Journal of Information Security, 2016, 7, 215-223
Published Online April 2016 in SciRes. http://www.scirp.org/journal/jis
http://dx.doi.org/10.4236/jis.2016.73017

Scientific Research Publishing

A Comparison of Malware Detection Techniques Based on Hidden Markov Model

Saja Alqurashi, Omar Batarfi

Information Technology Department, King Abdul Aziz University, Jeddah, Saudi Arabia
Email: ssmalqurashi@kau.edu.sa, obatarfi@kau.edu.sa

Received 25 January 2016; accepted 19 April 2016; published 22 April 2016

Abstract

Malware is a software which is designed with an intent to damage a network or computer resources. Today, the emergence of malware is on boom letting the researchers develop novel techniques to protect computers and networks. The three major techniques used for malware detection are heuristic, signature-based, and behavior based. Among these, the most prevalent is the heuristic based malware detection. Hidden Markov Model is the most efficient technique for malware detection. In this paper, we present the Hidden Markov Model as a cutting edge malware detection tool and a comprehensive review of different studies that employ HMM as a detection tool.

Keywords

Malware, HMM, Detection Tool, Obfuscation Techniques, Metamorphic

1. Introduction

Malware is a software which is developed for malicious intent [1]. Malware can steal sensitive data from a computer or it can infect files one after the other, and spreads the infection throughout the computer. Malware needs to be caught and removed from the infected computer promptly to avoid the leakage of sensitive data or any other malicious activity in the computer. Malwares can be classified into viruses, worms, Trojans, spywares, adwares, Rootkits, etc. [2].

To escape detection tools, malware developers evolve new techniques [3] [4]. They write malware in a fashion that it changes its appearance and alters its code on each infection—thus changes its appearance on each infection. Therefore, the signature-based detection schemes cannot detect them. The various techniques employed by malware developers are discussed below.

How to cite this paper: Alqurashi, S. and Batarfi, O. (2016) A Comparison of Malware Detection Techniques Based on Hidden Markov Model. *Journal of Information Security*, **7**, 215-223. http://dx.doi.org/10.4236/jis.2016.73017

2. Obfuscation Techniques

Obfuscation techniques that change the syntax of program without change the semantic that make the malicious code more difficult to be analyzed and understand while preserving its semantics and functionality [5]. Obfuscation can be achieved by register exchange where registers and variables within the file are switched but the code remains the same; instruction swap, which substitutes instructions with equivalent instructions; instruction permutation; transposition that reorders the independent instructions; dead code insertion where "do nothing" instructions are inserted in the code [6].

2.1. Code Encryption

Refers to hiding the malicious code. Encrypted malware comprises of decryption/encryption engine, decryption key and encrypted malicious code [2]. Encrypted malwares can be classified.

2.2. Oligomorphic

The malware developers encrypt the code in a manner that it can change its decryptors lightly [2]. However, this malware has some limitations and can be detected by signature-based detection tools [7].

2.3. Polymorphic

The polymorphic malware encrypts itself with a different encryption algorithm/key in every infection. It can use a large number of encryption algorithms/keys and thus makes its detection very difficult and time consuming [3].

2.4. Metamorphic

The metamorphic malware is capable of changing itself to a completely new instance that does not have anything common to its original [2]. This behavior makes it the most complicated malware to detect.

3. Malware Detection Techniques

Malwares are worldwide epidemic and malware detection techniques (MDTs) serve as first line of defense against them. The effectiveness of a malware detection tool is based on the techniques it uses. Malware detection techniques can be primarily divided into the following three classes.

3.1. Signature-Based Techniques

Signature-based detection is widely used to detect known malware. Although it is very effective, but becomes ineffective if there is even a very small change in the code, which in turn changes its signature. Furthermore, this scheme requires the signature database to be updated on regular basis to detect new malware [6].

3.2. Behavior-Based Techniques

These techniques constantly observe program behavior to determine whether it is harmful or not. Thus, these techniques can also detect unknown malware. However, the effectiveness of these techniques is not yet proven. The experiments show that these techniques have high false positive ratio [3]. Moreover, these techniques take more time in detection process.

