2013 IEEE 37th Annual Computer Software and Applications Conference Workshops

A Multi-Order Markov Chain Based Scheme for Anomaly Detection

Wenyao Sha*, Yongxin Zhu*, Tian Huang*, Meikang Qiu†, Yan Zhu*, Qiannan Zhang*
*Shanghai Jiao Tong University
School of Microelectronics
shawenyao@gmail.com, {zhuyongxin, ian_malcolm}@sjtu.edu.cn,
newzhuyan@gmail.com, michellezhang@sjtu.edu.cn
†University of Kentucky
Dept. of Electrical and Computer Engineering
mqiu@engr.uky.edu,

***Abstract*—This paper presents a feasible multi-order Markov chain based scheme for anomaly detection in server systems. In our approach, both the high-order Markov chain and multivariate time series are taken into account, along with the detailed design of training and testing algorithms. To evaluate its effectiveness, the *Defense Advanced Research Projects Agency* (DARPA) Intrusion Detection Evaluation Data Set is used as stimuli to our model, by which system calls and the corresponding return values form a two-dimensional input set. The calculation result shows that this approach is able to produce several effective indicators of anomalies. In addition to the absolute values given by an individual single-order model, we also notice a novelty unprecedented before, i.e., the changes in ranking positions of outputs from different-order ones also correlate closely with abnormal behaviours. Moreover, the analysis and application proves our approach's efficiency in consuming reasonable cost of time and storage.**

***Keywords*—Markov chain, Kth-order Markov chain, multivariate time series, anomaly detection.**

I. Introduction

A. Motivation

It has been recognized recently that we are living in the era of cloud computing. Despite lack of warranted trust, commercial, industrial and financial users tend to simply follow the cloud computing vogue by hosting their entire applications and storage on servers. Thus, taking care of both business and personal data, servers expose critical safety and availability issues, the invulnerability of which are of major importance to both individuals and the society. However, during catastrophic disasters such as intrusion, crash or breakdown, the anomalies must be first discovered before any actual remedy could come to its aid. Being recessive at the early stage, such problems would not exhibit distinct traits and often lead to delayed responses and irrecoverable results.

Fortunately, a server is ideal instance whose behavior manifests regularity statistically. This lays the foundation for any anomaly detection algorithm based on machine-learning or data-mining, all of which adopt the idea of extracting the patterns within the (massive) training set and thus raise the alarm on the deviate ones.

B. Existing Work

Previous and related work of anomaly or intrusion detection implements a great variety of approaches since [1]. Application of typical data-mining algorithms such as K-nearest neighbour or clustering could be found in [2], and neural networks in [3]. Hidden Markov Model also proves to be a useful technique as shown in [4]. In [5]–[8], theories of the classic Markov model are applied so as to detect anomalous patterns in the system, using *the ordering property of events* as proposed in [6]. [9] introduces the high-order Markov chain as an extension. Several approaches are then introduced to overcome the formidable cost that seems highly likely to come with it, including the hybrid model in [9] or support vector machine in [10].

On the other hand, as originally proposed in [12], the time series of system calls are now by common consent a powerful tool in identifying the nature of a system's behavior. Due to system calls' privileges, a large number of researches of intrusion detection, including [13] and [14] are based on exploiting, modelling, or learning from the audit data of system calls. In 1998, the Cyber Systems and Technology Group of MIT Lincoln Laboratory conducted a 7-week simulation of intrusion in the background of daily usage, and then released all their data named as the "1998 DARPA Intrusion Detection Evaluation Data Set" [15] on their website.

C. Our Contributions

The major contribution of the paper is our approach based on multi-order Markov chains, which reveals that the combination of mixed-order Markov chains would bring considerably interesting and substantial improvement over any single-order one with fairly reasonable cost. Utilizing not only the ordering property, this approach effectively suits the application of anomaly detection in addition to its first practice in rainfall modelling in [11]. In our practice, the relative ranking positions between probabilities from multi-order models serve as a new effective indicator for anomalies, as the ascending order suggests normal, while the descending one exhibits anomalous.

