2014 6th International Conference on Cyber Conflict
P.Brangetto, M.Maybaum, J.Stinissen (Eds.)
2014 © NATO CCD COE Publications, Tallinn

An Automated Bot Detection System through Honeypots for Large-Scale

Fatih Haltaş
Cyber Security Institute
The Scientific and Technological Research
Council of Turkey
Ankara, Turkey
fatih.haltas@tubitak.gov.tr

Abdulkadir Poşul
Cyber Security Institute
The Scientific and Technological Research
Council of Turkey
Kocaeli, Turkey
abdulkadir.posul@tubitak.gov.tr

Erkam Uzun
Computer Engineering
TOBB University of Economics
& Technology
Ankara, Turkey
euzun@etu.edu.tr

Bakır Emre
Cyber Security Institute
The Scientific and Technological Research
Council of Turkey
Kocaeli, Turkey
bakir.emre@tubitak.gov.tr

Necati Şişeci
Cyber Security Institute
The Scientific and Technological Research
Council of Turkey
Kocaeli, Turkey
necati.siseci@tubitak.gov.tr

Abstract: One of the purposes of active cyber defense systems is identifying infected machines in enterprise networks that are presumably root cause and main agent of various cyber-attacks. To achieve this, researchers have suggested many detection systems that rely on host-monitoring techniques and require deep packet inspection or which are trained by malware samples by applying machine learning and clustering techniques. To our knowledge, most approaches are either lack of being deployed easily to real enterprise networks, because of practicability of their training system which is supposed to be trained by malware samples or dependent to host-based or deep packet inspection analysis which requires a big amount of storage capacity for an enterprise. Beside this, honeypot systems are mostly used to collect malware samples for analysis purposes and identify coming attacks.

Rather than keeping experimental results of bot detection techniques as theory and using honeypots for only analysis purposes, in this paper, we present a novel automated bot-infected

machine detection system BFH (BotFinder through Honeypots), based on BotFinder, that identifies infected hosts in a real enterprise network by learning approach. Our solution, relies on NetFlow data, is capable of detecting bots which are infected by most-recent malwares whose samples are caught via 97 different honeypot systems. We train BFH by created models, according to malware samples, provided and updated by 97 honeypot systems. BFH system automatically sends caught malwares to classification unit to construct family groups. Later, samples are automatically given to training unit for modeling and perform detection over NetFlow data. Results are double checked by using full packet capture of a month and through tools that identify rogue domains. Our results show that BFH is able to detect infected hosts with very few false-positive rates and successful on handling most-recent malware families since it is fed by 97 Honeypot and it supports large networks with scalability of Hadoop infrastructure, as deployed in a large-scale enterprise network in Turkey.

Keywords: *Botnet, honeypots, NetFlow analysis, machine learning*

1. INTRODUCTION

Attackers, astutely, perform their attacks in a well-organized and automated way by leveraging infected zombie machines, for which, enterprise network is preferable basin [1], [2]. Since, infected machines are key players in Cyber-attacks, cleansing them is one of main goals for Active Cyber Defense Systems, thereby, initial step is inevitably, diagnosis. In other words, effectuating a practical and scalable system with capability of withstanding expeditiously growing and enhancing malwares to identify infected machines in an enterprise network has high priority in technologies to be improved within the technical domain of Active Cyber Defense.

There has been extensive work on identifying infected machines, mostly rely on host-based analyses that are feeble against today's malwares with complicated hiding techniques. Acknowledging that they retain a significance role in intrusion analysis, resting only on them can be imprudent as many intruders run wild inside the network in which host machines are armored with at least a couple of host-based security solutions, while those solutions do not provide any clue to system administrators.

In the meantime, numerous researches suggest the use of network related data to detect infected machine or benefit them as auxiliary to host-based systems [3], [4] and [5]. Some detection methodologies might require deep-packet inspection that is overcharge for an enterprise and not successful in the scenario of encrypted communication preferred as command and control channel by malwares. Availability of raw data and time-scalability of processing DPI data are important obstacles to deploy an automated detection system within an enterprise network. To surmount these issues, some of detection system methodologies ([6], [7], [8]) are developed to identify infected machines by using NetFlow standard data that is widely stored in an enterprise network [9].

Because of the limited information within NetFlow data, researchers should conduct a wise statistical analysis to conclude it with malicious activity detection. For that matter, some researchers suggest to include malware families' statistical NetFlow values by leveraging the machine learning techniques and training their systems through beforehand-created models [6], [8]. Some of the challenges here are creating a successful model by using malwares that might feed detection system featly and feature selection that creates utilitarian models. Moreover, they are mostly lack of being deployed in an enterprise network because of modeling unit that requires to be trained by most recent malwares and should be kept up-to-date.

Aforementioned limitations of current automated bot-detection technologies and stealthiness of recently introduced bots, which are not only send spam or conduct DoS attack but also steal sensitive data over encrypted C&C channels [1], [10], inspire us to design a more applicable, scalable and self-updated automated individual bot detection system with high detection rate, indeed, it was a corollary of a need for such system to an enterprise network in Turkey.

