Performance Evaluation and Accuracy Improvement for Anomaly and Bot Detectors

By
Mobin Javed
2008-NUST-MSCSE-18

Supervisor
Dr. Syed Ali Khayam
NUST-SEECS

A thesis submitted in partial fulfillment of the requirements for the degree of Masters in Communication Systems Engineering (MS CSE)

In
School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST),
Islamabad, Pakistan.

(November 2011)

Approval

Certified that the contents and form of thesis entitled **"Performance Evaluation and Accuracy Improvement for Anomaly and Bot Detectors"** submitted by Ms. **Mobin Javed** have been found satisfactory for the requirement of the degree.

Advisor: **Dr. Syed Ali Khayam**

Signature: _____

Date: _____

Committee Member 1: **Dr. Fauzan Mirza**

Signature: _____

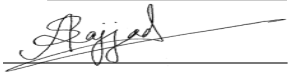
Date: _____

Committee Member 2: **Dr. Hassaan Khaliq**

Signature: _____

Date: _____

Committee Member 3: **Mr. Ali Sajjad**

Signature: 

Date: _____

To my parents and grandparents for their love, support and prayers!

Abstract

Due to the ever-rising network security threats, the domain of intrusion detection is progressing at a fast pace, and new detection techniques are being proposed quite frequently. However, little effort is being expended into carrying out comparative evaluations of the techniques and summarizing the body of knowledge that exists. Consequently, there is a lack of guidelines based on the state of art, which can help shape the design of future detectors. In this thesis, we make an effort to address this gap in the research literature in two sub domains of intrusion detection: (i) entropy based anomaly detection, and (ii) botnet detection.

First part of the thesis focuses on entropy based anomaly detection systems (ADSes). Although entropy based measures have been widely used in ADSes to quantify behavioral patterns, and these measures have shown promise in detecting diverse set of anomalies present in networks and end-hosts, it is unclear if full potential of the entropy tool is being exploited. We survey and investigate the usage of entropy for anomaly detection and show that the full potential of entropy-based anomaly detection is currently not being exploited because of its inefficient use. We highlight three important shortcomings of existing entropy-based ADSes and propose efficient entropy usage to mitigate these shortcomings.

Second part of the thesis focuses on botnets, which are regarded as the most significant security threat facing the Internet today because of their massive computing power and bandwidth.

Botnets are used today to launch large scale distributed denial of service attacks, harvest loads of personal information and generate vast amounts of spam. Security response to the botnet threat is however in its infancy; over the past years, few host and network-based bot detection techniques have been proposed in research literature. However, a comprehensive and judicious performance comparison of these techniques has not been performed, mainly because of the unavailability of open-source bot detector implementations and labeled bot datasets. We perform the first comparative evaluation of prominent bot detection techniques on a dataset containing traffic patterns of a diverse set of IRC bot malware, and release our dataset and open source detector implementations publicly for future performance evaluations by the research community. Based on the evaluation results, we highlight strengths and weaknesses of different techniques and outline the state-of-the-art in bot detection research. We also propose promising guidelines that can be used to improve the performance of bot detectors.

Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgment has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: **Mobin Javed**

Signature: _____

Acknowledgments

First of all, I'm extremely grateful to Allah Almighty for providing me the opportunity and strength to carry out this research in the exceptional environment of the Wireless and Secure Networks research lab (WiSNet) at NUST-SEECS.

I'm highly indebted to my advisor, Dr. Ali Khayam, without whose supervision this work was not possible. Advisee meetings with him have shaped a great deal of how this works looks in it's current form. This work has also benefitted a lot from the valuable feedback from my evaluation committee, Dr. Fauzan Mirza, Dr. Hassaan Khaliq Qureshi, and Mr. Ali Sajjad.

No doubt, putting together a thesis becomes much easier with the love of a supportive family. I'm extremely fortunate to have parents who are as passionate about my career as myself. They provided me with all the possible comfort and support, so that I could work on my thesis with full focus. I'm also thankful to my sister, Iqra Javed, for proofreading the thesis.

Finally, I'm highly thankful to my friends and colleagues at WiSNet for their contribution in the form of support. The fun times spent with them was a source of regaining the energy required to work on the thesis.

Mobin Javed

Table of Contents

List of Figures

List of Tables

Part I

Entropy Based Anomaly Detection

Chapter 1

Introduction

Malware and network attacks have become significant threats for today's networks and hosts. These attacks not only compromise information confidentiality and integrity but can also cause disruption of service resulting in financial losses of the order of billions of dollars. With our increasing reliability on computer and internet for tasks ranging from personal level (e.g. buying tickets, communicating with friends) to business level (e.g. important data transfer, e-commerce), it is imperative that adequate security protection is in place. Consequently, protecting and defending computers and computer networks has become a mainstream research area.

In order to combat the rapidly evolving cyber security threats, various intrusion detection systems have been designed and have now become an intrinsic component of today's host and network security infrastructure. According to Symantec's Annual Threat Report 2009, Symantec created 2,895,802 new malicious code signatures in 2009, a 71 percent increase over 2008. The 2009 figure represents 51 percent of all malicious code signatures ever created by Symantec [39]. This signifies that a high number of new attacks are being developed daily by the intruders, and a signature based intrusion detection technique which compares the observed data against a database of rules will clearly

fail keep up with the new attacks being developed by attackers everyday.

In order to mitigate the shortcoming of signature based IDSes, anomaly detection systems, a class of intrusion detection systems, capable of detecting zero day attacks, model benign behavior and look for deviations in this benign profile to detect attacks. Modeling and comparing the benign profile presents two major design: (i) which features should be selected for modeling the benign profile, and (ii) how to compare the benign model to the observed realtime data. A diverse range of host and network features, such as protocol type, length of packet, processes running on the host, is available to the ADS for analyses and monitoring. However, it is not practically possible to operate on such a high dimensional feature set, since this would result in unmanageable complexity. While choosing a comparison model for benign and observed data, low complexity is a primary objective in order to enable detection with minimum delay. With the rapid increase in bandwidth to Gbps, this consideration has become even more important to ensure the detection of attacks in a timely manner. Numerous research in the last few years, has focused on the use of information theoretic measures for the detection of anomalies [19]–[31]. This study relates to one such particular class of ADSs - entropy based ADSs.

1.1 Motivation and Problem Statement

Entropy based measures have been widely used in ADSes to quantify behavioral patterns. The entropy measure has shown significant promise in detecting diverse set of anomalies present in networks and end-hosts. Consequently, it has proved to be a powerful information theoretic tool. However, other variants of entropy measure remain largely unexplored which if applied to the domain of anomaly detection can result in significant

accuracy improvement.

A recent comparative evaluation of network based anomaly detectors shows that the entropy based anomaly detector, Maximum entropy, performs quite well whereas another entropy based anomaly detector, PCA, gives poor performance, even though the second detector uses a richer feature set [16]. This behaviour is quite surprising, because intuitively the second anomaly detector is expected to perform better. Preliminary investigation resulted in the observation that the two anomaly detectors differ in the way they use the entropy measure. Given the findings of the above study, we feel that a survey of entropy based anomaly detection was necessary to investigate the use of entropy measure in ADSs. A comparative performance evaluation of the ADSs can help us devise promising guidelines for the use of entropy for anomaly detection. The objectives of this study are:

1. To investigate the current usage of entropy in the domain of anomaly detection and gauge whether it is being used efficiently.

2. To propose promising guidelines for future use of entropy in anomaly detection.

The major goal of this thesis is to summarize the existing body of knowledge via experimental evaluation and consequently to develop a better understanding of the usage of entropy. We formulate the problem statement of part I of thesis as,

> *To carry out a judicious comparative evaluation of existing entropy based detectors on a comprehensive dataset, and to identify promising guidelines for future detectors.*

1.2 Contribution

We investigate the usage of entropy in anomaly detection and perform performance evaluation of six prominent entropy based host and network anomaly detectors [8, 9, 11, 13, 5, 14] on publicly available datasets [12, 15, 18]. We show that the full potential of entropy-based anomaly detection is currently not being exploited because of its inefficient use. We highlight important shortcomings of existing entropy-based ADSes and propose efficient entropy usage to mitigate these shortcomings.

1.3 Organization of Part I

Part I of the thesis is organized as follows,

Chapter 2 provides essential background information on the field of anomaly detection. It also discusses the entropy measure and provides an insight into how they can be used for the detection of anomalies.

Chapter 3 describes our evaluation framework. The six detectors evaluated in this study are discussed in detail. This chapter also describes the characteristics of datasets used in this study.

Chapter 4 discusses the our performance evaluation results, and proposes accuracy improving guidelines.

Chapter 5 concludes this part of the thesis and frames future directions.

Chapter 2

Background

2.1 Intrusion Detection

Intrusion detection refers to the processes used in discovering unauthorized uses of computer devices and networks. The history of intrusion detection dates back to 1986 with James Anderson proposal for the usage of audit trails for the detection of malicious activity.

IDSes are categorized as network-based, host-based or joint network-host based classifiers depending on the point of deployment and usage of host data or network traffic. Host based intrusion detection systems use the internal features of operating system such as system calls to detect malicious behaviour and are deployed on individual end hosts. Network based intrusion detection systems, on the other hand, collectively analyze the network data (e.g. packets or flows) of multiple computers and are generally deployed on routers.

The first intrusion detection systems were *signature based intrusion detection systems*. These systems use a signature(rule)-base to detect intrusions. The rules can be based on the header fields as well as packet content. Snort [1] and Bro [2] are examples of signature based intrusion detection systems. Although signature based systems are capable of detecting existing attacks

very well, a major limitation of these systems is their inability to detect 'previously unseen' anomalies. Another drawback of these systems is the need to continually update the signature database.

To mitigate this shortcoming, a new class of IDSes capable of detecting zero-day attacks was proposed. This class is known as *anomaly detection systems* (ADSes) and works on the premise that patterns in data that do not conform to expected behavior present potential threats to the entity under observation. Generally, the expected behavior is derived from the benign traces that are considered to be free of any anomalies.

Another class of intrusion detection systems employs both components: a rule based detection engine and an anomaly detection engine. Next Generation Intrusion Detection Expert System (NIDES) [4] is an example of such an intrusion detection system.

2.2 Entropy measures

The proposal for use of entropy in intrusion detection dates back to 2001 when Lee and Xiang proposed the use of information theoretic measures for data partitioning and setting parameters for existing intrusion detection models [5]. Since then, a wide range of entropy based anomaly detectors have been proposed in literature [6], [7], [8], [9], [11],[13], [14]. In this section, we provide theoretical foundations of the Shannon entropy measure as well as its variations.

