Hybrid Approach to System call Pattern Based Anomaly Detection

Kulkarni Sagar¹, Kahate Sandip²

¹Student, Department of Computer,
²Assistant Professor, Department of Computer,
SPCOE, Oitur, Pune, Maharastra, India,

Abstract— As computer networks is the most rapidly growing area of interest more efforts are placed to improve computer security. Hackers or intruders are increasing day-by-day to find the system vulnerabilities that are result of design flaws of software and to target your system using different contemporary techniques to steal or damage private and valuable data. Although a large number of tools are available for preventing damage to your system from network threats, it is very difficult to protect your system. So there is a need to have a tool that is capable of detecting attacks that arises from inside or outside network and thus IDS is an absolute choice of many people. Intrusion detection system can be Host-based or Network-based. Many researches on Host based IDS have done but they failed to achieve better detection rate with corresponding reduced false alarm rate. Here we propose host-based anomaly intrusion detection methodology using semantic theory on modified definition of word and use rarity index to overcome problem of incomplete training traces. Applying semantic theory to system calls sequences in order to reflect activities hidden in high-level programming languages which help to understand program anomaly behaviour. Enhancing definition of word reduces training time. Our decision is based on average mismatch rate and rarity of patterns. For evaluating our proposed system we use publicly available data sets like DARPA 98 and if possible UNM data set and/or ADFA-LD data set.

Keywords— Anomaly Detection, Intrusion Detection, System call, Semantic Theory.

I. INTRODUCTION

New computing devices such as smart phones, laptops, etc are available at an affordable price and easy to use. Even this development in information and telecommunication technology have many benefits, they also have pitfalls because it has given boost to malicious activity. There are number of readymade hacking tools, scripts available on internet which are easily accessible by normal computer user which could be used to exploit remote computer security. Recent malicious activities such as leakage of actress nude pictures from cloud in 2014, Sony play Station data theft [4] which steals number of username, password, and credit card details, recent hacking activities on Indian embassy web sites are some examples.

There are security measures such as Firewalls, access control lists, and encryption. Although a large number of tools are available for preventing damage to your system from network threats it is very difficult to protect your system. Hackers or intruders are increasing day-by-day to assess the vulnerabilities in your system for targeting your system. There are number of malicious events which are unknown to security investigators because Unknown attack are the most difficult to detect as there is no knowledge of such attack to discover. Security researchers attempt to limit the number of undiscovered zero-day attacks by actively working to locate and patch the vulnerabilities before black hat hackers take advantage of them, but the success of these efforts is debatable.

No system is totally foolproof. However we can try to detect these intrusion attempts so that action may be taken to repair the damage [7]. If there are attacks on a system, we must detect them as soon as possible otherwise serious damage could happen. Here the intrusion detection comes into the picture. Intrusion detection system is of two types Host-based or Network-Based. Many researches have been done on both these types but especially Network-based IDS [9].But Network based IDS have limitations as because data is mostly encrypted and it is not possible to check every packet on network. There are certain attacks that could be detectable only by analysing host computers. This motivates us towards host based IDS.

This paper introduces a new model of host based anomaly detection based on system call patterns. Here we apply semantic theory proposed by Creech [4] to understand the higher level programming activities with modified definition of word. Secondly we solve the problem of incomplete
training traces using rarity index based statistics. Our phrase generation is also controlled rather than uncontrolled as in paper [4] which decreases training time of proposed system. The decision of anomaly is taken on the basis of rarity index measure of phrases and mismatch. For validating and evaluating out proposed system we use publically available datasets like DARPA 98 and UNM. Even though DARPA 98 is old dataset which does not cover recent attacks traces but it will help us to compare our performance with other system [4, 13].

The rest of paper is organized as follows: Section 2 covers literature review. Section 3 presents algorithm and theories which supports proposed system. Section 4 contains design details and assumptions and Section 5 contains concluding remarks.

II. LITERATURE

We can consider Intrusion Detection System (IDS) as similar to a burglar alarm which is configured to monitor access points and hostile activities. It generates an alarm whenever someone enters into the system does some nonsense activity. An IDS is a specialized tool that knows how to read and interpret the data they are analysing [9].

