Machine Learning Based Botnet Detection

Vaibhav Nivargi Mayukh Bhaowal Teddy Lee
{vnivargi, mayukhb, tlee21}@cs.stanford.edu

CS 229 Final Project Report

I. INTRODUCTION

A Botnet [1] is a large collection of compromised machines, referred to as *zombies* [2], under a common Command-and-Control infrastructure (C&C), typically used for nefarious purposes. Botnets are used in a variety of online crimes including, and not limited to, large scale DDoS attacks, Spam, Click Fraud, Extortion, and Identity theft. The scale and geographical diversity of the machines enlisted in a Botnet, coupled with easily available source code, and support from communities, as well as mercenary Botmasters providing Botnet services for rent, have resulted in Botnets becoming a highly sophisticated and effective tool for committing online crime in recent times [3][4]. Botnets with thousands and millions of nodes have been observed in the wild, with newer ones being observe every day [10].

The lifecycle of a Botnet is depicted in Fig.1. The initial enlisting occurs by exploiting a known vulnerability in the Operating systems running on the machines using a wide variety of mechanisms like worms, Trojans, P2P file sharing networks, and exploits of common Windows vulnerabilities, etc. Once compromised, the bot is programmed to connect to a central location (typically an IRC [11] server), where the Botmaster could login and issue commands to the logged in bots. This mechanism essentially means that the communication is free, as broadcast is taken care of by the IRC channel. Most Bots additionally ship with multiple IRC and DNS addresses, meaning taking down one such IRC channel does not in general impair the activities of the Botnet.

Originally, most techniques to thwart Botnets have been *reactive*, reducing their effectiveness significantly, e.g. using Honeypots to trap and study malware, etc. Of late, significant research has been made into the dynamics of a Botnet, and more *proactive* techniques have been suggested [5] [6] [7] [8]. A wide variety of features and watermarks for network activity are employed to predict Botnet activity, including TCP syn scanning, DNS monitoring, and extensive models of Botnet attack and propagation [9]. Despite all these concerted efforts, Botnets remain an unsolved problem for the online community.

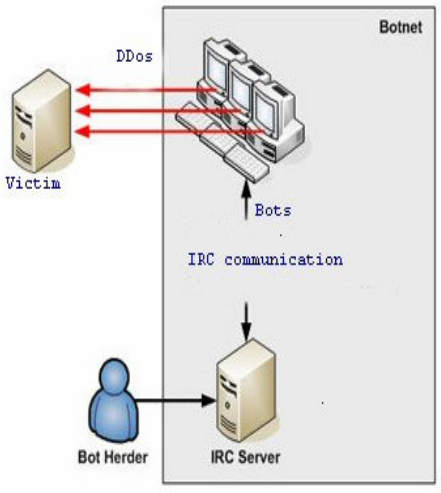

Fig 1. Botnet in action

II. DATA

We had two separate data sets to collect for the purpose of our experiments. The first set included a large number of binaries/executables labeled as botnet binaries and benign binaries. We acquired the botnet binaries from the computer science department at John Hopkins University. This is the same data they used for botnet related work [18]. As far as the benign binaries are concerned we randomly picked them up from Unix and windows machines, namely from /usr/bin and from windows system32 directory. This data was comprehensive and well represented.

We also needed labeled IRC logs for our experiments. While benign IRC logs are easily available on the web, botnet affected IRC logs are not readily available because of privacy and legal issues. The benign IRC logs were collected from several publicly available IRC logs from IRC channels like Wikipedia, linode, and kernelnewbies. These represent a diverse collection of logs, with different purposes. Some of these logs

also have automated bots for channel maintenance operations.

Obtaining malicious IRC logs proved to be extremely hard. There are several independent Security forums who actively track botnets [12]. They monitor and mine Botnet IRC logs for analysis and mitigation. Due to privacy and security issues, we were unable to obtain this data, which clearly represents a rich set of real world Botnet IRC logs, and which would have definitely provided more qualitative as well as quantitative results. Several other potential sources setup their own private infrastructure for collecting such training data [5].