Entropy: Entropy is the average amount of information (or the uncertainty) associated with the outcomes of a random variable. This random variable constitute the observed space in the anomaly detection process. For example, source/destination ports, system calls, connection status etc. Let this random variable be X with outcomes belonging to the set ω=1,2,...,λ. The

underlying probability distribution is characterized by the parameter p. The entropy associated with this random variable is given by:

$$H(X) = \sum_{x \in \omega} p(x) \log_2 \frac{1}{p(x)}. \tag{2.1}$$

A random variable with equi-probable outcomes has the highest entropy since all outcomes have the tendency to occur with the same probability. However, this is highly unlikely in real world scenarios. Thus, some outcomes of the random variable are more probable than the others hence defining the average uncertainty associated with the random variable under observation.

Benign user behavior, pertaining to the applications being used within the network or at the endpoint, has an associated uncertainty that defines the benign state. However, during the course of an anomaly, the information associated with the underlying outcomes of the random variable gets perturbed, resulting in considerable variations in entropy, which is consequently used for the detection of these intrusions.

Joint Entropy: The entropy of a vector of random variables is the joint entropy which uses joint probability distributions, instead of marginals, for entropy calculation. Numerous variations of the Shannon entropy measure have also been proposed for anomaly detection in research literature [33], [34].

Conditional Entropy: Conditional Entropy deals with multiple random variables. Given two random variables X and Y with sample spaces ω and $\tilde{\omega}$ respectively, conditional entropy provides a measure of the amount of information in Y not given by X.

$$H(Y|X) = \sum_{x \in \omega} \sum_{y \in \tilde{\omega}} p(x, y) \log_2 \frac{1}{p(y/x)}. \tag{2.2}$$

It employs the use of joint and conditional probability distributions and is thus employed when multiple features are used by the ADS for anomaly detection. If there is no information overlap between the random variables, the conditional entropy assumes the value zero.

Relative Entropy: Relative Entropy, also known as information divergence or the Kullback-Leibler (KL) divergence, is a distance measure that represents the dissimilarity between two probability distributions. Let the two distributions be *p(x)* and *q(x)*, then the relative entropy between the two distributions is given by,

$$D(p||q) = \sum_{x \in \omega} p(x) \log_2 \frac{p(x)}{q(x)}. \tag{2.3}$$

During the anomaly detection process, ADS is trained on benign traffic profiles while it is run on real-time data during actual deployment. During the course of an anomaly, the real-time distribution changes considerably relative to the baseline distribution. This is due to the lack of the knowledge of the the attacker regarding the topology or the services running inside the network. Thus a measure of the difference between the two probability distributions can signify the presence of an anomaly.

Mutual Information: Mutual information is a measure of the information overlap between two random variables (or distributions). If one of the random variable is known (e.g. X and consequently *p(x)*) then the other can be predicted if the information overlap between the two random variables is considerable. Mutual information between two random variables X and Y, is given by

$$I(X;Y) = H(X) - H(X|Y), \tag{2.4}$$

where one of the random variables (say X) represents the benign baseline distribution while the other represents the real-time distribution. Greater the mutual information between these two random variables, greater the similarity between the baseline and the real-time distributions and lesser is the probability of the presence of an anomaly and vice versa.

Entropy Rate: Average entropy measure provides the per sample average of the amount of information associated with a random process. Let a random process constitutes n random variables. The average entropy measure is then the average of the joint entropy of these n random variables constituting the random process.

$$\bar{H_X}(n) = \frac{1}{n} H(X_1, X_2, ...X_n). \tag{2.5}$$

Asymptotic evaluation of the average entropy measure is the entropy rate, given by:

$$H_X = limit_{n\to\infty} \bar{H_X}(n). \tag{2.6}$$

Thus, the entropy rate provides the average amount of uncertainty associated with a collection of random variables constituting a random process. Another important measure of the entropy rate of a process is the conditional entropy rate (CER). CER measures the conditional entropy of the process at time index n, given the history of the process:

$$\grave{H_X} = limit_{n\to\infty} \grave{H_X}(n) = limit_{n\to\infty} H(X_n|X_1, X_2.....X_{n-1}). \tag{2.7}$$

2.3 Entropy Usage in Anomaly detection

This section builds on the previous section by categorizing the ways in which entropy measures canbe used for anomaly detection.

Figure 2.1 on the next page shows a classification of the usage of entropy in anomaly detection. Anomaly detectors proposed in research literature either employ the use of single feature or multiple features for the detection process. Thus these classifiers can use marginal, joint and/or conditional distributions pertaining to these features. Moreover, detectors using single features may analyze a single feature or multiple features in isolation.

Once the features to be analyzed have been chosen, the corresponding distribution(s) may be the non-parametrized histograms or these may be the conditional distribution(s). In case of single feature, a conditional distribution signifies time/space correlation. In case of multiple features, it signifies the mutual information (MI).

Comparison of benign baseline and the real-time distributions can be performed in either of the following two ways, 1) Using entropy value comparison (referred to in Figure 2.1 as EV); or 2) Using relative entropy (RE) which performs a symbol by symbol comparison. For conditional distributions, the above comparisons are referred to as conditional entropy value (CEV) and relative conditional entropy (RCE) comparison respectively. A third tool, entropy rate (ER), has also been used for the comparison of conditional distributions in ADS research literature.

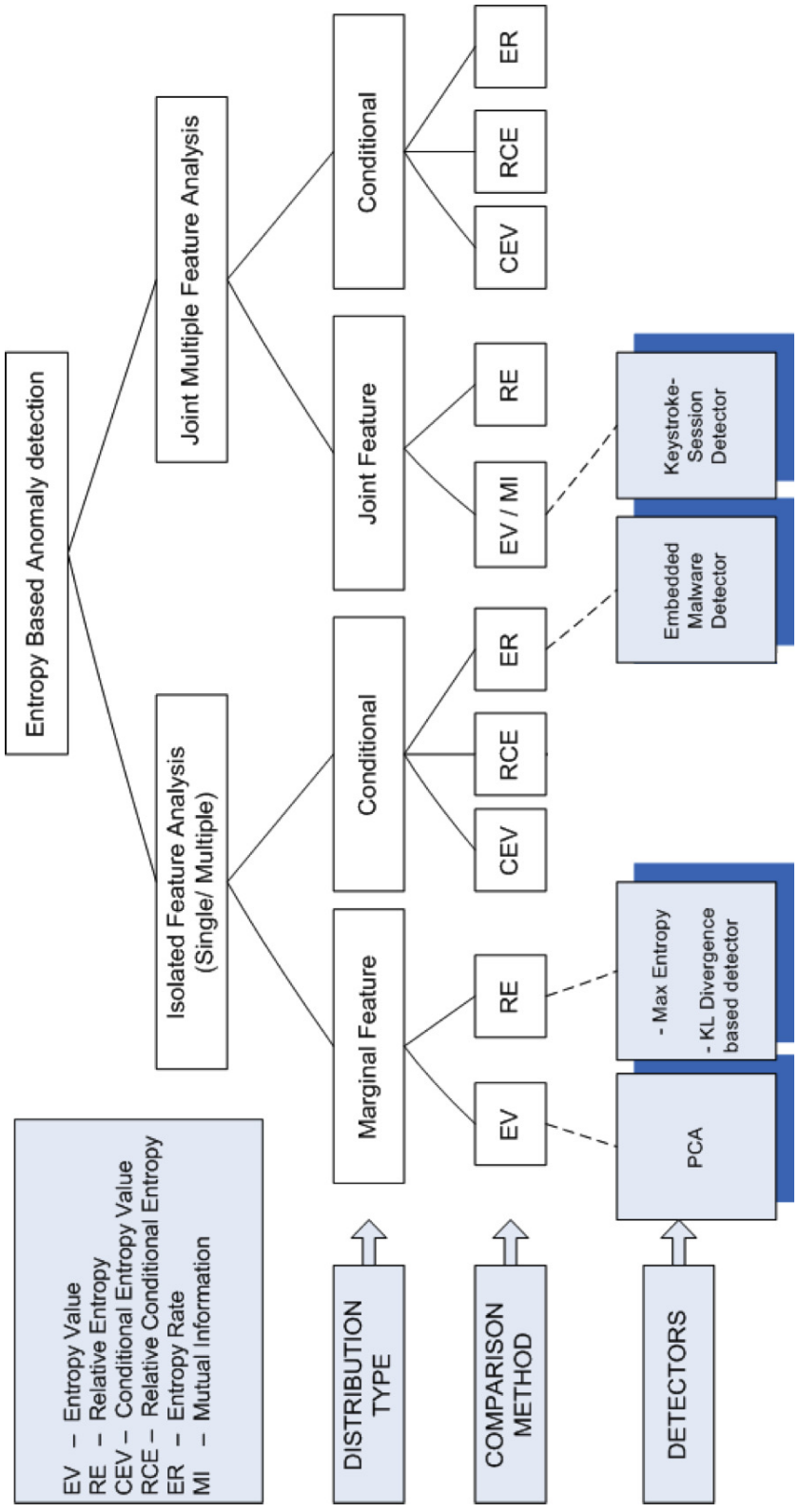


Figure 2.1: Entropy Based Anomaly Detection Classification.

Chapter 3

Evaluation Framework

In the first section of this capture, we provide a detailed description of the anomaly detectors used in this study. These detectors employ entropy, or its variations, for the detection of intrusions. In the second section, we describe the datasets used in evaluation.

3.1 Entropy Based Anomaly Detectors

Ayesha et al. show that network anomaly detectors are affected by the choice of entropy measure they employ [16]. We extend work on this finding and investigate whether there is a difference in the performance of other categories of anomaly detectors or not. To this end, we evaluate entropy based anomaly detectors from all the three categories i.e. host based, network based and joint host network based ADSs.

3.1.1 Entropy Based Network Anomaly Detection Systems

While many papers discuss the use of entropy for network anomaly detection, we limit the detailed discussion and evaluation to only those general purpose ADSs which are capable of detecting a wide range of anomalies. The specialized ADSs not discussed

here, perform the detection of only specific types of anomalies. For example, [6] discusses the use of entropy for DDoS attack detection, [7] suggests the use of entropy for the detection of Blaster and Witty worm by specifically analyzing the source and destination IP addresses and source ports entropies.

