A Data Mining Approach for Detection of Self-Propagating Worms

Mohd Fadzli Marhusin[1], Chris Lokan[1], Henry Larkin[2], David Cornforth[3]
[1]University of New South Wales, ACT 2600, Australia
{m.marhusin,c.lokan}@adfa.edu.au
[2]University of Aizu, Japan
research@logic.nu
[3]CSIRO Energy Technology, Mayfield West NSW 2304, Australia
david.cornforth@csiro.au

Abstract— **In this paper we demonstrate our signature based detector for self-propagating worms. We use a set of worm and benign traffic traces of several endpoints to build benign and worm profiles. These profiles were arranged into separate *n*-ary trees. We also demonstrate our anomaly detector that was used to deal with tied matches between worm and benign trees. We analyzed the performance of each detector and also with their integration. Results show that our signature based detector can detect very high true positive. Meanwhile, the anomaly detector did not achieve high true positive. Both detectors, when used independently, suffer high false positive. However, when both detectors were integrated they maintained a high detection rate of true positive and minimized the false positive.**

Keywords: self-propagating worm, worm detector, signature based detector, anomaly detector.

I. INTRODUCTION

The term "malware" refers to threats posed from code execution that causes damage or renders the system security useless [1]. In the early computer age, malware were created with limited objectives, whereas recent paradigms are also driven by profit-gain and information-gain motives. While some malware remain stationary until it is triggered, worms, such as *Code Red, Witty, Nachi, MyDoom, SoBig, Zotob, Witty, Blaster* are kind of malware that self-propagate from one host to another [2].

In this paper, we present a framework for detecting self-propagating worms using signature based and anomaly detectors. The contributions of this paper are as follows: we present a data mining approach for detecting self-propagating worms using benign and worm tree to form a signature based detection. We created our own tree structure based on a generic framework to store those signatures. When a tie-break is needed, an anomaly-based detection is used to make a final decision whether a given signature should be deemed as worm or benign. The anomaly detector bases its decision on comparing a signature against signatures in a fix-sized window of recent sessions. We report the result of both detectors and their integration and also compare their performance against some related work in the literature.

II. RELATED WORK

Basically there are several types of intrusion/malware detections [3-5], namely anomaly detection, signature based detection integration of both of these (called hybrid detection); and specification based detection.

Anomaly detector flags any activity that behaves or runs differently from the recorded profile. Although it has the ability in detecting novel attacks, it tends to generate high rate of false alarms. However, some authors [6], [7] claim that their anomaly detector alone could achieve outstanding results.

Signature based detection is a renowned detection type for malware and intrusion detections that uses known rules to detect known and unknown patterns, but share some characteristics to be detected. Basically a signature based detection represents knowledge in the form of "if <*condition*> then <*conclusion*>". However this technique is susceptible to a slight variation of the attack signature and also to an unknown attack.

Lakhina *et al.* [8] used entropy as a tool to detect anomalous traffic based on features of IP and port distributions. They claimed that using feature distributions can naturally divide anomalous from benign traffic and also uncover new anomaly types. The outcome of their experiment indicated that feature distribution is a key element to achieve a promising performance. However, their technique does not take into consideration the value of the features. For instance, there were repetitive port distributions over time; they did not take the values of the port into account when they used that technique.

Khayam *et al.* [6] proposed the use of Kullback-Leibler (KL) divergence measure to quantify perturbations in port distribution. KL is an information theoretic measure of the closeness between two probability distributions. They measured the perturbation of traffic in time-based windows. The basic idea of the technique is that they used the value they got in perturbation caused by worms, and compared it against existing recorded normal port distributions. Although they achieved 100% accuracy at most of the endpoints, their technique suffers from worms that propagate at a very low rate.

Shafiq *et al.* [7] suggested that intelligent consideration of several traffic features could improve detection. The features to be considered include burstiness of session arrivals, spikes in traffic volume, entropy of destination IP

978-0-7695-3838-9/09 $26.00 © 2009 IEEE
DOI 10.1109/NSS.2009.88

addresses, and divergence of port distribution. They considered fixed time windows of 30 seconds.

The time based window approach shows impressive outcomes but at some costs. Its drawback is that the cost to analyze a collection of ports in a window is much larger than using fix-sized window. Also, a dependency on an anomaly detector alone will make the cost consistently imposed to the system. Hence, we interested to evaluate the detection performance by using a fix-sized window rather than time based window, as implemented by previous authors. We are also interested to use signature based detector and only invoke the anomaly detector at certain conditions.