3.3. Heuristic Techniques

These techniques mainly use machine learning and data mining methods to identify the behavior of the running program. The major methods that have been used so far include Naive Bayesian, Neural Network and Hidden Markov Model [1] [3] [4] [8].

In recent work [1] [3] [4] [9], Hidden Markov Model (HMM) has been shown as an efficient machine learning model to detect malware. HMM is based on statistical analysis of different states of a code, where each state has an associated probability of transition to other states.

HMM is a process modeling technique where each state of the process is defined in terms of its input data and its corresponding probability of occurrence. HMM is initially trained for different states of a malicious code

along with its statistical properties. Based on this learning, HMM can detect, with high probability, the malicious codes of the same family. Thus, the more the training of HMM with different variants, the higher the chances of detection.

The following section gives us more details how HMM work.

4. Hidden Markov Model

4.1. Hidden Markov Model Overview

Hidden Markov models (HMMs) are generally used for statistical pattern analysis. Also can be used in speech recognition [10], malicious code detection [11] and biological sequence analysis [12].

Hidden Markov model is a statistical model that has states and known probabilities of the state transitions is called a Markov model [13]. In such a Markov model, the states are visible to the observer. But a hidden Markov model (HMM) has states that are not directly observable [10]. HMM is a machine learning technique. HMM acts as a state machine. Every state is associated with a probability distribution for observing a set of observation symbols. The transition between the states has fixed probabilities. We train an HMM using the observation sequences to represent a set of data [13]. We can match an observation sequence against a trained HMM to determine the probability of seeing such a sequence. If the probability is high, the observation sequence is similar to the training sequences. HMMs are used in protein modeling, also can be used for software piracy detection [14]. As mentioned in [13], the notations used in the hidden Markov models as following (**Figure 1**):

T = length of the observation sequence
N = number of states in the model
M = number of distinct observation symbols
Q = distinct states of the Markov Model
V = set of possible observations
A = state transition probability matrix
B = observation probability matrix
π = initial state distribution
O = observation sequence

A hidden Markov model is defined by the matrices A, B and π. An HMM is denoted as $\lambda = (A, B, \pi)$.

The following three problems can be solved efficiently using HMM algorithms:

Problem 1: Given a model $\lambda = (A, B, \pi)$ and an observation sequence O, we need to find $P(O|\lambda)$. That is, an observation sequence that can be scored to see how well it fits a given model.

Problem 2: Given a model $\lambda = (A, B, \pi)$ and an observation sequence O, we can determine an optimal state sequence for the Markov model. That is, the most likely hidden state sequence can be uncovered.

Problem 3: Given O, N, M, we can find a model λ that maximizes probability of O. This is the training of a model in order to best fit an observation sequence.

These three problems can be efficiently solved by the following three algorithms:

- The Forward algorithm
- The Backward algorithm
- The Baum-Welch re-estimation algorithm

The forward-backward algorithm is for calculating the probability of being in a state qi at time t given an observation sequence O [13]. The forward algorithm, or α pass, determines $P(O|\lambda)$. The algorithm can be stated as follows.

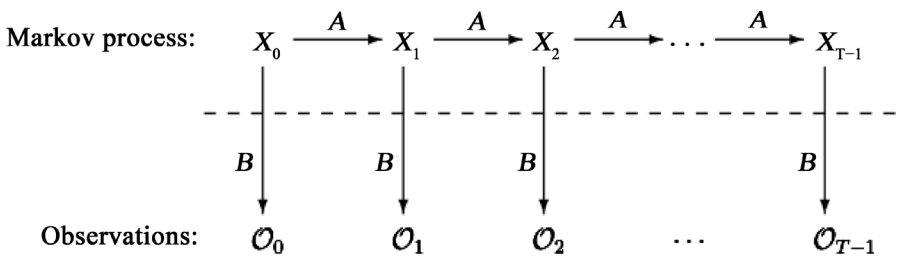

Figure 1. Represents a generic view of a hidden Markov model [13].

