Journal of Universal Computer Science, vol. 22, no. 4 (2016), 475-493
submitted: 1/10/15, accepted: 30/3/16, appeared: 1/4/16 © J.UCS

A Novel Similar Temporal System Call Pattern Mining for Efficient Intrusion Detection

Vangipuram Radhakrishna
(VNR Vignana Jyothi Institute of Engineering and Technology, Hyderabad, India
radhakrishna_v@vnrvjiet.in)

Puligadda Veereswara Kumar
(University College of Engineering, Osmania University, Hyderabad, India
pvkumar58@gmail.com)

Vinjamuri Janaki
(Vaagdevi Engineering College, Warangal, India
janakicse@yahoo.com)

Abstract: Software security pattern mining is the recent research interest among researchers working in the areas of security and data mining. When an application runs, several process and system calls associated are invoked in background. In this paper, the major objective is to identify the intrusion using temporal pattern mining. The idea is to find normal temporal system call patterns and use these patterns to identify abnormal temporal system call patterns. For finding normal system call patterns, we use the concept of temporal association patterns. The reference sequence is used to obtain temporal association system call patterns satisfying specified dissimilarity threshold. To find similar (normal) temporal system call patterns, we apply our novel method which performs only a single database scan, reducing unnecessary extra overhead incurred when multiple scans are performed thus achieving space and time efficiency. The importance of the approach coins from the fact that this is first single database scan approach in the literature. To find if a given process is normal or abnormal, it is just sufficient to verify if there exists a temporal system call pattern which is not similar to the reference system call support sequence for specified threshold. This eliminates the need for finding decision rules by constructing decision table. The approach is efficient as it eliminates the need for finding decision rules (2^n is usually very large for even small value of n) and thus aims at efficient dimensionality reduction as we consider only similar temporal system call sequence for deciding on intrusion.

Keywords: Intrusion, Malicious, System Call Pattern, Temporal, Similarity, Vulnerability
Categories: D.4.6, H.3.1, H.3.2, H.3.3, H.4

1 Introduction

Conventional security approaches prevent access or entry into the system and are mainly designed to make the attacks unsuccessful. These security mechanisms act like a wall or simple lock which prevents entry in to the system. However, if such an attack becomes a success, then this attack turns into an intrusion. In such a situation, these conventional tools cannot detect intrusion and fail to restrict any access to the

system. This brings a need to dynamically track the system behavior to detect intrusion. Intrusion detection systems are designed to address these issues.

In this paper, we address the importance of finding temporal system call association patterns to detect any possibility of an intrusion. For this we consider all the normal events or system call traces of processes showing normal behavior over a period of time or time slots such as day, week, month etc. The idea is to represent the dataset as a process-system call database of temporal transaction set. From this database, we find the similar temporal association patterns w.r.t normal reference sequence specified by network administrator. Once temporal association patterns are found, we obtain the process-system call pattern matrix. These temporal system call patterns found are considered as the feature vector for classification process.

Naturally, the temporal patterns which are not present in the normal process set have the possibility of intrusion. When any new process is initiated, the system call trace is considered as input and the temporal patterns in feature vector are searched for their existence and their presence or absence is recorded. The new process is hence reduced to the dimensionality of feature vector.

To find whether the process is normal or abnormal it is just sufficient to verify if there exists a temporal system call pattern which is not similar to the reference system call support sequence and user specified threshold. This eliminates the need for finding decision rules by constructing decision table. The importance of this approach coins out from the fact that the process of finding temporal system call patterns is a single database scan approach and eliminates extra overhead in performing multiple scans and also eliminates the need for finding decision rules (2^n is usually very large for even small value of n) and aims at efficient dimensionality reduction as we consider only similar temporal system call sequence. A brief literature survey is given in section-2 and proposed approach in abstract view is discussed in section-3. This is followed by outlining procedure to find similar temporal system call patterns in section-4 with case study in section-5. A sample case study for intrusion detection using proposed approach is discussed in section-6 and finally we conclude the work in section-7.

2 Literature Review

One of the major objectives of a typical information security system is to achieve the principle of “defense in depth”. In essence, the objective is to design, develop and deploy a multi-layer defense system which has the ability to prevent the exploit, identify and stop the attack, discover agents causing threat and subsequently process them accordingly so as to avoid possibility of security breach. In terms of computer security, the term ,“vulnerability“ refers to the attack surface which facilitates the attacker to target the downfall of systems information assurance. In simple sense, we may classify software risk as a vulnerability. Vulnerabilty may be at different levels. Application level vulnerability refers to inability of application which causes it to be exploited to compromise on its security. This initiates possibility of cyber crime which target confidentiality, availability and integrity. In general, 90% of vulnerabilities coin in application layer.

Software vulnerabilities may be at network or application level. [Aljawarneh, 2010] discusses the application level security vulnerabilities. They coin the failure of

firewalls in ensuring data integrity at application level. The authors [Aljawarneh, 2010] propose a novel semantic validation service to prevent application level security vulnerabilities, so as to secure the web system even in a situation where input validation modules are all bypassed. The author work in [Aljawarneh, 2011] coins security issues in cloud.

The importance of temporal pattern mining in intrusion detection is the present re-search interests among researchers working in areas of security, data mining, and information retrieval. The two most common approaches of intrusion detection are anomaly and misuse detection. The major limitation of misuse based network intrusion detection is that it is not suitable and inefficient to recognize new intrusion pat-terns as that of anomaly based approach which uses the concept of false alarming. The approach of intrusion detection using interval based techniques is addressed in [Qiang, 2002] using the concept of rule generation in a bi-temporal database. In [Jones, 2001], the concept of temporal signature is used for intrusion detection. In [Alexandr, 2002] Alexandr Seleznyov et.al, show the importance temporal patterns in the context of intrusion detection. Even though, sequential patterns are important in the context of intrusion detection, temporal patterns may also be used to extract the user behaviour to recognize legitimate users. Most anomaly detection approaches use statistical information about events or find sequential patterns and do not consider the temporal information and behaviour. Such temporal information if found can be very useful in performing anomaly detection. The authors work [Alexandr], addresses the importance of considering temporal information along with statistical information of events and sequential patterns to detect abnormal behaviour patterns. The approach in [Xin, 2005] includes using the system call sequences for host based intrusion detection. This work is an extension of the work presented at ACM ICEMIS 2015 conference [Radhakrishna, 2015].

