Denial-of-Service, Probing, User to Root (U2R) & Remote to User (R2L) Attack Detection using Hidden Markov Models

Ali Alharbi
School of Engineering and Computer Science
Oakland University
Rochester, Michigan
aoalharb [AT] oakland.edu

Sulaiman Alhaidari
School of Engineering and Computer Science
Oakland University
Rochester, Michigan

Mohamed Zohdy
School of Engineering and Computer Science
Oakland University
Rochester, Michigan

***Abstract*—at present, an increase in the level of insecurity due to unauthorized attacks and an increase in the crime rate. There is hence the need for secure systems that can ensure high-level security to guarantee safety and trusted communication. To ensure secure data communication over the internet and any other network there is the need to get rid of the threat of intrusions and misuses. Potential attackers pose great threats to the information security of the systems. Users seek to control these threats, recognition of attacks to increase safety. The attacks that affect a large number of computers in the world daily can be classified into four attacks and these include Probing, Denial of Service (DoS), User to Root (U2R), Remote to User (R2L) attacks.**

Researchers have found the study of attack detection and prevention of computers from them to be a major concern throughout the world. Owing to the desire of many researchers, this paper seeks to propose the idea of HMM based approach in the detection of Probing, DoS, U2R and R2L attacks on the system. To accomplish this, the paper will examine the matter of attack detection in computer systems, the significance of the computer security problem in general and attack detection in specific is also examined. Four salient types of attacks will then examined in detail to establish how they work. This will be followed by a brief examination of how hidden Markov models (HMM) is of use in this context and finally an investigation of the strategies behind and various constituent sub-problems of HMM.

Keywords-component; — Hidden Markov Model, Probing, Denial of Service, User to Root, and Remote to User attacks, NSL-KDD

I. INTRODUCTION

Computer systems and networks form the underpinning of much of modern society. Without assured integrity and continuity of operations of our electronic infrastructure, the world's engines of commerce, travel, defense, electric power, and logistics are impaired or altogether crippled. More specifically, malware that makes use of the various attack techniques detailed below is capable of compromising the confidentiality, integrity, and availability of information systems [20]. Confidentiality is compromised if the malware succeeds in stealing information and forwarding it to unauthorized recipients. Integrity is impaired if data can be modified either in place or in transit or, in fact, destroyed. Finally, availability is harmed if the malware succeeds in curtailing legitimate users' endeavors to take advantage of the data resources, communicative capabilities, and/or processing power afforded by these information systems [22]. Thus, it is critical to understand the classes of attacks that are launched against computer systems.

1) Denial of Service (DoS) attacks render system resources unavailable to legitimate users. These attacks are achieved by flooding target hosts or networks with carefully structured traffic in a manner calculated to expend the targets' resources and thereby either suspend or crash their operations. Typical DoS attacks include the Smurf attack and the SYN flood attack. In the Smurf attack, the attacker transmits ICMP broadcast packets with spoofed source IP addresses that refer to the attacked network. The responses generated to these broadcast packets feedback multiplicatively to disable the host. Admittedly, a Smurf attack is relatively easy to defend against by having a firewall solution that recognizes that the inbound packets are Martians, that is, arrivals at unexpected interfaces that look like the results of unexplained routing errors [15]

Assuming that the firewall is constructed a top a dual-homed host, it seems patently clear that protocol data units (PDUs) that purport to originate from nodes within the intranet should not be arriving at the firewall's external interface. It is straightforward for such PDUs to be detected as somehow spurious and attendantly discarded before TCP resources are expended to set up connections with their ostensible transmitters. By way of contrast, in the SYN

flood attack, requests for TCP connections via the three-way handshake are initiated in great numbers but never completed. These operations tie up the target's ports such that it is unable to process any more inbound requests [13].

2) Probe or probe-response, attacks are a new threat for collaborative intrusion detection systems. A probe is an attack which is deliberately crafted so that its target detects and reports it with a recognizable "fingerprint" in the report. The attacker then uses the collaborative infrastructure to learn the detector's location and defensive capabilities from this report. Probing attacks are intended to disclose information about the vulnerabilities of a target system by performing carefully chosen sequences of operations and observing how the system or the intrusion detection system (IDS) that fronts it behaves in response. Some probing attacks strive to map out the IP address space of the target as a prelude to a campaign of spreading malware.

