International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249 – 8958 (Online), Volume-9 Issue-4, April 2020

Anomaly Detection in Control Data of Programming Execution by Context Sensitive Hidden Markov Model

Jidiga Goverdhan Reddy, Sammulal Porika

Abstract:: The inconsistency is a major problem in security of information in computer is two ways: data inconsistency and application inconsistency. These two problems are raised due to bad structure of design in programming and create security breaches, vulnerable entries by exploiting application codes. So we can discover these anomalies by design of anomaly detection system (ADS) models at system programming (coding) levels with the help of machine learning. The security vulnerabilities (anomalies) are frequently occurred at potential code execution by exploitation or manipulation of instructions. So, in this paper we have specified various forms of extensions to our work to detect wide range of anomalies at coding exploits and use of a machine learning technique called Context Sensitive-Hidden Markov Model (CS-HMM) will improve the overall performance of ADS by discovering the correlations between control data instances. In this paper we are going to use Linux OS tracing kits to collect the necessary information such as control data instances (return addresses) collected from system as part of artificial learning. The results evaluated through practice on various programs developed for work and also uses of some Linux commands for tracing, finally compared performance of all those input datasets generated live (artificially). After that, the CS-HMM is applying to datasets to scrutinize the anomalies with similarity-search and correlation of function control data of program and classification process determines the anomalous outcomes.

Keywords: Anomaly, Anomaly detection, Hidden Markov Model, Linux Tracing, Return address.

I. INTRODUCTION

The anomalies such as control flow anomalies and data flow anomalies are commonly raised in execution of a contaminated program contains some exploits or manipulated coding lines. Present days we are observing that chance of taking control of program by injecting different malicious definitions and alteration of assembly codes are helping towards attacker's control. Hence we need to update the program definitions and alert on programming leakages. The most common attacks finding on memories by corrupting the memory resisdents with help of knowing address boundaries and chances to exploit the control sequence of software applications running currently. But this kind of problems are always to be lead for creation of potential security breaches if not solve as early possible in the unsafe memory based languages used in computer environment. Such kind of error pro occurances are creating overhead in the program development and also large mainainance is necessary for industries producing large no. of cyber security applications. The creation of anomalies by executing a program in legitimate or non-legitimate, those are executing in legitimate also doubtfull due to stealing of login credentials, so this kind of entries perform some vulnerable actions smashes the data and control parts. The attacker's vulnerable actions are very difficult to model in traditional methods of anomaly detection systems (ADS). So that we need to merge the concepts machine leaning to the system coding exploits to model the anomalies. The machine learning is very usefull to apply on data extracted at system coding and execution context. The adorned features of machine learning-HMM [4,5] are used in our work due to much innovative presentations and outcomes derivations, but not possible through system level exclusively. The extraction of contrlo data is called return addresses (RA) in this paper, these are helps us to pointing out single unit anomalies called as point anomaly as collected. The source of extraction of control data is system's stack; it is very stand alone place to gather abnormal points through observations.

The memory segmentation units also corrupted by different vulnerable programs due to infected code, shell code bundle, other overflow attacks. The HMM based platform is additive work to our existing work due to improvement of performance [2]. As earlier mentioned that the call-stack is raw material source for our concept, it is store the all control pointers, running data instances of process running currently. So that the sequence of control pointers and list of occured data observations to be stored into some predefined structures by using some advanced Linux tracing tricks on memory segments. For quick data collection in Linux environment, the backtrace (bt) is very easiest method and also other tricks such as PTRACE, STRACE, LTRACE and DTRACE methods are helpful to collect the data related instances and control related instances.

A. Anomaly Detection

Anomaly detection is a generally an uninterrupted practice of identify the intrusive behavior of a system users or audit data or application data instances are generated during normal runs. Then the deviation element is consider as anomaly due to its unauthorized behavior showing with known and correct.

Revised Manuscript Received on April 25, 2020.
* Correspondence Author
Jidiga Goverdhan Reddy*, Lecturer, Dept. of Technical Education ,Government of Telangana State and Research Scholar, JNTU University Hyderabad, India, jgreddymtech@gmail.com
Sammulal Porika, Professor, Dept. computer engg, JNTUH Nachupally, Jagitial, Affiliated to JNTU University, Hyderabad, TS State, India, sammulalporika@gmail.com

The Anomaly (Intrusion) detection [14] is main classified approach of intrusion detection systems possible to detect observations of bad behavior going to compromise the security of system compare to normal behavior shown correctly in previous attempts.