IDS is categorized into two main categories either Host-based or Network-Based based on data they interpret [9]. There are other categories like Distributed-IDS, Hybrid-IDS but they are derived categories. Both these IDS are categorized based on the technique they use to detect intrusive activity. These techniques are misuse based and anomaly based. Misuse based IDS utilizes the threat signature database to detect intrusions. They provides zero false positive alarm but having no capability to detect attacks whose signature does not exists in database we call them as "zero-day attacks" [1, 9]. While anomaly detection works by specifying normal behaviour and any large deviation from this normal behaviour is considered as an intrusion. Anomaly detection have capability to detect previously unseen attacks as opposed to misuse detection [7]. Even though anomaly detection approach is promising it suffers from two problems first is high training overhead and second is high FPR. Training overhead is due to time required to analyse normal behaviour of a system using different techniques. High false positive rate occurs because of either training is given on incomplete normal profile or selected baseline. Even though large number of researches shows that it is possible to reduce high FPR to manageable level with increased detection rate.

HIDS and NIDS is implemented by choosing information that they will analyse for taking decision. NIDS uses network statics, network packet header, etc [9]. On the other hand HIDS uses information available in host computer like log files, system calls, CPU usage, etc [4]. HIDS using log files have been tried with some level of success but they suffers from three problems firstly log files represents data that is generated by demon programs which is diluted [4]. Secondly managing log files is difficult because they grow very large and finally attacker can erase its footprints from log files [4].

Next source of information is system call sequences which is proved as trustable source of information [1]. As system call is a entry point into kernel, any activity of process can be recorded for further analysis of intrusion detection also attacker can not erase its footprint of system call traces [2]. They do not presents problem like in log file analysis. This is firstly proposed by Forrest in her work [1]. After Forrest [1, 2] work many researches have been conducted on HIDS using system call pattern analysis to detect intrusion with reasonable success.

Earlier research on system call pattern analysis was done by Forrest et.al [1] here she proposed a model in which normal profile of program is collected using sliding window on program traces and builded database normal profile. In testing time fixed sequences are extracted using sliding window and compared with database. If a match is found then it is called as normal sequence otherwise a mismatch is counted. When a mismatch count grows beyond the threshold she flagged that sequence as anomalous. She observed that short sequences are good discriminator of program behaviour. In paper [2] Forrest et.al proposed extension to earlier work [1] where she modified
the definition of mismatch that considers only local mismatch count according to her observation that anomaly occurs in burst rather than entire system call trace. To compute the similarity between two sequences “Hamming distance” is used.

Warrender et al [23] Proposed extension to tide model proposed by Forrest et.al. He introduced a method of simple enumeration of observed short system call sequence time delay embedding. In this method profile of normal behaviour is build by enumerating all unique, contiguous sequences of predetermined length l that occurred in training system call traces. He created database of normal profile with same methods proposed by Forrest [1]. At testing time, sequences from the test system call trace are compared to those in the normal profile. Any sequence not found in the profile is called mismatch which could indicate anomalous behaviour. The larger the number of mismatches a test trace has, the more likely the trace is anomalous. For each locality frame the anomaly signal is defined as the number of mismatches observed within this frame. They call this quantity a Locality Frame Count (LFC). The larger the LFC is more likely the corresponding locality frame is part of an anomaly. They set threshold on the LFC. If the LFC reaches or exceeds the threshold then corresponding locality frame is anomalous. The larger the threshold is the less intrusion it detects causing false negative alarms [3].

In [4] Warrender et al proposed extension to “STIDE” using frequency threshold called "t-STIDE”. He proposed that a proper criterion for accessing rarity of system call is to measure the frequency of system calls. This method keeps track of how often each system call sequence in the normal database appears in the training data. Once all the training data have been processed each sequence's overall frequency can be determined. As for STIDE model, sequence from test traces is compared to those in normal database. But in t-STIDE rare sequences are counted as mismatches in t-STIDE model. In their experiment, rare is defined as any sequence accounting for less than 0.001% of the normal training data. These mismatches are aggregated into locality frame count as in STIDE model. The LFCs are used as anomaly signal of the locality frame of system calls. Warrender et.al [4] reports that t-STIDE consistently under-performs in comparison to competing methods. This is due to their definition of rarity [4].

Wespi et.al [11] proposed novel approach which uses variable length patterns for creating database of normal profile. He addresses issue of fixed length patterns and observed fixed length patterns are not good discriminator for modelling normal profile. He used a novel algorithm called Teiresias for building normal profile of program. The normal database consists of longest variable length patterns that can model normal profile of program. While in experimental setup he chooses maximum pattern length as 20 because longer sequences of system call longer sequences are not representative true nature of corpus. He counts number of mismatches occurred and if the number of mismatches is beyond the threshold he flagged that sequence as anomalous.