Maximum Entropy Detector

The maximum entropy approach proposed in [8] first builds a normal packet classes distribution from training data using the maximum entropy principle. Packet classification is performed in two dimensions: protocol and the destination port number. The detailed packet classification of $\Omega = 2348$ packet classes used by this detector is shown in Figure 3.1. Baseline distribution for these packet classes is obtained by maximum entropy estimation; the underlying principle is to choose the most uniform distribution under the constraints derived from training data. The approach thus produces the least biased estimate possible based on the partial information in the training data by minimizing the KL divergence between the empirical distribution $\tilde{P}$ and the estimated baseline distribution P of the packet classes. The criteria for estimated baseline distribution is given by,

$$P = \arg\min_{P} D(\tilde{P}||P) \tag{3.1}$$

$$\text{where} \quad D(P\|\tilde{P}) = \sum_{\omega\in\Omega} P(\omega)\log_2 \tfrac{P(\omega)}{\tilde{P}(\omega)}. \tag{3.2}$$

Maximum entropy estimation consists of two steps: 1) Feature selection and 2) Parameter Estimation. In the feature selection step, the best feature functions that minimize the difference between model and empirical distribution are chosen. The set of feature functions are indicator functions which take on a

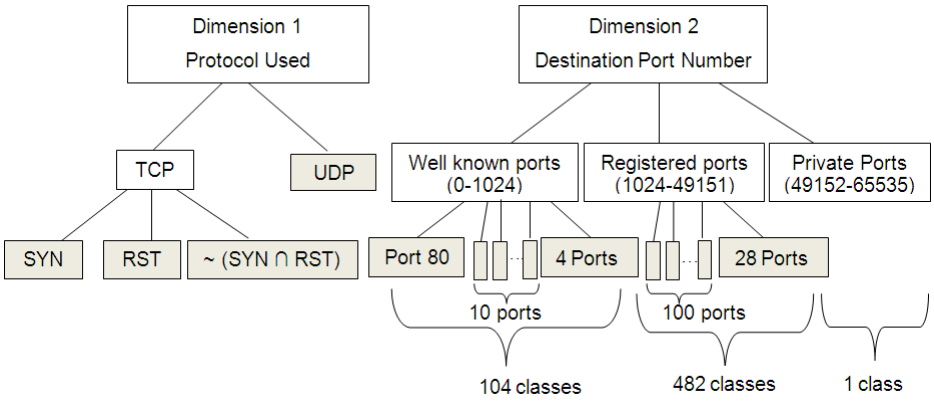


Figure 3.1: Packet Classification in Maximum Entropy Detector.

value *true* if the packet belongs to the feature class and *false* otherwise. The model distribution should have maximum entropy and the same expected value for each feature as that in the training data i.e. $E_P(f_i) = E_{\tilde{P}}(f_i)$. In the parameter estimation step, the weights of the feature functions are adjusted. Weight of all the feature functions are updated whenever a new feature is added. When the KL divergence between empirical and model distribution reaches a self-defined limit, the process of adding functions is terminated.

Once the model is built, this baseline distribution is compared with the observed real-time distribution by relative entropy measure in W windows using sliding window approach. An alarm is generated if the divergence is greater than threshold d in more than h time windows for a specific feature class.

Principle Components Analysis(PCA) based Subspace Detector

This technique is capable of detecting anomalies which cause perturbation in multiple features (source and destination IPs and ports) as well as anomalies spanning multiple flows. Individual feature entropy matrices comprising of values of entropy of that feature in a particular timebin for a particular flow are

first constructed. Then a three dimensional matrix t x p x k which indicates entropy value at time t for flow p of the traffic feature k is constructed. This matrix is illustrated pictorially in Figure 3.2. PCA is used to reduce the high dimensional data by selecting an $m <= p$ dimensional subspace.

The detection of anomalies is achieved via the standard subspace method. Each row at time t can be represented by the sum of normal and residual subspaces. Anomalies are then detected by examining the size of the residual subspace vector; large values of this vector indicate anomalous conditions. Using this method, the timebin in which the anomaly occurred is detected. Next, the participating flows and the features which were perturbed due to the anomaly are identified. For a flow k, the feature state vector can be written as

$$h = h^* + \Theta_k f_k \tag{3.3}$$

where h^* denotes the typical entropy vector, Θ_k specifies the components of h belonging to the flow k and f_k is the amount of change in entropy due to flow k. The following equation is employed to identify the anomalous flow l:

$$l = \arg \min_k \min_{f_k} \|h - \Theta_k f_k\|. \tag{3.4}$$

This method is continued recursively, until all the flows associated with the anomaly have been identified.

KL Divergence based worm detector

This technique, proposed in [11], analyzes the *relative entropy* (KL divergence) of benign and observed source, destination port distributions at an endpoint to detect anomalies at the particular endpoint. Empirical distributions of the training data are used as benign distributions. Observed distributions are computed on a 20 sec window basis by counting the number of times a particular port is used during the window.

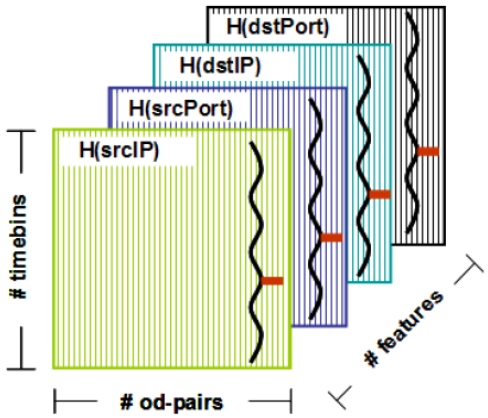


Figure 3.2: Multi-variate, multi-way data to analyze [9].

Let the source and destination port sets be denoted by S and D respectively. The benign source and destination port distributions are then represented by $X = \{p_i, i \in S\}$ and $Y = \{q_j, j \in D\}$. The *KL divergence* between the benign and currently observed port distributions is given as,

$$D(X_n \| X) = \sum_{i \in S_n} \frac{p_i^n}{p_n} \log_2 \frac{p_i^n / p_n}{p_i / p}, \tag{3.5}$$

$$D(Y_n \| Y) = \sum_{j \in D_n} \frac{q_j^n}{q_n} \log_2 \frac{q_j^n / q_n}{q_i / q}, \tag{3.6}$$

where $p = \sum_{i \in S} p_i$ and $q = \sum_{j \in D} q_j$ respectively represent the aggregate source and destination port frequencies observed in the benign profile.

3.1.2 Entropy based Host Anomaly Detection Systems

Embedded Malware Detector

A malware hidden in the content of benign files is known as embedded malware. To detect such malware, authors in [13] lever-

age the fact that benign files exhibit regularity in byte sequences. This regularity is modeled by conditional byte distributions.

Conditional byte distributions are more capable of detecting the malware than the n-gram distributions; an n-gram is the sequence of n symbols. The n-gram distribution is thus a joint distribution of the individual 1-grams. The redundant information contained in the n-gram distribution is removed by the conditional n-gram distribution by reducing the underlying sample space and thus depicts a more precise picture of the benign file.

First, the probability transition matrix containing the values of probability of the next byte given the first byte is computed as,

$$P = \begin{bmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,255} \\ p_{1,0} & p_{1,1} & \cdots & p_{1,255} \\ \vdots & \vdots & \ddots & \ddots \\ p_{255,0} & p_{255,1} & \cdots & p_{255,255} \end{bmatrix}.$$

Next, a first order 256 state Markov chain is built using the probability transition matrix as shown in Figure 3.3. In order to detect the variations in the conditional byte distributions, [13] does not use relative entropy to perform a blockwise comparison because computing the relative entropy of a large number of conditional distributions results in high complexity. The authors instead make use of the information theoretic measure, entropy rate which gives the expected entropy of the 256 state Markov chain to perform a blockwise comparison. Since the entropy rate, H_x, given by equation 2.6 in Section 2 does not exist in general, modified entropy rate of the 256 state Markov chain is given as,

$$R = \sum_{i=0}^{255} \pi_i H(X_i), \tag{3.7}$$

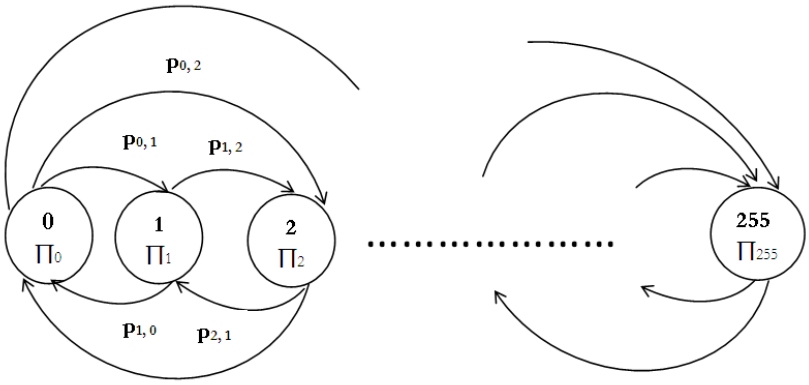


Figure 3.3: 256 state Markov chain in the Embedded Malware detector.

where π_i represents the probability of random variable X_i and $H(X_i)$ is the entropy of the random variable X_i.

Entropy rate of the infected file is perturbed at the malware embedded locations and thus results in the detection of malware.

3.1.3 Entropy based Joint Host-Network Anomaly Detection Systems

Keystroke Session Mutual Information Based Detector

Joint distributions of multiple features can be helpful in revealing subtle anomalies. An anomaly detector which makes use of such a technique was proposed in [14]. The authors note that marginal distributions of the keystrokes or network sessions are incapable of detecting malicious sessions. Instead, the histogram of the keystrokes which are used to initiate network sessions is skewed; perturbation in this distribution can easily reveal the malicious sessions. The benign profile is thus modeled by the joint distribution of the keystrokes and network sessions. Anomalies are detected by the change in entropy of the observed distribution.

Table 3.1: Ports used by malware in Endpoint Dataset

Worm	**Ports Used**
Zotob.G	445, 6667, 1171
Dloader-NY	135,139
Forbot-FU	445
My-Doom.A@mm	3127-3198
Rbot-AQJ	139,769
Witty	4000
CodeRedv2	80
Sim Src Port	2200

3.2 Datasets

In this section, we provide details regarding datasets used for performance evaluation. Two publicly available datasets were used in this study: (i) endpoint dataset [15], and (ii) embedded malware dataset [13]. The first dataset was used to evaluate network based ADSes whereas the other two were used for host based ADSes performance evaluation.

3.2.1 Endpoint Dataset

This dataset was collected at the `WiSNet lab` and comprises 12 months *session-level* traffic collected at 13 network endpoints comprising home, university and office machines [15]. A *session* corresponds to bidirectional communication between two IP addresses. Communication between the same IP address on different ports is considered part of the same network session. For this study, we use six weeks subset of the original dataset.

Attack Traffic

Attack traffic was generated by infecting virtual machines with the following set of diverse malware: Zotob.G, Forbot-FU, Sdbot-AFR, Dloader-NY, SoBig.E@mm, My-Doom.A@mm, Blaster,

Rbot-AQJ, RBOT.CCC, Witty, CodeRedv2 and Sim Src Port. These malware have varying attack ports, scan rates and transport protocols as listed in Table 3.1. The attack traffic mainly comprises of outgoing portscans with the scan rate ranging from 0.68 scans per second (sps) to 46.84 scans per second.

Background Traffic

Background traffic in this dataset comprises of traffic from machines operated by home users, researchers and administrative staff running a variety of applications. Mean session rate recorded on the lowest and highest rate endpoint were 0.21 and 1.92 sessions/sec respectively.

Merged Traffic

For each malware, attack traffic of fifteen minutes duration was inserted in the background traffic of each endpoint at a random time instance. This operation was repeated to insert 100 non-overlapping attacks of each worm inside each endpoints background traffic.

