Hades: A Hadoop-based Framework for Detection of Peer-to-Peer Botnets

Pratik Narang, Abhishek Thakur, Chittaranjan Hota
Dept. of Computer Science and Information Systems,
Birla Institute of Technology and Science-Pilani, Hyderabad Campus,
Hyderabad, India
{p2011414, abhishek, hota}@hyderabad.bits-pilani.ac.in

ABSTRACT

This paper presents `Hades`, a Hadoop-based framework for detection of P2P botnets in an enterprise-level network, which is distributed and scalable by design. The contributions of this work are two-fold: Firstly, our work uses the Hadoop-ecosystem to adopt a 'host-aggregation based' approach which aggregates behavioral metrics for each Peer-to-Peer (P2P) host seen in network communications, and uses them to distinguish between benign P2P hosts and hosts infected by P2P botnets. Secondly, we propose a distributed data-collection architecture which can monitor *inside-to-inside* LAN traffic, as opposed to relying solely on the NetFlow information available at a backbone router which cannot see the LAN communications happening in the network.

1. INTRODUCTION

In the past few years, botnets have emerged as the largest threat to modern networks. With enterprise-level networks regularly generating billions of events and gathering Terabytes of data each day, tracking malicious activity inside a network is nothing less than the proverbial *needle in the haystack* problem. The evolution of Peer-to-Peer (P2P) based botnets, which have a distributed and decentralized architecture, has created further challenges in their detection.

Traditional methods of botnet detection used signature-based or port-based approaches [11]. Such techniques were rendered useless by botnets which did not operate over fixed ports or used encryption. Although a lot of research has gone into detection and take-down of botnets (P2P or otherwise), not much work has been seen which identifies botnets based on their network behavior.

Detecting P2P botnets is a challenging task because P2P botnet traffic can very easily blend with benign P2P traffic in a network, like that of Skype, BitTorrent, eMule etc. Although some past work evaluated the detection of P2P botnets in Internet traffic [4], the detection of P2P botnets in the presence of benign P2P traffic is a challenging scenario. Furthermore, scalable detection approaches have received very little attention (such as in [10]).

Most of the previous work utilizing network behavior of botnets uses the traditional 5-tuple flow-based analysis of network traces. The 'flow-based' approach classifies packets on the basis of the 5-tuple: `source IP, destination IP, source port, destination port, protocol`. Some of the recent work has used 5-tuple based flow analysis along with supervised [7] and unsupervised [10] machine learning techniques or other statistical measures [9] in order to separate P2P botnet traffic from benign traffic.

The 'flow' information is typically obtained in the form of Cisco's NetFlow (or by using tools like Argus[1]) from a backbone router of an enterprise. Large-scale networks may involve multiple border routers. The NetFlow data collected at one router will not give a complete picture of the communications which happened to and from the network. A distributed data collection approach – where data collectors sit closer to the nodes in the network – can give a much better view of the communications. Such a distributed approach is especially beneficial and essential for the detection of smart P2P bots *inside* the perimeter of a network, which talk to each other and send upgrades to themselves on LAN in a P2P fashion, and limit communication to the outside world via one or two peers only. The activity of such bots, which communicate to each other on LAN in a P2P fashion, cannot be detected by traditional 'flow-based' approaches which only monitor the data crossing the backbone router(s).

Flow-based approaches are known to suffer from another drawback [6] that many modern P2P applications and P2P botnets randomize their port numbers and switch their communication between TCP and UDP. The classical 'flow' definition relies of port numbers and protocol, and cannot give the true picture of the communications that the P2P hosts inside the network are engaged in.

In this work, we present `Hades`, which is an acronym for '**H**ost-**a**ggregation based **de**tection **s**ystem' for P2P botnets. `Hades` utilizes the distributed computing power of the Hadoop ecosystem to parse large network traces and extract 'behavioral' features for every P2P host seen in network communications. The extracted feature-set is then used to train supervised machine learning models which can differentiate P2P botnets from P2P applications. `Hades` does not require signature-based detection approaches, Deep Packet Inspection (DPI) or a 'seed' information of bots obtained

The 20th International Conference on Management of Data (COMAD), 17th-19th Dec 2014 at Hyderabad, India.
Copyright ©2014 Computer Society of India (CSI).