Abbess *et al.* [9] proposed a combination of pattern matching and protocol analysis approaches. The first method worked on multi-pattern matching strategy. The latter used a decision tree that was adaptive to the characteristic of the network traffic.

Williamson [10] proposed a rate limiting behavior to detect scanning viruses at host level. Their detection was simple with the assumption that scanning viruses will actively attempt to contact many computers in the network. This will usually cause a burst in the intensity of requests to make outgoing connections.

Gu *et al.* [11] measured the maximum entropy estimation of network traffic to get a baseline traffic distribution and relative entropy which were used to detect anomalous traffic.

Honeypot is also another strategy in detecting worm and intrusive activities. It is one or a set of machines deployed as a tool for a larger detection system to detect, analyze, and gather intelligence about attack activities. Honeystat [12] is one example.

III. THE CLASSIFICATION SYSTEM

Our detection system as depicted in Fig. 1, suggests that all incoming sessions will undergo checking against the benign and worm trees. These trees are decision trees, integrated as a signature based detector. Note that the trees use a signature which is split into 5 parts. When an incoming signature to be tested in the trees, it will be split into 5 parts first and each part will be matched against those parts available in the trees. In order to have a perfect match 5/5 rate in the trees, the detector may need to search the entire nodes. Having lesser match rate will shorten the searching effort.

We tested match rate 3/5, 4/5 and 5/5 and discovered that 4/5 is the best setting to achieve a good accuracy while reducing the searching cost. Thus, 4 is set as a minimum threshold *T*. The total matches returned by a tree when comparing the 5 parts of the signature is denoted as *D*. Given *T*=4, if the benign tree produces a match rate *D*/5 that surpasses a preferred threshold *D*>=*T* while the worm tree produces a match rate lesser than *D*/5, the session will be allowed to pass. When the worm tree recorded a match rate *D*/5 that surpasses a preferred threshold *D*>=*T* while the benign tree produces a match rate lesser than *D*/5, the session will be blocked. If both trees return rates where none of them surpass the required threshold, the anomaly detector will have its final say. The same would also be undertaken if both trees have tied match rates even though the threshold was surpassed. Details of the anomaly detector will be explained in section IV.

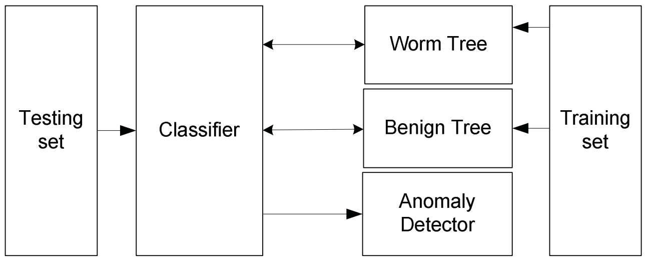

Figure 1. Classification system

IV. METHODOLOGY & ANALYSIS METHOD

We rely on session based information of endpoints to detect a self propagating worm. In our work, an endpoint refers to an individual computer system or device that acts as a network client and serves as a workstation or personal computing device [13]. This section describes the details of our work.

A. Data Collection

We obtained a dataset from [14] as used by [6,7]. They collected network profiles from 13 endpoints for over 12 months. Those endpoints' records consist of session traffic of home users, research students, technical staff and administrative staff. A session is defined as a two-way communication between two IP addresses. The dataset had a total signature of 1,881,234 benign sessions. The session data had the following 6 fields [7]:

- Session id: 20-byte SHA-1 hash sub names of host and remote IP address.
- Direction: a byte flag of 4 types; incoming/outgoing unicast, incoming/outgoing broadcast.
- Protocol; packet's transport layer protocol.
- Source port: packet' source port.
- Destination port: packet's destination port.
- Timestamp: millisecond resolution of session initiation.

Their worm dataset contained a total of 1,437,119 session signatures from 12 different types of worms. They were: 1) *Blaster*, 2) *Dloader-NY,* 3) *Forbot-FU*, 4) *MyDoom-A*, 5) *Rbot-AQJ*, 6) *RBOT.CCC*, 7) *Sdbot-AFR*, 8) *Sim Src Port*, 9) *CodeRed II*, 10) *Witty*, 11) *SoBig.E*, and 12) *Zotob.G*.

B. Dataset Transformation

We replaced the long session identifiers with unique integer values. We also decided to choose only the first five fields for our analysis.