3.2.2 Embedded Malware Dataset

The embedded malware dataset was generated by `nexginrc` by inserting malware in 1800 benign files [13]. Files in this dataset are of six common types: DOC, EXE, MP3, PDF, ZIP and JPG. The choice of these six file types provides a diverse testing platform consisting of compressed, uncompressed, executables and document files. 300 files of each of the six types were used thus making a total of 1800 benign files. The benign files were collected from the web as well as from the local network of the `nexginrc` lab. The file sizes range from 3KB to 20MB with an average size of 2MB.

Table 3.2: Malware statistics in Embedded Malware Dataset

Major Category	**Minor Category**	**Quantity**	**Average Size** (kilobytes)	**Minimum Size** (bytes)	**Maximum Size** (kilobytes)
Backdoor	Win32	3,444	285.6	56	9, 502
Constructor	DOS	178	104.2	62	7, 241
Constructor	Win32	172	398.5	371	5,971
Email Flooder	-	148	343.5	1, 430	4,262
Email Worm	Win32	935	73.5	14	8,762
Exploit	-	242	101.1	370	1, 912
Flooder	-	154	168.1	486	981
IRC Worm	-	485	34.4	56	1, 072
Nuker	-	140	188.1	4, 000	680
Trojan	BAT	649	20.2	12	708
Trojan	DOS	971	27.0	4	1, 818
Trojan	Win32	983	125.4	12	2, 998
Virus	Boot	1,514	32.5	108	1, 490
Virus	DOS	16,236	18.7	5	1, 860
Virus	MS Office	2,596	53.5	118	4, 980
Virus	Win32	991	44.3	175	1, 018
Worm	Win32	153	110.5	97	2, 733

37,420 malware samples of different types including viruses, worms, trojans, spyware and exploit codes were used from `VX Heavens virus database` [3]. The details of the malware used are presented in table 3.2. The malware size ranges from 4 bytes to 14 MB.

Two datasets were generated by `nexginrc` - first by inserting the malware in the benign files and second by encrypting the malware using ROT-13 cipher and then inserting in the benign files. The malware in each dataset were inserted after the file header to ensure the proper opening of file. We report the results on the dataset in which malware was encrypted before infecting the benign files.

Chapter 4

Accuracy Evaluation and Improvement

In this chapter, we discuss the performance evaluation results and propose accuracy improving guidelines.

4.1 Evaluation Results Discussion

Figures 4.1(a) and 4.1(b) show the accuracy evaluation of the four entropy based network anomaly detectors, Maximum entropy, KL divergence, Keystroke session, and PCA on the endpoint dataset. Recall that the endpoint dataset contains traffic from thirteen machines and each machine was infected with twelve malware. The graphs present results for the average detection rate across twelve malware for the thirteen individual endpoints in order to give an insight into the performance consistency.

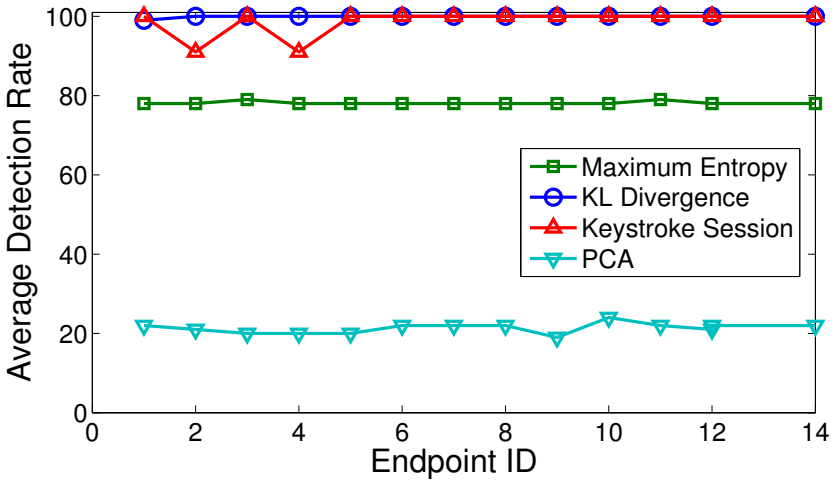
We observe that there is quite a lot of variation in the performance of the four detectors. Keystroke session detector achieves the highest accuracy of 100% detection rate and 0% false alarm rate. The high accuracy of keystroke session detector illustrates the power of using multiple feature conditional distributions and mutual information for the detection of anomalies. Keystroke

session detector is followed by the KL divergence detector which uses the KL divergence of source and destination ports for detection of anomalies. Note that, maximum entropy detector performs poor as compared to KL divergence detector. This is due to two factors: (i) Maximum entropy detector works only on destination port packet classes whereas KL divergence detector looks at two features: source and destination ports, and (ii) The entropy measure used by KL divergence detector is more powerful than that used by Maximum entropy detector. PCA detector fails to offer acceptable accuracy and only achieves a maximum detection rate of 22% with a 12% false alarm rate (the number of principal components used in the evaluation is two). This performance is explained by the fact that PCA uses entropy value comparison which is a weak comparison tool because it is incapable of differentiating between distributions with the same amount of uncertainty. Relative entropy on the other hand takes individual symbol values into account and is therefore capable of achieving greater accuracy.

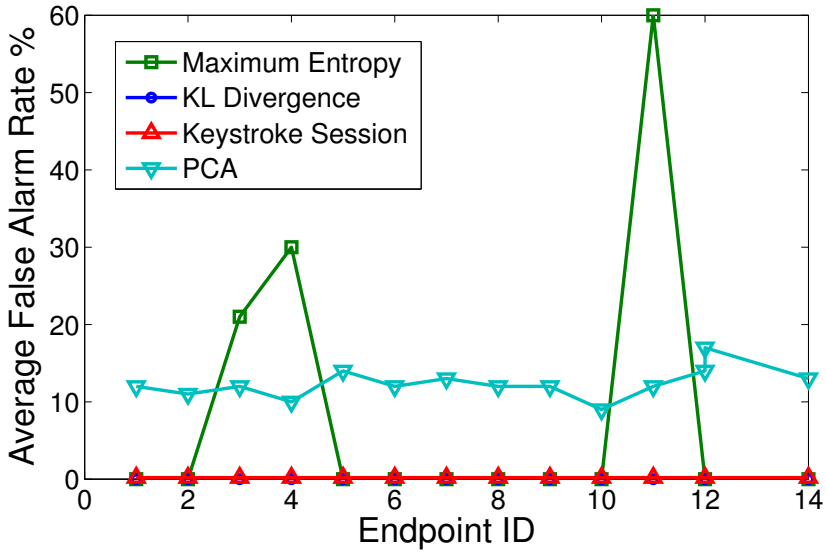
Figure 4.2 shows the evaluation results of embedded malware detector on the `nexginrc` embedded malware dataset. Embedded malware detector is capable of achieving 70-100% detection rate with a maximum false alarm rate of 32%. This is due to the fact that this detector does not use simple conditional entropy value comparison. It instead uses the entropy rate measure which gives an appropriate weight to each state; each state (conditional entropy value) is weighted by the probability of being in that state.

4.2 Accuracy Improving Guidelines

The previous section shows that entropy serves as a promising tool for anomaly detection. However, the detectors which fail to use it efficiently offer poor performance. In this section we



(a) Average detection rate



(b) Average false alarm rate

Figure 4.1: Evaluation results of the Maximum Entropy, Keystroke Session and KL divergence detectors on Endpoint dataset.

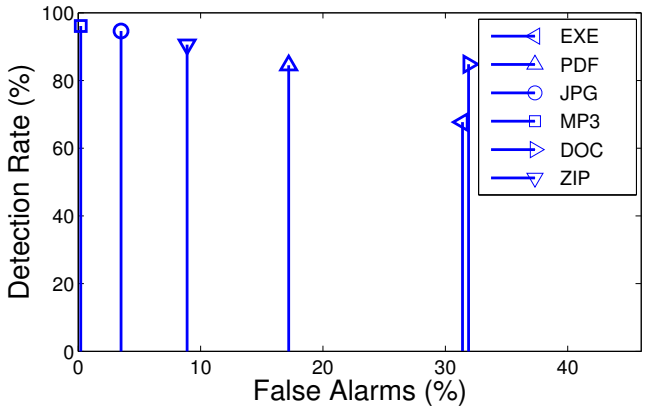
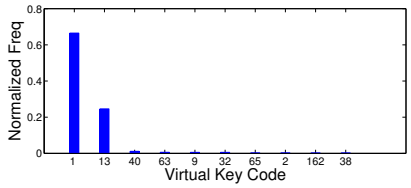


Figure 4.2: Accuracy evaluation of the embedded malware detector.

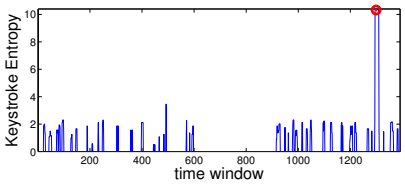
provide promising guidelines for the future use of entropy for anomaly detection based on the learnings from the performance evaluation of the existing entropy based detectors.

4.2.1 Feature correlation should be retained

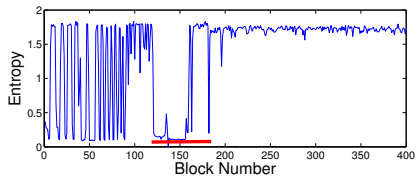
Significant correlation exists across traffic and/or host features. However, this correlation is missed out in the marginal feature distributions. Consequently, detectors which perform isolated feature analysis are unable to detect anomalies which perturb this correlation. This correlation can be modeled using joint/conditional multiple feature distributions and then performing entropy analysis to detect the anomalies. The increased performance resulting from the use of feature correlation is testified by the keystroke session detector which performs the best among all the entropy based detectors. This is due to the reason that the histogram of keystrokes which are used to initiate network sessions is skewed [see Fig. 4.3(a)] and perturbation in this metric can easily reveal the presence of an anomaly. While analyzing the entropies of the marginal keystroke distribution and/or the marginal session distribution is clearly not



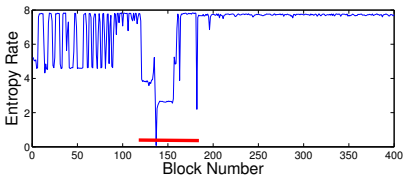
(a) Histogram of session-keystrokes



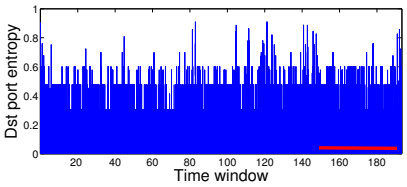
(b) Joint Entropy of sessions and keystrokes



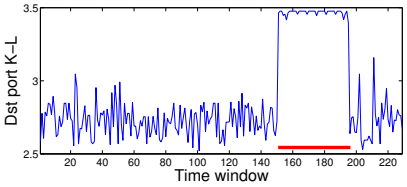
(c) Entropy of an infected file



(d) Entropy Rate of an infected file



(e) Destination Port Entropy



(f) Destination Port Relative Entropy

Figure 4.3: Examples to support the limitations of the current use of entropy.

useful, Fig. 4.3(b) shows that quantifying these features using joint (session-keystroke) entropy easily detects anomalous activity (Rbot-AQJ worm in this case).