Secondly, we take into accounts a new category of inputs (the return values of system calls) to improve the effectiveness of the multi-order Markov chain based approach and form a two-dimensional model. In the application of anomaly detection, the conventional notion of using system calls to identify a system's behaviour [12]–[14] is insufficient in that it does not take into account the resulting status of execution. To improve this common approach, it would be shown in this paper that

978-0-7695-4987-3/13 $26.00 © 2013 IEEE
DOI 10.1109/COMPSACW.2013.12

the return value also plays a role together with the time series of system calls.

The remainder of the paper is organized as follows. Section 2 forms the problem, reviews the basics of the Markov chain model and introduces an approach to achieve the high-order and multivariate generalizations with fairly minor variation. A simplified example of model training and testing is illustrated in Section 3. Section 4 discusses some detailed issues involving algorithm design and complexity. Results and evaluation based on the data set is given in Section 5, and a general conclusion is presented in Section 6.

II. Methodology Overview

A. Problem Formulation

Fig. 1 demonstrates a typical piece of audit data, where the operating system tries to open or close a file, read or write something and create or terminate a process. How could we learn from it and differentiate the malicious ones? In Fig.1, such operations belong to the very same category of system calls that might be used by any programs, regardless of their intention. Therefore, to compensate the uncertainty in anomaly detection, results from multiple indicators or even multiple models are desired to issue a real-time warning for detrimental operations. In this paper, we address the above question by deploying the multi-order Markov chain based model.

B. The Classical Markov Chain

Generally, Markov chain is a discrete-time model of stochastic process with finite state space, and it shares several remarkable similarities with the approaches for anomaly detection as already evidenced by [5]–[10].

1) The Markov Property and Time-homogeneous Assumption: Theoretically, in order for the model to hold, it is implicitly assumed that all the observation sequences satisfy the characteristics of both Markov property and time-homogeneousness, a reasonable and necessary simplification based on the potential regularity and periodicity of a server system. Consider the state space S with m states in total:

$$S = \{s_1, s_2, \cdots, s_m\} \tag{1}$$

and the observation sequence X_n (training sequence):

$$X_n \in S,\ n = 1, 2, ..., N_X \tag{2}$$

If X_n possesses Markov property, then it satisfies that the present state results solely from the last one whatever transitions the system might have gone through before:

$$\begin{aligned} &\mathrm{P}\left(X_n = s_{i_n} | X_{n-1} = s_{i_{n-1}}, \cdots, X_2 = s_{i_2}, X_1 = s_{i_1}\right) \\ &\quad = \mathrm{P}\left(X_n = s_{i_n} | X_{n-1} = s_{i_{n-1}}\right)\ for\ n \geq 2 \end{aligned} \tag{3}$$

On the other hand, the property of time-homogeneous (or stationary) ensures that the way how current state results in the next is independent of time:

$$\mathrm{P}\left(X_{n+1} = s_j | X_n = s_i\right) = \mathrm{P}\left(X_n = s_j | X_{n-1} = s_i\right) \tag{4}$$

so every single step of system variation would conform to a unified transition matrix, while at the same time rendering it blind to any time-sensitive anomalies.

2) The Model of Markov Chain: From the training sequence X_n in formula (2), both the initial probability distribution matrix Q, where each element q_i represents the initial probability of the corresponding state s_i:

$$Q = \left[\ q_1 \quad \cdots \quad q_i \quad \cdots \quad q_m\ \right],\ where\ q_i = \mathrm{P}\left(X_n = s_i\right) \tag{5}$$

and the transition probability distribution matrix P, where each element p_{ij} represents the transition probability from s_i to s_j:

$$P = \left[\ p_{ij}\ \right]_{\ m \times m},\ where\ p_{ij} = \mathrm{P}\left(X_n = s_j | X_{n-1} = s_i\right) \tag{6}$$

could be obtained.

3) Support for New Sequences: For new observation sequence Y_n (test sequence):

$$Y_n \in S,\ n = 1, 2, ..., N_Y \tag{7}$$

its probability of support from the model equals the product of the initial probability and succeeding transition probabilities:

$$\mathrm{P}\left(Y_1, \cdots, Y_{N_Y}\right) = \begin{cases} q_{Y_1}, & N_Y = 1 \\ q_{Y_1} \prod_{n=2}^{N_Y} p_{Y_{n-1}Y_n}, & N_Y \geq 2 \end{cases} \tag{8}$$

In a probabilistic sense, the value of support serves as a quantitative indicator for anomaly by measuring how significant the test sequence deviates from the training data.