3 Proposed Method

3.1 Algorithm

Figure.1 depicts abstract representation of proposed approach. The approach includes following steps

1. Represent audit logs as a temporal database of disjoint time stamped transaction sets. We view running process as a time stamped transaction having process-ID and system calls of running process may be viewed as items of transaction. Each process is represented as a system call sequence. The time slice may be day, week, month etc.
2. Choose a reference system call support time sequence and dissimilarity threshold. These are called subset specifications usually specified by network administrator. The reference sequence must be chosen to represent normal support sequence based on normal events.
3. Apply the algorithm in Section 4, to discover similar temporal system call patterns w.r.t reference sequence. These system call patterns are called normal patterns which are temporally similar and indicate normal system behavior.
4. Represent these temporal patterns as a feature vector and obtain the process-system call pattern matrix. The matrix gives information of presence or absence

of the corresponding system call pattern. To perform supervised learning we may use these temporal system call sequence pattern as features for process-system call representation.

5. Compare the system call sequences of new process with temporal patterns discovered in step-4. If there is a system call pattern(s) in the incoming process which is not similar to the reference system call support sequence, classify the process as Abnormal or Attack process indicating possibility of intrusion.

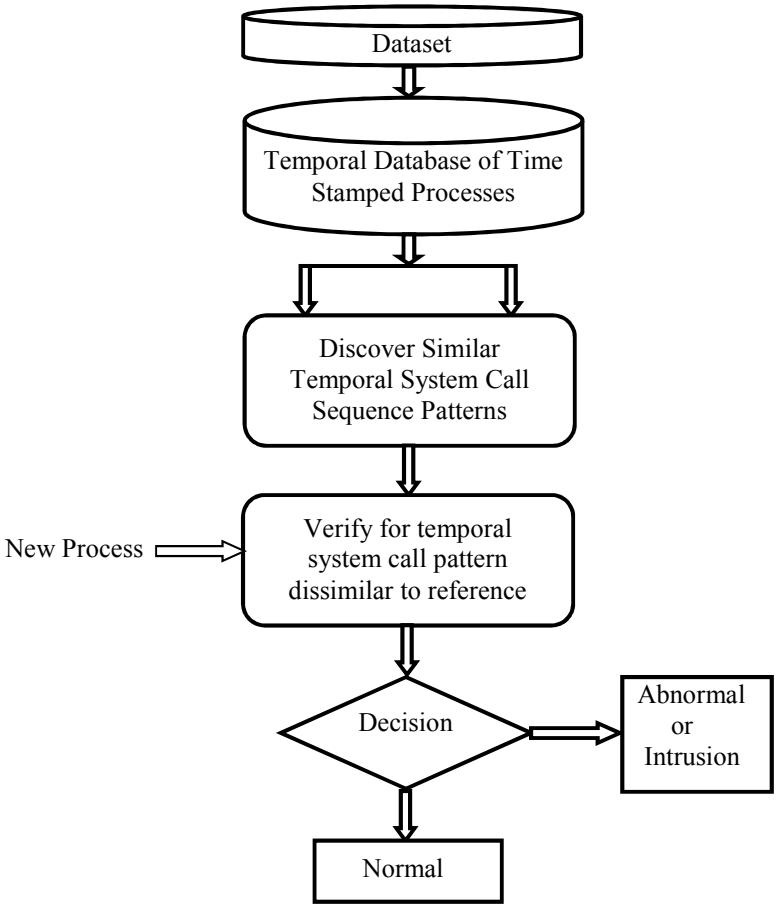

Figure 1: Proposed Approach for Intrusion Detection

In summary, we consider all the normal events or system call traces of processes showing normal behaviour over a time period or time slots such as day, week, month etc. The idea is to represent the dataset as a process-system call database of temporal transaction set. From this database, we find the temporal system call association patterns. Once temporal association patterns are found, we obtain the process-system call pattern matrix. These temporal system call patterns discovered may be considered

as the feature vector for classification process. Naturally, the temporal patterns which are not present in the normal process set have the intrusion possibility. In general, to find whether the process is normal or abnormal it is just sufficient to verify if there exists a temporal system call pattern which is not similar to the reference system call support sequence and user specified threshold.

This eliminates the need for finding decision rules by constructing decision table. The importance of this approach coins out from the fact that the process of finding temporal system call patterns is a single database scan approach and eliminates extra overhead in performing multiple scans and also eliminates the need for finding decision rules (2^n is usually very large for even small value of n) and aims at efficient dimensionality reduction as we consider only similar temporal system call sequence. Section 4 outlines proposed approach to find similar temporal system call patterns w.r.t a given reference system call support sequence and threshold.

4 Discovering Similar Temporal System Call Patterns

4.1 Problem Definition

Given a finite set of system calls, I and disjoint set of time slots and temporal database of time stamped process , a reference system call support sequence and user specified threshold value, the objective is to find set of all system call patterns which are considered temporally similar w.r.t reference vector. Each process is represented as a 2-tuple with elements timestamp and set of system calls for that process-ID. The distance measure, denoted by $f_{similarity}$ (P, Q) :$\rightarrow R^n$, where the parameters P and Q are numeric sequences is used as dissimilarity function. The objective is to " Find the set of all system call patterns, *I* , which are subsets of I such that each of these temporal system call pattern represented by *I* , satisfy the condition $f_{similarity}$ (S_I, Reference) $\leq \theta$ where S_I is the sequence of system call support values of *I* at time slots t_1 , t_2 ……., t_n

4.2 Research Objective

We have the following research objectives.