[1]http://qosient.com/argus/

from a blacklist of IPs. Hades just relies on the header information in the network and transport layer, and extracts statistical features which quantify the 'P2P' behavior of different kinds of P2P applications.

Hades addresses certain limitations of past works and makes the following contributions:

1. Hades is built on top of the Hadoop ecosystem, which is a *de facto* standard for big data analytics. Since it utilizes the power of distributed computing through Hadoop [2], Hades is scalable by design.

2. We propose a distributed data collection architecture wherein data collectors are distributed at multiple locations inside an enterprise network and sit close to the peers, say at an Access switch or a Wi-fi access point. This approach allows *inside-to-inside* communication view, which can be vital for detecting P2P botnets inside a network which communicate to each other over LAN.

3. Hades does not rely on the traditional 'flow' definition. We adopt a Host-aggregation based approach which obtains statistical features *per host* for all P2P hosts involved in network communications.

In order to facilitate reproducible research, we also discuss the implementation aspects of Hades in detail.

2. SYSTEM DESIGN AND IMPLEMENTATION DETAILS

The system design of Hades employs the libpcap library for collecting and parsing network traces. It utilizes the Hadoop ecosystem for aggregation of host-based data and for building scalable models for the detection of P2P botnets. Hades has been implemented on top of the Hadoop ecosystem with the open-source projects of Apache Hive [8] and Apache Mahout [1]. The system architecture of Hades is given in Figure 1.

2.1 Distributed Data Collection

Instead of relying upon NetFlow data obtained at a backbone router, Hades proposes a distributed data-collection technique wherein data collectors sit close to the peers inside the network perimeter. As mentioned above, this allows Hades to have a view of the conversations which happen inside-to-outside (or vice-versa) as well as inside-to-inside.

The implementation of Hades has multiple data collectors distributed inside the network perimeter. The initial deployment of the system has data collectors deployed at Wi-fi access points within the University campus of the authors. The multiple data collectors consist of commodity-grade hardware machines with 2 GB RAM, 2 CPU cores and 200 GB of disk space. The Wi-fi access points used are NetGear N150 and Belkin N150. Each data collector uses a libpcap library based module to capture traffic in the form of network traces .pcap files. Each data collector runs an automated parser module (built with libpcap library and Python) which parses the network traces and extracts packet-level features of interest to us. Features are extracted from the IP header and TCP/UDP header, and no DPI is required. For this work, the features extracted from each packet are:

1. Time-stamp of the packet

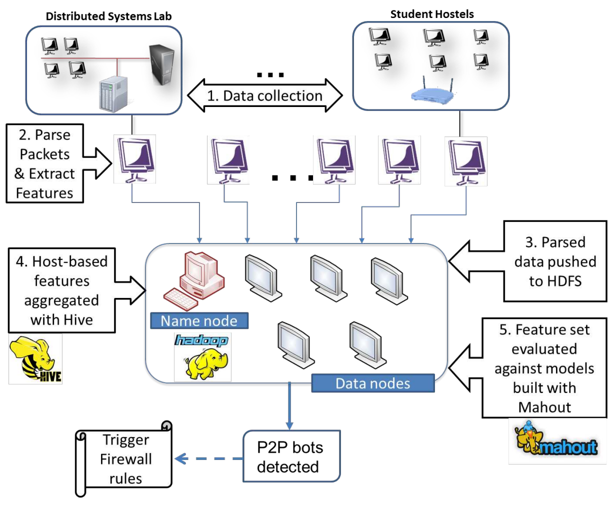
Distributed Systems Lab
Student Hostels
1. Data collection
2. Parse Packets & Extract Features
3. Parsed data pushed to HDFS
4. Host-based features aggregated with Hive
Name node
hadoop
Data nodes
5. Feature set evaluated against models built with Mahout
HIVE
mahout
Trigger Firewall rules
P2P bots detected