C. High-order and Multivariate Markov Chain

Markov chain does not make any specification about what kind of state s_i really is. Therefore, by constructing new states out of the original ones, a generalized variation that applies to the high-order or multivariate conditions could be derived.

1) High-order Markov Chain: For sequence X_n that satisfies the Kth-order Markov property, where the current state depends on not only the previous statuses but also the K preceding ones:

$$\begin{aligned} &\mathrm{P}\left(\mathrm{X_n} = \mathrm{s_{i_n}} | \mathrm{X_{n-1}} = \mathrm{s_{i_{n-1}}}, \cdots, \mathrm{X_2} = \mathrm{s_{i_2}}, \mathrm{X_1} = \mathrm{s_{i_1}}\right) \\ &= \mathrm{P}\left(\mathrm{X_n} = \mathrm{s_{i_n}} | \mathrm{X_{n-1}} = \mathrm{s_{i_{n-1}}}, \cdots, \mathrm{X_{n-K}} = \mathrm{s_{i_{n-K}}}\right) \\ &\qquad for\ n \geq K+1 \end{aligned} \tag{9}$$

construct new sequence X_n^* from X_n:

$$X_n^* = \underbrace{\left[\ X_n \quad \cdots \quad X_{n-K+1}\ \right]}_{K} \in S^*\ for\ n \geq K \tag{10}$$

Obviously, X_n^* is the first-order equivalent of X_n which satisfies the classical Markov property defined in formula (3). The new state space S^* comprises totaly m^K states:

$$S^* = \{s_1^*, \cdots, s_i^*, \cdots, s_{m^K}^*\} \tag{11}$$

where each state s_i^* is a certain permutation of the K original states:

$$s_i^* = \left[\ s_{\theta_K(i)} \quad \cdots \quad s_{\theta_j(i)} \quad \cdots \quad s_{\theta_1(i)}\ \right] \tag{12}$$

Without loss of generality, the permutation $\theta_j(i)$ could be defined as a K-digit base m number:

$$\theta_j\left(i\right) = \frac{1}{m^{j-1}} \left\{ \left[i - \sum_{\omega=1}^{j-1} \left(\theta_\omega\left(i\right)\ m^{\omega-1}\right) \right] \mod\ m^j \right\} \tag{13}$$

...	10:09:30 open() success	10:09:30 ioctl() success	10:09:30 close() success	10:09:33 close() failure	10:09:33 kill() failure	10:09:40 fork() success	10:10:00 exit() success	...

Fig. 1. The time series of system calls

Similarly to formula (5), the initial matrix of all m^K initial states is as follows:

$$Q^* = [\ q_1^* \quad \cdots \quad q_i^* \quad \cdots \quad q_{m^K}^*\ ] \tag{14}$$

where q_i^* represents the initial probability distribution of s_i^*:

$$q_i^* = \mathrm{P}\left(X_n^* = s_i^*\right)\ for\ n \geq K \tag{15}$$

Slightly different from the classical model, the total amount of possible transitions is limited to $m^K \times m$ instead of the entire $m^K \times m^K$ ones in the first-order condition, so the new transition matrix would be:

$$P^* = \left[\ p_{ij}^*\ \right]_{\ m^K \times m} \tag{16}$$

where:

$$p_{ij}^* = \mathrm{P}\left(X_n = s_j | X_{n-1}^* = s_i^*\right) \tag{17}$$

For test sequence Y_n in formula (7), construct new sequence Y_n^* in the same way:

$$Y_n^* = \underbrace{[\ Y_n \quad \cdots \quad Y_{n-K+1}\ ]}_{K} \in S^*\ for\ n \geq K \tag{18}$$

whose probability of support is given by:

$$\mathrm{P}\left(Y_K^*, \cdots, Y_{N_Y}^*\right) = \begin{cases} q_{Y_K^*}^*, & N_Y = K \\ q_{Y_K^*}^* \prod_{n=K+1}^{N_Y} p_{Y_{n-1}^* Y_n}^*, & N_Y \geq K+1 \end{cases} \tag{19}$$

2) Multivariate Sequences: Intuitively, an instant solution for multiple sequences is to apply the model for each of them, but in this way, the variables' latent correlations that might be the very factor that contributes to anomalies are lost. Therefore, combing them into a new space would be a practically superior method.