The anomaly detection is incorporating the prior knowledge of intrusion occurrences can easy to handle the different variations of attacks compared to signature based ADS only model the attacks pre-trained and static in the nature. Hence to tracing of novel attacks is excellent than known, the dynamic nature of ADS is not compromise to handle the unpredictable behavior showing continuously to elevate the zero day attacks. The signature (misuse) based IDS showing poor results towards dynamic nature of applications, so that it may show high false positives. Hence in this paper the ADS are selected for identifying the anomalies in control data instances while running of infected programs. In this kind of work, exclusive occurrences of control data anomalies or sequence of control anomalies is depending on the program and coding exploits. The correct hypothesis of program is showing correct sequence of control flow is normal, otherwise treated as also anomaly. There are so many traditional systems are developed previously such as expert systems, programs state modeling and Petri nets , system call based , simple rule based and traditional string matching with use of signature based IDS.

The ADS with mixing of system level control concepts with application level will be more accurate and maximum outcomes expected by well exploration of both machine learning and system data [10, 11]. We have presented novel ADS with machine learning HMM with good experimental analysis giving adorned discovering of point and continuous anomalies with different combinations. Now the ADS are only help us to work safe to showing anomalies by using machine learning approaches such as supervised, un-sup'd, semi-based models such a popular decision tree based, NN based, SVM, HMM, and other classification and clustering based can show well results on system's raw control data.

II. RELATED WORK: HMM IN ADS

In our work, we use a Hidden Markov Model (HMM) to model the extracted data from Linux tracing tools and perform the anomaly detection on the control data table to shows the anomalies or deviated behavior. The HMM is very good machine learning based on two-way stochastic development with a finite no. of hidden states. There is a lot work done in anomaly detection by HMM in earlier decades. All the models are developed based on either control flow or data flow instances. The detailed work of system calls based HMM given in [10, 12, 16]. The HMM is applied on sequence of system calls given well showing of results comparison in [13]. The majority of work done through HMM on system call is to recognize the no .of system calls, length, structural inference, system call token or id search. The window boundary is included for system calls in [12] given robust procedure to model the data flow sequence tracing of system calls. Also develop a HMM with pre-processing approach to remove unnecessary raw data of system call patterns to improve the overall performance [14]. Predefined state positions of system calls used with multiple HMMs and before taking a model of HMM deciding the anomalous pattern by training multiple HMMs [3].

The HMM-based ADS [11] developed for knowing the anomalies by two level invisible states for relative probability for sequence form of system calls of sendmail dataset is tune working . Today we are phasing real time frauds in card transactions, similarly some models were developed by HMM to deviate the fraud transactions. The majority people working on internet attacks, so the HMM is very powerful in this case to categorizes the internet attacks which showing complex behavior [6]. The hamming distance separator is used in discrimination of system calls that showing abnormality to real one in some cases, but lack in performance compare to previous work.

The system calls and function calls combination is giving good outcomes as result mentioned for anomalies on separation of forensic data [17]. The problem of hidden state to overcome by developing multiple HMMs among the combined results to be elevated for system calls dataset [15]. Many of HMM machine learning is applied in different datasets belongs to real-time applications such as image, DNA, RNA, pattern classification, medical, video surveillance special and many [7, 8, 9]. But in this work, we have utilized the strength of HMM context model process for showing the result of abnormal control data points in the normal execution of program executed either in legitimate or non-legitimate mode.

The original CS-HMM developed with an idea of working for a modeling of occurrences of sequence symbols in the data with multiple types of emissions in the state model. He stated to model long range with proposing a dynamic programming concepts in state sequence [1].

III. WORKING MODEL

In this section, we are giving the proposed blue print of work by taking the mathematical modeling concept called "Hidden Markov Model". The HMM are many types formulated previously in anomaly detection for different data related to scientific and computer applications. In the computer and network field, the anomaly detection is performed on system scope is very narrative about system calls, library calls…etc belongs to execution process of environment. The present concept showing in this based on HMM is applied at system's control data to knowing the unknown data instances occurred. The proposed HMM is elaborated by giving basic concepts of HMM below.