Figure 1: Hades: System Architecture

2. Source IP

3. Destination IP

4. Time-to-live (TTL) value

5. Transport layer protocol (TCP/UDP)

6. TCP or UDP payload length (as applicable)

The extracted features are stored in a .csv file at each data collector. Instead of transferring large .pcap files, these .csv files are periodically transferred from all data collectors to the Hadoop Distributed File System (HDFS) [3]. For the purpose of data sanitization, all packets which are found to *not* contain a valid IPv4 header are removed (E.g- corrupted packets). Presently Hades does not support IPv6. The present approach of Hades also disregards all packets corresponding layers below the IP layer, such as ARP broadcast messages. The implications of this choice will be further discussed in section 4.

2.2 Host data aggregation

The Hadoop cluster deployed for Hades consists of a 'name node' Virtual Machine with 8 GB RAM, 8 CPU cores and 200 GB disk space, and ten 'data node' Virtual Machines, each having 2 GB RAM, 2 CPU cores and 200 GB of disk space. Each Virtual Machine runs Ubuntu 12.04 Operating System.

Packet-level data obtained from multiple data collectors is aggregated *per host* for every host seen in network communication. The packet-level data is stored in Hive in the form of external tables. Hive commands are written in HQL (Hive query language) which is very similar to SQL [8]. The Hive command used to create the table for storing packet-level data is given here:

```
CREATE EXTERNAL TABLE packet_data (
    timestamp DECIMAL, ip_source STRING,
    ip_destination STRING, ttl INT,
    proto INT, payload_length INT )
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION '/user/hdfs/PacketDump';
```

For the task of detecting P2P botnets, we aggregate the following statistical features over a time-period T (say, one hour) for every P2P host inside the network:

1. **Number of distinct destination hosts contacted:** P2P hosts are involved in sharing content and downloading download different chunks of a file from different peers across the globe. A benign P2P host might be involved in downloading a certain file (or its chunk) from Adelaide, uploading a file to another peer at Birmingham, and download music content from a peer at California. As a normal user of the Internet engaged in P2P file sharing, a benign P2P host is expected to contact a number of peers in the different parts of the world, with no specific pattern involved in the destinations contacted. Moreover, due to the sheer size of these networks, most interactions in P2P file sharing networks are one-time transactions, where peers who share content with each other may never interact again. But, the behavior of hosts infected by P2P botnets gives a contrast. Bot-peers do not engage in file sharing or downloads. Rather they regularly and repeatedly contact their set of bot peers to receive or propagate commands and updates. Thus the number of unique destination hosts contacted by bot-peers are expected to be less as compared to benign P2P hosts.

2. **The total volume of data sent from the source host:** As stated above, benign P2P hosts are engaged in file transfers, downloads, uploads etc. whereas bot hosts are not expected to be engaged in these activities. The volume of data sent from a benign host is, quite clearly, expected to be more than the exchange of data seen at bots which are primarily involved in exchange of Command & Control information.

3. **The average of the TTL value of the packets sent from the source host:** A user of P2P file sharing systems who is involved in downloading some music content will not bother whether the seeding peers happen to be from his home country or some other part of the world. Rather, while the user might himself be situated in India, he may download one part of the file from a peer in China, another chunk from a peer in Holland, and another from a peer in Australia. Since the file requests of benign P2P users travel all over the world, these requests typically have high TTL values associated with them. In contrast, bot hosts tend to repeatedly contact their set of bot-peers. For the sake of efficient design and avoiding latency/overheads, botmasters would not want their bots to talk to peers in different parts of the world. Bots are expected to engage in communication with other bot peers near to them. This leads to the requests sent by bots having lower TTL values when compared to requests seen from benign P2P hosts.