1) For $t = 0,1,\cdots,T-1$ and $i = 0,1,\cdots,N-1$
2) $\alpha_t(i) = P(O_0, O_1, \cdots, O_t, x_t = q_i \mid \lambda)$

The probability of the partial observation sequence up to time t is $\alpha_t(i)$. Using the forward algorithm, $P(O \mid \lambda)$ can be computed as follows:

1) Let $\alpha_0(i) = \alpha_i b_i(O_0)$, for $i = 0,1,\cdots,N-1$
2) For $t = 1,2,\cdots,T-1$ and $i = 0,1,\cdots,N-1$ compute

$$\alpha_t(i) = \sum_{j=1}^{n-1} \alpha_{t-1}(j) a_{ij} b_i(O_t)$$

3) Then $P(O \mid \lambda) = \sum_{j=0}^{n-1} \alpha_{t-1}(i)$

The **backward algorithm** helps to find a most likely optimal state sequence. This algorithm can be stated as follows:

1) For $t = 0,1,\cdots,T-1$ and $i = 0,1,\cdots,N-1$ define
2) $\beta(i) = P(O_{t+1}, O_{t+2}, \cdots, O_{T-1}, x_t = q_i, \lambda)$

Then $\beta_t(i)$ can be calculated in following steps:

3) Let $\beta_{T-1}(i) = 1$, for $i = 0,1,\cdots,N-1$
4) For $t = T-2, T-3, \cdots, 0$ and $i = 0,1,\cdots,N-1$, compute

$$\beta_t(i) = \sum_{j=1}^{n-1} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$$

For $t = 0,1,\cdots,T-2$ and $i = 0,1,\cdots,N-1$ define

$$\gamma_t(i) = P(x_t = q_i \mid O, X)$$

The relevant probability up to time t is given by:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O \mid X)}$$

The most likely state at any time t is the state for which $\gamma_t(i)$ is maximum. **The Baum-Welch algorithm** helps in iteratively re-estimating the parameters A, B, π [13].

It provides efficient way to best fit the observations. The number of states N and number of unique observation symbols M are constant. However, other parameters like A, B and π are changeable with row stochastic condition. This process of re-estimating the model is explained as follows:

1) Initialize $\lambda = (A, B, \pi)$ with an appropriate guess or random values. For example $\pi = 1/N$, $A_{ij} = 1/N$, $B_{ij} = 1/M$.
2) Compute $\alpha_t(i)$, $\beta_t(i)$, $\gamma_t(i)$ and $\gamma_t(i,j)$ where $\gamma_t(i,j)$ is a digamma. The digammas can be defined as:

$$\gamma_t(i) = \alpha_t a_{ij} b_j \frac{O_{t+1}\beta_{t+1}(j)}{P(O \mid \lambda)}$$

$\gamma_t(i)$ and $\gamma_t(i,j)$ are related by:

$$\gamma_t(i) = (x+a)^n = \sum_{j=0}^{n-1} \gamma_t(i,j)$$

3) Re-estimate model parameters as: For $i = 0,1,\cdots,N-1$ let

$$\pi_i = \gamma_0(i)$$

For $i = 0,1,\cdots,N-1$ and $j = 0,1,\cdots,N-1$, compute:

$$a_{ij} = \sum_{t=0}^{T-2} \gamma_t(i,j) \Big/ \sum_{t=0}^{T-2} \gamma_t(i)$$

For $j = 0,1,\cdots,N-1$ and $k = 0,1,\cdots,M-1$, compute:

$$b_j(k) = \sum_{\substack{t\in\{0,1,\cdots,T-2\} \\ O_{t=k}}} \gamma_t(j) \Big/ \sum_{t=0}^{T-2} \gamma_t(i)$$

4) If $P(O|\lambda)$ increases go to Step 3.

Considering the potential and effectiveness of HMM-based MDTs, we have studied their strengths and weaknesses. In this paper, we examined 10 HMM-based MDTs and compared them to facilitate their selection for designing and developing secure systems. In addition, we also present a comprehensive literature review of MDT.

