Thus, overall accuracy of this stage is 96.8% (30 out of 31 hosts).

In Module 3, using heuristics from Section 3.3, we differentiate between legal P2P hosts and P2P hosts running bot code. As you can see, almost all hosts running Storm or Waledac bot code have been identified correctly with accuracy of 93.7% (15/16). Furthermore, one legal P2P host running Skype application was misclassified as malicious host with false positive rate of 7% (1/14).

Table 2 shows our threshold settings for our implementation.

<table>
<thead>
<tr>
<th>Threshold</th>
<th>Corresponding feature</th>
<th>Threshold value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Θ1</td>
<td>DNS packets</td>
<td>&lt;5</td>
</tr>
<tr>
<td>Θ2</td>
<td>Distinct destin. subnets</td>
<td>≥100</td>
</tr>
<tr>
<td>Θ3</td>
<td>Distinct destin. IPs</td>
<td>≥600</td>
</tr>
<tr>
<td>Θ4</td>
<td>runtime capturetime</td>
<td>0.6</td>
</tr>
<tr>
<td>Θ5</td>
<td>matching IPs/Total IPs</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Θ1 represents the number of any DNS packets exchanged during 10 minutes time interval. Θ2 represents the number of distinct destination subnets. Furthermore, Θ3 represents the number of distinct destination IP addresses. The last two features are introduced to differentiate between normal P2P and malicious P2P traffic. Θ4 set to 0.6 means that botnets exhibit communication more than 60% of the capture time. Moreover, Θ5 set to 0.7 means that 70% of destination IPs are same within the botnet.

Note that these thresholds are targeted to be set by network administrator.

V. Conclusion and Future work

Our contribution from this work can be described from multiple perspectives. First of all, we have developed unsupervised method for botnet detection, meaning we do not require any labeled data for training the system. Secondly, the accuracy of the system can be compared to the state-of-the-art detection methods. Furthermore, threshold setting makes it
customizable for network administrators. Lastly, our system is inherently scalable due to development in Hadoop environment.

In the future work we plan to extend the system into application profiling framework. Also benchmark of the system on large volume dataset in a cluster environment is required.

References