1. To find set of all those temporal system call patterns which are similar to a specified reference not exceeding the specified threshold. The temporal system call patterns and reference vectors in our case are multi-dimensional vectors which are sequence of support values computed for each time slot.
2. To perform only single scan of input temporal process-system call database.
3. To design formal expressions which can estimate the minimum and maximum bounds on system call support values of temporal system call pattern considered.

4.3 Terminology

We introduce following terms and conventions in this paper.
Here we refer to system call temporal pattern and itemset interchangeably.

4.3.1 Itemset

An item set is a subset obtained by chosen combination of items from a finite set of items. For a finite set of items,I ,whose size is |I|, we can obtain $2^{|I|}$ -1 , itemset combinations including empty set. In our case, Itemset is also called as System Call Set. It is treated as global feature set.

4.3.2 Negative system call pattern (Negative Itemset)

An itemset whose support is found for probability of its non-existence is called negative itemset. For example, all those itemsets represented as $\bar{A}$, $\overline{AB}$, $\bar{B}$ are called negative itemsets.

4.3.3 Positive system call pattern (Positive Itemset)

An itemset whose support is found for probability of its existence is called positive itemset. Item sets represented as X, AB, Y, XYZ are called positive itemsets.

4.3.4 Negative Support

The support value obtained considering negative system call pattern (itemset) is called as negative support.

4.3.5 Positive Support

The value of probability obtained considering positive system call pattern (itemset) is called as positive support.

4.3.6 System call pattern support sequence

The system call pattern support sequence (itemset support sequence) obtained for a given system call sequence pattern is an n-tuple denoted by a sequence represented mathematically as $S_\theta\ (I) = < S_{t_1}\ , S_{t_2}\ , S_{t_3}\ ,\ \ldots\ldots S_{t_n} >$. Here each S_{t_i} represents support value of system call pattern I computed for time slot t_i . Formally, $S_\theta\ (I) =$ U_n { S_{ti} | time slot, t_i varies from t_1 to t_n} where U_n is union of all support values of system call sequence pattern computed for each time slot, t_i .

4.3.7 Negative System call pattern support sequence

The pattern support sequence, S_Θ , of an itemset I′ ⊆ I denoted by, $\boldsymbol{S_\theta(I')}$, is said to be the negative system call pattern support sequence denoted by $\boldsymbol{S_{\theta_N}(I')}$, if support of each element, $\boldsymbol{S_{t_i}}$, in the sequence is computed for the absence of itemset I′ in time stamped temporal process–sytem call database. The negative support sequences denote probability of the system call pattern not existing in time slots t_1, t_2 ,…….. t_n.

4.3.8 Positive System call pattern support sequence

The pattern support sequence, S_Θ , of an itemset I’ ⊆ I denoted by, $S_\theta(I')$, is said to be the positive system call pattern support sequence, denoted by $S_{\theta_P}(I')$, if support

of each element, S_{t_i} , in the sequence is computed for the existence of system call sequence pattern, I′ in the database.

4.3.9 Lower Lower Bound

It is the formally defined as the distance computed between the minimum possible system call pattern support sequence (LBSTS) and the reference sequence [Vangipuram,2015].

4.3.10 Upper Lower Bound

It is the formally defined as the distance computed between the maximum possible system call pattern support sequence (UBSTS) and the reference sequence vector [Vangipuram,2015].

4.3.12 True Distance

Distance computed between system call pattern support sequence, S_Θ and reference sequence vector, R_Θ.

4.3.13 Pruning

The process of elimination of temporal system call pattern violating the specified constraints is termed as pruning. In general, any pattern, denoted by I, is said to be temporally similar, if and only if each of its subsets are also temporally similar.

4.4 Discovering Similar Temporal Association Patterns

In this section we outline the approach to find similar system call temporal association patterns for a given reference support sequence and user specified threshold.

Input: Let L, be a finite set of all system calls, $D_{temporal}$ be disjoint temporal database of time stamped process-system calls defined over a finite set of disjoint time slots, N indicating total number of items in set L, I denote itemset which is subset of L and is of size, k and $f_{similarity}$ be the distance measure used to estimate the dissimilarity between two vector sequences with respect to a user defined threshold, θ.

Output: Set of all temporal system call patterns, $\bar{I}$, which are subsets of L such that each of these patterns represented by I', satisfy the condition $f_{similarity}(S_I, R) \leq \theta$ where S_I is the sequence of support values of itemset I' at the time slots t_1 , t_2 ……., t_n

Stage-1: compute probability of singleton temporal system call patterns
Find probability of each positive and negative singleton temporal pattern. This must be computed for every time slot. These probability values are also called positive and negative support values of singleton system calls.

Stage-2: compute support sequence of temporal system call sequence patterns
Find positive system call support sequence and negative system call support sequence represented by S_{θ_P}, S_{θ_N} for positive and negative system call sequence patterns

respectively. The temporal system call patterns, we consider are categorized into 3 types according to size of temporal patterns, denoted by |S|. We consider three cases i.e temporal patterns of size=1, |S|=1; temporal pattern of size, |S|=2; temporal pattern of size, |S| > 2

Stage-3: Temporal system call patterns of size, |S|=1 (A, B, C ….)
Find Euclidean distance between each singleton temporal system call pattern and reference sequence. If computed distance ≤ user-specified dissimilarity value (threshold), consider such temporal system call patterns to be similar (i.e normal system call sequence pattern). If computed distance violates threshold constraint by deviating from reference sequence, such system call sequence patterns are treated temporally dissimilar.

Since, the Euclidean distance do not support monotonicity property, We choose retain all such temporal patterns whose approximate upper lower bound value, ULB_{approx} is less than user specified threshold value. This is mainly done to compute temporal patterns of size, |S|≥2 and also for the reason the upper lower bound distance preserves monotonocity property.