Nevertheless we acquired data from Northwestern University where the department of CS is conducting research on wireless overlay network architectures and botnet detection [7]. The data regarding botnet IRC logs was not comprehensive in the sense that it was IRC traffic over a small amount of time. A larger and more comprehensive dataset could have established our results and hypothesis more conclusively.

III. APPROACHES

We tried a 2 stage approach to solve this issue. These methods are complementary and we can combine them for better results. They are as follows:

3.1 Botnet Binary Detection

There are several stages in the lifecycle of a Botnet where a Machine learning based solution can be deployed to thwart its effectiveness. During an early stage, a Binary detection and classification system can warn of a potentially malicious executable which might endanger a host machine. There has been some work already in this area [7] and we leverage on top of their work to classify Botnet executables which propagate as worms on networks scanning for vulnerable machines for infecting them, and enlisting them into the Bot network.

Unlike Virus scanners like Norton AV, or McAfee, a Machine learning solution can perform this classification without the need for explicit signatures. Identifying such binaries without explicit signatures needs recognizing common features and correlations between these binaries, e.g. a Botnet executable will be a self-propagating worm with a simple IRC client built in. Presence or absence of such a feature is an indicator that such a binary might potentially be a Botnet executable. We used supervised learning techniques on groups of benign executables vs. Botnet executables.

3.1.1 Features

We are focusing on binary profiling, and hex dumps for feature extraction. Identifying the strain of these binaries might also give an insight about the location of the Command-and-Control center and about the botnet capabilities in general. We used n-grams (more specifically 4-grams) of hexdump of the binaries as our features. For example if the hexdump is ff 00 12 1a 32, our features will be ff 00 12 1a and 0012 1a 32. We extracted around more than a million features. We then used chi-square to select around 10,000 most informative features. For each feature, we find its value as follows:

	feature=00121a32	feature≠00121a32
class=botnet	A	C
class≠botnet	B	D

$$\chi^2(botnet, f) = \frac{N(AD-CB)^2}{(A+C)(B+D)(A+B)(C+D)}$$

Now we select the top 10,000 features with the highest chi-square scores.

3.1.2 Classification

We used several classification algorithms to classify the binaries into malicious/benign. The models we used are as follows:

1. Multinomial Naïve Bayes
2. linear SVM
3. kNN
   This makes use of similarity metrics (e.g. cosine similarity) to find k nearest neighbors and classifies the point under consideration into the majority class of its k nearest neighbor [15]. We used k=5.
4. Logistic Regression
5. Multiboost Adaboost - Class for boosting a classifier using the MultiBoosting method. MultiBoosting is an extension to the highly successful AdaBoost technique for forming decision committees. MultiBoosting can be viewed as combining AdaBoost with wagging. It is able to harness both AdaBoost's high bias and variance reduction with wagging's superior variance reduction. Using C4.5 as the base learning algorithm, Multi-boosting is demonstrated to produce decision committees with lower error than either

AdaBoost or wagging significantly more often than the reverse over a large representative cross-section of UCI data sets. It offers the further advantage over AdaBoost of suiting parallel execution. See [17] for more details.

6. J48 Decision tree -This is an entropy based approach for generating a pruned or unpruned C4.5 decision tree. For more information see [16]. In general, if we are given a probability distribution P = (p_1, p_2, .., p_n) then the *Information conveyed by this distribution*, also called *the Entropy of P*, is:

$$I(P) = -(p_1 * \log(p_1) + p_2 * \log(p_2) + ... + p_n * \log(p_n))$$

If a set T of records is partitioned into disjoint exhaustive classes C1, C2, .., Ck on the basis of the value of the categorical attribute, then the information needed to identify the class of an element of T is **Info(T)** = I(P), where P is the probability distribution of the partition (C1, C2, .., Ck):

$$P = (\frac{|C_1|}{|T|}, \frac{|C_2|}{|T|}, ..., \frac{|C_k|}{|T|})$$

If we first partition T on the basis of the value of a non-categorical attribute X into sets T_1, T_2, .., T_n then the information needed to identify the class of an element of T becomes the weighted average of the information needed to identify the class of an element of T_i, i.e. the weighted average of Info(T_i):

$$Info(X,T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} * Info(T_i)$$

Consider the quantity Gain(X,T) defined as

$$Gain(X,T) = Info(T) - Info(X,T)$$

This represents the difference between the information needed to identify an element of T and the information needed to identify an element of T after the value of attribute X has been obtained, that is, this is the gain in information due to attribute X. We can use this notion of **gain** to rank attributes and to build decision trees where at each node is located the attribute with greatest gain among the attributes not yet considered in the path from the root.