The attacker probes various ports in order to observe what subset of traffic is being monitored at which IP addresses, which enables the attacker to target subsequent pernicious activity toward those less tightly managed hosts. The attacker can also transmit PDU, or packets, into the network and observe via traceroute the details of their delivery. This furnishes the attacker with potentially invaluable information about the structure of the network and what ranges of addresses are reachable from where. Yet another probe attack is more directly targeted at spotting weaknesses in the IDS per se and discovering how they can be exploited to penetrate it as a prelude to attacking the systems that it defends [10]

3) A *User to Root* attack the attacker first succeeds in gaining a foothold on the remote system in the form of a user session, ideally in the form of an interactive shell or TELNET window. By combining a variety of traditional techniques, the attacker endeavors incrementally to escalate his privileges until he achieves super-user permissions [8]. An example of an escalation technique is the use of stack-smashing, which feeds a packet to a set-UID-to-root program that corrupts its address space so that a return-from-subroutine instruction results in the spawning of a set-UID-to-root command shell [1]. It is surprising how many archaic mechanisms remain in both UNIX and Windows that enable attackers rapidly to escalate their privileges. For example, in an archaic implementation of local e-mail, an attacker could link /etc/password to root's personal mailbox. He can then use the command mail root to construct and install a bogus login with super-user permissions [19]. This attack takes advantage of the fact that e-mail was not built according to the least privilege principle [16].

4) A *Remote to Local* attack (sometimes also referred to as a Remote to User attack) is conceptually similar to the user-to-root attack but is more modest in its ultimate ambition. Such an attack is transacted when an attacker sends packets to the target host that are intended to disclose vulnerabilities that would enable the attacker to exploit a local user's privileges. Such vulnerabilities would ideally be found in such utilities as xlock, guest, xnsnoop, phf, and Sendmail [7]. Admittedly, obtainment of a foothold as an authorized user can serve as an important prelude to waging a subsequent user-to-root attack.

The paper is organized as followed, in the second section; the related works that used HMM methods are examined. Third section, involves the examination of the relative utility of competing analytical techniques—the hidden Markov model (HMM) in terms of practical experiments in their application and more detailed exploration of their respective natures. Afterwards, the method used for setting up and executing the experiment will be explained. This is then followed with an evaluation of the results obtained from the experiment. The last section will be a conclusion.

II. Related Works

The work by Jain & Abouzakhar involved the study of HMM and SVM in anomaly intrusion detection (183). The work identified a new method that can identify the distinguishable TCP services utilizing the J48 decision tree algorithm. Each of the trained model was evaluated using Forward and Backward algorithms. [4]

Bhole & Patil analyzed the anomaly detection based IDS utilizing the privilege transition flows (29). Compared to WEKA tool, the HMM was found to be a more accurate generative model and provides better detection rate. The model gives better accuracy because it makes use of anomaly-based detection technique. [11]

Security can be offered using intrusion detection systems. Devarakonda et al. discussed both the HMM and Bayesian Network. The work found the IDS model to possess high order in the detection and classification of the normal and intrusions attacks. [6]

Threats of potential attackers has become a major concern making information security a must for any organization. The study by Khosronejad et al. conducted an analysis of HMM and C5.0. The authors posit that both HMM and C5.0 are more accurate intrusion detection techniques. [5]

III. Hidden Markov Models

HMM suffer from three classic problems. The first of these is the evaluation problem: given a hidden Markov model and an output sequence, how does one calculate the probability that the model actually generated the output sequence? The second is the decoding problem: given a model and an attendant output sequence, what is the most likely model history, that is, the sequence of states and state transitions that was responsible for generating the output sequence? The third is the learning problem: based upon a model and a variety of output sequences, how can the model parameters be deduced from those output sequences so that they correspond to the model with a high degree of fidelity.

1) The evaluation problem consists of both the forward algorithm and the backward algorithm. According to the forward algorithm, the various conditional probabilities are first determined based upon an initialization, induction, and termination procedure. This enables the calculation of the conditional probability of generation of a given output sequence to be completed in TN^2 operations. When the backward algorithm is instead applied, a set of backward variables, or betas, is derived to represent the probabilities of partial observation sequences. This requires the completion of a two-step procedure that entails initialization and induction and that, like the forward algorithm, terminates after TN^2 operations [9].