If the same data is used to build and evaluate the detectors, the results would be unrealistic. Therefore, to evaluate the ability of the detectors to classify new and unseen sessions, we split the datasets, worm and benign respectively, into training and testing sets using a *k*-fold cross validation scheme where *k*=10.

C. *Tree Structure*

In our discovery oriented data mining method with *n*-nary worm and benign trees [15], new arriving sessions will be searched for matches inside the large pool of existing signatures stored in those trees.

Those worm and benign trees were constructed using the training datasets from the *k* fold sets. The ID3 [15], and its variants decision tree's induction algorithms were well developed to operate on generalized rules. In our case we needed to have an *n*-ary tree structure used to store the 5 parts of a given signature; each node was required to keep one part of the signature respectively. A simple hash table can also be used but we postponed that for future work.

A rule contains 5-fields added under the root node. Each field of the rule was stored into a single node in a right sequence. If two or more similar rules added into the tree, the existing nodes containing matching parts were reused whenever possible. Only the different parts were constructed as new nodes in the tree. The bottom most nodes were also merged, to represent consecutive ranges of values in a single node by storing the lower bound (LB) and the upper bound (UB) of the range. This will ensure that the size of the tree is always minimized at the lowest level nodes (upper level nodes could be merged in a similar way; this is future work). E.g.: given four rules (4.6.1034.1.1158), (4.6.1034.1.1168), (4.6. 1034.1.1169) and (4.6.1034.1.1170), we can construct the tree shown in figure 2.

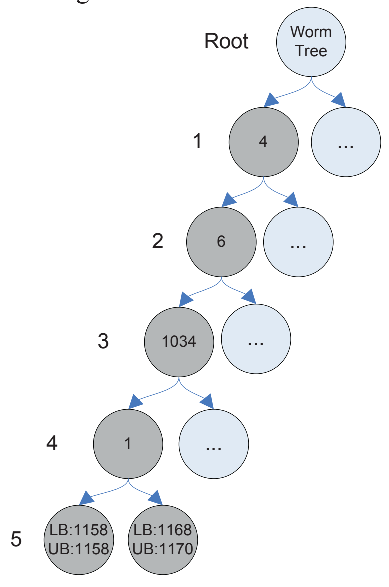

Figure 2. Sample worm tree structure

We identified the least varied field to be the first child node in the sequence, followed by the larger ones. We found that the following sequence of a rule was the best order: (direction, protocol, destination port, session id, source port). We also merged the bottom most nodes in the tree so that any continuous number could be put into range of numbers having a lower bound and upper bound stored in a single node. The impact on minimum or no redundant rule is vital in reducing the search space in the pattern matching phase.

A drawback of this tree-based structure is that two or more rules that differ only at a higher level will be replicated entirely at lower levels. Another drawback of using trees is the cost taken to process the training data to build the trees. However, the overhead is quickly balanced by a speedier detection [10].

D. *Training and Testing Datasets*

The worm and benign trees were constructed using the training datasets, using a *k*-fold (*k*=10) cross validation scheme. The data was divided into ten subsets. Each subset was held out in turn to be a testing set; performance on that testing set was evaluated using worm and benign trees constructed using the other 90% of the data.

When building the trees, we limited the number of training samples of every endpoint and worm used in each *k* to the first block of 1000 or 10000, if the total signatures were greater than 10000.

To form testing sets, for each *k* set, we inserted a sum of about 2040 or 6240 depending on the total size of the benign size from the *k* set and non-overlapping testing worm signatures into endpoint sessions. Each worm signature was inserted into the benign endpoint sessions at a random interleaved size of minimum 3 and maximum 5. Taken from the k set, we set only 30 signatures for the first worm, 100 for the next 5 worms and the remaining worms were set to 1000 signatures each.

E. *The Signature Based Detector*

Searching for matching patterns in a huge tree based structure can be too costly if the search strategy is not meticulously designed. A simple search can be quickly accomplished if a search condition is to find the next part of a signature given only that the current part of the signature matches. Otherwise, if a search condition is to find the next part of a signature regardless of whether the current part of the signature matches or not, the search needs to traverse either all nodes in the tree or enough of the tree until the required detection rate is obtained. Since we wanted the second condition to apply, we introduced a minimum detection threshold of *T*=4.

That will minimize the need to check all nodes in the trees, and in many cases the search may not even traverse half the size of the tree e.g.; when a detection rate is already obtained. If our system is implemented at the host level, the size of the benign tree was not too large, assuming that most of everyone's PC is not shared and everyone gets attached to only a set of websites or other Internet-based services. If this system is implemented at the gateway level, the trees should only maintain small amount of recent data.