For the purpose of our experiments we did not make use of any separate testing data set. To stay unbiased we used 10 fold cross validation to get the results. We used off the shelf softwares such as weak[19] and libsvm[13] for experimental results.

3.2 IRC log based detection

IRC has played a central role in the simplicity and effectiveness of a Botnet. Using a public communication channel, the Botmaster can use a simple command interface to communicate with a huge number of compromised zombie nodes, instructing them to carry out his orders.

There are two phases to this approach: First, to separate IRC traffic from other traffic. This is a reasonably solved problem [5][6]. The second step comprises of identifying Botnet traffic in the IRC traffic. Hence the problem now boils down to a text classification problem.

To be able to differentiate a benign IRC log from an IRC log manifested with Botnet activity, we used features involving both dialogues and IRC commands. Then using these features, we experimented with a variety of machine learning algorithms on them. In particular, we ran algorithms such as Naïve Bayes, SVM, J48 decision trees, kNN, etc. with 10 fold cross validation. The main categories of features we used included:

- *Number of users*: An IRC channel with Botnet activity should contain an unusually large number of users.
- *Mean / variance of words per line and characters per word in dialogues*: Bots usually do not produce dialogues that resemble human dialogue.
- *Number and frequency of IRC commands*: We have noticed through examination of the logs that there tends to be a large number of IRC commands at small intervals in Botnet manifested logs. One possible explanation would be the immense number of joins and exits from the result of accommodating a huge number of users in one channel.
- *Number of lines, dialogues, and commands*: In a benign IRC log, the number of dialogues should be much greater than the number of commands. And as mentioned above, a Botnet manifested log tends to contain an immense number of IRC commands.

IV. RESULTS AND EVALUATION

4.1 Botnet Binary Detection

The results obtained from the botnet binary based detection approach are summarized in Fig. 2. Clearly all the models performed reasonably well. Special mention must be made about Naïve Bayes which performed remarkably well although it is one of the simplest of models. SVM performed good too. However some models like kNN gave an accuracy of 96.4 which was lower than that of others. We will discuss about these results in details in the discussion section. In particular our evaluation metric included accuracy, F1 score and kappa measure.

$$Accuracy = \frac{\#Correctly\ predicted\ \ data\ points}{\#Total\ data\ points}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

$$Kappa = \frac{Observed\ Agreement - Chance\ Agreement}{Total\ Observed - Chance\ Agreement}$$

Model	Accuracy	F1 score	Kappa score
NB	.982	.981	.963
Linear SVM	.982	.981	.963
J48 – decision tree	.976	.975	.95
kNN	.964	.963	.926
Logistic	.982	.9815	.963
MultiboostAB	.976	.975	.951

Fig. 2 ***Performance of binary based detection***

Model	Accuracy	F1 score	Kappa score
NB	.70	.457	.705
Linear SVM	.976	.945	.97
J48 – decision tree	.992	.982	.99
kNN	.992	.982	.991
Logistic	.989	.974	.987
MultiboostAB	.967	.926	.963

Fig. 3 Performance of IRC based detection

4.2 IRC log based detection

The results obtained from the IRC log based detection approach are summarized in Fig. 3. A large chunk of the problem here involves text classification, and all algorithms, barring Naïve Bayes, perform encouragingly well.

V. DISCUSSION

In the binary based classification approach, most of the linear classifiers such as NB and linear SVM performed remarkably well. kNN however performed relatively bad. This can be explained on the basis of linearity of the data set. kNN is a non-linear model and hence it has less bias and a high variance unlike NB which is linear and has high bias and less variance. Given our data set was linear, it was therefore no surprise that kNN exhibited a higher generalization error compared to NB or linear SVM.

In the IRC log based approach, most classifiers performed similarly, except for Naïve Bayes, which was significantly worse. One possible explanation would be that there are dependencies in the features. These dependencies may be resolved with more training data. Since the set of IRC logs we obtained was not very large and comprehensive, it is possible that currently correlated features are not actually correlated in a larger dataset.