Stage-4 : Temporal system call patterns of Size, |S|=2 (AB, AC, BC….)
Set, |S| = |S|+1. This step involves finding support sequences of temporal patterns which is followed by finding upper-lower, lower - lower and lower bound distances of temporal itemset w.r.t reference sequence. This involves generating temporal patterns of size, |S|=2, from temporal association patterns retained in step-3. All generated patterns shall be of the form $I_i\ I_j$ where I_i is singleton temporal item of length one and I_j represents item not present in I_i. Since, we do not scan the database, we choose to find maximum and minimum possible support sequence of temporal itemsets of size, |S|=2 respectively. Here, for each temporal pattern of the form $I_i\ I_j$, I_i and I_j is the temporal itemset of size, |S|=1, whose support sequence is already found in Step-3.

To compute the support sequences of itemset of size, |S|=2, we use the expression,

$$I_i\ I_j = \frac{1}{2}[\ I_i + I_j - I_i\ \overline{I_j} - I_j\ \overline{I_i}\ ] \quad (1)$$

To compute , temporal support time sequence for itemset of the form , $I_i\ I_j$, we must compute upper and lower bound support sequence vectors of $I_i\ \overline{I_j}$ and $I_j\ \overline{I_i}$ using the generalized procedure for an itemset $I_i\ I_j$ discussed in section 3.6 , then obtain minimum and maximum possible support sequences for itemset, $I_i\ I_j$ of size, |S|=2. This is followed by finding the upper lower bound (ULB) , lower lower bound (LLB) and lower bound (LB) distance for the itemset $I_i\ I_j$. If , lower bound distance , LB< θ , then consider it as similar temporal association pattern, otherwise treat as temporally dissimilar. Alternately if, the upper lower bound distance value (ULB) of $I_i\ I_j \leq \theta$, then, retain such patterns, to find support sequence of temporal patterns of size, |S|>2.

Stage-5: Temporal system call patterns of size, |S|>2

Set |S| = |S|+1. Generate all possible temporal system call patterns of size, |S|>2, from the temporal association patterns of size, |S|=K-1, retained in previous step.

Now, the temporal itemset combinations generated will be of the form $I_i\ I_j$ where I_i must be mapped to first |S|-1 sequence of items and I_j indicates, the singleton temporal item of length equal to one, not present in I_i. For, temporal patterns of size, |S|>2, we have a peculiar situation. This is because, when |S|=1, we know true support values of singleton temporal patterns. This finishes first scan. For |S|=2, we do not compute true support sequences, but we obtain the maximum and minimum possible support sequences of temporal patterns as in step-4. So, for temporal patterns of size, |S|>2, such as |S|=3, 4, 5….N, we have four cases to be considered for computing itemset support sequences which are obtained using equation (2) below. This is shown given by the equation.2 considering itemset of the form $I_i\ I_j$

$$I_i\ I_j = \begin{cases} \frac{1}{2} * \{ (I_i)_{UBSTS} + I_j - [(I_i)_{UBSTS} * \overline{I_J}]_{UBSTS} - [I_j * \overline{(I_\iota)_{UBSTS}}]_{UBSTS} \} \\ \frac{1}{2} * \{ (I_i)_{UBSTS} + I_j - [(I_i)_{UBSTS} * \overline{I_J}]_{LBSTS} - [I_j * \overline{(I_\iota)_{UBSTS}}]_{LBSTS} \} \\ \frac{1}{2} * \{ (I_i)_{LBSTS} + I_j - [(I_i)_{LBSTS} * \overline{I_J}]_{UBSTS} - [I_j * \overline{(I_\iota)_{LBSTS}}]_{UBSTS} \} \\ \frac{1}{2} * \{ (I_i)_{LBSTS} + I_j - [(I_i)_{LBSTS} * \overline{I_J}]_{LBSTS} - [I_j * \overline{(I_\iota)_{LBSTS}}]_{LBSTS} \} \end{cases}$$

` (2)

From these support sequences, obtain the maximum and minimum support sequence of temporal itemsets of size, |S|>2 respectively. These are called maximum support time sequence and minimum support time sequence of itemset, $I_i\ I_j$ of size, |S|>2. Now, find the upper lower bound and lower lower bound values, lower bound values for the itemset $I_i\ I_j$. If the value of lower bound < θ , then consider it as similar temporal association pattern, otherwise treat such itemsets as temporally dissimilar. However, if, the approximate upper lower bound distance value of $I_i\ I_j$ is less than θ, then, consider all such itemsets of the form, $I_i\ I_j$, to generate itemset support sequences of next level. Itemsets of size, |S|=|S| +1. Repeat step-5 till size of temporal itemset is equal to number of items in the itemset I or till no further item sets can be generated.

Stage-6: Output set of all similar temporal association patterns w.r.t reference support sequence satisfying user specified constraints.

4.5 Generating System Call Support Sequence Bounds

Generating the support time sequences is very crucial to find the similar temporal association patterns. To generate the upper bound and lower bound support sequences we follow the procedure outlined in [V.Radhakrishna, 2015]. However the computation of support time sequence for a given temporal pattern is carried in different approach using the Equation.1 and Equation.2.

The earlier approaches for finding support time sequence of an itemset in literature requires support values of all its subsets and this requires scanning database for actual support values at previous stage in case the next stage temporal association pattern need to be found. In our proposed approach, computation of support time sequences for itemset combination $I_i\ I_j$ requires computing support time sequences for itemset denoted by $I_i\ \overline{I}_j$ and $I_j\ \overline{I}_1$. This eliminates need to maintain support of all k-1 subsets of item set of size k.

