BOTNET: Lifecycle, Architecture and Detection Model

Divya Nanda
Department of Computer Engineering, Pune University, India
divya.nanda01@gmail.com

Pooja Wadhwa
Department of Computer Engineering, Pune University, Pune
socutediawadhwa@gmail.com

Sanjay Singh
Department of Computer Engineering, Pune University, Pune
singh_sjy@yahoo.co.in

Deepak Kumar
Department of Computer Engineering Pune University, Pune
deepakkumaratwork@rediffmail.com

***Abstract* - Global Internet threats are undergoing a profound transformation from attacks designed solely to disable infrastructure to those that also target people and organizations. Behind these new attacks is a large pool of compromised hosts sitting in homes, schools, businesses and governments around the world. These systems are infected with a *bot* that communicates with bot *controller* and other bots to form what is referred to as a *botnet.* BOTNET is a large network of compromised computers used to attack other computer systems for malicious intent [1]. So, Botnets are networks of malware-infected machines that are controlled by an adversary, and which are the cause of a large number of problems on the internet. They are increasing faster than any other type of malware and have created a huge army of hosts over the internet. By coordinating themselves, they are able to initiate attacks of unprecedented scales. We start from the definition and essential properties of botnets. As defined above, a botnet is a coordinated group of malware instances that are controlled via C&C communication channels. The essential properties of a botnet are that the bots communicate with some C&C server/peers, perform malicious activities, and do so in a similar or correlated way. Accordingly, our detection system clusters similar communication traffic and similar malicious traffic, and performs cross cluster correlation to identify the hosts that share both similar communication patterns and similar malicious activity patterns. These hosts are thus bots in the monitored network.**

This paper presents an approach to understand, design and implement a Botnet Detection System. In order to achieve this, a detailed analysis of the current Botnet Models, its architecture, threat and impact is studied and Botnet Detection software, called "Bot Digger" is to be designed and implemented. Botnets are now the key platforms for many Internet attacks, such as spam, distributed denial-of-service (DDoS), identity theft and phishing. Most of the current botnet detection approaches works only on specific command and control (C&C) protocols (e.g., IRC, HTTP, etc.) and structures (e.g., centralized, unstructured etc.), can become ineffective as botnets change their C&C techniques. The aim of this paper is to research about the Botnets and develop a Botnet detection System, by using a general detection framework that is independent of botnet C&C protocol and structure.

I. INTRODUCTION

Internet Security is one of the major concerns all over the world and a lot of it is being weakened through Botnets operating worldwide. The Botnet is commandeered by a "botmaster" and utilized as a "platform" for attacks and activities such as spam, phishing, identity theft etc. In order for a botmaster to command a botnet, there needs to be a command and control (C&C) channel through which bots receive commands and coordinate attacks and fraudulent activities. The C&C channel is the means by which individual bots form a bot*net.* We start by exploring the life cycle of Botnet which consists of a linear sequence of stages.

A) *Botnet Life-Cycle*

There are six stages associated here and the final stage i.e attack success, is reached only after all previous stages have been successfully carried out.

1. *The Conception Stage*:

- Motivation – The motivations of a botmaster could be classified as: Money, Entertainment, Ego, Cause, Entrance to social groups, and Status.
- Design – To design the desired botnet, several aspects are carefully considered during this process, especially those regarding the bot infection and botnet communications. However, the key decision on the design i.e. architecture of the botnet architecture could be: *Centralized* – unique command & control (C&C) server, *distributed* or *P2P based*, all the bots of the botnet act simultaneously as servers and clients, or *hybrid* or *unstructured*.
- Implementation – Once the botnet is conceptually conceived and designed, the last process of this stage is the own implementation of the bot code, following a traditional software development process.

2. *The Recruitment Stage (or Infection Stage)* **–** will deploy the botnet software for its operation in a real environment. A user may be infected from execution of an attachment in a fake email or opening of a binary resource downloaded from a P2P network.

3. *The Interaction Stage***,** this stage refers to all the interactions performed during the botnet operation. One of the main differences between botnets and other type of malwares is the existence of communications by using C&C messages.