4.2. HMM as Malware Detection Tool

4.2.1. Traditional HMM Detector

Wong and Stamp in [3] [4] proposed a detection tool based on HMM to detect metamorphic virus. In their work, the authors compared their tool with commercial anti-virus tools for the morphed viruses generated by four virus engines: Next Generation Virus Creation Kit (NGVCK), Second Generation virus generator (G2), Virus Creation Lab for Win32 (VCL32), and Mass Code Generator (MCGEN). The commercial anti-virus could not detect morphed virus generated by NGVCK, whereas HMM based detector not only classified each virus to a particular family but showed a detection rate of 97% with only 3% false positive detections [3] [4]. **Figure 2** show the HMM score used to classify the family virus, non-family virus and normal file.

The authors in [6] developed a code obfuscation engine that creates metamorphic viruses, which cannot be detected by any signature based technique, and validate detection based on HMM. The authors created 120 different variants of seed viruses. The code morphing engine randomly shuffles the virus assembly code by dividing it into different blocks and also inserts some blocks of dead code in the original code. These metamorphic viruses were scanned using commercial anti-virus (McAfee), which failed to detect any of these viruses. Then, these metamorphic viruses were tested using the proposed HMM detector. For all of the viruses, the score significantly varied from that for normal files. Thus, the HMM-based detector identified viruses from their similarity score only [6].

Another work in [15] developed metamorphic engine to produce highly diverse morphed copies of base viruses. The code obfuscation techniques used to develop engine are dead code insertion, equivalent instruction substitute and transposition. These metamorphic viruses were tested against a commercial anti-virus and HMM detector. The results showed that HMM detector is effective to identify metamorphic viruses with high accuracy [15].

Authors in [8] proposed a code emulator designed to detect dead code in metamorphic virus. This emulator enhanced the effectiveness of HMM detector. The following graph shows us the architecture of the Code Emulator Process Flow. This emulator has capability to emulate all important CPU registers and can filter out changes in the instructions/subroutines caused by code obfuscation. The proposed emulator effectively identified 15%, 25% and 35% morphed viruses [8].

Shabana et al. [14] used HMM to detect software piracy (**Figure 3**). In their work, the authors [14] used a metamorphic generator to create morphed copies of a basic code. The authors extracted opcode sequences from these copies and trained HMM on these sequences. Based on this learning, HMM computed similarity score of

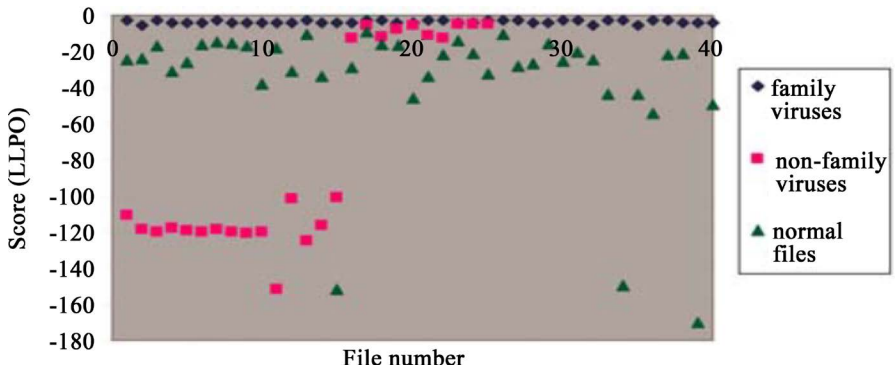

Figure 2. HMM score (LLPO).