It was possible also to have a simple search with earlier condition but we needed to build more trees amounted as factorial to the number of the field in a signature where each tree will have field values uniquely permutated. However, we decided to consider that option for future work.

F. *The Anomaly Detector*

We discovered that some signatures were found as benign and worm at the same time. As the signature-based detector could not make a decision, an anomaly detector that analyzed recent traffic could help to finalize the tie-break.

In data mining [16], the window based technique called sliding window model is a technique that is ideal to analyze stream data. The technique observes patterns in recent data, rather than the entire streams seen so far. A long stream of traffic contains patterns that change over time, so reliance on the entire stream to make a decision is unreliable. Instead the decision is based on recent data.

We investigated the effectiveness of the entropy technique and we discovered that entropy could only suggest how much an entire window of signatures is different from a normal window. An entropy value does not tell us whether one questionable session in a particular window is benign or worm. However, we noticed that the calculations towards getting an entropy value are very useful to give the answer that we are interested in. Therefore, we included that algorithm for our anomaly detector.

Regardless of whether the tree based detector is able to make a decision or not, we always maintained the most recent window of w sessions in a buffer. As a new session appeared, it was stored into the buffer and the oldest session in the buffer was removed. Each time the anomaly detector was invoked, it used the sessions stored in the buffer to draw its conclusion.

We considered the frequency of port numbers in the buffer. If the session in question had port numbers matching those with the highest frequency in the window, and some other conditions were met, the session was considered a worm, otherwise it was considered benign.
We experimented with several conditions:

- Should there be a minimum frequency of matching port numbers, in a fixed window of w sessions, to deem a session as a worm? (One fifth of the window size was used for the minimum.)
- Should there be a maximum frequency of matching port numbers, in a fixed window of w sessions, to deem a session as a worm? (One third of the window size was used for the maximum.)
- Considering source and port destination numbers separately, should they both suggest that a session is a worm to deem it to be a worm, or is one enough?
- Windows of 15, 20, 30 and 40 sessions were tested in this study.

G. *Performance measures*

We evaluated the performance of the classification system based on its rates on:

- True Positive (TP): The fraction of worm traffic correctly classified as worms.
- False Negative (FP): The fraction of benign traffic wrongly classified as worms.

V. Result and Discussion

We evaluated the performance of our system using each combination of parameter values, and performed an analysis of variance to identify which parameters made a difference to TP and/or FP. Whether one or both ports needed to match made a difference in many situations. Requiring both to match lowered both TP and FP. On balance, requiring only one to match seems best, as that way the TP rate goes up to over 90%. The use of minimum and maximum frequencies together made a difference in some endpoints. It either made no difference, or if it did make a difference the best performance comes a window size of 40, a minimum frequency of 8, and a maximum of 13. Thus, the best way to configure the system appears to be with these parameter values just: window size of 40, and only one port is enough to deem a worm. With this configuration, the performance for each endpoint is shown in Table I. The TP rate is over 90% for 8 of 13 endpoints. For the other 5 it ranges from 74% to 82%. The FP rate varies. It is particularly good at endpoint 13. It is unusually high at endpoints 5 and 10. Endpoint 10 has very high TP and also high FP.

Detection based only on Anomaly does badly with TP and only attains 50% on two endpoints. Detection based only on signature does very well for TP, but also produces high FP rates on many endpoints. Hence, the anomaly detector does really have a significant role to bring the FP rate under control while achieving good TP rate. Overall TP rate averages 88%, overall FP rate averages 12%.

TABLE I. PERFORMANCES BASED ON ENDPOINTS

Endpoint	True Positive			False positive		
	Anomaly	**Signature**	**Both**	**Anomaly**	**Signature**	**Both**
1	0.403	0.985	0.813	0.171	0.238	0.135
2	0.072	0.987	0.737	0.090	0.577	0.154
3	0.224	0.933	0.788	0.093	0.228	0.055
4	0.307	0.999	0.817	0.383	0.249	0.110
5	0.211	0.984	0.902	0.055	0.616	0.291
6	0.164	0.931	0.774	0.195	0.403	0.092
7	0.087	0.973	0.919	0.196	0.427	0.120
8	0.121	0.998	0.986	0.293	0.207	0.087
9	0.285	0.964	0.905	0.165	0.282	0.071
10	0.157	1.000	0.998	0.198	0.684	0.268
11	0.519	0.977	0.906	0.061	0.633	0.057
12	0.045	1.000	0.974	0.198	0.345	0.143
13	0.561	0.999	0.944	0.025	0.220	0.013
Mean	**0.243**	**0.979**	**0.882**	**0.163**	**0.393**	**0.123**