4. *The Marketing Stage***,** At this point, the botnet has been created and it is plenty of functionality after the previous stages. Now, the botmaster needs some motivation to use it. The expected economical profit is usually obtained by - Selling the botnet code or, Renting the botnet code or its services.

5. *The Attack Execution Stage* **–** The final goal of a botnet is the execution of an attack.

II. BOTNET ARCHITECTURE

1. Star or Centralized C&C Topology

The Star topology relies upon a single centralized C&C resource to communicate with all bot agents. Each bot agent is issued new instructions directly from the central C&C point [2].

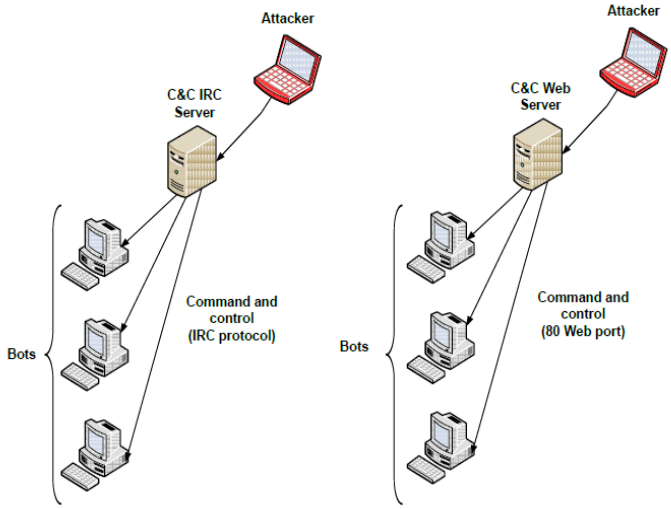

Figure 1: Centralized C&C Servers

2. Multi-Server C&C Topology

Multi-server C&C topology is a logical extension of the Star topology, in which multiple servers are used to provide C&C instructions to bot agents. These multiple command systems communicate amongst each other as they manage the botnet. Should an individual suffer fail or be permanently removed, commands from the remaining servers maintain control of the botnet [2].

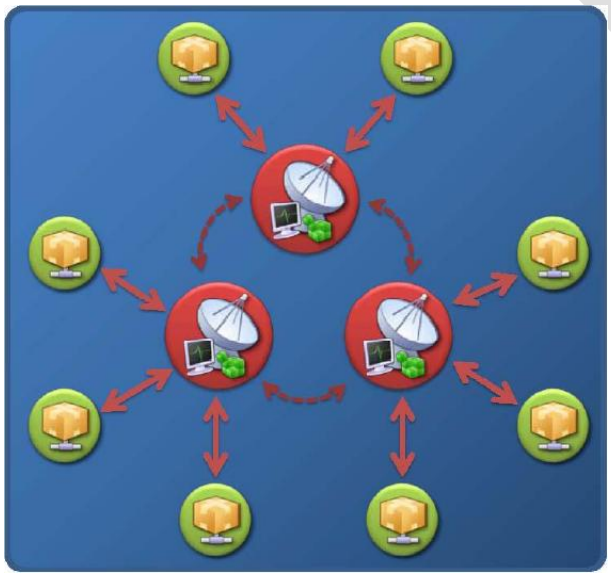

Figure 2:Multi-server C&C Topology

3. Hierarchical C&C Topology

A Hierarchical topology reflects the dynamics of the methods used in the compromise and subsequent propagation of the bot agents themselves. The command instructions suffer latency issues making it difficult for a botnet operator to use the botnet for real-time activities [2].