4.5.1 Computation of Upper and Lower Bound System Call Support Time Sequences

Let $\mathbf{S(I_i) = < S_{i1}\ , S_{i2}\ , S_{i3}\ , \ldots\ldots\ S_{im}\ >}$ and

$$S(I_j) = < S_{j1}\ , S_{j2}\ , S_{j3}\ , \ldots\ldots\ S_{jm}\ > \quad (3)$$

be the support time sequences of items I_i and I_j .

The upper bound and lower support time sequences of itemset $I_i\ I_j$ are computed using the equations below

$$\text{UBSTS}(I_i\ I_j) = < min(S_{i1}, S_{j1}), \min(S_{i2}, S_{j2}), \min(S_{i3}, S_{j3}) \ldots\ldots\ldots\ldots \min(S_{im}, S_{jm}) > \quad (4)$$

$$\text{LBSTS}(I_i\ I_j) = < max(S_{i1} + S_{j1} - 1, 0), max(S_{i2} + S_{j2} - 1, 0), max(S_{i3} + S_{j3} - 1, 0) \ldots\ldots\ldots\ldots max(S_{im} + S_{jm} - 1, 0) > \quad (5)$$

4.5.2 Computation of System Call Pattern Upper lower Bound (ULB)

Let R=< r_1 , r_2 , r_3 , …….r_m > also represented as R= < r_i | i ←1 to m > be a reference sequence. Let, itemset upper bound support time sequence, be represented by U = =< U_1 , U_2 , U_3 , …….U_m >.

Let the notations R^{Upper} and U^{Lower} indicate the subsequence of reference and upper support time sequences of length k , such that for each i varying from 1 to k the condition $R_i > U_i$ holds true, then the upper lower bound distance value is computed as ULB-distance(R, U) = Euclidean distance between vectors R^{Upper} and U^{Lower} of length, k.

4.5.3 Computation of System Call Pattern Lower lower Bound (LLB)

Let R=< r_1 , r_2 , r_3 , …….r_m > also represented as R= < r_i | i ←1 to m > be a reference sequence .Let, itemset lower bound sequence be denoted by L = =< L_1 , L_2 , L_3 , …….L_m >.

Further, if we assume R^{lower} and L^{upper} to be the subsequence of reference and lower support sequences of length k , such that for all i varying from 1 to k the condition $R_i < L_i$ holds valid, then the lower lower bound distance value is computed as LLB-distance(R, L) = distance between support sequences R^{lower} and L^{upper} of length, k.

4.5.4 Lower Bound distance (LB)

The minimum bound distance alsoc called lower bound defined as sum of upper lower bound and lower lower bound distances. Mathematically, Lower-bound distance formally as

LB = ULB distance + LLB distance (6)

Process-System Calls in SESSION-1		**Process-System Calls in SESSION-2**	
Process_ID	**System Calls**	**Process_ID**	**System Calls**
P1	fork	**P11**	open, close
P2	fork open close	**P12**	open
P3	fork, close	**P13**	fork, open, close
P4	fork	**P14**	fork, open close
P5	fork open close	**P15**	close
P6	close	**P16**	fork, open, close
P7	close	**P17**	fork, close
P8	fork open close	**P18**	close
P9	close	**P19**	open
P10	close	**P20**	open, close

Table 1: Sample Process-System Call Network Database

	System Call	**Probability in Session-1**	**Probability in Session-2**
Positive Probability	fork	0.60	0.40
	open	0.30	0.70
	close	0.80	0.80
Negative Probability	$\overline{\textbf{fork}}$	0.40	0.60
	$\overline{\textbf{open}}$	0.70	0.30
	$\overline{\textbf{close}}$	0.20	0.20

Table 2: System Call Probability in Different Sessions

Normal Reference Sequence	**Probability in Session-1**	**Probability in Session-2**
Ref	0.40	0.60

Table 3: Reference Sequence

5 Case Study - Finding Temporal Similar System Call Patterns

Consider the Table.1 depicting sample process system call information, N_D consisting finite set of system calls {fork, open, close} of processes executing over two sessions performed at the time slots t_1 and t_2 .

Assume the reference sequence (normal) is <0.4, 0.6>.

Step-1: Initially, we start by finding the positive support value of singleton system calls fork, open, close and the corresponding negative support of system calls $\overline{\textbf{fork}}$, $\overline{\textbf{open}}$, $\overline{\textbf{close}}$ for each session. The support values at different sessions i.e at time slots $\boldsymbol{t_1}$, and $\boldsymbol{t_2}$ is shown in the Table.2 for all the positive and negative singleton system calls.

Step2: Obtain the Positive and Negative Support Sequences (S_{θ_P}, S_{θ_N}) of singleton system calls from support values of positive and negative items obtained in step-1. We can obtain positive and negative support sequences of singleton system calls as shown in Table.4 for sessions w.r.t time slots t_1 and t_2.

Positive System call , I	Positive Support Sequence, S_{θ_P}	Negative System call, I'	Negative Support Sequence, S_{θ_N}
fork	<0.60,0.40>	$\overline{\textbf{fork}}$	<0.40,0.60>
open	<0.30,0.70>	$\overline{\textbf{open}}$	<0.70,0.30>
close	<0.80,0.80>	$\overline{\textbf{close}}$	<0.20,0.20>

Table 4: Positive and Negative Support Sequence of Singleton System Calls

Step-3: Find Level -1 Similar Temporal System Call Patterns
Compute maximum possible lower bound distance and actual distance as shown in Table.5 distance for all singleton system call patterns w.r.t reference pattern sequence. Mark all temporal patterns which are similar as ✓ and dissimilar **×** .