4.2.2 Temporal/Spatial correlation should be retained

Benign behavior generally exhibits significant temporal and spatial patterns. Thus, temporal/spatial correlation analysis proves

useful in the detection of subtle anomalies. Entropy fails to take into account this correlation. To leverage this correlation, conditional feature distributions must be used in addition to the marginal feature distributions. Embedded malware detector provides a proof of concept example of this guideline. Fig. 4.3(c) shows the block-wise (block size = 1KB) entropy of a `PDF` file which is infected by an embedded executable malware. It is evident that entropy is unable to provide clear perturbations required for detection. On the other hand, entropy rate [Fig. 4.3(d)], which models and accounts for the spatial/temporal correlation, provides very clear perturbations at the infected file blocks; entropy rate quantifies the average entropy of conditional distributions.

4.2.3 Randomness quantification is not enough

Entropy cannot distinguish between differing distributions with the same amount of uncertainty; e.g., entropy of the normalized distribution of a source producing 90 packets on port 80 and 10 packets on port 21 is the same as a source producing 900 packets on port 6666 and 100 packets on port 6667. Thus anomalies which do not perturb randomness go undetected. This limitation arises due to the fact that entropy does not take the individual symbol values into account. It is therefore important to perform a symbol-by-symbol comparison between benign and observed distributions. This can be achieved by 1) computing the relative entropy of the distributions or 2) incorporating some notion of weight by using entropy rate. Fig. 4.3(e) shows a case where the Blaster worm cannot be detected in the destination port entropy time-series; endpoint traffic dataset from [16] is used for this experiment. This limitation arises due to the fact that entropy does not take the individual port numbers into account.Fig. 4.3(f) shows that K-L divergence time series

of destination port is perturbed due to the presence of `Blaster` worm. Chapter 4 also shows that detectors which use KL divergence and entropy rate measures are capable of achieving greater accuracy.

Chapter 5

Conclusion and Future Work

We surveyed the usage of entropy for anomaly detection and developed a taxonomy of the surveyed detectors. We then evaluated the detectors on publicly available datasets and studied the results to understand the variations in the performance of the evaluated detectors. Our study has found that the detectors which perform poorly often use the *entropy measure* inefficiently. Based on this observation, we highlight three important shortcomings of existing entropy-based ADSes and propose guidelines for efficient entropy usage.

One avenue for future work is to carry out a detailed comparative design evaluation by modifying the detectors to mitigate the shortcomings. Evaluating the modified detectors will illustrate the performance improvements achieved by the use of powerful entropy measures.

Part II

Botnet Detection

Chapter 6

Introduction

The magnitude of malware threats has risen considerably in the past few years due to the emergence of botnets - networks of zombie computers which can be commanded to carry out coordinated large scale malicious activities. Contemporary botnets contain millions of bots [38] which provide extensive bandwidth and processing power to carry out malicious activities, such as distributed denial of service attacks, information harvesting, click fraud, spamming and cyber warfare. Recent examples include the botnet-based Web War I attack on Estonia's Internet infrastructure [45], Blue Security being thrown out of business [45], and Google paying 90 million dollars to settle a click fraud lawsuit [46]. Moreover, Symantec reported that more than 90% of email spam is generated through botnets [38]. Despite a serious commercial and research interest in bot detection, botnet sizes are increasing rapidly with alarming estimates of 10-15% worldwide infected hosts reported by recent studies [41, 42].

6.1 Motivation and Problem Statement

Botnet detection has emerged as a vibrant area of research and a number of bot detectors have been proposed recently [47, 48, 49, 50, 51, 52, 53, 54, 55, 56]. These techniques rely on different

traffic and host features to detect bots, including network flow data, DNS traffic, system calls, etc. However, no effort has been expended in comparing the performance of these detectors, mainly because of an unavailability of detector implementations and bot datasets. Consequently, there is a gap in our current understanding of which bot detection approaches are better than others and why.

In this thesis, we take a first step towards filling the above gap in the botnet research literature. We perform comparative evaluation of prominent network based detectors and release the open source implementations and dataset publicly. A comparative evaluation on a public dataset will help the research community: 1) understand strengths and weaknesses of existing techniques; 2) identify promising features and guidelines for bot detection; and, most importantly, 3) open doors for future research which can use the publicly available datasets, implementations and features/guidelines for performance benchmarking.

We therefore formulate the problem statement of part II of thesis as,

> *To carry out a judicious comparative evaluation of existing network based bot detectors on a comprehensive dataset, and to identify promising guidelines for future bot detectors.*

6.2 Contribution

To unveil the weaknesses of bot detectors, in this study, we test how well the detectors perform under the simplest bot detection problem. Centralized botnets which use a single command and control server pose the easiest detection problem because group activity is observed in network traffic as a result of bot communication. Despite their ease of detection, Internet Relay Chat

(IRC) based botnets, the most primitive type of centralized botnets, are estimated to be currently 47% of the total share due to their ease of use and efficiency [38]. Therefore, in this study, we choose IRC based botnet detection as the performance evaluation platform.

We compare the accuracies of three network-based [47, 48, 49] on a dataset comprising of four prominent IRC botnets, Sdbot, GTbot, Rxbot and Spybot. Attack traffic is generated by modifying the binaries and issuing a mix of all the available commands from an IRC Command and Control server under our control. The attack traffic is then merged with real-time background traffic at two varying data rates. In addition to evaluating how well a detector performs, we also test if it is resilient to variation in background traffic rate and encryption of botnet traffic. Our evaluation shows that a detector's accuracy may vary under one or both of the above mentioned scenarios. We therefore, propose guidelines to mitigate the accuracy degradation due to the two factors.

Salient contributions of this study are thus:

1. Collection of a comprehensive IRC botnet traffic dataset.

2. Development of an open-source library of botnet detectors which can be used for repeatable performance benchmarking by future detectors.[1]

3. Quantification and comparison of the accuracies of existing bot detection techniques under a variety of botnet attacks.

4. Identification of promising guidelines that can be used by future bot detectors.

[1]Background and attack datasets are available at `http://www.wisnet.seecs.nust.edu.pk/downloads/datasets/IRCbots`. Botnet detector implementations are also available at `http://www.wisnet.seecs.nust.edu.pk/downloads/botdetectors`

6.3 Organization of Part II

Part II of the thesis is organized as follows,

Chapter 7 provides essential background information on botnets.

Chapter 8 describes in detail our dataset collection setup and the bot detectors used in this study.

Chapter 9 discusses our performance evaluation results and proposes accuracy improving guidelines.

Chapter 10 concludes this part of the thesis and frames future work.

Chapter 7

Background

Bots are programs that are covertly installed on a victim's computer and allow an attacker to remotely control the targeted computer through a communication channel, such as Internet relay chat (IRC), peer-to-peer (P2P), or HTTP. Computers infected with bot malware are known as *zombie* computers. A *botnet* is a network of such zombie computers. The attacker controlling the botnet is known as *botmaster*. Botnets allow botmasters to carry out large scale coordinated malicious activities by issuing commands through the command and control (C & C) channel.

7.1 Botnet Lifecycle

First a machine is infected with bot malware. The bot infection phase is similar to other malware infections and results from an exploit, visting a malicious URL, opening a malicious email attachment etc.

After a bot binary has downloaded, it sits until it's execution. Once executed, the bot process tries to connect to the command and control server. The location of C & C server is embedded in the bot binary either statically by storing an IP address(es) or dynamically by storing a DNS name. The bot masters today

mainly use the latter technique in order to provide resiliency against disruption.

Once the bot infected machine is connected to the C & C server, the botmaster can issue commands to the newly turned bot to participate in malicious activities such as DDoS, spamming, email harvesting etc. The botmaster can also instruct the bot to recruit more bots.

7.2 Types of Botnets

Botnets are categorized on the basis of the type of C & C channel they use.

Centralized Botnets use a single C & C channel to communicate with all the bots. Internet Relay Chat based botnets are an example of the centralized botnets.

Decentralized Botnets use a peer-to-peer architecture for their C & C communication. P2P based botnets are examples of decentralized botnets.

Centralized architecture is more efficient for the botmaster. However, it suffers the drawback of easy detection and disruption due to the single point of failure.

7.3 History of Botnets

Bots came into being as useful tools for carrying out automated tasks. The first bot was an IRC based virtual user that could play games on an IRC channel on a user's behalf. With the passage of time, however, the malicious minds discovered the potential of botnets to earn profit by engaging in malicious activities.

Table 7.1 shows the timeline of emergence of various botnets. IRC botnets are the most primitive kind of botnets which ap-

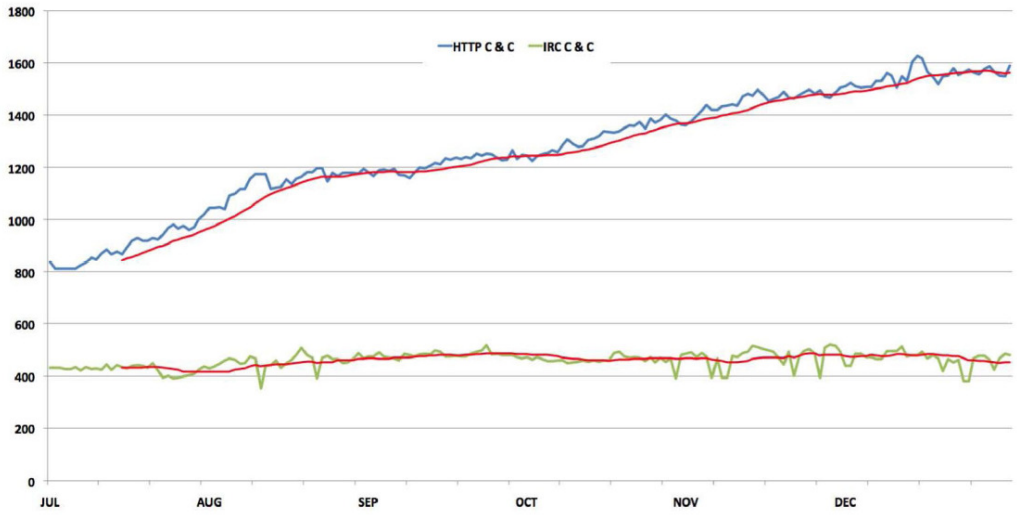


Figure 7.1: An analysis of recent botnet activity [40].