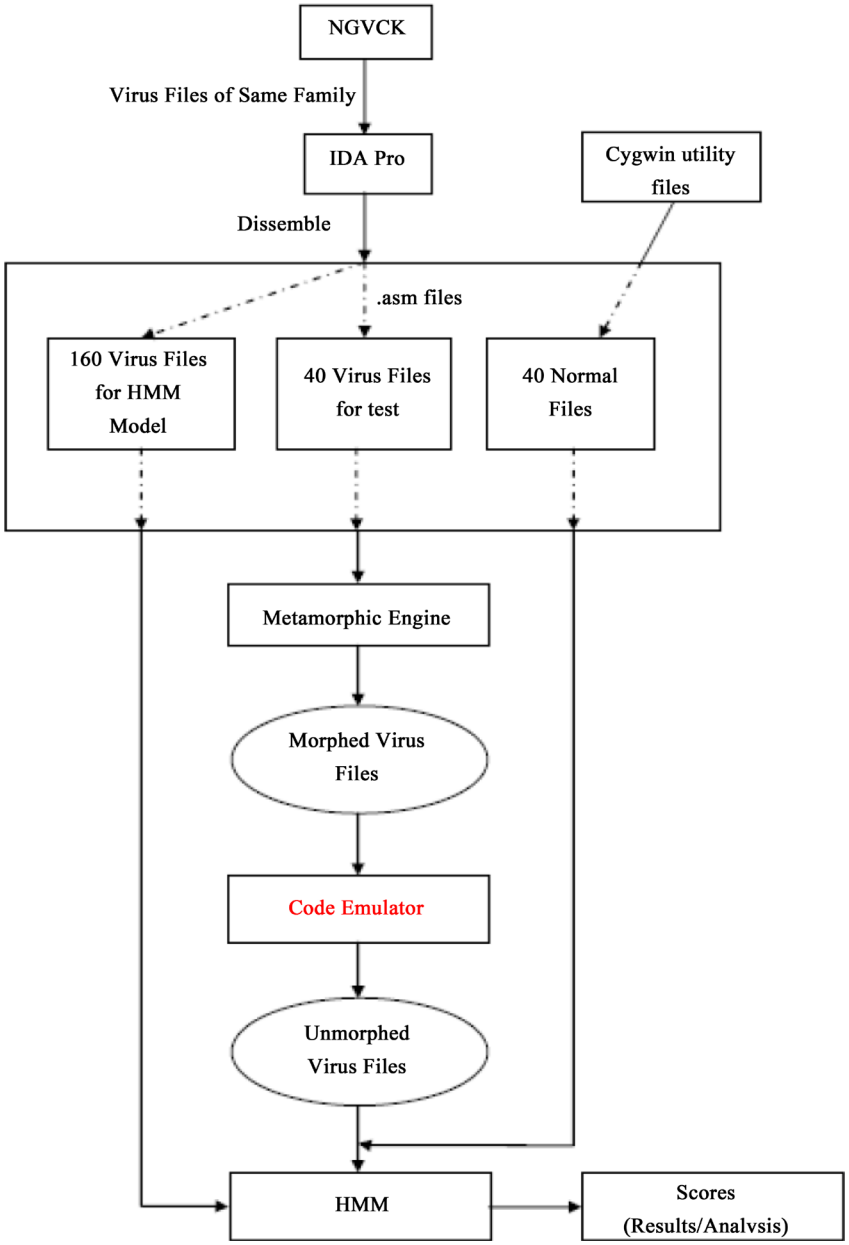

Figure 3. Code emulator process flow [8].

the suspicious program to the basiccode. The higher the similarity score, the more likely it is that the suspicious program is an altered version of the basiccode, and vice versa. This approach proved to be highly effective for detecting morphed viruses.

4.2.2. Dueling HMM Detector

As mentioned in [3] [4], HMM detector is effective for detecting metamorphic viruses. The work in [16] explored HMM in more depth. The authors built models for different compilers, extracted salient features of those models and used them in their proposed model to detect metamorphic viruses. In its classical implementation, HMM has been used to mark a file as infected if its virus code similarity exceeds a given threshold [4] [5]. However, authors' proposed approach marks the suspect file as a virus only if the HMM estimates a high probability by comparing the suspect file with several developed HMMs. This approach proved to be highly effective in detecting the malware that even used advanced metamorphic techniques. On the other hand, this approach is

deficient in balancing false negative and false positive results. In addition, this approach suffers from high performance overheads as compared with the work in [4] and [5]. To overcome this problem of high overheads, the work in [13] brings forth a hybrid approach of dueling HMM and traditional HMM.