A. Proposed CS-HMM

The proposed HMM is based on memory state, where the memory plays a role of processing emission sate from the state model. First the working model of our task is shown in Fig.1.

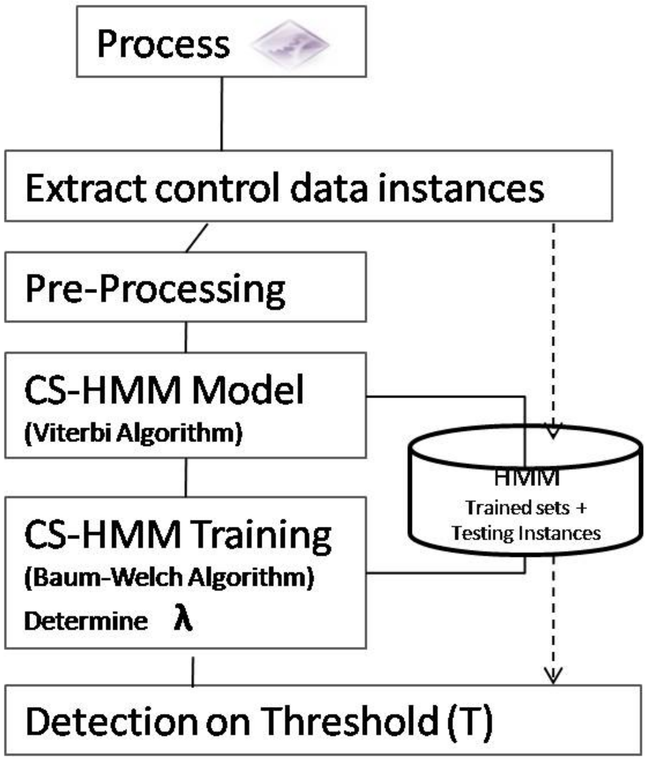


Fig. 1.Working model of our proposed system.

From the Fig.1, the executed process of Linux program is designed exclusively for developing the HMM model. The process is a general program initially; later experimental work is carried out on Linux built-in processes. The control data instances are extracted from running process by taking some pre-defined tracing tools developed on Linux platform such PTRACE, Backtrace, STRACE. But in our work, we used these three types due to its individual extraction process to get more accurate control data. The control data instances are basically controlling the flow of program. In this individual and sequence of control data instances are part of work and these are inputs for the HMM model.

The control data instances are represented by C.

C={FC-RA, FCP}, where FC-RA is set of return addresses are extracted for each function call made during process running and FCP is a set of fabricated paths created between two function calls. FC-RA = {$R_{I,,J}$} is set of return addresses {R_1, R_2 ,R_3 ……R_n belongs to process J. FCP = {{$P_{I,,J}$}$_N$} is set of function call paths {P_{12}, P_{13}, ….P_{21}, P_{22}, ….P_{31}, P_{32}, …….. P_{ij}} belongs to process N. where the $P_{I,,J}$ is path established during the calling in particular sequence of from function I to function J . The $P_{I,J}$ is an arbitrary length of sequence may not fixed and it is very for each path made in the calling sequence where each $P_{I,,J}$ is the set contains set of intermediate return addresses occur between R_I and R_J of $P_{I,,J}$ then $P_{I,,J}$ = { R_I , R_{i+1}, R_2 ,….R_3 …… R_7 ,…..R_{j-1} , R_J }.

The pre-processing is done on set of C by removing unnecessary control instances occurred and repetition takes place while normal runs of program. The C control data instances are collected through various tracing tools, so that consolidation of data is necessary to avoid false positives.

Let we take HMM model for our work is designed for CS-HMM. The basic HMM (**λ**) is denoted as follows.

$$\text{CS-HMM}(\lambda) = (Q, V, A, B, \pi) \qquad (1)$$

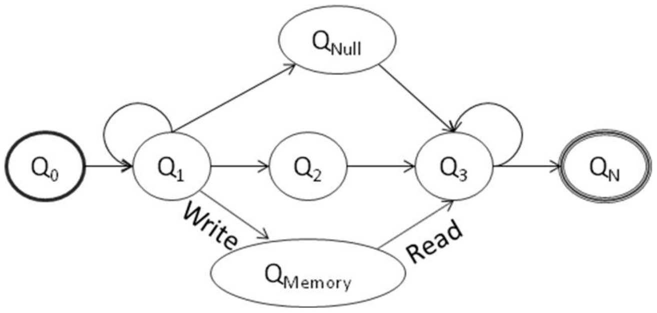


Fig. 2.HMM Model for context-sensitive state (Q3) with memory concept.