For a D-dimensional sequence, if the original state spaces are noted as S_1, S_2 to S_D and each consists of m_1, m_2 or m_D states respectively, the corresponding state space of combination comprises totaly m^D states:

$$m^D = \prod_{i=1}^{D} m_i \tag{20}$$

and the new state space S^D is given by:

$$S^D = \left\{s_1^D, \cdots, s_i^D, \cdots, s_{m^D}^D\right\} \tag{21}$$

each state s_i^D of which is a combination of the original states:

$$s_i^D = \left[\ s_{i_1} \quad \cdots \quad s_{i_j} \quad \cdots \quad s_{i_D}\ \right],\ 1 \leq i_j \leq m_j \tag{22}$$

where each element belongs to one of the original state spaces:

$$s_{i_j} \in S_j,\ j = 1, 2, ..., D \tag{23}$$

Hence, the multivariate sequences are able to comply with the classical univariate Markov chain approach.

TABLE I. MAPPING FROM $\{A, B\}$ TO S

$\{A, B\}$	S
$\{a_1, b_1\}$	s_1
$\{a_1, b_2\}$	s_2
$\{a_1, b_3\}$	s_3
$\{a_2, b_1\}$	s_4
$\{a_2, b_2\}$	s_5
$\{a_2, b_3\}$	s_6

TABLE II. MAPPING FROM $\{S, S\}$ TO S^*

$\{A, B\}$	S
$\{s_1, s_1\}$	s_1^*
$\{s_1, s_2\}$	s_2^*
$\{s_1, s_3\}$	s_3^*
...	...
$\{s_6, s_5\}$	s_{35}^*
$\{s_6, s_6\}$	s_{36}^*

Moreover, the variation of high-order and multivariate model could be applied jointly, creating a D-dimensional Kth-order Markov chain. In our practice discussed later, both of these generalizations are implemented as the Multi-order Markov Chain based model, where we utilize the system calls and their return values as a two-dimensional input ($D = 2$) of the 1st, 2nd and 3rd-order Markov chain ($K = 1, 2, 3$).

III. AN ILLUSTRATIVE EXAMPLE

For clarity, an example of model training and testing is shown step by step as follows. Suppose we have the raw training data shown in Fig. 2, and a two-dimensional ($D = 2$)

10:09:30 kill() success	10:09:30 fork() failure	10:09:30 kill() success	10:09:33 fork() failure	10:09:33 open() failure

Fig. 2. The raw input series (training set)

model of 2nd-order ($K = 2$) is desired.

First, the two-state space A and three-state space B could be obtained:

$$A = \{success,\ failure\} \tag{24}$$

$$B = \{open(),\ kill(),\ fork()\} \tag{25}$$

For simplicity, the training sequences derived directly from 2 are noted as X_1 and X_2 respectively:

$$X_1 = a_1, a_2, a_1, a_2, a_2 \tag{26}$$

$$X_2 = b_2, b_3, b_2, b_3, b_1 \tag{27}$$

Now we create the table for mapping from the input multivariate sequence to a univariate one as shown in TABLE I, and construct a univariate equivalent sequence X (and the implicit state space S):

$$\begin{aligned} X &= \{a_1, b_2\}, \{a_2, b_3\}, \{a_1, b_2\}, \{a_2, b_3\}, \{a_2, b_1\} \\ &= s_2, s_6, s_2, s_6, s_4 \end{aligned} \tag{28}$$

Then, the first-order equivalent sequence X^* (and the implicit state space S^*) is given using the mapping in TABLE II:

$$\begin{aligned} X^* &= \{s_2, s_6\}, \{s_6, s_2\}, \{s_2, s_6\}, \{s_6, s_4\} \\ &= s_{12}^*, s_{34}^*, s_{12}^*, s_{34}^* \end{aligned} \tag{29}$$