Wen-Hu Ju and Vardi [3] proposed a model based on rarity-of-occurrence index of short commands and system call sequence for profiling user behaviour and processes. They measure rarity of short sequences based on frequent usage. They take individual commands (or system call) as well as transition between them to formulate the scores. They consider fixed length short sequences of command as "Local Command Pattern". They broke the command sequence into several distinct LCP and assigns scores to each LCP based on its rarity. These scores of LCP are then aggregated to find test sequence is normal or anomalous [3]. They applied the concept of rarity on STIDE [6] model. According to them frequency based measure is not suitable for defining rarity of system call sequences as some processes a few sequences contribute for large portion of training data. Therefore frequencies of other normal sequences tend to look insignificant and can be flagged anomalous. Rarestness of system call sequence is determined by number of traces containing it in training data.

G. Creech [3] proposed a model based on semantic system call analysis using dis-contiguous system call analysis where they formed word using sliding window of different length and based on these words they formed phrases with their count.
Phrases with their count are given to decision engine which in turn gives result. They applied different decision engines on these features and gets promising results. But their training time is so large and it is not feasible to deploy in online environment. Also their definition of word has problems.

A k-Nearest Neighbour (k-NN) classifier based approach is introduced by Liao [10] to classify program behaviour as normal or intrusive. Program behaviour, in turn, is represented by frequencies of system calls. Each system call is treated as a word and the collection of system calls over each program execution as a document. These documents are then classified using k-NN classifier, a popular method in text categorization. Ye [16] attempts to compare the intrusion detection performance of methods that used system call frequencies and those that used the ordering of system calls. The names of system calls were extracted from the audit data of both normal and intrusive runs, and labelled as normal and intrusive respectively. Since both the frequencies and the ordering of system calls are program dependent, this over simplification limits the impact of their work.

Other approaches considers modelling system call traces on process-based intrusion detection involves Finite State automata (FSA). In paper [17], a high-level specification language called Auditing Specification Language (ASL) is developed for specifying normal and abnormal behaviours. These two approaches both are misuse detection, which have the same shortcomings lie in hard to create compact models of attacks and vulnerable to novel attacks. Ref. [18] uses data-mining technique, called RIPPER, to generate associate rule of normal profile. Unlike most researchers who concentrated on building individual program profiles, Asaka [18] introduces a method based on discriminate analysis. Without examining all system calls, an intrusion detection decision was made by analysing only 11 system calls in a running program. But its feasibility needs to be tested. In paper [15] combined approach of intrusion detection using system call sequence with their argument analysis is performed. Using system call sequence with argument increases accuracy but complexity analysis also increases.

Our approach combines different theories to build normal profile of a process and decision is taken collectively based sequence popularity and mismatch occurred while forming these phrases. We also address the problem of incomplete training by varying threshold for observed sequences. We reduce training time and can be used in online environment. Following section gives theories used to implement proposed system.

III. PRELIMINARIES

A. Use of Semantic Theory

Semantic theory provides various methods for parsing programming languages, and allows for a robust approach to computer system design [4]. According to semantic theory, sets of rules can be used to describe sequences of patterns. From a natural language perspective, these rules formulate grammar of the language and allow the construction of valid sentences within that rule set. Semantic theory suggest that system call traces which are formed by execution of programs written in higher language can be analysed semantically means inside system call traces there must be semantic relationship [4]. These grammars can also be used to identify invalid, or ungrammatical, sentences. And it is this fact which suggests their applicability to intrusion detection systems.

Semantic theory defines two key classes, namely terminating semantic units and non-terminating semantic units. These definitions are easily applied to human language, where a terminating unit is usually synonymous with a "word" and a non-terminating unit is usually quoted as a "sentences" [4]. The field of semantic analysis allows for many different types of grammar, the simplest of which is a CFG. This type of grammar is most commonly used for computer systems, in applications ranging from programming language development to complex artificial reasoning. The extension of semantic theory to system call analysis considers the unique system calls as letters, a sequence of system calls as word, and collections of system call sequences as phrases. Under the structures of a
CFG unique system calls become terminating units and words become non-terminating units.