The preponderant concern for both the forward and backward algorithms is to determine the likelihood of a history of observations, O = {o1, o2, o3 . . .}, for a model λ = {A, B, π}. Every possible sequence of states ends up being scrutinized and the attendant probability, P (O | λ) = ΣQ P (O | Q, λ) P (Q | λ), evaluated. In the forward procedure, probabilities of occurrence of partial observation histories are characterized by a set, { αt(i) }, where αt(i) = P(o1, o2, . . ., ot, qt = Si | λ). These can be computed via an induction procedure that embraces three discrete phases: initialization, in which a1(i) = πi bi(o1); induction per se, in which αt+1(j) = [Σαt(i) αij] bj(ot+1); and termination, in which P(O | λ) = ΣαT(i). The backward procedure instead derives the set { βt(i) }, defined as P(ot+1, ot+2, . . ., oT | qT = Si, λ). The induction procedure for deriving this set entails only two steps: initialization, in which βt (i) is arbitrarily set to unity; and induction per se, in which βt (i) = Σj αij βj (ot +1) βt+1(j) [9]. Forward algorithm

$$P(V^T|\lambda) = \sum_{j=1}^{N} \alpha_j(T) \qquad (1)$$

Backward Algorithm

$$P(V^T|\lambda) = \sum_{j=1}^{N} \beta_j(1)\pi_j b_{jv(1)} \qquad (2)$$

2) The decoding problem intends to find an optimal path through a set of hidden states based upon consideration of a set of observations. The Viterbi technique, which is an example of a dynamic programming algorithm, is commonly employed in finding the longest, or critical path through an assumed directed acyclic graph (DAG). The algorithm derives the path that maximizes the conjoint probability of transit and completes in O (NQ^2) time [3].

Where

$$P(w_i|V^T,\lambda) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N}\alpha_j(t)\beta_j(t)} \qquad (3)$$

The decoding problem can be characterized algorithmically as follows [18]:

create a path probability matrix *viterbi[N+2,T]*
for each state *s* **from** 1 **to** *N* **do** ; initialization step
  $viterbi[s,1] \leftarrow a_{0,s} * b_s(o_1)$
  $backpointer[s,1] \leftarrow 0$
for each time step *t* **from** 2 **to** *T* **do** ; recursion step
  for each state *s* **from** 1 **to** *N* **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s}$
$viterbi[q_F,T] \leftarrow \max_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$; termination step
$backpointer[q_F,T] \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$; termination step
return the backtrace path by following backpointers to states back in time from $backpointer[q_F,T]$

3) The learning problem is an aspect of the Viterbi algorithm. Both supervised and unsupervised learning techniques exist. Pursuant to a trivial initialization phase, linear parameters are derived that maximize the probability of correct classification of the indicated training set. Any classification errors are fed back in order to recalculate the parameters by a mathematical technique such as a gradient descent procedure [17] to achieve a superior subsequent estimate [14]. The initialization, estimation, and update phases that cooperate within the learning algorithm can be characterized in PDL form as follows [18]:

initialize *A* and *B*
iterate until convergence
E-step
$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\alpha_T(q_F)} \quad \forall\, t \text{ and } j$$
$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1})\beta_{t+1}(j)}{\alpha_T(q_F)} \quad \forall\, t,\ i, \text{ and } j$$
M-step
$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\sum_{k=1}^{N}\xi_t(i,k)}$$
$$\hat{b}_j(v_k) = \frac{\sum_{t=1 s.t.\ O_t=v_k}^{T}\gamma_t(j)}{\sum_{t=1}^{T}\gamma_t(j)}$$
return *A*, *B*

It should be pointed out by way of contrast that the traditional machine learning (ML) models are rather more simplistically structured [21]. Under the most common ML paradigm, observations that constitute a training set are represented as vectors in a hyperspace of arbitrary dimensionality termed Kotelnikoff space. Training the machine properly to classify samples typically involves the

derivation of a linear discriminant function that manages to separate Kotelnikoff space into decision regions [17]. If linear classification is impossible, the problem is recast to take advantage of the neural net formalism [17] that is more algorithmically affine to HMM and its constituent procedures.