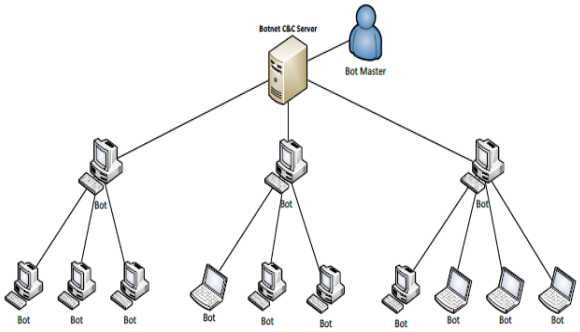

Figure 3: Hierarchical C&C Topology

4. Peer-to-Peer Topology

Millions of users are daily sharing programs, movies and games. Each host periodically connects to its neighbor to retrieve orders from the Botmaster. The Botmaster only need to connect to one of the Bots (peer) to send his commands all over the network [2].

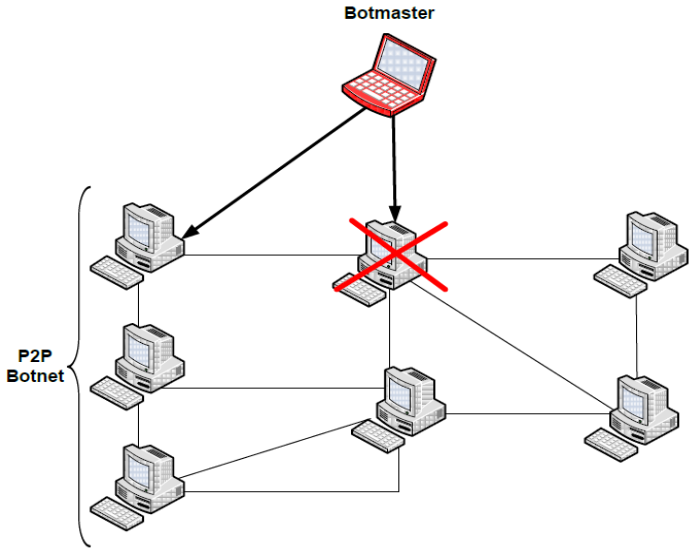

Figure 4:P2P Hybrid C&C Topology

5. Unstructured or Random C&C Topology

Each Bot has the ability to scan the internet in order to find another Bot. Random botnets are highly resilient to shut down and hijacking because they lack centralized C&C and employ multiple communication paths between bot agents [2].

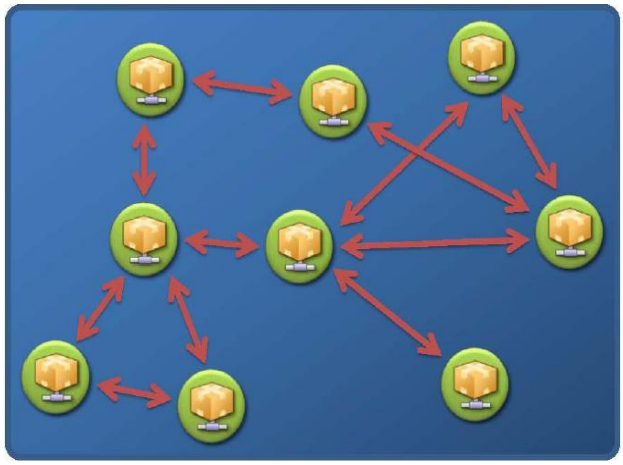

Figure 5: Random C&C Topology

Along with the Botnet Architecture, another very important factor guiding the making of a powerful Botnet is the type of communication protocol being used. Below table summarizes our study of different communication protocol based botnets.

TABLE 1: COMPARISION OF COMMUNICATION PROTOCOLS

Protocol	Ease of Setup	Efficiency	Effectiveness	Robustness
IRC	Simple	High	High	Low
HTTP	Medium	High	High	High
P2P	Complex	Low	Medium	High

III. BOTNET DETECTION

Botnet detection and tracking has been a major research topic in recent years. Researchers have proposed a few approaches [3,4,5] to detect the existence of botnets in monitored networks. Almost all of these approaches are designed for detecting botnets that use IRC or HTTP based C&C. For example, Rishi [4] is designed to detect IRC botnets using known IRC bot nickname patterns as signatures. Another more recent system, BotSniffer [5], is designed mainly for detecting C&C activities with centralized servers. According to our studies, the botnet detection techniques can be classified into three, namely,

- Honeypot[6]
- Passive network traffic monitoring and analysis, and
- Based on traffic application.