The host-aggregated features described above are stored in another table:

```
CREATE TABLE host_data (
    host STRING, destinations DECIMAL,
    avg_ttl DECIMAL, volume BIGINT )
ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,’
LINES TERMINATED BY ‘\n’ STORED AS TEXTFILE;
```

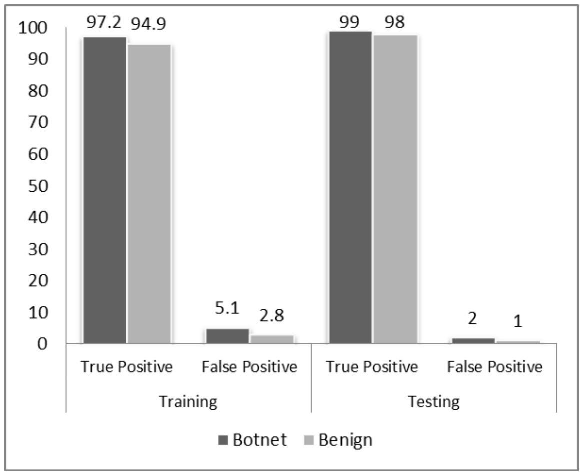

Figure 2: True Positive rate and False Positive rate with training and testing data for Random forests of ten trees

Since the packet-level data arrives from different data collectors periodically, a Hive script is run periodically to convert it into host-aggregated form and store it in the table `host_data` created above. The Hive script given below uses a ‘GROUP BY’ operation to obtain data in host-aggregated format:

```
INSERT INTO TABLE host_data
SELECT ip_source, COUNT (DISTINCT ip_destination),
       AVG(ttl), SUM(payload_length)
FROM packet_data
GROUP BY ip_source;
```

2.3 Detecting P2P bots from hosts

The host-based features extracted above are used to train and test supervised machine learning models. Apache Mahout is used for this purpose. Mahout is a fairly new tool, and at present does not offer many machine learning algorithms. Further, many of Mahout's algorithms (for classification and clustering) do not run as MapReduce jobs. Parallelized implementations are important for scalability of `Hades` over large datasets. Thus, for this work, we stick to the Random Forest implementation of Mahout which is a parallelized implementation (in contrast to other implementations like Linear regression, AdaBoost etc., which are not). More details on the data used are given in the next section.

Results generated from `Hades` can be used to alert a network administrator for suspicious activity in the network, trigger rules to a Firewall, and/or log or drop botnet traffic. This way `Hades` can be used by network administrators as an assisting tool which is ‘P2P-aware’.

3. EVALUATION AND RESULTS

For evaluation of `Hades`, we use P2P data obtained from a recent work at the University of Georgia [7]. Our dataset consists of network traces of two P2P applications, namely Vuze and Frostwire, and two P2P botnets, namely Storm and Waledac. The data of P2P applications was generated by Rahbarinia et al. by running these applications in their lab environment for a number of days. The data of P2P botnets was obtained by them from third-parties, and corresponds to real-world traces of these botnets.

The size of this dataset was around 20 GB (14 GB for Vuze and FrostWire, and 6 GB for Storm and Waledac) in `.pcap` format, and around 10.5 GB (6.6 GB for Vuze and FrostWire, and 3.9 GB for Storm and Waledac) when parsed to `.csv` format.

After extracting host-based features from each application, we created a 'labeled' dataset. Instances belonging to P2P hosts (Vuze or Frostwire) are labeled 'benign', while the instances belonging to P2P bots (Storm or Waledac) are labeled 'malicious'. This dataset was split into training and testing data in 2:1 ratio. With the training data, Random Forest models were built for different number of trees in each run. The models were then evaluated for their accuracy with the test data. Due to limitations of space, we only present the results for number of trees in the forest equal to 10. Figure 2 shows the accuracy obtained for the training and testing data with a Random Forest of 10 trees. Our system could detect bot-infected hosts with a True Positive rate of 97% and 99%, and a low False Positive rate of 5% and 2% over training and testing datasets respectively.