B. Use of Semantic Theory

According to this model the system call sequence is said to be as popular if it encounters in “vocabulary” of most training traces. In contrast a system call sequence is "rare" if it is in the vocabulary of few traces and "unique" if it is in vocabulary of only one trace. According to Wen-Ju Hu & Vardi [3] the user’s action is limited to particular frequent command sequence which is popular in community. We can use this theory to tackle with the problem of incomplete training traces. Earlier work tries profile normal behaviour based on training traces but practically we know training samples are limited which does not cover all normal behaviour of user. Frequency based measure is not suitable for defining rarity of system call sequences as some processes a few sequences contribute for large portion of training data. The theory helps us take decision of anomaly. Here we hypothesized that if the sequence of previous sequence of words are popular then the mismatch rate between words should be small and if the previous word sequence is rare then there is high probability that next sequence will be either rare or mismatch. This conclusion is derived from the fact that is user knows rare command in operating system then the probability of next command to be rare increases. So we must set different thresholds for popular and rare sequences. The rarity is may be because of incomplete training so we increase the mismatch threshold with respect to rarity. 

Rarity index is defined as,

\[ \delta(G, U) = \begin{cases} \frac{1}{2} \sum_{i=1}^{U} I_{R_{m_{i}}}, & \text{if } \sum_{i=1}^{U} I_{R_{m_{i}}}(C) \geq 0 \\ 1, & \text{if } \sum_{i=1}^{U} I_{R_{m_{i}}}(C) = 0 \end{cases} \]

Where I is indicator function,

\[ \delta(I_{R_{m_{i}}}(C)) = \begin{cases} 1, & \text{if } C \text{ is one of the subsequence of } R_{m_{i}} \\ 0, & \text{if } C \text{ is not one of the subsequence of } R_{m_{i}} \end{cases} \]

Smaller the \( \delta(C) \) is the more training traces contain the sequence and less abnormal the sequence is. Negative rarity index indicates that the sequence has been seen in training traces while positive sequence indicates that the sequence is new and can be abnormal [3].

C. Definition

1. Let \( T=\{ \text{architecture specific system calls} \} \).
2. Let \( N=\{ \exists w \in T : y = w_{i} ; w_{j} ; w_{k} \ldots \} \) or in other words \( N=\{ \text{variable length words that can represent normal profile} \} \)
3. Consider \( G \) represents a known normal trace and \( A \) represents a known anomalous trace.

Here \( T \) represents terminating units and \( N \) represents non-terminating syntactic units.

D. Syntactic Development

Assuming context free grammar we can define syntax of System call trace as,

\[ \exists w \in T : G \rightarrow wG' \]

Using production rule 1, a sequence of training data \( \{ G_{0}, G_{1}, \ldots \ldots, G_{N} \} \) yields new set as, \( B = \{ \exists w_{i}, w_{j} ; w_{k} \ldots \ldots, w \in T | w_{i}, w_{j} ; \ldots, \in [G_{0} \ G_{1} \ldots \ldots G_{N}] \} \). This set \( B \) represents phrases created by sequence of system calls. All the phrases formed using this grammar theoretically.

E. Semantic Hypothesis

The following inequality was hypothesised as the core element here, where \( G_{n} \) represents previously unseen normal trace:

\[ | \{ \exists w \in B : G_{n} \} | \geq | \{ \exists w \in B : A \rightarrow xA' \} | \]

This equation indicates that occurrence of valid semantic units extracted from new normal trace should be greater the and practically contain vast amount of phrases but we control the phrase generation by using generating only those traces those are observed training samples.

IV. IMPLEMENTATION

Proposed system architecture is shown in Fig.1. There are two improvements over last research done by G. Creech [4], firstly we are constructing words using Teiresias algorithm which will improve the definition of word and our method of phrase generation is controlled which results in less training overhead. The proposed system work in three steps, firstly normal variable length pattern database we call it as word dictionary is generated. Secondly phrase generation from these words with corresponding phrase rarity index count is done and finally decision is taken on the basis of mismatch count and rarity index. Following section gives detailed description proposed methodology.
F. Constructing Variable Length Words

Constructing variable length words is done by collecting system calls traces some process being monitored and extracting patterns with Teiresias. Teiresias is pattern matching algorithm initially developed for discovering rigid patterns in unaligned biological sequences.