The performance for each separate worm is shown in Table II. The FP rate per worm basis is very stable, at about 10% for every worm. The type of worm does not make much difference to FP rates, with any type of detector. TP rate varies enormously. *MyDoom-A, Sim Src Port, CodeRed II* and *Witty* have poor TP rates, especially *MyDoom-A* and *CodeRed II*. Signature based detection has high TP rates for all worms except *CodeRed II*.

TABLE II. PERFORMANCES BASED ON WORM

Worm	True Positive			False positive		
	Anomaly	Signature	Both	Anomaly	Signature	Both
1	0.275	0.998	0.959	0.156	0.364	0.108
2	0.125	0.996	0.898	0.149	0.373	0.106
3	0.207	1.000	0.994	0.159	0.376	0.116
4	0.035	0.914	0.208	0.152	0.372	0.107
5	0.133	0.988	0.821	0.155	0.378	0.107
6	0.232	0.992	0.981	0.155	0.384	0.107
7	0.388	0.999	0.994	0.151	0.391	0.113
8	0.528	1.000	0.572	0.137	0.419	0.118
9	0.398	0.655	0.331	0.135	0.385	0.087
10	0.531	1.000	0.635	0.134	0.372	0.100
11	0.361	0.997	0.984	0.135	0.324	0.086
12	0.217	0.999	0.992	0.143	0.315	0.078
Mean	**0.286**	**0.962**	**0.781**	**0.147**	**0.371**	**0.103**

Figure 3 and 4 shows a comparison of our system with Khayyam *et al.*'s K-L/SVM [7] Maximum Entropy [18], and Rate Limiting [11]. The figures show the results per endpoint ID. However, the endpoint IDs of ours may not accurately match with others because the labels in the dataset were changed since we received the dataset from [15].

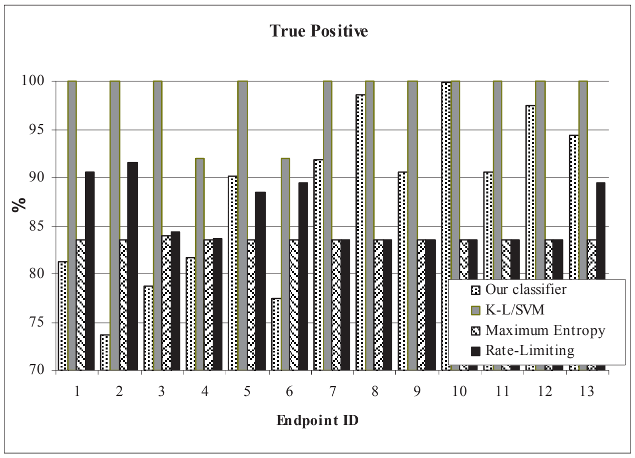

Figure 3. True positive rate

The signature based detector alone produced high TP but also high FP. It was because sessions coexisted in worm and benign datasets. Also, when the training and testing data was split, there was lack of representation of training data in the testing data because the data pattern were not well distributed over a wide range of the dataset due to pseudo-random behavior of users who used the endpoints over time.

Meanwhile, the anomaly detector alone did not really perform well. It caused low TP. This is due to the use of a fixed size window, which prevented the detector from detecting abnormal sessions' intensity over a time window using the same anomaly detection algorithm. When we integrated both detectors, their strengths were complemented by each other.

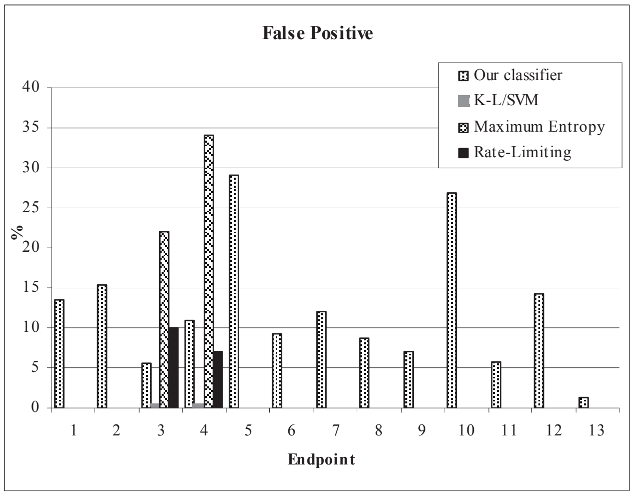

Figure 4. False positive rate

Merging the nodes operations at lowest level nodes in both trees did not reduce many nodes. From 2,427,795 benign signatures used in the entire endpoints and their *k* sets, only 15903 nodes was cut off. That is equivalent to only 0.0065%. With average=122.3 and standard deviation=186.1, most of the cut off nodes were from endpoint 3, 6, 8, and 9 while others almost 0.