Evidently, the initial matrix Q^* (1 by 6×6) with three nonzero elements ($q_{12}^* = 1/2,\ q_{21}^* = 1/4,\ q_{34}^* = 1/4$):

$$Q^* = \begin{bmatrix} \cdots & \frac{1}{2} & \cdots & \frac{1}{4} & \cdots & \frac{1}{4} & \cdots \end{bmatrix}_{1\times 36} \tag{30}$$

and the transition matrix P^* (6×6 by 6) with three nonzero elements ($p_{12,2}^* = 1/2,\ p_{12,4}^* = 1/2,\ p_{34,6}^* = 1$):

$$P^* = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \frac{1}{2} & 0 & \frac{1}{2} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & 1 & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{36\times 6} \tag{31}$$

are established. Therefore, the possibility of every test sequence constructed this way could be determined using formula (19). For instance, given two pieces of test data shown in Fig. 3 and Fig. 4, again the test sequences derived are noted

10:09:30	10:09:30	10:09:30
kill()	fork()	open()
success	failure	failure

Fig. 3. The raw input series (test set 1)

10:09:30	10:09:30	10:09:30
kill()	fork()	fork()
failure	failure	success

Fig. 4. The raw input series (test set 2)

as Y_1, Y_2, Z_1, Z_2:

$$Y_1 = a_1, a_2, a_2 \tag{32}$$

$$Y_2 = b_2, b_3, b_1 \tag{33}$$

$$Z_1 = a_2, a_2, a_1 \tag{34}$$

$$Z_2 = b_2, b_3, b_3 \tag{35}$$

Similarly, we have:

$$Y = \{a_1, b_2\}, \{a_2, b_3\}, \{a_2, b_1\} = s_2, s_6, s_4 \tag{36}$$

$$Y^* = \{s_2, s_6\}, \{s_6, s_4\} = s_{12}^*, s_{34}^* \tag{37}$$

$$Z = \{a_2, b_2\}, \{a_2, b_3\}, \{a_1, b_3\} = s_5, s_6, s_3 \tag{38}$$

$$Z^* = \{s_5, s_6\}, \{s_6, s_3\} = s_{30}^*, s_{33}^* \tag{39}$$

According to formula (19), the probabilities of them are:

$$\mathrm{P}(Y^*) = q_{12}^* \times p_{12,4}^* = 1/2 \times 1/2 = 1/4 \tag{40}$$

$$\mathrm{P}(Z^*) = q_{30}^* \times p_{30,3}^* = 0 \times 0 = 0 \tag{41}$$

As a result, the latter test set is expected to be more "abnormal" than the former one.

In addition, the ordinal indices of the new state space could conform to any mapping other than TABLE I and TABLE II as long as they are unique.

IV. Algorithm Design and Implementation

Several further considerations and solutions about selecting model orders, dealing with zero-probability events and storing sparse matrix, etc. are discussed in detail.

A. Order of the Model

In our approach, the trade-off between model complexity and generalizability comes in the form of weighing the appropriate order for it. System of higher-order tend to neutralize the idealization brought by the assumption of Markov property and thus provides a more sensitive response over anomalous exceptions. In other words, high-order Markov chain generates larger support for frequent patterns within the training set while produces smaller support for rare ones at the same time. On the contrary, high-order models might lead to overfitting for noise or idiosyncrasy of the training sequence, and loses the ability to generalize outside it. In consequence, Markov chain models of up to the third order are utilized in our simulation.

B. Zero-probability Events

Due to the fact that a single zero in the multipliers would ultimately dominate the whole output value and the incomplete nature of any training sequence, here a solution proposed in [5] is implemented by replacing the zeros with a value far smaller than any other element (yet still essentially different from the real zero).

C. Time Complexity

The training stage involves calculating the initial and transition matrices and takes as much time as $O\left(m^{K+1}\right)$ approximately, which grows either polynomially with the amount of states or exponentially with the order of Markov chain. On the other hand, the test stage requires nothing more than a series of multiplication and consumes about $O(N_Y)$'s time when no overflow of bits occurs, which is linearly proportional to the length of the test set.

Even in the cases where multiplying are considered unacceptably time-consuming, it would be highly feasible to replace both transition and initial matrices with their logarithm values in training stage, causing all multiplication of the test stage substituted by addition, one of the basic commands supported by any instruction set architecture.