System Call	Temporal System call Support Sequence	ULB	Actual Distance
fork	<0.60,0.40>	**0.20**✓	0.28 ×
open	<0.30,0.70>	**0.10**✓	0.14 ✓
close	<0.80,0.80>	**0.00**✓	0.45 ×

Table 5: Computation of ULB and actual distance of Singletons w.r.t Ref

Step-4: Generate Temporal Patterns of Size, |S| > 2
1. System Call Pattern: [fork, open]
Consider computation of support sequence for system call pattern [fork, open]

Here $I_i = \text{fork}$ and $I_j = \text{open}$. We may obtain [fork, open] using two ways, either using the expression

$$[\text{fork, open}]_{\text{MIN}} = \frac{1}{2}[\text{fork} + \text{open} - \overline{\text{fork}} * \text{open}_{\text{UBSTS}} - \text{fork} * \overline{\text{open}}_{\text{UBSTS}}]$$

$$= \frac{1}{2}[< 0.6,0.4 > + < 0.3,0.7 > - < 0.3,0.6 > - < 0.6,0.3 >] = <0.0,0.1>$$

$$[\text{fork, open}]_{\text{MAX}} = \frac{1}{2}[\text{fork} + \text{open} - \overline{\text{fork} *} \text{open}_{\text{LBSTS}} - \text{fork} * \overline{\text{open}}_{\text{LBSTS}}]$$

$$= \frac{1}{2}[< 0.6,0.4 > + < 0.3,0.7 > - < 0.0,0.3 > - < 0.3,0.0 >] = < 0.3, 0.4 >$$

The computation of lower bound distance of [fork, open] is shown in Table.6. In this case, both the maximum possible lower bound and Lower bound distance of system call pattern [fork, open] do not satisfy threshold, so the system call pattern [fork, open] is not temporally similar pattern.

2. System Call Pattern: [open, close]

Consider computation of support sequence for system call pattern [open, close]

Here $I_i = \text{open}$ and $I_j = \text{close}$. We may obtain [open, close] using two ways, either using the expression

$$[\text{open, close}]_{\text{MIN}} = \frac{1}{2}[\text{open} + \text{close} - \overline{\text{open}} * \text{close}_{\text{UBSTS}} - \text{open} * \overline{\text{close}}_{\text{UBSTS}}]$$

$$= \frac{1}{2}[< 0.3,0.7 > + < 0.8,0.8 > - < 0.7,0.3 > - <0.2, 0.2>$$

$$= <0.1, 0.5 >$$

$$[\text{open, close}]_{\text{MAX}} = \frac{1}{2}[\text{open} + \text{close} - \overline{\text{open}} * \text{close}_{\text{LBSTS}} - \text{open} * \overline{\text{close}}_{\text{LBSTS}}]$$

$$= \frac{1}{2}[< 0.3,0.7 > + < 0.8,0.8 > - < 0.5,0.1 > - <0.0, 0.0>$$

$$= <0.3, 0.7 >$$

Table.7 gives the computation of lower bound distance of [open, close]. Since both the ULB and LB of pattern [open, close] satisfy threshold, the system call pattern [open, close] is temporally similar.

3. System Call Pattern: [fork, close]

Similarly system call pattern [fork, close] is similar temporally as shown in Table.8

Step-5: Temporal patterns of size, |S| =3

Consider system call patterns [fork, open], [open, close], [fork, close] with reference sequence. Here ✓ and ✗ indicate satisfying and not satisfying the threshold constraint respectively i.e threshold ≤ 0.2. Since only [fork, close] and [open, close] system call

patterns satisfy the threshold constraint, and [fork, open] does not satisfy threshold constraints, system call pattern [fork, open, close] is not temporally similar.

Temporal System call Pattern	**Max-Min Bound (d1)**	**Min-Min Bound (d2)**	**Min. Bound Value**	**Decision**
$[fork, open]_{min}$ = <0.0,0.1> $[fork, open]_{max}$ = <0.3, 0.4>	- 0.2236 ✖	0 -		
Minimum Bound = d1+ d2	0.2236	0	0.2236	✖
True distance			0.32	✖

Table 6: Minimum Bound Distance of Temporal Pattern [fork, open]

Temporal System call Pattern	**Max-Min Bound** (d1)	**Min-Min Bound** (d2)	**Min. Bound Value**	**Decision**
$[open, close]_{min}$ = <0.1,0.5> $[open, close]_{max}$ = <0.3, 0.7>	- 0.1 ✓	0 -		
Minimum Bound = d1+ d2	0.1	0	0.1	✓

Table 7: Minimum Bound Distance of Temporal Pattern [open, close]

Temporal System call Pattern	**Max-Min Bound (d1)**	**Min-Min Bound (d2)**	**Min. Bound Value**	**Decision**
$[fork, close]_{min}$ = <0.4,0.2> $[fork, open]_{max}$ = <0.6, 0.4>	- 0.2 ✓	0 -		
Minimum Bound = d1+ d2	0.2	0	0.2	✓

Table 8: Minimum Bound Distance of Temporal Pattern [fork, close]

System Call Pattern	**ULB**	**LLB**
[fork,open]	✖	✖
[open,close]	✓	✓
[fork, close]	✓	✓

Table 9: Upper and lower bound distance of [fork, open, close]

So the similar temporal system call patterns are **[open], [open, close] [fork, close].** All system call patterns other than these patterns must be treated as dissimilar temporal system call patterns.