Thunga and Neelisetti [17] proposed an intelligent classification approach based on HMM to identify viruses of metamorphic family. In the proposed approach, the authors train multiple HMMs, one for each virus family. They represent "n" opcodes sequences in an executable file as "n-gram". The selection of n-gram affects the overall accuracy of the proposed approach. The n-gram sequence in the metamorphic virus is analyzed by comparing with the trained HMMs one by one using a "log-likelihood similarity measure". A virus is considered to be a part of the family for which the "log-likelihood similarity measure" is the highest. The authors observed 90% accuracy for each family of viruses. The proposed method is easily scalable for a higher number of virus families. In addition, parallel implementation of the comparison phase makes the method very efficient [17] (**Figure 4**).

4.2.3. Hyper HMM Detector

Researchers are still working to enhance the HMM detector. The authors in [18] proposed to use genetic algorithm with HMM. In HMM based techniques Baum-Welch method is used for solving one of the three problems [15], *i.e.* estimating the parameters of the corresponding HMM given an output sequence. In this work the authors used genetic algorithm to solve the problem. The Baum-Welch algorithm is linear in nature and suffers from the local optima problem. The selection of genetic algorithm over traditional Baum-Welch method lies in the non-linearity of genetic algorithm.

There are two reasons to choose genetic algorithm over Baum-Welch algorithm [18]:

- Genetic algorithm is a part of random and evolutionary algorithms and does not suffer from the local optima problem but Baum-Welch algorithm suffers from this problem.

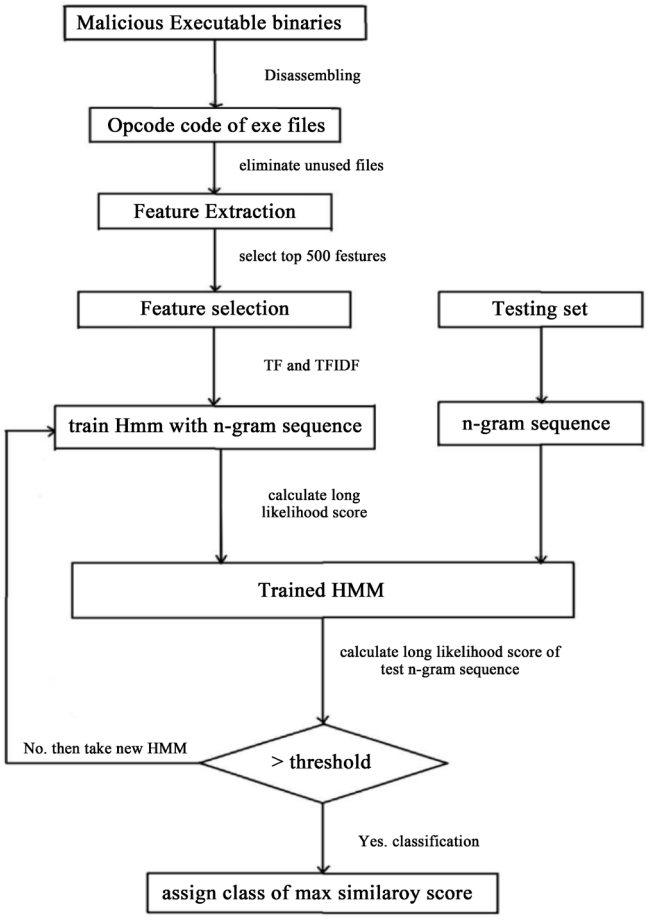

Figure 4. Work flow of the proposed HMM based classifier [17].

- Genetic algorithm works on set of populations (comprising of numbers of chromosomes at time, while the Baum-Welch algorithm works on individuals, so the time required to train the model increases and set of opcodes remains shallow.