IV. METHODOLOGY

A. Dataset

Detecting different types of cyber-attacks is the main aim of this study. The NSL-KDD dataset is a revision of the KDD '99 dataset that extracted from the KDD'99 dataset to solve some of the inherent problems [19]. The size of NSL-KDD dataset is smaller than original KDD99. In each record of the set, there are 41 different attributes/Features and one more attribute for class assigned to each record as either an attack type or normal as shown in Table I. The NSL-KDD dataset contains 23 types of attack in the training set and 17 additional attack types in the testing set (New attacks that are not included in the training set) that classified into four major categories, DoS, Probe, R2L and U2R as shown in table II & III in the training set and testing set respectively. Table IV shown the attack classes and the quantity of the attack in each categories.

TABLE I. Total list of features in NSL-KDD dataset

Feature #	Feature Name	Feature #	Feature Name
1	Duration	22	is_guest_login
2	protocol type	23	count
3	service	24	srv_count
4	flag	25	serror_rate
5	src_bytes	26	srv_serror_rate
6	dst_bytes	27	rerror_rate
7	land	28	srv_rerror_rate
8	wrong_fragment	29	same_srv_rate
9	urgent	30	diff_srv_rate
10	hot	31	srv_diff_host_rate
11	num_failed_logins	32	dst_host_count
12	logged_in	33	dst_host_srv_count
13	num_compromised	34	dst_host_same_srv_rate
14	root_shell	35	dst_host_diff_srv_rate
15	su_attempted	36	dst_host_same_src_port_rate
16	num_root	37	dst_host_srv_diff_host_rate
17	num_file_creations	38	dst_host_serror_rate
18	num_shells	39	dst_host_srv_serror_rate
19	num_access_files	40	dst_host_rerror_rate
20	num_outbound_cmds	41	dst_host_srv_rerror_rate
21	is_host_login	42	Attack/Normal

B. Feature Pruning

Feature Pruning is one of the most essential steps in building our model. In order to build a suitable detection algorithm with the ability to detect attacks with a high accuracy, the most effective and significant features need to be extracted or pruned. However, the NSL-KDD data set has 41 Features, from which reducing the feature set to a smaller set of features and selecting the significant features will improve the computation time, obtain higher accurate detection rates and minimize the effect of noise when training occurs. For this study, we wrote a feature-pruning algorithm and the process is described as follows:

Step 1: Loading the dataset values (Full Feature set).
Step 2: Standardizing the data in a normal distribution (a mean of zero and standard deviation of one).
Step 3: Combining the data to one sequence of observation for all the features
Step 4: Calculating the Model parameters
Step 5: Calculating the accuracy by using the Viterbi algorithm.
Step 6: Checking the accuracy, if it was equal or better, go to Step 7.
Step 6: Removing a feature from the set, repeat step 6
Step 7: Done. The final exactly selected features obtained with the highest accuracy rate.

TABLE II. Attack types in NSL-KDD train dataset

Attack	**Attack Type**
DoS	Back, land, Neptune, pod, smurf, teardrop
Probe	Satan, ipsweep, nmap,portsweep
R2L	guess_passwd, ftp_write, imap, phf, multihop, warezmaster, spy
U2R	buffer_overflow, loadmodule, perl, rootkit

C. Initializing HMM Parameters

The next step after obtaining the significant features set and before training an HMM is to initialize the parameters. Technically, the HMM parameters could be initialized random and then determined or estimated over several training iterations by using the Baum-Welch training algorithm (also known as Forward-Backward algorithm). It is necessary to start with a rough guess to determine the parameters of HMM (the transition probability matrix and emission probability matrix). Once they are determined, they can be re-estimated by applying the Baum-Welch algorithm and find the more accurate parameters and obtain the HMM best describes the observed sequence. Then, this trained HMM can be used to the testing set to ensure it is able to detect the proper states. Finger 2 shown the flow chart of the proposed approach.

TABLE III. Attack types in NSL-KDD test dataset

Attack	Attack Type
DoS	Back, land, Neptune, pod, smurf, teardrop, apache2 , mail bomb , process table , udpstorm
Probe	Satan, ipsweep, nmap,portsweep, mscan, saint
R2L	guess_passwd, ftp_write, imap, phf, multihop, warezmaster, spy, httptunnel, named, sendmail, snmpgetattack, xlock, xsnoop
U2R	buffer_overflow, loadmodule, perl, rootkit , ps, snmpguess, sqlattack, worm, xterm

TABLE IV. The quantity of each attack categories in NSL-KDD training and test dataset