From 13,336,713 worm signatures used in the entire *k* sets, only 117 nodes were cut off. That was equivalent to only 0.0000087%. The cut off nodes were well balance across endpoints with average=0.9 and standard deviation=0.3 indicating that those worms generated unique signatures.

VI. Conclusion

In this paper, we present results of an experiment that employed a signature based detector and anomaly detector. We explored the performances of the systems by testing them with several different settings. We discovered that with certain settings, the proposed signature based detector managed to detect worms at a very good rate but at a high FP rate. While an anomaly detector alone is also weak, the integration of both detectors working in tandem formed a stronger detection mechanism.

The main result of this paper is that by combining more than one malware detection scheme, the resulting detection can be greatly strengthened. This suggests a way forward to provide the next generation of malware detection.

Our future work includes investigating the use of a window of a fixed duration, rather than a window containing a fixed number of sessions. We also would like to investigate the performance in term of loading and searching speeds when using the tree and the hash table.

References

[1] P. Szor, The Art of Computer Virus Research and Defense February 13, 2005: Addison-Wesley Professional 744.

[2] R. Ford, Malcode mysteries revealed [computer viruses and worms] IEEE Security & Privacy Magazine, 2005 3(3): p 72-75.

[3] A. Sundaram, An Introduction to Intrusion Detection, in Crossroads : The ACM Student Magazine 1996.

[4] P. Ning, and S. Jajodia, Intrusion Detection Techniques, in H Bidgoli (Ed), The Internet Encyclopedia December 2003, John Wiley & Sons.

[5] R. Sekar, *et al.*, Specification-based anomaly detection: a new approach for detecting network intrusions, in Proceedings of the 9th ACM conference on Computer and communications security 2002, ACM: Washington, DC, USA.

[6] S.A. Khayam, H. Radha, and D. Loguinov Worm Detection at Network Endpoints Using Information-Theoretic Traffic Perturbations in Communications, 2008 ICC '08 IEEE International Conference on 2008.

[7] M.Z. Shafiq, S.A. Khayam, and M. Farooq, Improving Accuracy of Immune-inspired Malware Detectors by using Intelligent Features, in Genetic and Evolutionary Computation Conference (GECCO) 2008, ACM Press: Atlanta, USA.

[8] A. Lakhina, M. Crovella, and C. Diot, Mining anomalies using traffic feature distributions, in Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications 2005, ACM: Philadelphia, Pennsylvania, USA.

[9] T. Abbes, A.A. Bouhoula, and M Rusinowitch Protocol Analysis in Intrusion Detection using Decision Tree in Proc International Conference on Information Technology, Coding, and Computing (ITCC '04) 2004.

[10] M.M. Williamson, Throttling viruses: restricting propagation to defeat malicious mobile code in Computer Security Applications Conference, 2002 Proceedings 18th Annual 2002.

[11] Y. Gu, Y, A. McCallum, and D Towsley, Detecting Anomalies in Network Traffic Using Maximum Entropy Estimation, in ACM/Usenix IMC 2005.

[12] D. Dagon, *et al.*, HoneyStat: Local Worm Detection Using Honeypots, in Recent Advances in Intrusion Detection 2004 p 39-58

[13] EndpointSecurityorg Endpoint Security Homepage 2008 [cited; Available from: http://wwwendpointsecurityorg/Documents/What_is_endpointsecuritypdf.

[14] Endpoint Worm Scan Dataset 2008 [cited; Available from: http://wwwnexginrcorg/indexphp?option=com_content&view=category&layout=blog&id=7&Itemid=24.

[15] L. Rojach, and O. Maimon, Data Mining With Decision Tree Series in machine Perception and Artificial Intelligence 2008: World Scientific Publishing 244.

[16] J. Han, and M. Kamber, Data Mining Concepts and Techniques 2006, San Francisco: Morgan Kaufmann.