Where, Q is a group of states generally called hidden states by default in all types of HMMs. The hidden states are classified into single, double (pair), context (memory)-sensitive emission states shown in Fig.2.

$$Q = \{ Q_0 , Q_1 , Q_2 , Q_3 , \ldots\ldots Q_N , Q_{memory} \} \qquad (2)$$

Where, Q_0 and Q_N are basic starts showing start and end process in the model. Q_1 is double-(pair) emission state connected to observation state includes memory V , Q_1 is null state to continue the model sequence, Q_2 is a single emission connected to observation sequence V, Q_3 is a context sensitive (memory based) state whose emission symbols are determined by context of special state is memory state Q_{memory} . The both pair and context states are use the memory portions of computer to store the emission sequence by write and read instructions. Q_{memory} is state for storing the emission sequences processing during between Q_1 , Q_3. The Q_{memory} is a basically a queue selected for movement of symbols and where as stack is used in [1].

The V is group of M unique observation symbols { V_0 , V_1 , V_2 , ……V_M } to form the observation sequence emitted from basic hidden sates. The emission process from hidden states can determine the observation sequence.

A is the Probability distribution at time t, for hidden state transition is notated by A_{ij}

For example, $P(Q_{i+1} = x \mid Q_i = y) = t(x, y)$ (3)

$A = |a_{ij}|$ and $\Sigma \mid a_{ij}| = 1$, for all j=1….N (4)

For CS-HMM, the probability is depending on the context of symbols stored in memory or not. Then

$P(Q_{i+1} = x \mid Q_i = y, Q_{memory}) = t(x, y)$ (5)

B is a probability distribution at time t, for state j for observation symbols x_i as:

$B = |b_j(k)|$ and $\Sigma \mid b_j(k)| = 1$, for all k=1….M (6)

The emission probability for CS-HMM is as follows: for context (memory) state, the emission probability is depending on memory status as follows

$P(V_i = V \mid Q_i = Q, Q_{memory}) = e(V/Q, Q_{memory})$ (7)

Now, the initial probability for states is given by $\pi = \pi_i$ at time interval t_0, the HMM basic learning and detection is determined by selecting the suitable algorithm.

The criteria for evaluation is decided by either forward/reverse recursive method to generate the sequence of observations from model and Viterbi algorithm is used to decoding the sequence of observations to determine the anomalous behavior shown in the function call list and its sequence, finally Baulm-Welch is used maximize the probability of generating observation sequence by compute the P(O/λ) efficiently for the model λ to decide the occurrences of normal or anomaly by setting up fixed threshold φ (T in Fig.1).

Lets now to realize the problem of building λ for function calls comparison and its sequence comparison by distributing the probability to state has exactly emit the suitable sequence to be part of modeling the control execution of program.

IV. EXPERIMENTAL SETUP

A. Datasets

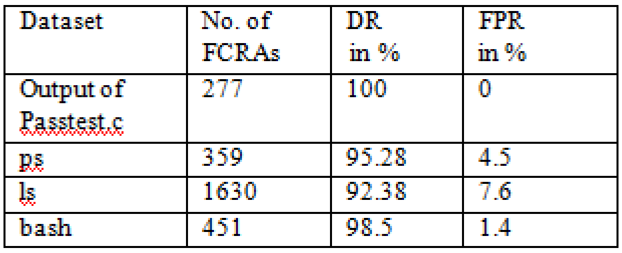
The Datasets are basically live data collected from running process of an artificial created program in C named as passtest.c and Three Linux built-in command process. We have two sets of datasets FCRA and FCP collected two times during normal runs of program and same to be executed while in attacking mode. The comparison of normal and attacking mode datasets are inputs to HMM (λ).