D. Space Complexity

As the amount of states and the order of model grow, the memory consumption for storing transition matrix alone could be formidable. However, when taking into account the sparse feature of the matrix, a binary bit of memory space is already sufficient enough for most elements of it. Together with the extra table declared exclusively for saving all nonzero values and their indices, the revised algorithm nonetheless succeeds in distinctly optimizing space complexity at the cost of limited sacrifice in time complexity.

During the our practice, the length of the non-zero value table is set to 10^4 while no overflow occurred, another proof of how such look-up approach manages to save storage consumption without compromising accuracy.

TABLE III. INPUT PARAMETERS

Parameter	Value
Orders	1,2,3
Training set amount	819472
Test set amount	100000×2
Sliding window width	200
ZERO	1×10^{-5}

TABLE IV. MULTI-ORDER EXECUTION SUMMARY

Order	1	2	3
Number of states	58	3364	195112
Number of transitions	3364	195112	11316496
Typical runtime (training)	3s	3s	5s
Typical runtime (test)	25s	38s	56s

V. RESULTS AND EVALUATION

For validation, the 1998 DARPA Intrusion Detection Evaluation Data Set [15] are processed with such model. Starting in June 1998, MIT Lincoln Laboratory conducted a 7-week simulation of TCP attacks in the background of normal computer operations, recording detailed system information including event time, system call, return value, etc.

A. Multi-order Markov Chain

All 819472 data of system calls from the first week's Friday are used as the training set here, while data from other Fridays are used as test sets. A more detailed summary of input parameters and execution could be found in TABLE III and TABLE IV. It is also noteworthy that although there are up to 243 types of posssible system calls, only 57 of them were encountered during the training stage. In this case, treating all other system calls as a single category would result in a simplified yet equivalent space comprising totally 58 states.

Calculation results using two different test sets are shown in Fig. 5 and Fig. 6, where the horizontal axis represents the ordinal number of a sub training sequence with 200 data, while the vertical axis represents the negative logarithmic values of support from the model. The blue, red and green curves stand for the first, second and third-order model respectively. For most of the time, the curves are maintained at a relatively low level, following the descending order from the first-order model to higher ones. Take the third-order model in Fig. 5 for instance, the green curve usually oscillates between 20 and 80, which indicates an average one-step transition probability of approximately 0.2 to 0.4 due to the negative logarithm. In view of the total 58 possible transitions, a probability of 0.2 to 0.4 would by no means be regarded as "rare" and thus agrees with the universal rule of the normality having a greater probability. However, at some point, the curve rises sharply, reaching roughly 330 in the tail of Fig. 5, together with the abrupt reverse of the originally descending order. Moreover, the sixth Friday shown in Fig. 6 witnesses a more dramatic situation in which the third-order result soars to 1000, or 1E-5 in terms of the one-step transition probability, the exact same value as the pre-defined constant ZERO in TABLE III.

According to the record by Lincoln Lab, both of the extreme values depicted in Fig. 5 and Fig. 6 coincide with a TCP attack chronologically, whereas the latter one resulted in an unprecedented system crash on July 10th, 1998.