1. Honeypot based detection

Generally it consists of a computer, data or a network site that appears to be part of network, but is actually isolated and monitored, and which seems to contain information or a resource of value to attackers. Honeypots are mostly useful to understand botnet technology and characteristics, but do not necessarily detect bot infection.

2. Traffic Application based Detection

Botnet detection techniques based on traffic application classification are usually guided by botnet and C&C control protocol e.g. if one is only interested in IRC-based botnets then traffic will be classified into IRC and non-IRC groups.

3. Passive network traffic monitoring and analysis based detection

Botnet detection techniques based on passive traffic monitoring have been useful to identify the existence of botnets. These techniques can be classified as being signature-based, anomaly-based, DNS-based, and mining-based that will be described and summarized in this section respectively.

A. *Signature-based Detection*

Knowledge of useful signatures and behavior of existing botnets is useful for botnet detection. For example, Snort is an open source intrusion detection system (IDS) that monitors network traffic to find signs of intrusion. However, this solution is not useful for unknown bots.

B. Anomaly-based Detection

Anomaly-based detection techniques attempt to detect botnets based on several network traffic anomalies such as high network latency, high volumes of traffic, traffic on unusual ports, and unusual system behavior that could indicate presence of malicious bots. Although anomaly detection techniques solve the problem of detecting unknown botnets, problems with anomaly detection can include detection of an IRC network that may be a botnet but has not been used yet for attacks, hence there are no anomalies.

C. DNS-based Detection

DNS-based detection techniques are based on particular DNS information generated by a botnet. DNS-based detection techniques are similar to anomaly detection techniques as similar anomaly detection algorithms are applied on DNS traffic. In order to access the C&C server bots perform DNS queries to locate the respective C&C server that is typically hosted by a DDNS provider. Thus, it is possible to detect botnet DNS traffic by DNS monitoring.

D. *Mining-based Detection*

One effective technique for botnet detection is to identify botnet C&C traffic. However, botnet C&C traffic is difficult to detect. Several data mining techniques including machine learning, classification, and clustering can be used efficiently to detect botnet C&C traffic.

IV. PROPOSED MODEL

A. Problem Statement and Assumptions

According to our definition, a botnet is characterized by both a C&C communication channel (from which the botmaster's commands are received) and malicious activities (when commands are executed). Regardless of the specific structure of the botnet (centralized or P2P), members of the same botnet (i.e., the bots) are coordinated through the C&C channel. This is largely due to the fact that bots are non-human driven, pre-programmed to perform the same routine C&C logic/communication as coordinated by the same botmaster. In the case, the botmaster chooses to divide a botnet into *sub-botnets*, for example by assigning different tasks to different sets of bots wherein each sub-botnet will be characterized by similar malicious activities and C&C communications patterns, and our goal is to detect each sub-botnet. Hence our assumption holds true in such a case too.

B. Objectives

The objective of our paper is to detect *groups* of compromised machines within a monitored network that are part of a botnet. We do so by making use of anomaly - based and mining (Clustering) detection approach of botnets.

We do not aim to detect botnets at the very moment when victim machines are compromised and infected with malware (bot) code. In this paper we are not concerned with the way internal hosts become infected (e.g., by malicious email attachments, remote exploiting, and Web drive-by download). We focus on the detection of groups of already compromised machines inside the monitored network that are part of a botnet.

C. BotDigger: Architecture

Bot digger's architecture consists of five main components: Control Center, Monitoring Engine, Clustering Engine, Correlation Engine and UI module. Control Center is the core of the Bot Digger system. It is responsible for managing all other components of the system. It is responsible for coordinating all the actions, executions and operations of each and every component for efficient and smooth functioning of the system. The Monitoring Engine is responsible for logging network flows in a format suitable for efficient storage and further analysis and for detecting suspicious activities (e.g., scanning, spamming and exploit attempts).