B. HMM λ Computation and Defection Criteria

The generation of λ for HMM in this context of Detection phase is as:

1. λ to be Computing: start with Q_0 and continue the initial probability with π and generate the observed symbols by transitioning from Q_0 to Q_N with the probability to execution of HMM π.
2. Calculate the emission probability for pair of FCRA.
3. The P_λ generate the pair{R_i, R_j}
4. Repeat step 2, 3 for generate the FCS and determine the probability P_λ to generate pair {FCS_i, FCS_j}.
5. To denote the probability of pair emitting with max π for λ is $P_\lambda(R_i, R_j) = \Sigma P_\lambda(\pi)$, $\pi \varepsilon \lambda (R_i, R_j)$
6. Maximizes the $P_\lambda(\pi)$.
7. Compare with predefined Threshold φ
8. If $P_\lambda(\pi) > \varphi$, normal otherwise detecting anomalous observation.
9. For FCS, we have also a new criteria to decide the anomalous or abnormal showing as follows:
10. Formulate the FCSs and attach the trained model to FCS,
11. Find the μ (mean) and σ (S.D) based on the probability of generated FCS over HMM λ with total no. of FCS.
12. Define the test sets to compare with training set with calculation of evaluation parameter distance ($Dist_i$) to the HMM model λ centroid.
13. Compare with predefined Threshold φ
14. If $P_\lambda(\pi)$ [FCS] > φ, normal otherwise detecting anomalous observation.

Table- I: Experimental output analysis of datasets in Linux



Dataset	No. of FCRAs	DR in %	FPR in %
Output of Passtest.c	277	100	0
ps	359	95.28	4.5
ls	1630	92.38	7.6
bash	451	98.5	1.4

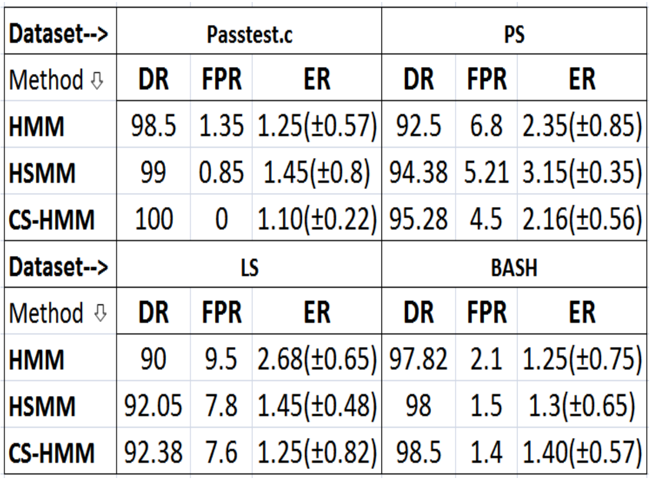
C. Results and Discussions

The experimental work is carried out on three datasets which provides the basic return addresses of function calls made in the program. One program is basic C program written with N number of functions created dummy and made a call entry in the stack to extract return addresses into dataset-1. The others are basic Linux built-in commands programs ls, ps and bash. The table-I showing full details of no. of return addresses along with performance measurements DR and FPR through execution of training phase in legitimate mode and detection phase in non-legitimate mode (attacking mode).

The above work can carry out by CS-HMM code written for simulation of procedure adapted. The function codes written separately for each state shown in fig.2 HMM. In training of HMM, Q_1 is the state emitting the pair of function calls and compute the probability difference between $FCRA_i$ and $FCRA_j$ along with observation symbol {0} by default showing as normal instance initially with 0.5 emission probability. The same pair of function call's return addresses is moving to state called Q_{memory} (Queue) and stored in FIFO order for further processing. The Q_{memory} (Queue) state fixed with N length of call values stored.Q_2 is the state used for emission of single symbols whenever required for comparison at context sensitive state Q_3.

The CS-HMM model development for our task is basically contains two components, one is dealing with FCRA and other one is FCS. In this paper, we have given maximum outcomes of FCRA given. The Table. II is showing the results of all datasets after evaluation by using anomaly detection confusion matrix parameters.