In this paper, we present BFH (BotFinder through Honeypots) automated bot-infected machine detection system, based on BotFinder [8], relying on exclusively NetFlow data and leveraging the capability of Honeypots on collecting topical malware samples and utilizing the scalability of Hadoop infrastructure and MapReduce programming logic. In particular, our system consists of three important units, which are Cyber Threat Monitoring Unit, modeling and matching units which trade on Hadoop system.

Cyber Threat Monitoring System (CTMS) unit is, basically, a comprehensive system, developed by our team within the scope of European Unions SysSec project [11]. In BFH system, we benefit its capability of collecting and classifying most recent malwares through 97 honeypots, beside this; it is cultivated with the extension of an aptitude for NetFlow generation of malware families. In a nutshell, this produces preliminary data in order to feed modeling unit.

Modeling and matching units of BFH are implemented based upon BotFinder's methodology with an additional feature analysis. Multi-faceted models are acutely crafted after execution of samples for each malware family in a controlled environment, handled as component of CTMS, through using NetFlow-based features that characterize a malware family communication pattern and by identifying similarities in following demeanors; (i) temporal behavior of flows, (ii) outgoing and incoming data size characteristics, (iii) duration of connections, (iv) communication regularity, (v) data accumulation regularity. These features are also calculated, during trace extraction part, on NetFlow data of investigated enterprise network and used in matching unit and worked out to identify bot-like machine activities. Since an enterprise network consists of a numerous number of hosts and large amount of flow records that should be stored long for better results, our system leverages the Hadoop infrastructure and map-reduce programming logic[12].

An extensive evaluation of BFH is provided in a large-scale enterprise network in Turkey, BFH is deployed. Modeling unit of BFH is automatically trained by caught and classified malwares which are still active, at least in Turkish Networks, as they are caught through 97 Honeypots, that are live more than four months. Based on models, BFH runs over subjected enterprise

network whose pcap data is logged for affirmation purposes for a month. Our evaluation demonstrates that BFH is able to detect malicious activity in the network traffic of bot-infected machines with high accuracy in a reasonable scale for an enterprise. In substance, contributions of this paper are as follows:

- We introduce BFH (BotFinder through Honeypots); a vigilant automated bot-detection system, which leverages capability of Honeypots on collecting recent malwares and scalability of Hadoop infrastructure to increase applicability to an enterprise.

- We present BFH (BotFinder through Honeypots) that strengthen BotFinder's model generating approach with extra feature analysis, examining similarities of stolen data size over time in C&C communication of a bot family.

- We consolidate that C&C communication traffic of bot families has some similarities, even in most recent bot families in the wild as they are caught lively and exploit these similarities on detecting bot-infected machine by only analyzing NetFlow data that provides successful detection even on encrypted or obfuscated traffic.

2. RELATED WORK

Botnet detection studies over network data include multiple approaches. However, to our knowledge, honeypots are not actively involved in the individual bot detection systems though yet they have been benefited. BotMiner [13], BotGrep [14] and BotTrack [7] typify the approach on correlating NetFlow data and detect P2P bots through their C&C topology. They propose to identify the hosts that build up P2P networks by clustering them and discriminate rogue and benign groups benefiting the information on infected machines, gathered from several sources such as IDS and honeypots. On that sense, they are instances of bot detection systems, utilizing the use of Honeypots. However, they are restricted with IDS signatures, which may be insufficient as attackers evolve bots shrewdly to be more disguised.

Aforementioned studies do not only capitalize on NetFlow analysis. Indeed, there exist only a few papers, specifically focusing on it. For instance, Livadas et al. [15] focuses on IRC-based botnets through classification methodology based on machine-learning. Francois et al. [7] leverages PageRank algorithm on NetFlow-based approach to detect P2P botnets [16]. They both focus on particular type of traffic.

On the other hand, BotFinder detects malware infections by exploiting traffic patterns characteristics of them, yet it should be extended to detect malwares, performing non-periodic communication patterns. While not conclusive, BFH proposes a way of smoothing this out by additional feature analysis. BFH provides practicality in an enterprise network by keeping training module updated via Honeypots. It is also a significant illustration of applicability of BotFinder that BFH is a live system with some improvements, deployed to an enterprise network. Furthermore, BFH upgrades infrastructure for scalability concern to Hadoop and processes data.

Lastly, Disclosure suggests a distinct approach to detect botnet over large-scale NetFlow analysis [6]. Disclosure exhibits similar approach to BotFinder on which BFH is based upon, yet, it detects C&C servers. Giroire et al. [17] has similar approach to BotFinder, albeit, it differs in malware detection methodology.