4. DISCUSSION & POSSIBLE EVASIONS

It was explained in Section 2 that `Hades` ignores messages below the IP layer, such as ARP broadcast messages. Its implications will be discussed here. In the Introduction, we argued on the case of 'smart' P2P bots which may exchange C & C with peers on a LAN and limit communication with the outside world to one or two peers. A bot-master may also configure such smart bots to utilize protocols lower than the IP layer – such as ARP messages – to facilitate communication between the bots on the same LAN. `Hades` will not be able to detect the communication of such bots since it does not deal with those messages in its present approach. Although no past work has touched upon this issue and no such botnets are known to exist at present, we argue that with the evolution of botnet detection mechanisms, bot-masters will also improvise their botnets in these ways to make them more efficient and harder to detect.

Further, the present approach of `Hades` is limited to detection of bots when bots and apps are *not* running on the *same host*. If a host which is running P2P applications is also infected with a bot, `Hades` will be unable to correctly classify it as an infected host.

5. CONCLUSIONS AND FUTURE WORK

This work presented `Hades`, an approach to collected P2P data inside a network in a distributed manner, and extract host-aggregated features to distinguish between P2P applications and botnets using supervised machine learning approaches. To the best of our knowledge, `Hades` is the first attempt at distributed data collection for the detection of P2P botnet traffic. The distributed data collection architecture proposed by us gives *inside-to-inside* visibility of traffic. With such an approach, `Hades` attempts to target the detection of 'smart' P2P bots. However, such botnets are not known to exist at present[2], and thus no network traces corresponding to such behavior could be evaluated. We plan to evaluate `Hades` on such data by generating synthetic botnet data.

[2]With the exception of Stuxnet [5], which we ignore here since it targets SCADA systems, and evaluating its detection with Internet traffic would not be possible

Acknowledgments

The authors would like to thank their teammates G. Sharath Chandra and Kamaldeep Singh for their efforts on Hadoop set-up and deployment.
This work was supported by Grant number 12(13)/2012-ESD for scientific research under Cyber Security area from the Department of Information Technology, Govt. of India, New Delhi, India.

6. REFERENCES

[1] Mahout: Scalable machine-learning and data-mining library. `http://mahout.apache.org`, Accessed on 30 November 2013.

[2] A. Bialecki, M. Cafarella, D. Cutting, and O. O MALLEY. Hadoop: a framework for running applications on large clusters built of commodity hardware. *Wiki at http://lucene.apache.org/hadoop*, 11, 2005.

[3] D. Borthakur. The hadoop distributed file system: Architecture and design, 2007. *Apache Software Foundation*, 2011.

[4] J. François, S. Wang, R. State, and T. Engel. Bottrack: Tracking botnets using netflow and pagerank. In *Proceedings of the 10th International IFIP TC 6 Conference on Networking - Volume Part I*, NETWORKING'11, pages 1–14. Springer-Verlag, Berlin, Heidelberg, 2011.

[5] L. O. Murchu. Stuxnet p2p component. `http://www.symantec.com/connect/blogs/stuxnet-p2p-component`. Accessed on 12th February 2014.

[6] P. Narang, C. Hota, and V. Venkatakrishnan. Peershark: flow-clustering and conversation-generation for malicious peer-to-peer traffic identification. *EURASIP Journal on Information Security*, 2014(1):1–12, 2014.

[7] B. Rahbarinia, R. Perdisci, A. Lanzi, and K. Li. Peerrush: Mining for unwanted p2p traffic. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 62–82. Springer-Verlag, Berlin, Heidelberg, 2013.

[8] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy. Hive: a warehousing solution over a map-reduce framework. *Proceedings of the VLDB Endowment*, 2(2):1626–1629, 2009.

[9] T.-F. Yen and M. K. Reiter. Are your hosts trading or plotting? telling p2p file-sharing and bots apart. In *Proceedings of the 2010 30th International Conference on Distributed Computing Systems*, ICDCS '10, pages 241–252. IEEE, 2010.

[10] J. Zhang, R. Perdisci, W. Lee, X. Luo, and U. Sarfraz. Building a scalable system for stealthy p2p-botnet detection. *Information Forensics and Security, IEEE Transactions on*, 9(1):27–38, 2014.

[11] Z. Zhu, G. Lu, Y. Chen, Z. Fu, P. Roberts, and K. Han. Botnet research survey. In *Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International*, pages 967–972. IEEE, 2008.