Table 7.1: Timeline of Botnets

Year	**Botnet**	**Type**
1998	GTbot	IRC
2002	SDbot	IRC
2002	Agobot	IRC
2003	Spybot	IRC
2003	Sinit	P2P
2004	Phatbot	P2P
2004	Rxbot	IRC
2004	Bobax	HTTP
2006	SpamThru	P2P
2006	Nugache	P2P
2007	Peacomm / Storm	P2P
2008	Kraken	P2P

peared around 1998. Due to the ease of detection of centralized botnets, bot masters started shifting towards decentralized botnets, such as P2P around 2003. According to Symantec's report, in 2008 the percentage of IRC botnets is estimated to be 47% [38]. Team Cymru's analysis of the recent botnet activity shows that although web based HTTP botnets are on the rise, the

number of IRC botnets tracked has not declined over the year 2009 [40]. Figure 7.1 shows this trend. According to a more recent report by Trendmicro, IRC based botnets are now being used as propagation tools for other types of botnets [35]. For example, Sdbot is being used by botmasters to spread spamming botnet *Cutwail* and social networking site Facebook based botnet, *Koobface*. Therefore, in this study we focus on the detection of primitive IRC based botnets.

Chapter 8

Evaluation Framework

In this chapter, we describe our evaluation framework — dataset collection setup and the detectors used in evaluation.

8.1 Bot Detectors

In this section, we provide a description of bot detectors used in the study. We evaluate *network based* bot detectors in this study in order to understand how well pure network based detection fares.

Network based bot detectors are categorized as either vertical correlation based or horizontal correlation based. Vertical correlation based detectors use individual behaviour to detect bots by correlating multiple stages of infection of a single machine. Horizontal correlation based detectors, on the other hand, detect botnets by correlating activities across multiple hosts.

In this study, we evaluate one vertical correlation based bot detector, BotHunter and two horizontal correlation based detectors, BotSniffer and BotMiner. We use the implementation of Bot Hunter provided by SRI International, and implement the other two detectors ourselves. We publicly release our implementations for future evaluation.

8.1.1 BotMiner

BotMiner monitors each host in two planes: 1) Communication (C plane), and 2) Activity (A plane) [47]. The C plane monitor simply logs the network flows and the A plane monitor detects hosts performing four malicious activities: scanning, spamming, binary downloading, exploit attempts. Next, clustering is performed in the two planes. The C plane clusterer first forms C-flows by aggregating the flows having the same protocol, source IP, destination IP and port during a fixed time period. X-means clustering algorithm is then applied to form the clusters based on four features: number of flows per hour (fph), number of packets per flow (ppf), average number of bytes per packet (bpp), average number of bytes per second (bps). The A plane clusterer groups hosts performing the same type of malicious activity together. In the last step, botnet score for each host is calculated by performing cross-cluster correlation and the hosts below the threshold are dropped.

8.1.2 BotSniffer

BotSniffer is a network packet based approach designed for the detection of bots which use chat based protocols such as IRC and HTTP [48]. Port independent protocol matchers based on keyword analysis of payload are first used to extract IRC and HTTP traffic. From this filtered traffic, response crowds are formed by grouping the traffic towards a particular destination IP and port. Response crowds are then analyzed for message and activity correlation. IRCPRIVMSG's within a response crowd are analyzed for correlation using the Response Crowd Homogeneity Check Algorithm which uses DICE algorithm to perform a bi-gram overlap analysis of the payload. Response Crowd Density Check algorithm which uses threshold random walk (TRW) to establish confidence is used to detect activity correlation and

thus detect the likelihood of the response crowd being a botnet.

8.1.3 BotHunter

Bothunter[49] analyzes the sequence of communication exchanges between a host and the internet. It models the infection as a loosely coupled sequence of five stages: inbound scanning, exploit usage, egg downloading, outbound bot coordination dialog and outbound attack propagation. These five stages will be referred to as *E1, E2, E3, E4* and *E5* respectively in the rest of this thesis. Suspicious outbound activity coupled with intrusion detection activity indicates a successful bot infection. BotHunter's detection architecture consists of three components, SCADE (Statistical Scan Anomaly Detection Engine), SLADE (Statistical Payload Anomaly detection engine) and Snort rule based detection engine. SCADE detects incoming and outgoing scans and SLADE performs a lossy n-gram payload analysis to look for divergences from benign byte distributions of certain protocols. Alerts from these three components are input to the bothunter correlation engine which uses a weighted equation to determine bot score of a host.

8.2 Dataset Collection

In this section, we describe the experimental setup used for the collection of the botnet and background network traffic.

8.2.1 Botnet Traffic

Four IRC based botnets namely `GTbot`, `SDbot`, `Spybot`, and `Rxbot` were used in this study. The source code of these bots is available on the web, and thus we were able to modify the location of C & C servers to point to servers under our control.

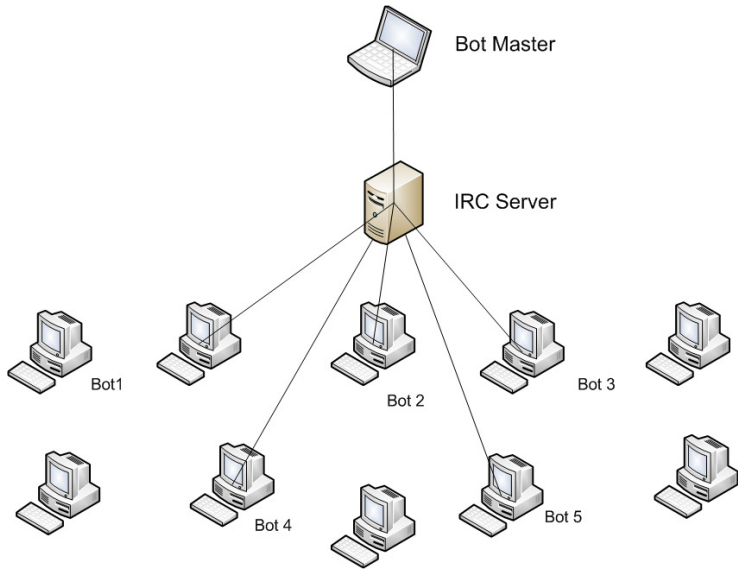


Figure 8.1: Botnet Dataset Collection Setup

Table 8.1: Commands in the dataset

GT	**SD**	**Rx**	**Spy**
login	login	login	login
info	status	id	start keylogger
scan (port 6667)	sysinfo	keylog on	list processes
scan (port 80)	netinfo	clone	opencmd
scan (port 8080)	download URL	driveinfo	cmd command
portredirect	sysinfo	capture screen (15 times)	keyboardlights
nick newnick	spy	get (15 times)	cd-rom 1
fileserver.access	udp	delete (15 times)	scan
update	stopspy	Dcc send	syn
cstats	ping	execute	info
scan (port 113)		httpserver	sendkeys
pac		httpstop	stopkeylogger
show.access			reboot
voiceall			
cstats			
icqpagebomb			
pac			
exit			

Table 8.2: IRC C & C traffic characteristics

	GT	**SD**	**Rx**	**Spy**
Number of infected machines	5	5	4	4
Number of Packets (IRC)	687	1657	386	2292
Time between first and last packet (secs)	2758.7	2121.5	1117.5	2101.04
Avg. packets per second	0.25	0.78	0.35	1.09
Avg. packet size (bytes)	185.41	130.82	186.81	165.95
Avg. bytes per second	46.17	102.16	64.25	181.03

Figure 8.1 shows our experimental setup. Four to five machines were infected for each bot malware in a controlled virtual network via two channels: (i) IRC DCC send file, and (ii) network share copying. The command and control server was set up at a home computer on a network different from those of bots. Botnet traffic was generated and captured for durations ranging 20 - 45 mins by issuing commands from our C & C server to the infected machines.

Table 8.1 lists the commands that were issued for each bot malware. As can be seen from the table, the traffic captured contains malicious commands such as flood attacks, information harvesting, scanning as well as seemingly benign commands such as getting the system information, restarting the computer, visiting a URL. Moreover, the GT and SD commands result mainly in network activity whereas Rx and Spybot commands initiate host activity. Table 8.2 lists the characteristics of IRC command and control traffic for the four bots. The botnet command and control traffic rate is quite low, ranging from 0.25 - 1.09 packets/sec.

8.2.2 Background Traffic

We captured background traffic at our research lab router. This provides a realistic evaluation scenario, since the network bot

Table 8.3: Benign traffic characteristics

	Benign trace 1	**Benign trace 2**
Number of Packets (IRC)	2521897	2970928
Time between first and last packet (secs)	4377.67	2924.27
Avg. packets per second	576.08	1015.95
Avg. packet size (bytes)	482.95	760.47
Avg. bytes per second	278216.89	2970928

detectors are correlation based enterprise solutions to be deployed at router. The lab router handles traffic from a total of 28 computers running different operating systems, applications and services. Inbound, outbound and internally routed traffic was captured at the lab router using port mirroring. The traces included traffic from activities such as peer-to-peer file sharing, software downloading from remote servers, web browsing, and real-time video streaming.

The background traffic was captured at two different times of the day - 11:00am and 4:15pm to obtain varying data rates and to test if the detectors' accuracy was affected by an increase in the background data rate. Note that 11:00am is the peak Internet activity time and at 4:00pm the traffic rate lowers down. From now on, we refer to the low rate traffic as *benign trace 1* and the other one as *benign trace 2*. The detailed traffic characteristics for the two benign traces are listed in Table 8.3.

For *benign trace 1*, the average traffic rates on the five endpoints whose traffic was merged with bot traffic varied between 1.8 to 202.31 packets/second with a standard deviation of approximately 89.25 packets/second. For *benign trace 2*, average traffic rates range between 0 (no network activity) and 21.66 packets/ second with a standard deviation of 10.11 packets/second.

8.2.3 Merged traffic

Merged traffic was obtained by merging the botnet traffic with the background traffic and consists of eight traces - each of the four bot traffic traces merged first with *benign trace 1* and secondly with *benign trace 2*. From now on, we refer to these traces as: GT_b1, GT_b2, SD_b1, SD_b2, Rx_b1, Rx_b2, Spy_b1, and Spy_b2.

Chapter 9

Accuracy Evaluation and Improvement

In this chapter, we describe the evaluation results of the three detectors: BotHunter, BotSniffer and BotMiner on the eight traces discussed in the previous chapters. In addition to evaluating which detector offers better accuracy, we analyze the detectors for the following two characteristics: 1) Is the detector's accuracy affected by background traffic rate?, and 2) Is the detector's accuracy affected by encryption?