3. SYSTEM OVERVIEW

BFH operates in two phases as training and investigation. Detection models based on statistical features are generated for each malware families in training phase. Investigation phase includes extracting same statistical feature extraction for test data and matching unit which compares test data with each of the malware family models to detect whether incoming data belongs to an infected machine or not.

Figure 1 shows an overview of the system architecture. In the training phase, after collecting malwares honeypots, a classification unit classifies the malwares in different families. Then, NetFlow data is generated after executing samples of each malware families. Afterwards, the trace extraction is conducted to the NetFlow data of each malware family and ordered connections are listed between internal and external IP addresses on a given destination port. After extracting trace data, six statistical features are calculated for each member of families. These features are the average time between the start times of two consecutive flows in the trace, the average duration of a connection, the number of bytes on average transferred to the source and to the destination, the Fourier Transformation over the flow start times in the trace and the ratio of outgoing data difference over time difference between the start times of two subsequent flows [8]. In the modelling unit, multiple binary classification models for each malware families are created by combining all the feature vectors of the members of corresponding family.

Finally, in the matching unit, each of the produced feature vectors for evaluated traces is subjected to classification via all detection models that are created in the training phase with a particular clustering algorithm in a sequential fashion. If any of the detection models raise an alert for an examined trace in the matching unit, it is assumed that the internal IP in this particular trace is infected.

4. SYSTEM DETAILS

A. Cyber Threat Monitoring System (CTMS)

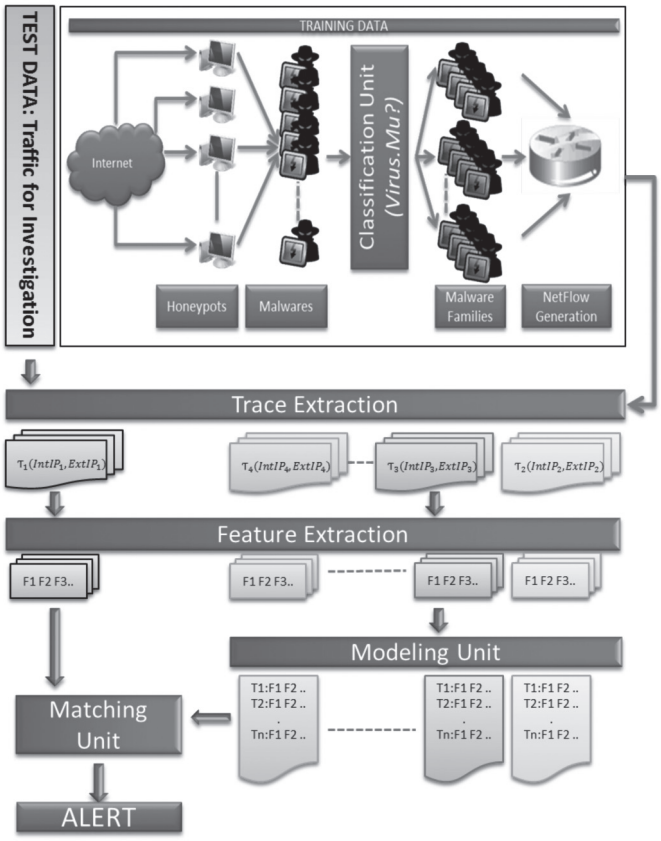
The input data we need for our detection models is collected through distributed sensors located in wide area as traffic capture. For collecting this input data we propose an infrastructure (CTMS) which comprises to two main parts; distributed sensors and malware detection centre. Malware detection centre is composed of sub modules such as virtualization servers which are hosted low and high interaction honeypots, network traffic monitoring systems such as NetFlow collection and aggregation unit, IDS and anti-virus scanners. In this step, it is important to correctly classify the collected input data through honeypots so that different samples of same malware family are analysed together. Thus, an actual classification unit which includes anti-virus scanners is used in this work.

B. Honeypots, NetFlow Generation and Classification Unit

1) Honeypots

The main feature of a honeypot is to collect attack records and malware samples by imitating networks, network services, operating systems or applications. Honeypots are classified depending on their abilities as low and high interaction or their roles as server sided and client sided. In our work, we use four types of honeypots as collectors (Col_A , Col_B) and generators (Gen_A , Gen_B) which are responsible for catching malwares from internet and generating malware communication, respectively. While URLs and attachments of spam mails and web crawlers are used as source for Gen_A , Gen_B and Col_A honeypots, other malwares are captured by Col_B honeypot. Gen_A honeypot is responsible for the execution of spam mail attachments whereas Col_A honeypot runs detected URLs from spam mails. Detailed explanations about these honeypots are as follows;

FIGURE 1: SYSTEM OVERVIEW



- Gen_A: Windows XP operating system running on a virtualization environment. Three Gen_A high interaction honeypots are used in our environment.
- Gen_B: Windows XP and Windows Vista operating systems running on a server with 2.5 GHz CPU and 2 GB RAM. Three Gen_B machines are used as sandbox.
- Col_A: A client side low interaction honeypot aimed at mimicking the behaviour of a web browser in order to detect and emulate malicious contents [18].
- Col_B: A server side low interaction honeypot that captures attack payloads and malwares [19].