	Training data set	Testing data set
Class	***The quantity of attack***	
DoS	45927	7458
Probe	11656	2421
R2L	995	2754
U2R	52	200
Normal	67343	9711
Total	125973	22543

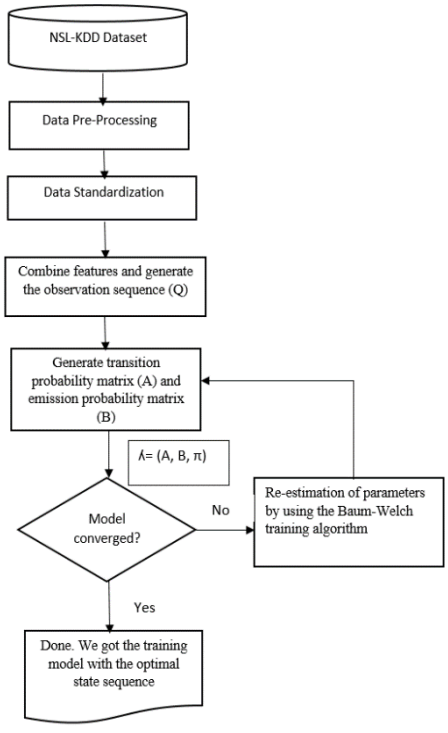

Figure 1. Flow chart of the proposed approach

TABLE V. Confusion matrix (actual vs predicted class)

		Predicted Class	
		Attack	***Normal***
Actual Class	***Attack***	TP	FN
	Normal	FP	TN

V. PERFORMANCE ANALYSIS AND EVALUATION

A. Method of performance testing

To evaluate the performance of our proposed model and how accurate the model was classifying and predicting the class label of attack and normal, we need to know the following four terms:

True Positive (TP): the number of attacks instances correctly identified as attack over all instances

True Negative (TN): the number of normal instances correctly identified as normal over all instances

False Positive (FP): the number of attacks instances incorrectly identified as normal over all instances

False Negative (FN): the number of normal instances incorrectly identified as attacks over all instances.

Confusion matrix for a two class case (Attack and normal) shown in Table V

For this study, the following performance measures is used to test the performance of the proposed model as shown in Table VI.

Accuracy: the ratio of the total number of correct predictions to the total number of all predictions. In our study, accuracy measures by using the Viterbi algorithm to generate a likely state sequence and compare it to the known state sequence to get TP, FP, FN, and TN. The accuracy can be calculated by using the following equation:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \qquad (4)$$

The error rate (misclassification rate)/ False Negative Rate (FNR): the ratio of the total number of misclassified instances to total number of all predictions. The error rate can be calculated by using the following equation:

$$Error\ rate = \frac{FP + FN}{TP + FP + TN + FN} \qquad (5)$$

Fall-out/ False Positive Rate (FPR). Number of negative instances that are incorrectly identified as attack. The Fall-out can be calculated by using the following equation:

$$Error\ rate = \frac{FP}{FP + TN} \quad (6)$$

The sensitivity/ True Positive Rate (TPR): the ratio of the total number of positive instances that are correctly identified as attack to all the positive instances. The Sensitivity can be calculated by using the following equation:

$$\text{sensitivity} = \frac{TP}{P} \quad (7)$$

The specificity/ True Negative Rate (TNR): the ratio of the total number of negative instances that are correctly identified as normal to all the negative instances. The Specificity can be calculated by using the following equation:

$$specificity = \frac{TN}{N} \quad (8)$$

The precision /Positive Predictive Value (PPV) and recall: The precision and recall measures are widely used for performance evaluation of machine learning classification methods . Precision is the ratio of the total number of positive instances that are correctly identified as an attack to the total number of detected instances. Whereas recall is the ratio of the total number of positive instances that are correctly identified as an attack to the total number of all the positive instances (it is the same as sensitivity). The precision and recall can be calculated by using the following equations

$$Precision = \frac{TP}{TP + FP} \quad (9)$$

$$Recall = \frac{TP}{TP + FN} \quad (10)$$

F measure: F measure is a testing score that considers both the precision and recall. It tells how much our model is precise. F measure can be calculated by using the following equation