Table- II: the Performance parameters of Experimental work on datasets by various HMMs (per 100%)



Dataset-->	Passtest.c			PS		
Method ⇩	DR	FPR	ER	DR	FPR	ER
HMM	98.5	1.35	1.25(±0.57)	92.5	6.8	2.35(±0.85)
HSMM	99	0.85	1.45(±0.8)	94.38	5.21	3.15(±0.35)
CS-HMM	100	0	1.10(±0.22)	95.28	4.5	2.16(±0.56)
Dataset-->	**LS**			**BASH**		
Method ⇩	DR	FPR	ER	DR	FPR	ER
HMM	90	9.5	2.68(±0.65)	97.82	2.1	1.25(±0.75)
HSMM	92.05	7.8	1.45(±0.48)	98	1.5	1.3(±0.65)
CS-HMM	92.38	7.6	1.25(±0.82)	98.5	1.4	1.40(±0.57)

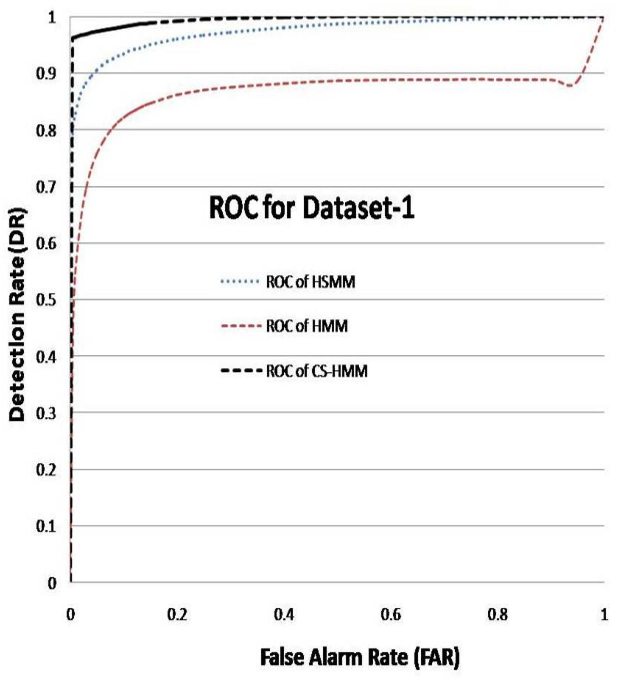


Fig. 3.ROC Curves for Dataset-1 (FCRA) , an output of passtest.c created for simulation.

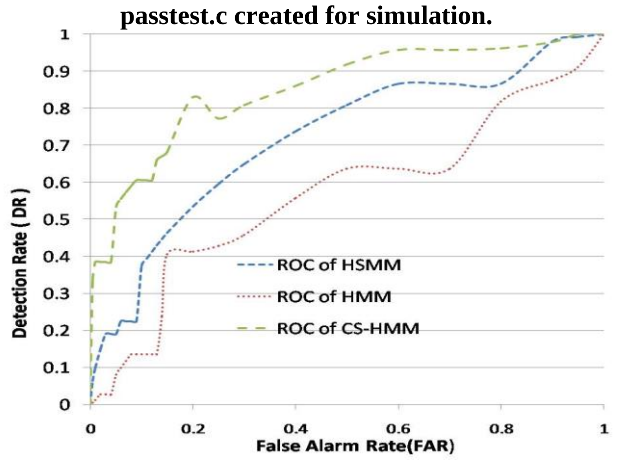


Fig. 4.ROC Curves for Dataset-2 (PS) command in Linux.

The Table-II results are outcome of FCRA only, but not for FCS. The CS-HMM is giving maximum performance then traditional HMMs in all aspects. The ROC curves given in the Fig.3 and Fig.4 respectively for dataset-1 generated from artificial created program to simulate the FCRA and its sequence model through CS-HMM. From Fig.3, if we observe that maximum DR and low FPR are notified for CS-HMM than normal HMM and Semi-HMM. Almost 100% DR is possible in my observations due to small no. of RAs and it is a clear dataset.

The Fig.4 showing the ROC results of dataset-2 (PS), the FCRAs are generated by using STRACE tracing tool. This set is bigger than first one, but showing good AUC. For this FPR is high for when using HMM. The CS-HMM results for remaining datasets of FCRAs and FCSs for all datasets also analyzed and got similar kind of results shown in Table.II and ROC for those not shown in this paper.

V. CONCLUSION

The CS-HMM is good for control data instances as proved in our paper. The results are depending on generation of no. of FCRA and its sequence of appearance to user and input for HMM model is variant in little bit. The pair of function calls return addresses are basically compared between training and detection phase in our concepts. There are thousands of calls made in Linux processes belongs to some commands, but for our experimental work, we took some clear data which is generated by different tracing tools such as STRACE, PTRACE, BACKTRACE. Finally prepared small sets of Linux datasets only consider in the work.