2) NetFlow Generation

Gen_A executes each malware samples five or more times in CTMS environment. After executing and generating trace of flow, we restore virtual machines to clean state and then Gen_A prepares itself for new malware execution. Same process is repeated for Gen_B. Nevertheless, this process is more complicated than the cleaning state in Gen_A because of requirement of operating system reinstallation on the server.

Since some malware families are virtual machine (VM) aware that can recognize the virtualized environment and alters behaviour accordingly, we alter the settings by changing original manufacturer information of the VM with a pseudo one, removing or changing registry keys containing VM keyword, changing MAC address identified as VM Ethernet cards, changing disk settings such as serial number, firmware number etc. and killing particular service threads which indicate VM existence to delude the VM-aware malwares.

3) Classification Unit (Virus.Mu?)

A custom malware classification module called *Virus.Mu?*, (meaning *"Is it virus?"* in Turkish) similar to VirusTotal which is multi engine online virus scanner [20], is implemented by using actual versions of various antivirus products from different vendors on isolated VMs [11]. After appending malware samples and suspicious documents gathered by honeypots to a queue, each antivirus product scans the queue. If a suspicious file in the queue is identified as malicious, it is tagged based on common keyword in virus naming scheme of corresponding vendor. Then, different naming scheme correlated with the same malware family are used to get exact family name. For instance, a *Waledac* malware sample is tagged as *Email-Worm.Win32.Iksmas.gen, Mal/WaledPak-A* and *Trojan:Win32/Waledac.gen!A* by Kaspersky, Sophos and Microsoft, respectively. In addition to *Virus.Mu?*, we use Suricata-IDS with the Emerging Threats Pro Ruleset (ETPro) which delivers network based malware threat detection rule set [21], [22]. This rule set contains newly detected malwares' signatures, thus, we can validate *Virus.Mu?* and IDS alerts in our development network.

C. Features

1) Trace Extraction

As a preliminary phase for some statistical and computational features we extract traces from NetFlow data. Traces, representing consequent flows in terms of chronological order are the most commonly used concepts in bot detection algorithms. Since we apply trace extraction unit both training and investigation data, we have to whitelist common Internet services such as

Microsoft, Google, Akamai, update services, file-sharing services such as SharePoint, DropBox etc. Another filtering process is applied by comparing the destination IP with most known C&C servers. Flows are eliminated and our trained models are more likely to capture only bot traffic. As a result, the filtering process in trace extraction has a mediate impact on malware detection results.

2) Feature Extraction

Later, we utilize statistical features such as average time interval, average connection duration, average number of source bytes per flow, average number of destination bytes per flow, communication regularity and outgoing data accumulation regularity. Each statistical feature is computed on subsequent flow pairs. Features are briefly as follow: ([8] gives detailed explanations for first five ones):

- **Average Time Interval:** It reflects the average time interval between two subsequent flows in the trace. This measure detects the periodic characteristics occurred in C&C connections. Most of the malwares intent to use a constant time interval or a random interval time within a constant value between two connection periods.
- **Average Duration of Connections:** Since a malware runs same process in each connection, it is expected that the duration of different connections of a malware might be similar and different than human-computer interaction. Therefore, computing this statistic helps to distinguish a malware connection from normal ones.
- **Average Number of Source and Destination Bytes per Flow:** As the same motivation with the previous feature, it is expected that a specific C&C server will send same commands to a target machine. Thus, the average number of bytes has a characteristic structure in a C&C trace. Similar consideration will be in charge in destinations bytes. A target machine will give a fixed response to a particular C&C server.
- **Communication Regularity:** We apply Fast Fourier Transform to the binary sampled C&C communication to detect communication regularities. While doing this we sample our connection start time as 1 and 1/4th of the smallest connection interval slops as 0. Afterwards, we compute the Power Spectral Density (PSD) of the Fast Fourier Transformation over our sampled trace and extract the most significant frequency. This helps us to detect even randomly varied C&C connections within a certain range to an extent.
- **Data Accumulation:** We apply a new feature in addition to [8] for detecting malwares with randomly changed duration within two subsequent flows in a trace. This measure is calculated as average value of the each ratio of data size difference between two subsequent flows to difference of start times of them. Since the connection times of such flows may be extended because of communication problems with C&C, the accumulated data amount, which is produced by victim and stolen by an attacker, in the following connection in such a case will grow up, especially in malware with keylogger payload. Thus, characterizing the accumulated data amount per second between two connections might exhibit similarities

D. Model Creation and Detection Unit

The basic assumption behind the usage of a machine learning algorithm in detection module is that malwares leave proprietary patterns of traffic or behaviour, which could be tracked over traces, within the target machine. Our desired outcome is to raise an alert if the NetFlow data gathered from investigated traffic includes a known pattern belongs well-known and actual malwares. Thus, we use a supervised machine learning algorithm based on several statistical features, explained in previous section, instead of one of the unsupervised algorithms which do not need any training data and are mostly used to cluster similar data within isolated groups.