This proposed work [18] enhances HMM and results in higher accuracy in detecting metamorphic viruses.

Annachatre et al. [1] proposed new approach using HMM to detect metamorphic viruses based on classifier algorithm. In previous works like [3] [4] and [16], the authors used HMM alone to detect metamorphic viruses. In contrast to these studies, Annachatre *et al*. used a classifier algorithm to enhance the effectiveness of HMM detector. The authors trained HMMs on a collection of malware. Then these models were evaluated and based on the resulting score the malware samples were separated into clusters using k-means clustering algorithm as a classifier. This work showed the highest accuracy as compared to previous works such as [3] [4] and [16].

Payandeh *et al*. proposed a HMM based *detection circle* in [18], where he characterized a virus family using three parameters: 1) the amount of virus similarities; 2) specifically-located character occurrence probability; 3) string occurrence probability. The proposed virus *detection circle* is developed on x, y coordinate axis, where x and y represent probabilities of the factors considered in the study. To evaluate the proposed approach, the authors used NGVCK kit to create morphed malware. Next, they calculated similarity scores of these viruses using Mishra's algorithm after making using Mishra method samples in each family are compared two by tow. The least, the most and the average quantity of samples comparison in each virus creation kit can be seen in the following **Table 1**. The proposed approach detected these malware with 91% accuracy. This work shows good result in malware detection but need to improve the detection speed.

Table 1. Min, Max and average kits similarities percentage.

	NGVCK	G2	VCL	PSMPC
Min	0.00000	0.12858	0.19909	0.00000
MAX	0.23099	1.00000	0.97642	0.96875
Average	0.04835	0.470112	0.650315	0.32857

5. Comparison

Article	Pros strengths	Cons/limitations
Analysis and Detection of Metamorphic Computer Viruses [3] [4]	Used HMM as detection and has ability to identify all malware.	Cannot classify malware to their family (HMM binary classification).
Code Obfuscation and Virus Detection [6]	HMM has ability to identify metamorphic viruses better than commercial AV.	-
Metamorphic Detection via Emulation [8]	Improved HMM detection using emulator to detect dead code in metamorphic malware.	-
Towards an Undetectable Computer Virus [15]	Proves how HMM detector is effective to identify metamorphic viruses with high accuracy.	Still HMM binary classification (classify malware to family and non-family).
Hidden Markov Models for Software Piracy Detection [14]	HMM detected piracy software with high accuracy.	Still HMM binary classification (classify malware to family and non-family).
Dueling Hidden Markov Models for Virus Analysis [16]	Achieved good result in detecting malware developed using advanced metamorphic techniques.	No balance between false positive and false negative. High performance overheads.
Identifying Metamorphic Virus Using n-grams and HMM [17]	Scalable for a number of HMMs (directly proportional to number of virus families).	-
Detecting Metamorphic Virus Using Hidden Markov Model and Genetic Algorithm [7]	Enhanced HMM using genetic algorithm to detect metamorphic viruses.	Still HMM binary classification (classify malware to family and nonfamily).
Hidden Markov Models for Malware Classification [1]	Improved malware detection by HMM using k-means clustering algorithm as a classifier.	Need to enhance the classifier.
Detecting Encrypted Metamorphic Viruses by Hidden Markov Models [18]	Characterized each family of malware in terms of three parameters: 1) string occurrence probability; 2) specifically-located character occurrence probability; 3) the amount of virus similarities. These parameters improved accuracy of malware detection using HMM.	The detection speed of was slower than traditional HMM. Also sample data is small to test the proposed solution.

6. Research Directions

- To detect malware based on clustering algorithms such as K-means, decision Tree etc. based on learning models such as HMM and Neural Network.
- To study how different detection approaches performs against each obfuscation techniques like dead code technique.

7. Conclusion

Malware is any code that is developed for malicious intent. The malware has rapidly become a major security threat for computing community and is the basic reason for the most of the security problems on internet. Recent malware can easily deceive the signature-based detection approaches by altering their code while keeping its function intact. To solve this problem, some machine learning techniques such as, HMM and Neural Network are used. In current study, we have compared several HMM based malware detection approaches. From this survey, we conclude that HMM is an efficient model to detect malware and further studies may focus on how to improve the classification of malware based on HMM as malware detection tool.