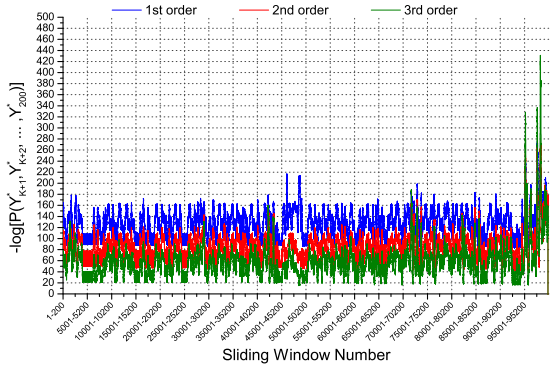

Fig. 5. Friday, Week 2

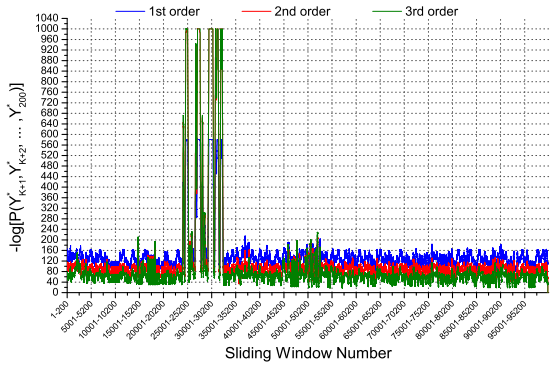

Fig. 6. Friday, Week 6

TABLE V. MULTIVARIATE INPUT PARAMETERS

Parameter	Value
Orders	1,2,3
Dimension	2
Training set amount	819472×2
Test set amount	(100000×2)×2
Sliding window width	200
ZERO	1×10^{-5}

TABLE VI. MULTIVARIATE EXECUTION SUMMARY

Order	1	2	3
Number of states	116	3364	1560896
Number of transitions	13456	1560896	181063936
Typical runtime (conversion)		15s	
Typical runtime (training)	3s	4s	5s
Typical runtime (test)	27s	44s	68s

B. Multivariate Sequence

Whenever a system call is executed, a certain binary value of either SUCCESS or FAILURE is returned. Along with the types of system calls processed above, the return values contribute to forming a kind of simplest two-dimensional time series. Accordingly, the summaries of input parameters and execution are given in TABLE V and TABLE VI.

The new results of the same Fridays are shown in Fig. 7 and 8. Although somewhat similar, these two sets of results could be distinguished in that where the univariate model in Fig. 5 gives 330 to indicate anomalies, the multivariate one generates 400 in Fig. 7. The greater values from the multivariate model naturally lead to a more sensitive anomaly detection considering the fact that when intrusion or attack

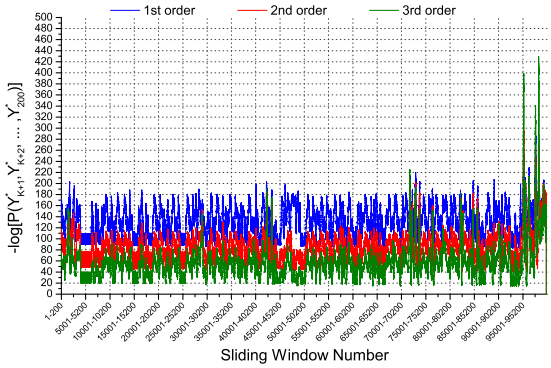

Fig. 7. Friday, Week 2 (two-dimensional)

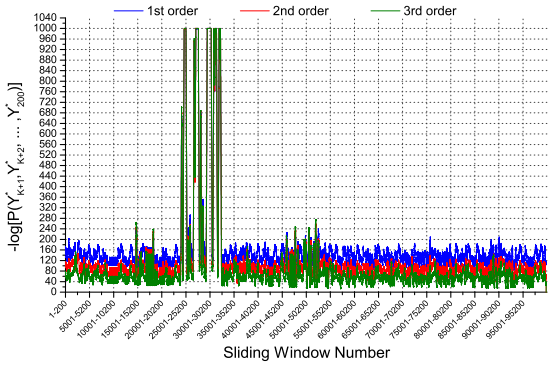

Fig. 8. Friday, Week 6 (two-dimensional)

occurs, it is more likely to have an extensive scale of FAILED system calls. The Other multivariate results in Fig. 8 fail to yield a better result for the curve in Fig. 5 is already at its theoretical maximum using the univariate approach.

C. Discussion

Although more of an empirical practice rather than theory-based proof, our results reveal several interesting properties of the multi-order model. Firstly, the training process assigns small (or ZERO) values to the rare transitions, and even smaller ones to the corresponding high-order transitions, resulting in a distinct inversion of their ranking order. Secondly, the time series of return values also contribute to identify groups of failed system calls. Either way, we are a tiny step closer to the ultimate goal of an ideal detection system in terms of accuracy and sensitivity. However, it is still possible to fabricate certain training set that leads to results with completely opposite properties. After all, it is what's written in the training set that defines the anomaly. In our study, the unique target object as a server system helps to relax such constraints to some extent.