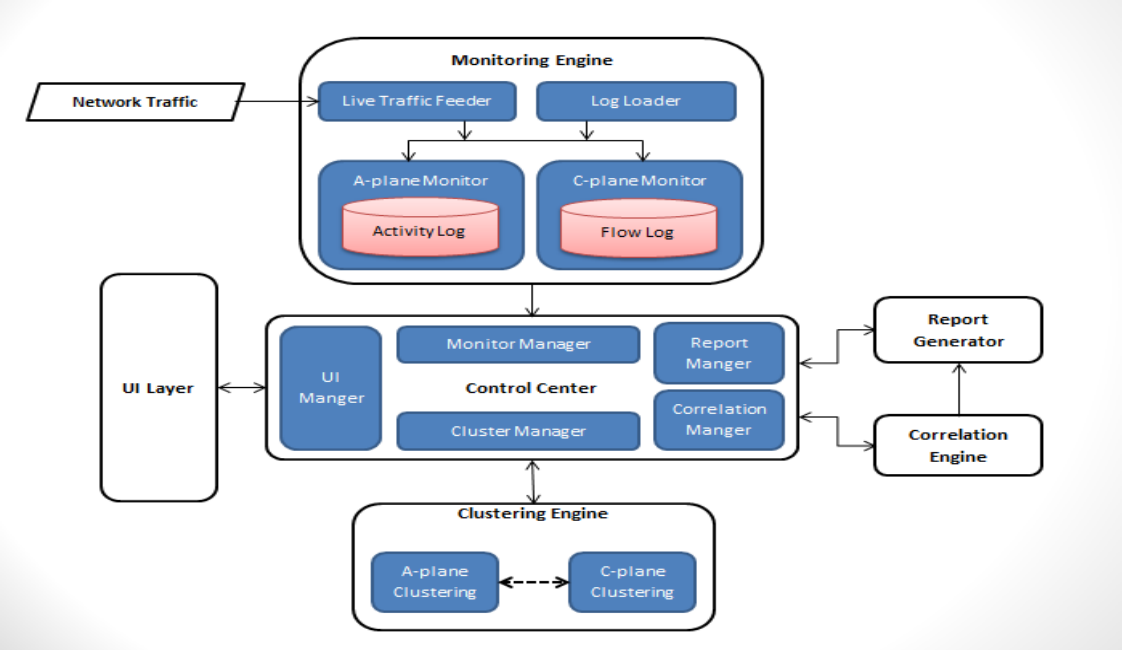

Figure 6:BotDigger's Architecture

A- Plane and C- plane Clustering is done in the Clustering Engine. We perform a two-layer clustering on activity logs (generated by A-plane monitor of Monitoring Engine). For the whole list of clients that perform at least one malicious activity during one day, we first cluster them according to the types of their activities (e.g., scan, spam, and binary downloading). This is the first layer clustering. Then, for each activity type, we further cluster clients according to specific activity features (the second layer clustering). C-plane clustering is responsible for reading the logs generated by the C-plane monitor and finding clusters of machines that share similar communication patterns. First of all, we filter out irrelevant (or uninteresting) traffic flows. This is done in two steps: basic-filtering and white-listing. Next, we further reduce the traffic workload by performing aggregation of related flows into communication flows (C-flows) as follows:

- Aggregation of traffic flows:

Given an epoch E (typically one day), all *m* TCP/UDP flows that share same protocol (TCP or UDP), source IP, destination IP and port, are aggregated into the same C-flow c_i.

$$c_i = \{f_j\}_{j = 1..m}$$

where each f_j is a single TCP/UDP flow.

Basically, the set $\{c_i\}_{i = 1..n}$ of all the n C-flows observed during E tells us "who was talking to whom", during that epoch.

- Vector Representation of C-flows

The objective of C-plane clustering is to group hosts that share similar communication flows. This can be accomplished by clustering the C-flows. In order to apply clustering algorithm to C-flows, we first need to translate them in a suitable vector representation.

We *extract* a number of statistical features from each C-flow c_i, and translate them into d-dimensional pattern vectors, $p_i \in R^d$.