References

[1] Annachhatre, C., Austin, T.H. and Stamp, M. (2015) Hidden Markov Models for Malware Classification. *Journal in Computer Virology and Hacking Techniques*, **11**, 59-73. http://dx.doi.org/10.1007/s11416-014-0215-x

[2] Bazrafshan, Z., Hashemi, H., Fard, S.M.H. and Hamzeh, A. (2013) A Survey on Heuristic Malware Detection Techniques. *The* 5*th Conference on Information and Knowledge Technology* (*IKT* 2013), Shiraz, 28-30 May 2013, 113-120. http://dx.doi.org/10.1109/ikt.2013.6620049

[3] Wong, W. (2006) Analysis and Detection of Metamorphic Computer Viruses. MSc, San Jose State University.

[4] Wong, W. and Stamp, M. (2006) Hunting for Metamorphic Engines. *Journal in Computer Virology*, **2**, 211-229. http://dx.doi.org/10.1007/s11416-006-0028-7

[5] Bayer, U., Moser, A., Kruegel, C. and Kirda, E. (2006) Dynamic Analysis of Malicious Code. *Journal in Computer Virology*, **2**, 67-77. http://dx.doi.org/10.1007/s11416-006-0012-2

[6] Venkatesan, A. (2008) Code Obfuscation and Virus Detection. MSc, San Jose State University.

[7] Dastidar, S.G., Mandal, S. and Barbhuiya, F.A. (2012) Detecting Metamorphic Virus Using Hidden Markov Model and Genetic Algorithm. *Proceedings of the International Conference on Soft Computing for Problem Solving* (*SocProS* 2011). India, 20-22 December 2011.

[8] Priyadarshi, S. (2011) Metamorphic Detection via Emulation Metamorphic Detection via Emulation. San Jose State University.

[9] Austin, T.H., Filiol, E., Josse, S. and Stamp, M. (2013) Exploring Hidden Markov Models for Virus Analysis: A Semantic Approach. 46*th Hawaii International Conference on System Sciences*, Wailea, 7-10 January 2013, 5039-5048. http://dx.doi.org/10.1109/hicss.2013.217

[10] Rabiner, L.R. (1989) A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proc. IEEE*, **77**, 257-286. http://dx.doi.org/10.1109/5.18626

[11] Annachhatre, C. (2013) Hidden Markov Models for Malware Classification. San Jose State University.

[12] Krogh, A. (1998) An Introduction to Hidden Markov Models for Biological Sequences. *Computational Methods in Molecular Biology*, **32**, 45-63. http://dx.doi.org/10.1016/s0167-7306(08)60461-5

[13] Stamp, M. (2004) A Revealing Introduction to Hidden Markov Models. Dep. Comput. Sci. San Jose State, 1-20.

[14] Kazi, S. (2012) Hidden Markov Models for Software Piracy Detection. San Jose State University,

[15] Desai, P. (2008) Towards an Undetectable Computer Virus. *Intelligence*, **1**, 402-427.

[16] Kalbhor, A., Austin, T.H., Filiol, E., Josse, S. and Stamp, M. (2014) Dueling Hidden Markov Models for Virus Analysis. *Journal in Computer Virology and Hacking Techniques*, **11**, 103-118.

[17] Thunga, S.P. and Neelisetti, R.K. (2016) Identifying Metamorphic Virus Using N-Grams and Hidden Markov Model. 2015 *International Conference on Advances in Computing, Communications and Informatics* (*ICACCI*), Kochi, 2015, 2016-2022.

[18] Payandeh, A. (2014) Detecting Encrypted Metamorphic Viruses by Hidden Markov Models. 2014 11*th International Conference on Fuzzy Systems and Knowledge Discovery* (*FSKD*), Xiamen, 2014, 973-977.