VI. Conclusions

We proposed an anomaly detection scheme based on multi-order Markov chain by observing that the relative positions between results from models of different orders provide a new effective indicator for anomalies. If the support from a low-order model exceeds that of the high-order one, it is implied that abnormity might have occurred in the system. To improve sensitivity, we combined multiple sequences as a multivariate one into a single model and proved that the return values of system calls also play an essential role in detection. In addition, a practical algorithm with both time and space efficiency was implemented, minimizing the possibility of being the source of anomalies itself. The time consumption of training stage does not even exceed 15 seconds for a data set as large as 1.6 million, and for models up to the third-order combined. As anomaly detection remains an intricate issue, our finding is another evidence that only the confluence of results would massively increase the confidence level due to the inherent coexistence of both merits and defects in any model.

VII. Acknowledgment

This paper is partially sponsored by the National High-Tech R & D Program (863) of China (2009AA012201) for Yongxin Zhu, and NSF CNS-1249223 of US for Meikang Qiu.

References

[1] Denning, D.E., "An intrusion-detection model," *IEEE Transactions on Software Engineering*. pp. 222 - 232, Feb. 1987

[2] Yihua Liao, V.Rao Vemuri, "Use of K-nearest Neighbor Classifier for Intrusion Detection," *Computers & Security*. 21(5): 439-448, 2002

[3] S. Saravanakumar, Amruth Kumar.A, Anandaraj.s, s.Gowtham, "Algorithms Based on Artificial Neural Networks for Intrusion Detection in Heavy Traffic Computer Networks," *Proc. of Int'l Conf. on Advancements in Information Technology*, pp.6-23, 2011

[4] C.V. Raman, Atul Negi, "A Hybrid Method to Intrusion Detection Systems Using HMM," *Distributed Computing and Internet Technology*. pp. 389 - 396, 2005

[5] Nong Ye, "A Markov Chain Model of Temporal Behavior for Anomaly Detection," *Workshop on Information Assurance and Security*. West Point, NY, June 2000.

[6] Nong Ye, Xiangyang Li, Qiang Chen, Syed Masum Emran, Mingming Xu, "Probabilistic Techniques for Intrusion Detection Based on Computer Audit Data," *IEEE Transactions on Systems, Man, and Cybernetics tPart A: Systems and Humans*. 31(4):266-274, July 2001

[7] Nong Ye, Yebin Zhang, Connie M. Borror, "Robustness of the Markov-Chain Model for Cyber-Attack Detection," *IEEE Transactions on Reliability*. 53(1):116-123, March 2004

[8] Robert Gwadera, Mikhail Atallah, "Markov Models for Identification of Significant Episodes," *Proceedings of the 5th International Conference on Data Mining*. 2005

[9] Wen-Hua Ju, Yehuda Vardi, "A Hybrid High-Order Markov Chain Model for Computer Intrusion Detection," *Journal of Computational and Graphical Statistics*. 10(2): 277-295, 2001

[10] Chuanhuan Yin, Shengfeng Tian, Shaomin Mua, "High-order Markov kernels for intrusion detection," *Neurocomputing*. 71(16-18):3247-3252, 2008

[11] Markus Stowasser, "Modelling Rain Risk: A Multi-order Markov Chain Model Approach", *The Journal of Risk Finance,* 13(1):45 - 60, 2011

[12] Stephanie Forrest, Steven A, Hofmeyr, Anil Somayaji, Thomas A, Longstaff, "A Sense of Self for Unix Processes," *Proceedings of IEEE Symposium on Security and Privacy*, pp. 120 - 128, 1996

[13] Gaurav Tandon, Philip Chan, "Learning Rules from System Call Arguments and Sequences for Anomaly Detection," *Workshop on Data Mining for Computer Security*. IEEE International Conference on Data Mining, pp. 20 -29, Nov. 19-22, 2003

[14] Wei Wang, Xiaohong Guan, Xiangliang Zhang, Liwei Yang, "Profiling program behavior for anomaly intrusion detection based on the transition and frequency property of computer audit data," *Computers & Security*. 25(7): 539 - 550, 2006

[15] Richard P, Lippmann, David J, Fried, Isaac Graf, et al, "Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation," *Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX)*, Vol. 2, pp. 12 - 26, 2000