A supervised machine learning algorithm in malware activity detection has to address several concerns such as generality, robustness on evasion techniques, stealthiness and timely detection [23]. Firstly, the generality of the detection module represents the ability of covering a wide spectrum of malware types in the training data. Secondly, the robustness refers to the ability of recognizing different and new types of smuggling methods. Thirdly, stealthiness requires detecting a malware attack without revealing ourselves to the attacker. Moreover, the detection algorithm has to operate in on-line fashion with a reasonable respond time and high detection accuracies. Since our system upgrades itself with daily collected data through a number of honeypots, classification models cover recent malware types and are getting robust on their evasion techniques. In our method, since we analyse the trace data in passive fashion without establishing an interaction with attacker, it is not possible to draw information about detection process to the attacker. Finally, the investigation data are gathered as NetFlow data and it is a trivial operation in terms of time consumption to extract traces and statistical features. Thus, the detection system in this work is suitable for on-line operation.

In the last decade several supervised machine learning algorithms such as Support Vector Machines (SVM), Artificial Neural Networks (ANN), Decision tree classifiers, Bayesian classifiers and random forest algorithms have been proposed in botnet detection and C&C server identification [6], [24]. On the other hand, similar algorithms like these ones could be customized for botnet detection with specific feature space as applied in [8]. In this case, such techniques require a clustering phase for creating classification models in training while they need a weighted scoring methodology to identify the cluster of the investigation data. In what follows next, we introduce our modelling and matching algorithms based on six statistical features. Detailed explanation about our detection algorithm is given in [8].

1) Model Creation

In common supervised machine learning algorithms, the size and attributes of the classes in the classifier model should be introduced before triggering the training process. For instance, labels represent the malware families should be included the detection model by associating them with the feature vectors created via the traces belong that malware family in the first place before training the model in SVM algorithm, and like so many others. However, this limits to introduce the actual and new malware families to the classifier model while dynamically updating that with daily incoming data from honeypots in our situation. Therefore, we use an un-supervised machine learning approach, CLUES (CLUstEring based on local Shrinking) algorithm [25], to create detection models for each malware families. We first calculate our six statistical features separately for each trace of the training data. Then, the trace-features are

clustered by using CLUES algorithm which allows dynamic sized clustering without selecting number of clusters. A cluster which includes large number of trace-features for a particular malware family, identified through malware classification unit, represents the one of the expected values for this feature for this particular malware family. A weight is associated with each cluster in the degree of representing that malware family. Eventually, six sets of weighted clusters are created for each malware families. The average value of all of each cluster weights for a family is assigned as *cluster quality*.

2) Detection

Each features of a trace belongs to investigation data is compared individually with each of the clusters of each of the detection models which represent malware families. For instance, the first feature of a trace (*T*) is in the scope of values belong to one of the clusters in a model (*M*), then, it counts a hit. Then, the weight associated with this cluster is added to that feature's *total hit score*. If another cluster for this feature in model-M raises a *hit*, its weight is added in the same way. Then, if this feature's *total hit score* exceeds the same feature's *total hit threshold*, it counts that this feature belongs to model-*M*. Same calculations are conducted for other features of trace-*T*. Eventually, if majority of the features of trace-*T* belongs to model-*M*, an alert raises about detection of infected machine by the malware family which has the classification model as model-*M*.

E. Distributed Processing

Hadoop Distributed File System (HDFS), is a purposefully developed system for handling large files through write-once and read many data-access patterns. It has two components; name node, which is responsible for metadata of file system and management and data node that is for block storage and retrieval of data. Hadoop provides MapReduce software framework. MapReduce programming model utilizes input and output (key, value) pairs to manage processing data on different nodes.

BFH processes exclusive traces and does not require correlating any of two, thus, calculating statistical feature is easy to be implemented in distributed way. Since, its modeling and matching unit focuses on traces between $IP_{internal}$ and $IP_{external}$ entities, MapReduce programming logic is a perfect match for our system as they can be used for key values.

MapReduce methodology provides grouping and partitioning utilities to manage to group flows based on multiple entities at the same time. BFH manipulates it to store the flows that have same ($IP_{internal}$, $IP_{external}$) entities, meaning once flow start times are sorted, it extracts traces automatically. Main overhead for Hadoop is moving data over network, reading and writing to disk, yet, this type of data storing, maximizes the possibility of keeping traces in one data node, minimize the possibility of moving data over network. Performance evaluation of our system is a complete work for another paper; thus, it is not discussed in this paper. However, [26] provides ground truth on how Hadoop can outperform for enough large scale networks.

5. EXPERIMENTATION

BFH is deployed in a part of large-scale enterprise network in Turkey which has about 15000 hosts as an extension of CTMS, actively running in a production environment. NetFlow data over this network is directly extracted from Cisco devices and stored on Hadoop clusters after dumping them to text file.