	SCP1	**SCP2**	**SCP3**	**SCP4**	**SCP5**	**SCP6**
System call patterns (SCP)	{open**, Ø** }	{fork**, Ø** }	{close**, Ø** }	{fork, open}	{fork, close}	{open, close}

Table 10: System Call patterns for sample dataset

Process	**System call patterns**						**Class**
	{fork, Ø}	**{open, Ø }**	**{close, Ø }**	**{fork, open}**	**{open, close}**	**{fork, close}**	
P1	1	0	0	0	0	0	Normal
P2	0	0	0	1	1	0	Normal
P3	0	0	0	0	0	1	Normal
P4	1	0	0	0	0	0	Normal
P5	0	0	0	1	1	0	Normal
P6	0	0	1	0	0	0	Normal
P7	0	0	1	0	0	0	Normal
P8	0	0	0	1	1	0	Normal
P9	0	0	1	0	0	0	Normal
P10	0	0	1	0	0	0	Normal
P11	0	0	0	0	1	0	Normal
P12	0	1	0	0	0	0	Normal
P13	0	0	0	1	1	0	Normal
P14	0	0	0	1	1	0	Normal
P15	0	0	1	0	0	0	Normal
P16	0	0	0	1	1	0	Normal
P17	0	0	0	0	0	1	Normal
P18	0	0	1	0	0	0	Normal
P19	0	1	0	0	0	0	Normal
P20	0	0	0	0	1	0	Normal

Table 11: Process-System Call Pattern Matrix

6 Intrusion Detection – Case Study

The temporal sequence system call patterns w.r.t sample input database are stated in Table.10 below. Table.11 shows the process system call matrix indicating similar temporal patterns w.r.t reference support sequence of interest. The process are all normal process, when considered legitimate runs.

Let the new incoming process be P_{new} = {fork, open, fork, open, close, close, close, fork}. The possible system call patterns for the process, P_{new} are listed below.

The temporal patterns existing in the sample input dataset are shown normally while the other patterns possible are shown in bold.

{fork, open}, **{open, fork},** {open, close}, **{close, close}, {close, fork}**

These patterns indicated bold are all abnormal patterns as they are not similar to reference sequence as that of the patterns {open};{open, close}and {fork, close}. The frequency of existence of these patterns in new process are as follows : **{fork, open}** – 2; {open, fork} -1; **{open, close}**-1 ;{ close, close}-2;{ close, fork}-1.

So, the new process is considered an attack process because their exists temporal patterns which are not similar to reference support sequence of interest and deviate from specified threshold value. The importance of this approach is that it eliminates the need for framing or finding decision rules using decision trees. It is also advantageous because it takes into consideration the sequence of system call patterns considering temporal behavior. Since the maximum calls per process in the sample dataset was three, we considered 2-call sequence.

Process	open	fork	{open, close}	{fork, close}	{close, fork}	{close, close}	{open, fork}	Class
P1	0	1	0	0	0	0	0	Normal
P2	0	0	1	0	0	0	0	Normal
P3	0	0	0	1	1	1	1	Attack
P4	0	1	0	0	0	0	0	Normal
P5	0	0	1	0	0	0	0	Normal
P6	0	0	0	0	1	1	1	Attack
P7	0	0	0	0	0	1	1	Attack
P8	0	0	1	0	0	0	0	Normal
P9	0	0	0	0	1	1	1	Attack
P10	0	0	0	0	1	0	1	Attack
P11	0	0	1	0	0	0	0	Normal
P12	1	0	0	0	0	0	0	Normal
P13	0	0	1	0	0	0	0	Normal
P14	0	0	1	0	0	0	0	Normal
P15	0	0	0	0	1	1	1	Attack
P16	0	0	1	0	0	0	0	Normal
P17	0	0	0	1	1	0	1	Attack
P18	0	0	0	0	1	0	1	Attack
P19	1	0	0	0	0	0	0	Normal
P20	0	0	1	0	0	0	0	Normal

Table 12: Process-System Call Pattern Matrix

So, the new process is considered an attack process because there exists temporal patterns which are not similar to reference support sequence of interest and deviate from specified threshold value. The importance of this approach is that it eliminates the need for framing or finding decision rules using decision trees. It is also advantageous because it takes into consideration the sequence of system call patterns

considering temporal behavior. Since the maximum calls per process in the sample dataset was three, we considered 2-call sequence.

In general, if the number of system calls in N, we may consider a maximum of (N-1) temporal Sequence. Consider the process shown below in Table.12 depicting process vs system call pattern. Without the need for finding decision rules, we can straight away say the process is normal or abnormal whenever we encounter a process which has abnormal temporal patterns. In this way, we can eliminate the need for finding decision rules which is space and time efficient. Further, the method aims at single database scan and avoids multiple scans when finding similar temporal patterns which are first of kind in literature.

7 Conclusions

In this work, the main idea is to represent the dataset as a time stamped process-system call database. From this representation of time stamped temporal database, we discover similar temporal system call association patterns which satisfy the specified constraints. After similar system call temporal association patterns are discovered, we obtain the process-temporal system call pattern matrix. These temporal system call patterns found are considered as the feature vector for classification process to detect possibility of intrusion. To find whether the process is normal or abnormal it is just sufficient to verify if there exists a temporal system call pattern which is not similar to the reference system call support sequence and user specified threshold. This eliminates the need for finding decision rules by constructing decision table. The importance of this approach coins out from the fact that the process of finding temporal system call patterns is a single database scan approach and eliminates extra overhead in performing multiple scans and also eliminates the need for finding decision rules (2^n is usually very large for even small value of n) and aims at efficient dimensionality reduction as we consider only similar temporal system call sequence. In future, the objective is to apply a novel similarity measure which considers temporal nature to find possibility of intrusion.

References

[Alexandr, 2002] Alexandr Seleznyov and Oleksiy Mazhelis. 2002. Learning temporal patterns for anomaly intrusion detection. In Proceedings of the 2002 ACM symposium on Applied computing (SAC '02). ACM, New York, NY, USA, 209-213.
DOI=http://dx.doi.org/10.1145/508791.508836

[Aljawarneh, 2011] Aljawarneh, S. (2011). Cloud Security Engineering: Avoiding Security Threats the Right Way. International Journal of Cloud Applications and Computing (IJCAC), 1(2), 64-70. doi:10.4018/ijcac.2011040105

[Aljawarneh, 2010] Aljawarneh, S., Alkhateeb, F., & Al Maghayreh, E. (2010). A semantic data validation service for web applications. Journal of theoretical and applied electronic commerce research, 5(1), 39-55.