We can describe this task as a projection function F: C-plane $\rightarrow R^d$.

The projection function F is defined as follows. Given a C-flow c_i, we computer the discrete sample distribution of (currently) four random variables:

a) Number of flows per hour(fph)
fph is computed by counting the number of TCP/IP flows in c_i that are present for each hour of that epoch E.

b) Number of packets per flow(ppf)
ppf is computed by summing that total number of packets sent within each TCP/IP flow in c_i

c) Average number of bytes per packets(bpp)
bpp is computed for each TCP/UDP flow $f_i \in c_i$ by dividing the overall number of bytes transferred within f_j by the number of packets sent within f_j.

d) Average number of bytes per second(bps)
bps is computed as the total number of bytes transferred within each $f_i \in c_i$ divided by the duration of f_j.

Given the discrete sample distribution of each of these four random variables, we compute an approximate version of it by means of a *binning* technique. For example, in order to approximate the distribution of *fph* we divide the x-axis in 13 intervals as $[0, k_1)$, $(k_1, k_2]$,…, (k_{12}, ∞). The values k_1, …, k_{12} are computed as follows.

a) Compute the overall discrete sample distribution of *fph* considering all the C-flows in the traffic for an epoch E.
b) Compute the quantiles $q_{5\%}$, $q_{10\%}$, $q_{15\%}$, $q_{20\%}$, $q_{25\%}$, $q_{30\%}$, $q_{40\%}$, $q_{50\%}$, $q_{60\%}$, $q_{70\%}$, $q_{80\%}$, $q_{90\%}$ of the obtained distribution.
c) And at last, we set $k_1 = q_{5\%}$, $k_2 = q_{10\%}$ etc.

Now, for each C-flow we can describe its *fph* (approximate) distribution as a vector of 13 elements, where each element *i* represents the number of times *fph* assumed a value within the corresponding interval $(k_{i-1}, k_i]$.

We can apply the same algorithm for *ppf*, *bpp* and *bps*, and therefore we map each C-flow c_i into a pattern vector p_i of $d = 52$ elements.

Now we explain how our Correlation Engine works. After obtaining the clustering results from A-plane (activities patterns) and C-plane (communication patterns), the idea is to crosscheck these clusters in the two planes to find out intersections that reinforce evidence of a host being part of a botnet.

For this, Let H is the set of hosts reported in the output of the A-plane clustering module, and $h \in H$. Let A1 be the cluster of hosts that were found to perform scanning and were grouped with h in the same cluster. Also, let A2 be a cluster related to exploit activities that includes h and other hosts that performed similar activities. A larger overlap between A1 and A2 would mean a larger possibility of a stronger[1] bot (h). Similarly, if h belongs to A-clusters that have a large overlap with C-clusters, then it means that the hosts clustered together with share similar activities as well as similar communication patterns

V. CONCLUSION AND FUTURE WORK

With the steep rise in computer network attacks mostly due to Botnets, has significantly highlighted the issue to work on effective and efficient remedy for Botnet. Through this paper we analyzed the existing Botnets, the detection techniques and proposed a novel network anomaly and mining based Botnet Detection System.

In our future work, we will study new techniques to monitor/cluster communication and activity patterns of botnets that will be more robust to evasion attempts. In addition, we plan to further combine different correlation techniques (e.g., vertical correlation and horizontal correlation), and develop techniques to work in very high speed and very large network environments.

VI. REFERENCES

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

[2] Botnet Detection Literature Review, Benoit Jacob, Edinburgh Napier University School of Computing 2008.

[3] J. R. Binkley and S. Singh. An algorithm for anomaly-based botnet detection. In *Proceedings of USENIX SRUTI'06*, pages 43–48, July 2006

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

[5] G. Gu, J. Zhang, and W. Lee. BotSniffer: Detecting botnet command and control channels in network traffic. In Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08), 2008.

[6] Honeynet Project and Research Alliance. Know your enemy: Tracking Botnets, March 2005. See http://www.honeynet.org/papers/bots