TABLE VI. Performance evaluation measures of the model

Measure	Formula	Expected Value
Accuracy	$\frac{TP + TN}{P + N}$	Maximum
Error rate	$\frac{FP + FN}{P + N}$	Minimum
Fall-out	$\frac{FP}{N}$	Minimum
Sensitivity/ recall	$\frac{TP}{P}$	Maximum
specificity	$\frac{TN}{N}$	Maximum
precision	$\frac{TP}{TP + FP}$	Maximum
F measure	$2 * \frac{Precision * recall}{Precision + reacall}$	Maximum

$$F\ meaure = 2 * \frac{Precision * recall}{Precision + reacall} \quad (11)$$

TABLE VII. Performance on the NSL-KDD dataset

Measure	Dataset	
	Training set	*Testing set*
Accuracy	88.18%	77.11%
Error rate	11.8%	22.9%
Fall-out	15.4%	34.4%
Sensitivity/recall	93.6%	98.4%
specificity	84.5%	65.5%
precision	80%	60.7%
F measure	86	74

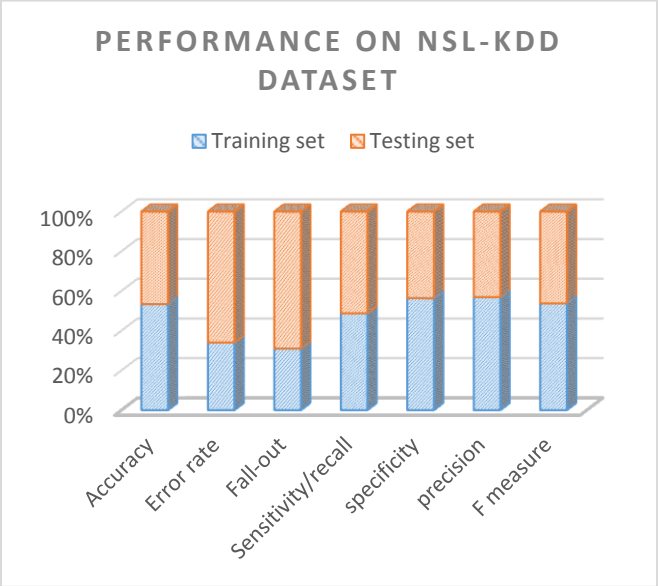

Figure 2. Comparison chart between training and testing sets

B. *Result*

By training an HMM and testing it on the NSL-KDD set, DOS, U2R, R2L, and Probe attacks were detected with greater than 88 percent accuracy on training set and 77 percent on testing set in most trial runs. Table VII summarizes the results of an experiment and Figure. 2 shows in a graphical way a comparison between training and testing sets

VI. CONCLUSION

In recent years, machine-learning methods are gaining the most attention in prediction due to its ability to learn, evolve, improve and adapt. Thus, in this paper, we presented our detection approach detection approach using

Hidden Markov Models (HMM). The result shows that we able to produce a great performance with maximum accuracy, minimum error rate and False Positive Rate in detection denial of service (DOS), Probe, remote to local (R2L), and unauthorized access to root (U2R) attacks. The detection result demonstrated that the proposed detection approach achieved achieves 88 % accuracy on the training set and 77% accuracy on the testing set.

REFERENCES

[1] Aleph One. "Smashing the Stack for Fun and Profit." Phrack 7.49, 1996, www-inst.eecs.berkeley.edu/~cs161/fa08/papers/stack_smashing.pdf.

[2] Farah Nutan, Hridoy Avishek, Muhammad Faisal, Rahman Abdur, Rafni Musharrat and Farid Dewan. "Application of Machine Learning Approaches in Intrusion Detection System: A Survey". International Journal of Advanced Research in Artificial Intelligence, vol 4, no. 3, 2015. The Science and Information Organization, doi:10.14569/ijarai.2015.040302. Accessed 6 July 2018.

[3] Freitas, Ana Teresa. "Hidden Markov Models". 2011. fenix.tecnico.ulisboa.pt/downloadFile/3779577326932/Modelos_prob_12.pdf. Accessed 6 July 2018.

[4] Jain, Ruchi, and Nasser S. Abouzakhar. "A Comparative Study of Hidden Markov Model and Support Vector Machine in Anomaly Intrusion Detection". Journal of Internet Technology and Secured Transaction, vol 2, no. 3/4, 2013, pp. 176-184. Infonomics Society, doi:10.20533/jitst.2046.3723.2013.0023. Accessed 6 July 2018.