For evaluation purposes, we evaluate BFH on a part of system, a network with ~8200 hosts and in daily measurement ~6300 concurrently active, which are more vulnerable to be infected as they provide services over internet (Table I) for three months. This network will be referred as "experiment network".

TABLE I: EXPERIMENT NETWORK INFORMATION

Total Number of Flows	322920000
NetFlow Size (GB)	41.4
Internal Host Count	~8200
Concurrently Active	~6300
Start Date	01-07-2013
End Date	30-09-2013
Length (Days)	92

A. Training Dataset

As SysSec Report [11] details the information on malwares caught by CTMS, our system is able to perform on a large amount of malware samples, however, to provide better estimation on performance, as Table II shows, six different malware families are discussed over time period of 15 days. Classified Malwares, caught via 97 honeypots are used to train our system. On each 15 days, traces and models are updated via accumulated malwares till that date. Table II shows sample and trace details of families over time.

1) Malwares

Carberp - Sophisticated, modular and persistent malware utilizing advanced obfuscation techniques to evade detection, removal and the ability to disable antivirus.

Hesperbot - A Trojan horse that opens a back door on the compromised computer and may steal information.

Tinba - Tiny Banking Trojan that steals information from the compromised computer.

Ramnit - A multi-component malware family which infects Windows executables as well as HTML files.

Gamarue - A malware that can download files and steal information about compromised computer.

Cridex - A malware that may be delivered via spammed malware. It captures online banking credentials entered via web browsers, downloads and executes files, and searches and uploads local files.

Aforemntioned malwares are most observed malwares within Turkish Network, thus, they have been selected in experiments.

B. Experiment

Experiment is conducted on experiment network after whitelisting for some external services that might exhibit regular behavior and increase FP rate, such as; Microsoft, Google, Akamai, update services, file-sharing services; SharePoint, DropBox etc. A BFH generated alert is analyzed by using full traffic capture, if symptoms are explicitly matched than it is signed as true alert. Meanwhile both network-based and host-based IDS/IPS alerts are also used for double-check. If there is no explicit symptom from neither full packet capture nor IDS/IPS solutions then blacklisting services are used to determine [27-31]. In case, none of these controls provide any infection implication, it is signed as False Positive, while this might not be completely true.

TABLE II: MALWARE FAMILY INFORMATION CAUGHT BY HONEYPOTS

	Start Date - End Date	01 Jul - 15 Jul	01 Jul - 31 Jul	01 Jul - 15 Aug	01 Jul - 31 Aug	01 Jul - 15 Sep	01 Jul - 30 Sep
		Number of Samples / Traces					
Malware Family	**Carberp**	3 / 8	4 / 9	18 / 18	32 / 24	42 / 31	52 / 35
	Hesperbot	4 / 4	6 / 6	9 / 10	11 / 13	14 / 21	19 / 27
	Tinba	20 / 24	32 / 30	34 / 38	38 / 45	46 / 52	49 / 62
	Ramnit	11 / 21	14 / 25	18 / 29	25 / 36	33 / 46	37 / 55
	Gamarue	25 / 24	28 / 29	31 / 35	34 / 39	38 / 43	43 / 51
	Cridex	12 / 20	16 / 25	21 / 33	25 / 39	32 / 46	36 / 50

1) Test Dataset

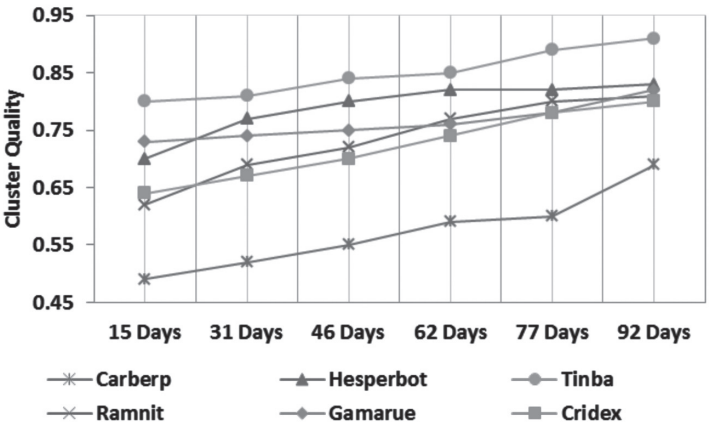
Bot Detection systems, mostly focus on off-line dataset analysis and one dataset of a large-scale enterprise network. However, in real scenarios, actively running bot detection systems are most likely to be analyzing weekly or monthly changing dataset. In our active system, created models via accumulated malwares are used to detect bots on NetFlow traffic that belongs to last four months. Since our NetFlow data changes over time, we focus on diverse dataset, which is NetFlow of each month. Consequently, our test dataset consists of three different NetFlow, stored in months: July 2013, August 2013, and September 2013.