REFERENCES

1. Yoon B.J, P P Vydyanathan " Context-sensitive HMMs for Long Range Dependencies in Symbol Sequences", IEEE Transactions on Signal processing, vol.54, no.11, pp. 4169-4184, 2006
2. Cho B.S. and Park, 'Efficient anomaly detection by modeling privilege flows using HMM, Computers and Security, 22(1), pp. 45-55, 2003.
3. Hoang, X D., Hu, 1. and Bertok, P., 'A multilayer model for anomaly intrusion detection using program sequences of system calls', The 11th IEEE International Conference on Networks, pp. 531-536 (2003).
4. Lane, T. HMMs for Human/Computer Interface Modeling, In Computer Journal of Internet Technology and Secured Transactions (JITST), Volume 2, Issues 1/2/3/4, 2013
5. Rabiner, L.R. (1989) 'A tutorial on HMM and selected applications in speech recognition', in proceedings of the IEEE, 77(2), pp. 257 -286.
6. Ourston, D., Matzner, S., Stump, W. and Hopkins, B. (2002) 'Application of HMMs to Detecting Multi-stage Network Attacks', 36th Hawaii ICSS Proceedings, 9, pp. 334-344.
7. S. Cho and S. Han. 'Two sophisticated techniques to improve HMM-based intrusion detection systems'. 6th International Symposium on Recent Advances in Intrusion Detection -RAID 2003.
8. I. M. Meyer and R. Durbin. 'Comparative ab initio prediction of gene structures using pair HMMs'. Oxford University Press, 2002.
9. L. Pachter, M. Alexandersson, and S. Cawley. 'Applications of generalized pair HMMs to alignment and gene finding problems'. Computational Biology, 9(2), 2002.
10. C. Warrender, S. Forrest, and B. Pearlmutter. 'Detecting intrusions using system calls: alternative data models'. In Proceedings of the IEEE Symposium on Security and Privacy, 1999.
11. Du, Y., Wang, H., and Pang, Y. A 'HMMs-based anomaly intrusion detection method'. In Intelligent Control and Automation, 5th World Congress vol. 5, IEEE, pp. 4348–4351, 2004.
12. Eskin, E., Lee, W., and Stolfo, S. J. 'Modeling system calls for intrusion detection with dynamic window sizes'. In DARPA Conference- II, 2001. DISCEX'01. vol. 1, IEEE, pp. 165–175.
13. Forrest, S., Hofmeyr, S. A., Somayaji, A., and Longstaff, T. A. 'A sense of self for Unix processes'. In Security and Privacy, 1996. IEEE Symposium Proceedings, pp. 120–128., 1996.
14. Hu, J., Yu, X., Qiu, D., and Chen, H.-H. 'A simple and efficient hidden markov model scheme for host-based anomaly intrusion detection'. Network, IEEE 23, 1 (2009), 42–47.
15. Khreich, W., Granger, E., Sabourin, R., and Miri, A. 'Combining HMMs for improved anomaly detection'. ICC'09. IEEE International Conference on (2009), IEEE, pp. 1–6.
16. Stephanie Forrest, Steven Hofmeyr, and Anil Somayaji. 2008. 'The evolution of system-call monitoring'. In Annual Computer Security Applications Conference, 2008 (ACSAC'08). IEEE, 418–430.
17. Sean Peisert, M. Keith. Analysis of computer intrusions using sequences of function calls. IEEE Trans. Dependable Secure Computing. 4(2):137–150, 2007.

AUTHORS PROFILE

Jidiga Goverdhan Reddy, Presently working as a Senior Lecturer in Department of Technical Education, Hyderabad, Government of Telangana State. Also a Research Scholar at JNT University and presently pursuing Ph.D as External at JNTU, Hyderabad, India. I did my B.Tech and M.Tech in Computer Science and Engineering from JNTU University, Hyderabad.

Sammulal Porika, Professor, Presently working at CSE Dept, JNTUH College of Engineering,, Karimnagar affiliated to JNTU Hyderabad, Telangana State India. He did his Ph.D from OU Hyderabad, Telangana State, sammulalporika@gmail.com