[Asaf, 2010] Asaf Shabtai, Uri Kanonov, and Yuval Elovici. 2010. Intrusion detection for mobile devices using the knowledge-based, temporal abstraction method. J. Syst. Softw. 83, 8 1524-1537. DOI=http://dx.doi.org/10.1016/j.jss.2010.03.046

[David, 2006] David J. Malan and Michael D. Smith. 2006. Exploiting temporal consistency to reduce false positives in host-based, collaborative detection of worms. In Proceedings of the 4th ACM workshop on Recurring malcode (WORM '06). ACM, 25-32. DOI=http://dx.doi.org/10.1145/1179542.1179548

[Guiling, 2005] Guiling Zhang. 2005. Applying mining fuzzy association rules to intrusion detection based on sequences of system calls. In Proceedings of the Third international conference on Networking and Mobile Computing (ICCNMC'05), 826-835. DOI=http://dx.doi.org/10.1007/11534310_87

[Hongpei, 2001] Hongpei Li, Lianli Chang, and Xinmei Wang. 2001. A Useful Intrusion Detection System Prototype to Monitor Multi-processes Based on System Calls. In Proceedings of the Third International Conference on Information and Communications Security (ICICS '01),Springer-Verlag, London, UK, UK, 441-450.

[Hsu, 2011] Hsu, W. F. (2011). A server side solution to prevent information leakage by cross site scripting attack.

[Joao, 2001] Joao B. D. Cabrera, Lundy Lewis, and Raman K. Mehra. 2001. Detection and classification of intrusions and faults using sequences of system calls. SIGMOD Rec. 30, 4 25-34. DOI=http://dx.doi.org/10.1145/604264.604269

[Jones, 2001] A. Jones and S. Li. 2001. Temporal Signatures for Intrusion Detection. In Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC '01). IEEE Computer Society, Washington, DC, USA, 252.

[Kyubum, 2006] Kyubum Wee and Sinjae Kim. 2006. Construction of finite automata for intrusion detection from system call sequences by genetic algorithms. In Proceedings of the 10th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining (PAKDD'06), Springer-Verlag, Berlin, Heidelberg, 594-602.

[Mohammad, 2009] Mohammad Akbarpour Sekeh and Mohd. Aizani Bin Maarof. 2009. Fuzzy Intrusion Detection System via Data Mining Technique with Sequences of System Calls. In Proceedings of the 2009 Fifth International Conference on Information Assurance and Security - Volume 01 (IAS '09), Vol. 1. IEEE Computer Society, Washington, DC, USA, 154-157. DOI=http://dx.doi.org/10.1109/IAS.2009.32

[Pistoia, 2015] Pistoia, M., Segal, O., & Tripp, O. (2015). U.S. Patent No. 8,984,642. Washington, DC: U.S. Patent and Trademark Office.

[Qiang, 2005] Qiang Duan, Chenyi Hu, and Han-Chieh Wei. 2005. Enhancing network intrusion detection systems with interval methods. In Proceedings of the 2005 ACM symposium on Applied computing (SAC '05), Lorie M. Liebrock (Ed.). ACM, New York, NY, USA, 1444-1448. DOI=http://dx.doi.org/10.1145/1066677.1067006

[Radhakrishna, 2015] V.Radhakrishna, P.V.Kumar, V.Janaki. "An Approach for Mining Similarity Profiled Temporal Association Patterns Using Venn Diagrams", Proceedings of ACM ICEMIS 2015, Aug 24th-26th, Istanbul, Turkey.

[Shi, 2007] Shi Pu and Bo Lang. 2007. An intrusion detection method based on system call temporal serial analysis. In Proceedings of the intelligent computing 3rd international conference on Advanced intelligent computing theories and applications (ICIC'07), Springer-Verlag, Berlin, Heidelberg, 656-666.

[Vangipuram, 2013] Vangipuram Radhakrishna, C.Srinivas, C.V.GuruRao. Document Clustering Using Hybrid XOR Similarity Function for Efficient Software Component Reuse. Procedia Computer Science, 2013; (17): 121-128.

[Vangipuram, 2014] Vangipuram Radhakrishna, Chintakindi Srinivas, and C. V. GuruRao. 2014. A modified Gaussian similarity measure for clustering software components and documents. In Proceedings of the International Conference on Information Systems and Design of Communication (ISDOC '14). ACM, New York, NY, USA, 99-104. DOI=http://dx.doi.org/10.1145/2618168.2618184

[V.R, 2015] V.Radhakrishna, P.V.Kumar, V.Janaki. "A Temporal Pattern Mining Based Approach for Intrusion Detection Using Similarity Measure", Proceedings of ACM International Conference on Engineering and MIS 2015, Aug 24th-26th , 2015

[V.Radhakrishna, 2015] V.Radhakrishna, P.V.Kumar, V.Janaki. "An Approach for Mining Similarity Profiled Temporal Association Patterns Using Gaussian Based Dissimilarity Measure", Proceedings of ACM ICEMIS 2015, Aug 24th-26th, Istanbul, Turkey.

[Xin, 2005] Xin Xu and Tao Xie. 2005. A reinforcement learning approach for host-based intrusion detection using sequences of system calls. In Proceedings of the 2005 international conference on Advances in Intelligent Computing - Volume Part I (ICIC'05), Vol. Part I. Springer-Verlag, Berlin, Heidelberg,995-1003

[Xin, 2010] Xin Xu. 2010. Sequential anomaly detection based on temporal-difference learning: Principles, models and case studies. Appl. Soft Comput. 10, 3 (June 2010), 859-867. DOI=http://dx.doi.org/10.1016/j.asoc.2009.10.003

[Xinguang, 2010] Xinguang Tian, Xueqi Cheng, Miyi Duan, Rui Liao, Hong Chen, and Xiaojuan Chen. 2010. Network intrusion detection based on system calls and data mining. Front. Comput. Sci China 4,522-528.DOI=10.1007/s11704-010-0570-9