Besides, complete traffic captures of this particular network are stored for 30 days to verify generated alerts, but, for storage limitations, it is deleted monthly. Therefore, in our experimental setup, detection rates and infected host are analyzed by using accumulated malware samples and traces after each 15 days to provide better understanding for contribution of Honeypots. More precisely, accumulated traces are used to train the system then created models are applied on subjected month's NetFlow data.

6. DISCUSSION

Figure 2 summarizes training dataset characterization for each family over time. This graphic illustrates, when number of samples increases, cluster quality for each family rises. This graphic implies that BFH, wisely, manipulates honeypots to increase cluster quality.

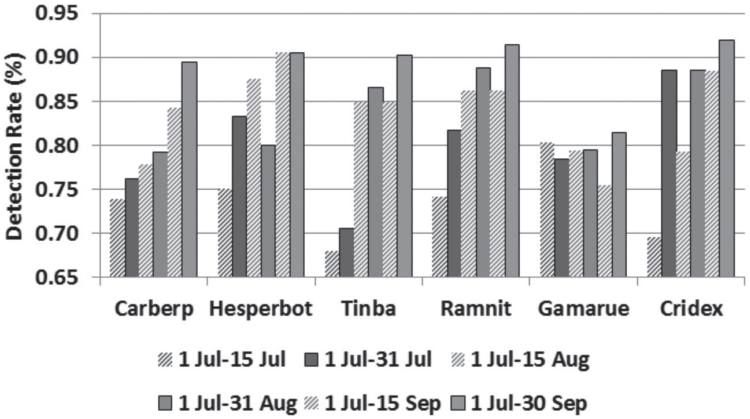
FIGURE 2: CLUSTER QUALITY OVER TIME



Interestingly, cluster quality of Carberp malware family is less than other malware families; main reason for this is that Carberp produces different number of traces from one sample. For example, in the first half of July, three samples are captured and eight traces are generated out of them. Beyond that, two factors can be considered as cause for this, one is that classification unit identifies some of the malwares as Carberp, yet, it belongs to a different family. Second, Carberp might have different variants, exhibiting diverse network characteristics.

Figure 3 is the BFH detection results. In this graphic, detection rate of each experiment on same dataset is highlighted with same color. First and foremost, Figure 3 reveals that BFH is able to detect bot-infected machines in worst case 68%, in which NetFlow data is limited to two weeks and number of samples of this particular family is less than a half of number of samples in September. Although, there is not false negative evaluation opportunity, for a detection system, having a few false-positives among a significant number of alerts (Figure 4) is an important indication of success, where BotFinder has detection rate from 49% to 87%, except Banbra family.

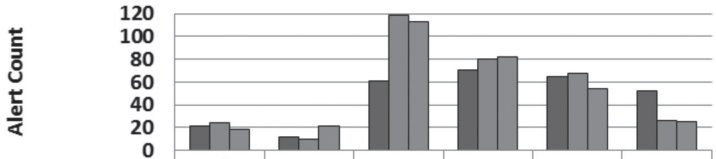
FIGURE 3: DETECTION RATE (EACH COLOR INDICATES RESULT OF EXPERIMENT ON SUBJECTED MONTH'S NETFLOW)



Secondly, detection rate for same dataset (tone-in-tone dyed) indicates, the more traces used in training, the more accurate detection, except Hesperbot in August. This family generates only 8 alerts with 1 FP on first half of August whereas it generates 10 alerts with 2 FPs. In real, it detects more bot-infected machines. Consequently, it highlights the vigilance of BFH on integrating Honeypots to bot-detection system.

Furthermore, when we compare detection rates for different datasets, in Figure 3, dashed columns of each family should be considered so as to infer that detection rates increase in monthly by improvement of samples and traces with a few exceptions, discussed on later section. Indeed, detection rate is expected to increase between second half of a month and first half of a month because system is trained with higher number of traces, yet datasets are different but hosts within the network same. However, Ramnit and Gamarue families have statistics that contradict to it. For instance; BFH has higher detection rate on end of August than beginning of September. Since experiment network involves around 8000 hosts with approximately 6300 concurrently active hosts, and active hosts are most likely to be different within different months while matching unit runs.

FIGURE 4: INFECTION ALERTS ON EACH DATASET OVER TIME



	Carberp	Hesperbot	Tinba	Ramnit	Gamarue	Cridex
Jul Test Data	21	12	61	71	65	52
Aug Test Data	24	10	119	80	68	26
Sep Test Data	19	21	113	82	54	25

7. CONCLUSION

This paper presented BFH, a live BotFinder-based automated bot-infected system through Honeypots. BFH does not require any host-based information, deep-packet inspection or any support from other network-based security deployments such as IDS/IPS. Instead, it relies on NetFlow data, uses behavioral and training-based approach so as to detect encrypted communications and avoid storage overhead, thus, it provides solution for large-scale. BFH is vigilant system, since training module of BFH is fed by samples caught via sophisticated honeypot system. BFH is deployed to a large-scale enterprise network in Turkey on Hadoop that provides scalability. Our experiment on subjected network shows that BFH is able to detect centralized bot-infected machines with high-accuracy; indeed, similar approach can be improved to detect P2P bots as future work.