9.1 Evaluation Results Discussion

Table 9.1 shows the detection results of BotHunter. An average detection rate of 38% was observed with zero false alarms. We investigated the missed detections and found that in some cases although Snort was generating alerts, but when input into the bothunter correlator failed to be detected. For example, in the case of GT bot, E4 alerts indicating potential bot commands were generated, but BotHunter failed to flag the infected machines as bots. This is due to the reason that the Bothunter correlation engine operates on time window basis, and if enough evidence is not collected in the specified window, then at the end

Table 9.1: BotHunter Detection Results

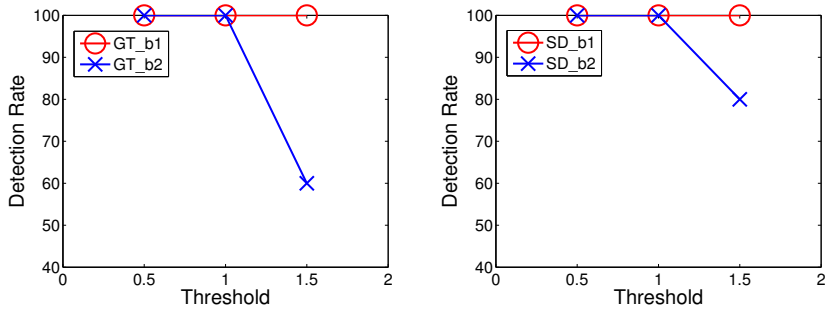
Trace	Events detected by Snort	Detection Rate	False Alarm
GT-b1	E3,E4	0% (0/5)	0%
GT-b2	E3,E4	0% (0/5)	0%
SD-b1	E2,E3,E4,E5,E6	100% (5/5)	0%
SD-b2	E2,E3,E4,E5,E6	100% (5/5)	0%
Rx-b1	E2,E3,E4	50% (2/4)	0%
Rx-b2	E2,E3,E4	50% (2/4)	0%
Spy-b1	E3	0%	0%
Spy-b2	E3	0%	0%

Table 9.2: BotSniffer Detection Results at threshold 0.5

Trace	Detection Rate	False Alarm
GT-b1	100% (5/5)	0%
GT-b2	100% (5/5)	0%
SD-b1	100% (5/5)	0%
SD-b2	100% (5/5)	0%
Rx-b1	100% (4/4)	0%
Rx-b2	100% (4/4)	0%
Spy-b1	100% (4/4)	0%
Spy-b2	100% (4/4)	0%

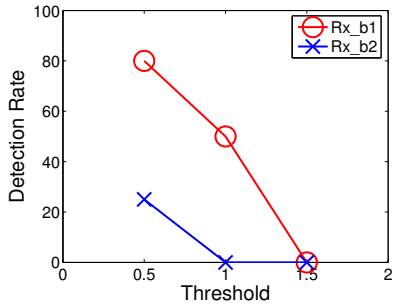
of time window, that evidence was discarded. Since the implementation of BotHunter provided by SRI International does not have any tunable parameter, we could not experiment with detection rates under different time windows. Table 9.1 also shows that Bothunter is unaffected by variation in background traffic rate. However, encryption of botnet traffic will affect Bothunters' accuracy due to it's reliance on a signature base to catch malicious activities. The SLADE component of BotHunter also analyzes the content of payloads.

Table 9.2 shows that BotSniffer achieves 100% detection rate on all the traces without generating any false alarms. The accurate detection of BotSniffer results from the Response Crowd Homogenity Check module which first separates out the IRC traffic and then performs similarity analysis on the payloads.

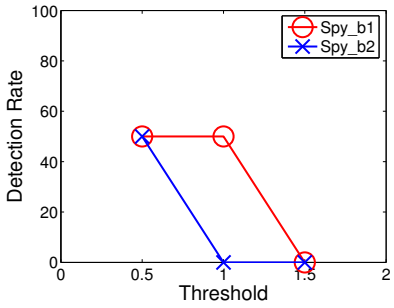


(a) GTbot

(b) SDbot



(c) Rxbot



(d) Spybot

Figure 9.1: BotMiner Detection Results

BotSniffer's performance although unaffected by variation in data rate, will severely degrade if the IRC traffic is encrypted. Further, this technque's power lies in similarilty correlation across different machines and fails to detect individual bots.

Figure 9.1 shows the detection results of BotMiner. For the traces *GT_b1* and *SD_b1*, 100 % detection rate is observed. However, for *Rx_b1* and *Spy_b1*, the detection rate falls as the overlap threshold is increased. This is because the GT and SD bot commands lead to network activity which results in well formed A-plane clusters. However for Rx and Spy, the commands primarily result in host based activity. Consequently, enough A plane activity is not observed in the network traffic. Moreover,

Table 9.3: Evaluation Summary

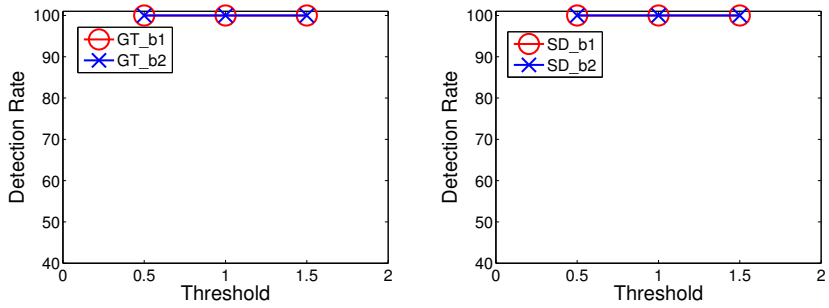
Detector	Average Detection Rate	Affected by Encryption?	Affected by Background Traffic Rate?
BotHunter	100%	Yes	No
BotSniffer	100%	Yes	No
BotMiner	83%	No	Yes

for the high data rate traces, an average detection rate of 68.75% at an overlap threshold of 0.5 which falls further as the threshold is increased. This reduced reduction on the high data rate trace is due to the fact that C plane clustering is not performed effectively.

Table 9.3 shows a summary of the bot detectors' performance. The detection rate shown is averaged over the eight traces. We conclude that the detectors fail to achieve good detection even on the most simple centralized botnet detection problem. Those achieving good detection rates are affected by encryption and high background traffic. In the next section, we propose guidelines to mitigate the effects of the above factors.

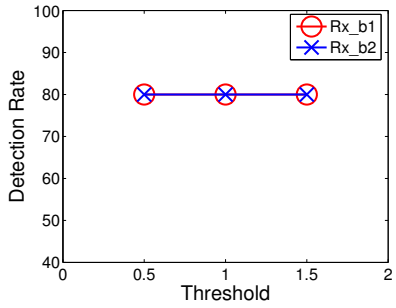
9.2 Accuracy Improving Guidelines

The evaluation results show that the following are the accuracy degrading factors in the bot detectors: 1) Payload encryption, 2) High background traffic rate leading to ineffective C plane clustering, and 3) Lack of sufficient activity evidence. Considering the first factor, since payloads can be easily encrypted, any payload analysis based bot detection scheme will fail. Therefore, for bot detection in the communication plane, flow information should be used. To mitigate the other two factors mentioned above, in this section, we propose accuracy improving guidelines for the detectors.

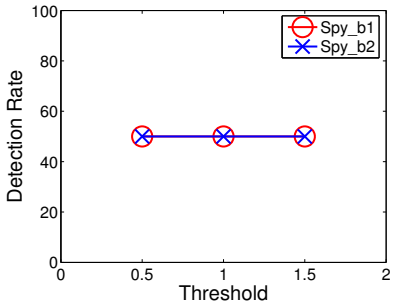


(a) GTbot

(b) SDbot



(c) Rxbot



(d) Spybot

Figure 9.2: BotMiner Detection Results

9.2.1 Introducing a traffic splitter

Based on the evaluation results, we note that the high accuracy of BotSniffer results from the fact that it first separates the IRC traffic for further analysis. This traffic splitting reduces noise and consequently, improves detection. Thus, traffic splitting is a promising generic module that can be introduced in any bot detector. Figure 9.2 shows the preliminary evaluation of BotMiner after this module was incorporated. It can be seen that the addition of this module results in improved detection rate. This is due to the reason that the separation of IRC traffic resulted in improved C plane clustering and consequently, bots

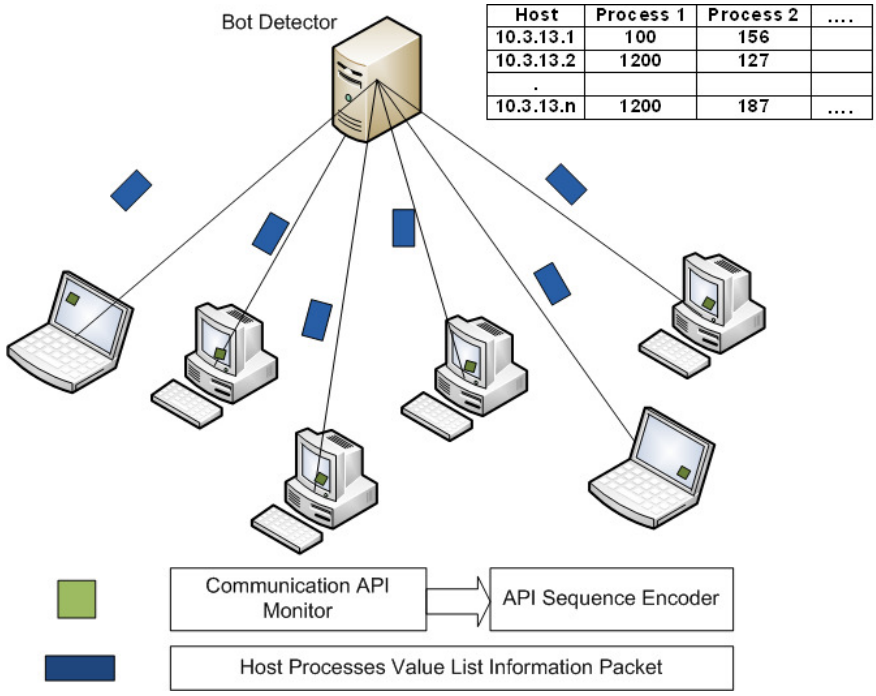


Figure 9.3: Host Activity Module

were detected at higher overlap thresholds.

9.2.2 Extracting activity evidence by analyzing host behaviour

A bot detectors' accuracy can be improved if it also takes into account bot infection indicating host activity in addition to network activity. Rxbot and Spybot are examples of bots whose malicious activity is observed on the infected bot machine rather than in the network traffic. To this end, we propose a framework through which machines in a network can communicate a summary of their host activity to a central machine running the bot detector.

Our proposal is based on the observation that the sequence of communication API functions called by a bot process on all the machines is the same. Therefore, our API monitor mod-

ule on a host machine monitors the following five API calls: *socket, send, sendto, recv, recvfrom* for every process running on the machine who has opened a socket for communication. In addition to being sufficient for similar host activity analysis, monitoring only the above mentioned API calls results in little complexity. Moreover, only those processes are monitored which are involved in network activity. The sequence of each monitored process is then encoded in the form of a number and communicated to the network component of the bot detector. The bot detector maintains a table containing entries for process scores of every host machine. The A planeclusterer in the bot detector can then analyze this table for similar sequence scores across the list of network hosts. The proposed architecture is shown in Figure 9.3. Preliminary evaluation of this module incorporated in BotMiner shows the successful detection of Rxbot and Spybot without raising any false alarms.