The other classifiers performed very well, all having accuracies greater than .96. Since SVM

and logistic regression had the best accuracies, F1 scores, and Kappa scores, it is likely that our dataset is linear.

In the logs that we collected with Botnet activity, there weren't strong attempts of log obfuscation by Botmasters. But suppose that there were strong attempts to produce human-like dialogues by the bots. Then to potentially fool our classifiers, the dialogues must produce similar averages and variances in terms of words per line and characters per word.

Pushing the above scenario to the extreme, suppose that the bots produce perfect dialogues. In this case, our classifier should still work due to one fundamental difference between benign logs and logs manifested with Botnet activity – there is an immense number of join and exit commands in a Botnet infested channel.

VI. CONCLUSION

In this course project we aimed to highlight a major problem plaguing the Internet today and a potential solution to the problem by employing novel Machine Learning techniques. Our results are encouraging, indicating a definite advantage of using such techniques in the wild for alleviating the problem of Botnets.

Conversion of the problem in the Text classification domain has opened up new avenues in feature extraction for our solution. Possible future enhancements include adding additional features to the IRC traffic classification engine. Also to solve the problem of training data which, in retrospect, seems to be the critical to the success of such a project, we propose setting up a farm of Virtual machines as a testbed for collecting and mining our own logs for training purpose.

Lastly, the end goal of such a tool should be to provide *proactive* help to system administrators. In this regard, it is easy to envisage an application which can run on a Gateway or a Router, and look at Network flow traffic, and classify it on-the-fly enabling automated blacklisting of involved machines in the network.

VII. ACKNOWLEDGEMENT

We take this opportunity to thank Elizabeth Stinson of Stanford Security Group for her help and support. We also extend our acknowledgements to Yan Chen of northwestern university for providing us with botnet IRC logs and to Andreas Terzis of JHU for botnet binary datasets. We further acknowledge our gratitude to Prof. Andrew Ng and the TAs of cs229 for their help and support.

VIII. REFERENCES

[1] http://en.wikipedia.org/wiki/Botnet

[2] http://en.wikipedia.org/wiki/Zombie_computer

[3]http://securitywatch.eweek.com/exploits_and_attacks/everydns_opendns_under_botnet_ddos_attack.html

[4]http://it.slashdot.org/article.pl?sid=06/10/17/002251

[5] W. Timothy Strayer, Robert Walsh, Carl Livadas, and David Lapsley. Detecting Botnets with Tight Command and Control**.** To Appear in 31st IEEE Conference on Local Computer Networks (LCN'06). November 2006

[6] Using Machine Learning Techniques to Identify Botnet Traffic, Carl Livadas, Robert Walsh, David Lapsley, W. Timothy Strayer

[7] *Yan Chen, "Towards wireless overlay network architectures"*

[8] Binkley- An Algorithm for Anomaly-based Botnet Detection .

[9] E. Cooke, F. Jahanian, and D. McPherson. The zombie roundup: Understanding, detecting, and disrupting botnets. In *USENIX SRUTI Workshop*, pages 39–44, 2005.

[10]http://www.vnunet.com/vnunet/news/2144375/botnet-operation-ruled-million

[11] http://www.irc.org/tech_docs/rfc1459.html

[12] www.shadowserver.org

[13] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[14] Detecting Botnets with Tight Command and Control, Carl Livadas, Robert Walsh, David Lapsley, W. Timothy Strayer.

[15] Aha, D., and D. Kibler (1991) "Instance-based learning algorithms", Machine Learning, vol.6, pp. 37-66.

[16] Ross Quinlan (1993). "C4.5: Programs for Machine Learning", Morgan Kaufmann Publishers, San Mateo, CA.

[17] Geoffrey I. Webb (2000). "MultiBoosting: A Technique for Combining Boosting and Wagging". Machine Learning, 40(2): 159-196, Kluwer Academic Publishers, Boston

[18] Moheeb Abu Rajab *et.al* (2006)*"A Multifaceted approach to understanding the botnet phenonmenon"*

[19] http://www.cs.waikato.ac.nz/ml/weka/