[5] Mahsa Khosronejad, Elham Sharififar, Hasan Ahmadi Torshizi and Mehrdad Jalali. "Developing A Hybrid Method Of Hidden Markov Models And C5.0 As a Intrusion Detection System". International Journal of Database Theory and Application, vol 6, no. 5, 2013, pp. 165-174. Science and Engineering Research Support Society, doi:10.14257/ijdta.2013.6.5.15. Accessed 6 July 2018.

[6] Nagaraju, Devarakonda, Pamidi, Srinivasulu, Kumari, V. Valli, Govardhan, A. "Intrusion Detection System Using Bayesian Network and Hidden Markov Model". Procedia Technology, vol 4, 2012, pp. 506-514. Elsevier BV, doi:10.1016/j.protcy.2012.05.081. Accessed 6 July 2018.

[7] Paliwal, Swati, and Ravindra Gupta. "Denial-Of-Service, Probing & Remote To User (R2L) Attack Detection Using Genetic Algorithm". International Journal of Computer Applications, vol 60, no. 19, 2012, pp. 0975 – 8887. pdfs.semanticscholar.org/060d/0c18c3f490720b62e40e7003aa7f75d50941.pdf. Accessed 6 July 2018.

[8] Ricardo, Gutierrez-Osuna. "L23: Hidden Markov Models." research.cs.tamu.edu/prism/lectures/pr/pr_l23.pdf. Accessed 6 July 2018.

[9] Shmatikov, Vitaly, and Ming-Hsiu Wang. "Security against Probe-Response Attacks in Collaborative Intrusion Detection". The University of Texas At Austin, 2007, www.cs.cornell.edu/~shmat/shmat_lsad07.pdf. Accessed 6 July 2018.

[10] T.Bhole, Ashish, and Archana I. Patil. "Intrusion Detection with Hidden Markov Model and WEKA Tool". International Journal of Computer Applications, vol 85, no. 13, 2014, pp. 27-30. Foundation Of Computer Science, doi:10.5120/14902-3394. Accessed 6 July 2018.

[11] Tsai Chih-Fong, Hsu Yu-Feng, Lin Chia-Ying, Lin Wei-Yang. "Intrusion Detection by Machine Learning: A Review". Expert Systems with Applications, vol 36, no. 10, 2009, pp. 11994-12000. Elsevier BV, doi:10.1016/j.eswa.2009.05.029. Accessed 6 July 2018.

[12] US-CERT. "Understanding Denial-Of-Service Attacks | US-CERT". Us-Cert.Gov, 2009, www.us-cert.gov/ncas/tips/ST04-015. Accessed 6 July 2018.

[13] Yemini, Yechiam. "Chapter 4: Hidden Markov Models." www.cs.columbia.edu/4761/notes07/chapter4.3-HMM.pdf. Accessed 6 July 2018.

[14] Comer, Douglas E. Internetworking With TCP/IP: Principles, Protocols, And Architecture:. 5th ed., Pearson Prentice Hall, 2006.

[15] Conrad, J., Gultekin, M., & Kaul, G. "Asymmetric Predictability of Conditional Variances". Review of Financial Studies, vol 4, no. 4, 1991, pp. 597-622. Oxford University Press (OUP), doi:10.1093/rfs/4.4.597.

[16] Duda, R., Hart, P., & Stork, D. Pattern Classification. 2nd ed., John Wiley and Sons Inc., 2001.

[17] Jurafsky, Dan, and James H. Martin. Speech and Language Processing. 3rd ed., Pearson Education International, 2017.

[18] Kochan, Stephen G, and Patrick H. Wood. UNIX Shell Programming. 3rd ed., Sams, 2003.

[19] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," *Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, 2009.

[20] A. &. A. S. &. A. A. &. Z. M. Alshammari, "Security Threats and Challenges in Cloud Computin," in IEEE CSCloud, 2017

[21] P. A. A. N. E. &. A. A. Z. M. Symmonds, "Application of Unsupervised Learning for Detection Cross-site Scripting (XSS) Security Breaches," International Journal of Computer and Information Technology, vol. 6, no. 6, p. 2279 – 0764, 2017.

[22] Alharbi, Ali, et al. "Sybil Attacks and Defenses in Internet of Things and Mobile Social Networks." UEBA 2018 IEEE Intelligence and Security Informatics (2018). (Submitted)