Chapter 10

Conclusion and Future Work

We carried out the first comparative performance evaluation of three prominent network based bot detectors. For this purpose, we collected a dataset containing *four* easiest to detect IRC bots. Our evaluation shows that pure network based bot detection from the vantage point of network router suffers two problems even on the easiest detection problem: (i) encrypted traffic, and (ii) high background traffic rates. We thus argue that pure network based bot detection is an incomplete solution and propose a low cost joint host-network bot detection solution. The host component monitors the communication API and periodically sends encoded API call sequences per process to the network component. The network component correlates and compelements this information with the network detection results to build confidence regarding bot infections. We incorporate the host activity module in the evaluated network detectors and show increased detection rate without any false alarms. We also propose traffic splitting to mitigate the performance degradation due to high background traffic rate.

In our work we evaluated the detectors only on the most primitive kind of bots i.e. IRC bots. One avenue for future work is evaluation of the modified detectors on other types of bots i.e. P2P and HTTP. Such an extensive evaluation will highlight any

shortcomings in detecting different kinds of botnets and will help develop techniques to detect the new kinds of bots.

Bibliography

[1] Snort, `http://www.snort.org/`

[2] Bro Intrusion Detection System, `http://www.bro-ids.org/`

[3] VX Heavens, `http://vx.netlux.org/`

[4] Next-Generation Intrusion Detection Expert System, `http://www.csl.sri.com/projects/nides/`

[5] W. Lee, D. Xiang, "Information-theoretic Measures for Anomaly Detection," in *IEEE Symp. On Security and Privacy '01.*

[6] L. Feinstein, D. Schnackenberg, R. Balupari, D. Kindred, "Statistical approaches to DDoS attack detection and response," in *DISCEX '03.*

[7] A. Wagner, B. Plattner, "Entropy Based Worm and Anomaly Detection in Fast IP Networks," in *IEEE WET ICE '05.*

[8] Y. Gu, A. McCullum, D. Towsley, "Detecting anomalies in network traffic using maximum entropy estimation," in *ACM Usenix IMC '05.*

[9] A. Lakhina, M. Crovella, C. Diot, "Mining anomalies using traffic feature distributions," in *ACM SIGCOMM '05.*

[10] D. Brauckhoff, B. Tellenbach, A. Wagner, M. May, A. Lakhina, "Impact of packet sampling on anomaly detection metrics," in *ACM SIGCOMM '06*.

[11] S. A. Khayam, H. Radha, D. Loguinov, "Worm Detection at Network Endpoints using Information-Theoretic Traffic Perturbations," in *ICC '08*.

[12] Next Generation Intelligent Networks Research Center, `http://nexginrc.org/Default.aspx`.

[13] M. Z. Shafiq, S. A. Khayam, M. Farooq, "Embedded Malware Detection using Markov ngrams," in *DIMVA '08*.

[14] S. A. Khayam, H. Radha, "Using Session-Keystroke Mutual Information to Detect Self-Propagating Malicious Codes," in *ICC '07*.

[15] `http://wisnet.seecs.nust.edu.pk/projects/ENS/DataSets.html`

[16] A. B. Ashfaq, M. Joseph, A. Mumtaz, M.Q. Ali, A. Sajjad and S. A. Khayam, "A Comparative Evaluation of Anomaly Detectors under Portscan Attacks," in *RAID '08*.

[17] M. Q. Ali, H. Khan, A. Sajjad, S. A. Khayam, "On Achieving Good Operating Points on an ROC Plane using Stochastic Anomaly Score Prediction," in *ACM CCS '09*.

[18] Computer Immune Systems Datasets, `http://www.cs.unm.edu/~immsec/data/synth-sm.html`.

[19] S. Ando, "Clustering needles in a haystack: An information theoretic analysis of minority and outlier detection," in *7th International Conference on Data Mining*.

[20] H. He, J. Wang, W. Graco, and S. Hawkins, "Application of neural networks to detection of medical fraud," in

Expert Systems with Applications, Volume 13, Number 4, November 1997 , pp. 329-336.

[21] Z. He, S. Deng, and X. Xu, "Outlier detection integrating semantic knowledge," in *Third International Conference on Advances in Web-Age Information Management '02.*

[22] Z. He, S. Deng, X. Xu, and J. Z. Huang, "A fast greedy algorithm for outlier mining," in *Proceedings of 10th Pacific-Asia Conference on Knowledge and Data Discovery '06.*

[23] Z. He, X. Xu, and S. Deng, "Discovering cluster-based local outliers," in *Pattern Recognition Letters*, Volume 24, Issues 9-10, June 2003, Pages 1641-1650.

[24] Z. He, X. Xu, and S. Deng, " An optimization model for outlier detection in categorical data," in *International Conference on Intelligent Computing '06.*

[25] Z. He, X. Xu, J. Z. Huang, and S. Deng, "A frequent pattern discovery method for outlier detection," in *Springer Berlin/Heidelberg*, Vol. 3129, 2004, pp. 726-732.

[26] Z. He, X. Xu, J. Z. Huang, and S. Deng, "Mining class outliers: Concepts, algorithms and applications," in *Springer Berlin/Heidelberg*, Vol. 3129, 2004, pp. 588-589.

[27] A. Arning, R. Agrawal, and P. Raghavan, "A linear method for deviation detection in large databases," in *2nd International Conference of Knowledge Discovery and Data Mining*, 1996.

[28] S. Chakrabarti, S. Sarawagi, and B. Dom, "Mining surprising patterns using temporal description length," in 24^{th} *International Conference on very large databases*, 1998.

[29] J. Lin, , Keogh, E., Fu, A., and Herle, H. V. 2005, "Approximations to magic: Finding unusual medical time series," in Proceedings of the *18th IEEE Symposium on Computer-Based Medical Systems*, IEEE Computer Society, Washington, DC, USA, 329-334.

[30] Lin, S. and Brown, D. E. 2003, "An outlier-based data association method for linking criminal incidents," in Proceedings of the *3rd SIAM Data Mining Conference*.

[31] Noble, C. C. and Cook, D. J. 2003, "Graph-based anomaly detection," in Proceedings of the 9th ACM *SIGKDD international conference* on Knowledge discovery and data mining. ACM Press, 631-636.

[32] V. Chandola, A. Banerjee, V. Kumar, "Anomaly Detection : A Survey," *ACM Computing Surveys*, Vol. 41(3), Article 15, July 2009.

[33] Y. Kopylova, D. Buell, C.-T. Huang, J. Janies, "Mutual Information Applied to Anomaly Detection", in *Journal of Communications and Networks*, Vol. 10, No. 1, 2008, pp. 89-97.

[34] A. Ziviani, M.L. Monsores, P.S.S. Rodrigues, A.T.A. Gomes, "Network Anomaly Detection using Nonextensive Entropy," in *IEEE Communications Letters*,vol.11, no. 12, Dec 2007, pp. 1034-1036.

[35] SdBot IRC Botnet Continues to Make Waves, December 2009, `http://us.trendmicro.com/imperia/md/content/us/trendwatch/researchandanalysis/sdbot_irc_botnet_continues_to_make_waves_pub.pdf`

[36] UnRealIRCd, `http://www.unrealircd.com/`

[37] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and D. Dagon, "Peer-to-peer botnets: Overview and case study," in Proceedings of *USENIX HotBots*, 2007.

[38] Symantec Internet Security Threat Report, Trends for 2008, Volume XIV, Published April 2009.

[39] Symantec Internet Security Threat Report, Trends for 2009, Volume XV, Published April 2010.

[40] Developing Botnets, `http://www.team-cymru.org/ReadingRoom/Whitepapers/2010/developing-botnets.pdf`

[41] How Many Bot-Infected Machines on the Internet, `http://www.avertlabs.com/research/blog/index.php/2007/01/29/how-many-bot-infected-machines-are-on-the-internet/`

[42] Emerging Cyber Threats Report for 2009, Georgia Tech Information Security Center, October 15,2008, `http://www.gtiscsecuritysummit.com/pdf/CyberThreatsReport2009.pdf`

[43] BBC News: WEBER, T., Criminals may overwhelm the web. `http://news.bbc.co.uk/1/hi/business/6298641.stm`, 2007.

[44] Attack of the Bots by Scott Berinato, `http://www.wired.com/wired/archive/14.11/botnet.html`.

[45] Michael Lesk, "The New Front Line: Estonia under Cyberassault," IEEE Security and Privacy, vol. 5, no. 4, pp. 76-79, July/Aug. 2007, doi:10.1109/MSP.2007.98

[46] L. Zhang and Y. Guan, "Detecting Click Fraud in Pay-Per- Click Streams of Online Advertising Networks," in *ICDCS*, 2008, pp. 7784.

[47] G. Gu, R. Perdisci, J. Zhang and W. Lee, "BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection," in Proceedings of the *17th USENIX security Symposium*, 2008.

[48] G. Gu, J. Z and W. Lee, "BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic," in Proceedings of the *15 Annual Network and Distributed System Security*, pp. 1-18, Jan, 2008.

[49] G. Gu, P. Porras, V. Yegneswaran, M. Frog, and W. Lee, "BotHunter: Detecting malware infection through ids-driven dialog correlation," in Proceedings of the *16th USENIX Security Symposium*, Boston, MA, August 2007.

[50] G. Gu, V. Yegneswaran, P. Porras, J. Stoll, and W. Lee, "Active Botnet Probing to Identify Obscure Command and Control Channels," in Proceedings of *Annual Computer Security Applications Conference (ACSAC)*, Honolulu, Hawaii, December 2009.

[51] J. R. Binkley and S. Singh, "An algorithm for anomaly based botnet detection," in Proceedings of the *2nd Conference on Steps to Reducing Unwanted Traffic on the Internet* (SRUTI), San Jose, CA, July 2006.

[52] H. Choi, H. Lee, H. Lee, and H. Kim, "Botnet detection by monitoring group activities in DNS traffic," in Proceedings of the *7th IEEE International Conference on Computer and Information Technology (CIT)*, Washington, DC, October 2007.

[53] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer, "Using machine learning techniques to identify botnet traffic," in *2nd IEEE LCN Workshop on Network Security (WoNS)*, 2006.

[54] Y. Al-Hammadi, U. Aickelin and J. Greensmith, "DCA for Bot Detection," in Proceedings of *Congress on Evolution Computation(CEC)* IEEE, 2008.

[55] E. Stinson, J.C. Mitchell, "Characterizing bots remote control behavior," in Proceedings *DIMVA 2007*

[56] L. Liu, S. Chen, G. Yan, and Z. Zhang,"BotTracer: Execution-based Bot-like Malware Detection", in Proceedings of the *11th Information Security Conference (ISC)*, Taipei, September 15-18, 2008.

[57] Enhance netstat - the code project. `http://www.codeproject.com/KB/IP/enetstatasp.aspx`