8. ACKNOWLEDGEMENT

This work was an extension of European Union SysSec project and it is funded by The Scientific and Technological Research Council of Turkey (TUBITAK).

REFERENCES:

[1] Cooke, Evan, Farnam Jahanian, and Danny McPherson. "The zombie roundup: Understanding, detecting, and disrupting botnets." Proceedings of the USENIX SRUTI Workshop. Vol. 39. 2005.
[2] Freiling, Felix C., Thorsten Holz, and Georg Wicherski. Botnet tracking: Exploring a root-cause methodology to prevent distributed denial-of-service attacks. Springer Berlin Heidelberg, 2005.
[3] Bayer, Ulrich, et al. "Scalable, Behavior-Based Malware Clustering." NDSS. Vol. 9. 2009.
[4] Bayer, Ulrich, Christopher Kruegel, and Engin Kirda. "Anubis: Analyzing Unknown Binaries." (2009).
[5] Coskun, Baris, Sven Dietrich, and Nasir Memon. "Friends of an enemy: identifying local members of peer-to-peer botnets using mutual contacts." Proceedings of the 26th Annual Computer Security Applications Conference. ACM, 2010.
[6] Bilge, Leyla, et al. "Disclosure: detecting botnet command and control servers through large-scale NetFlow analysis." Proceedings of the 28th Annual Computer Security Applications Conference. ACM, 2012.
[7] Franćois, Jérôme, Shaonan Wang, and Thomas Engel. "BotTrack: tracking botnets using NetFlow and PageRank." NETWORKING 2011. Springer Berlin Heidelberg, 2011. 1-14.
[8] Tegeler, Florian, et al. "BotFinder: finding bots in network traffic without deep packet inspection." Proceedings of the 8th international conference on Emerging networking experiments and technologies. ACM, 2012.
[9] Claise, Benoit. "Cisco systems NetFlow services export version 9." (2004).
[10] Franklin, Jason, et al. "An inquiry into the nature and causes of the wealth of internet miscreants." ACM conference on Computer and communications security. 2007.
[11] (SysSec) A European Network of Excellence in Managing Threats and Vulnerabilities in the Future Internet: Europe for the World.
[12] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51.1 (2008): 107-113.
[13] Gu, Guofei, et al. "BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection." USENIX Security Symposium. 2008.
[14] Nagaraja, Shishir, et al. "BotGrep: Finding P2P Bots with Structured Graph Analysis." USENIX Security Symposium. 2010.
[15] Livadas, Carl, et al. "Usilng machine learning technliques to identify botnet traffic." Local Computer Networks, Proceedings 2006 31st IEEE Conference on. IEEE, 2006.
[16] Page, Lawrence, et al. "The PageRank citation ranking: bringing order to the web." (1999).
[17] Giroire, Frederic, et al. "Exploiting temporal persistence to detect covert botnet channels." Recent Advances in Intrusion Detection. Springer Berlin Heidelberg, 2009.

[18] Thug: The Honeynet Project [Online]. Available: http://www.honeynet.org/taxonomy/term/218
[19] Provos, Niels, and Thorsten Holz. Virtual honeypots: from botnet tracking to intrusion detection. Pearson Education, 2007.
[20] VirusTotal [Online]. Available: https://www.virustotal.com/
[21] Jonkman, M. "Suricata IDS available for download." Message posted to marc. info (2009).
[22] "Emerging Threats," [Online]. Available: http://www.emergingthreats.net/.
[23] Stevanovic, Matija, and Jens Myrup Pedersen. "Machine learning for identifying botnet network traffic."
[24] Nogueira, António, Paulo Salvador, and Fábio Blessa. "A botnet detection system based on neural networks." Digital Telecommunications (ICDT), 2010 Fifth International Conference on. IEEE, 2010.
[25] Wang, Xiaogang, Weiliang Qiu, and Ruben H. Zamar. "CLUES: A non-parametric clustering method based on local shrinking." Computational Statistics & Data Analysis 52.1 (2007): 286-298.
[26] Lee, Youngseok, Wonchul Kang, and Hyeongu Son. "An internet traffic analysis method with mapreduce." Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP. IEEE, 2010.
[27] [Online]. Available: http://www.mtc.sri.com/live_data/attackers/.
[28] [Online]. Available: http://isc.sans.edu/sources.html.
[29] [Online]. Available: http://www.projecthoneypot.org/list_of_ips.php.
[30] [Online]. Available: http://mirror1.malwaredomains.com/files/BOOT.
[31] [Online]. Available: http://www.malwaredomainlist.com/hostslist/hosts.txt.