Suppose we observe the following system call 
\[ S = \{ s_a, s_b, s_c, \ldots, s_n \} \]
means consider one system call such as `lseek`, `read` etc. Initialize sliding window length \( k \) equals 2, and take out the iterant sequences, we get some patterns whose length is 2. Recodes the frequency of each pattern and add it into word dictionary. If \( k \) is smaller than \( n \) \( (n \) is the length of \( S \)) or a predefined value which is bigger than 2), initializes sliding window length \( k \) equals \( k+1 \), and repeats the extraction steps.

Here, we use \( Q \) to denote the patterns set. Which means \( p_j \) is the substring of \( q_i \). If the sequence of \( p_j \) is smaller than or equal to the sequence of \( q_i \), then we delete \( p_j \) from the normal database, and add \( q_i \) in it. Repeats the steps until all the patterns in \( Q \) are included. The extraction \( A \) stand for \( S_A \), \( B \) stand for \( S_B \), etc. Only the processes of \( k=3 \) and \( k=4 \) are described, while others are omitted.

Changing word definition reduces word dictionary size and creates exact definition of word. Again modifying definition of word is important because many programs are written in higher language which uses library functions of underlying language. This means whenever library function is called creates similar system call pattern with little or no variation. Even when program starts and finished execution it generates similar sequence of system call patterns with small variation. Using these unique words it is possible to build entire normal profile. If we considers each word as a function executed by process then using sequence of words we get information about the transition between them which is stronger representation of process behaviour.

The unique words generated may consist of long sequence of system calls which contains more semantic information but it poses some difficulties as it increases normal database size and manipulation becomes difficult. Database size reduces by using Lower patterns but such patterns have less semantic information. So the maximum word length allowed is determined by experiment.

G. Construction of Phrases

In second step we iterate through training traces and construct phrases consisting of one word up to \( l \). Here \( l \) represents the size of sliding window. The choice of \( l \) has importance because as stated longer phrases contain more positional information about word we call it as semantic information. The phrases that are building are controlled rather than uncontrolled as in paper [1] which tries to generate every possible phrase using words dictionary. This improvement reduces training time and stores phrases up to given length. Here we slide window of length \( l \) to gather consecutive words and stores them in phrases dictionary while doing so we also record the occurrence count of each phrase. We should also consider that longer phrases are not representative of true nature of corpus so we must keep the value of \( l \) to manageable level.

H. Decision Engine

In most of the previous works the decision of system call trace is taken using only one parameter like Forrest et.al [1] uses number of mismatches in
last $N$ sequences [10], Wen [11] uses definition of rarity of system call sequence to flag the sequence as anomalous if the threshold exceeds. But the problem with their approaches is that the above concepts gives false alarm when a new unknown trace is given because training samples may not containing such sequences.

Here our decision is based on two parameters first one if the mismatch rate in phrase of length $L$, second parameter considers rarity of sequences in phrases. There is theoretical reason for considering for considering average number of mismatches in phrases in trace because we must consider the fact that training samples may not contain every possible path of execution. So when such normal trace is given then it would contain $L$ word $M$ number of mismatches. Using mismatch count alone may give false alarm so to take decision we determine average number of mismatch words this gives us how much word mismatches occurred in last $k$ phrases. Then second parameter gives insights of how popular the given sequence is? If the given sequence is popular then mismatch rate should be low. Because mostly users use the same popular path while using programs. Our hypothesis is that when a anomalous trace is given then it would result in high average number of mismatches locally, high rarity of word sequences.

For evaluating intrusion detection system we are using DARPA 98 intrusion detection dataset. There are two parts of these data set first one is online and second one if offline. We choose offline evaluation. The system call traces are present inside BSM file. There is seven weeks of training data and 2 weeks of testing data. But the testing data is unlabeled so for evaluation purpose we use 7 weeks of training data for both training and testing purpose. We train our system using normal traces of each single process and test our detection rate. Experimental evaluation will provide proof to our proposed concept.

V. ALGORITHM

1: procedure GETINITIALWORDS(syscalltraces)  
2: for all traces do  
3: counter = 1
4: initWordDic ←empty
5: for system call in traces do  
6: word = syscall + nextcountercalls  
7: if word is in initWordDic then  
8: increment count of word  
9: else  
10: add word to initWordDic  
11: end if  
12: if counter > M + 1 then  
13: exit  
14: end if  
15: end for  
16: end for  
17: return initWordsDic  
18: end procedure

1: procedure CALLTEIRESIAS(initWordDic)  
2: for all words in initWordDic do  
3: currentWord = word  
4: if currWord is subString of anyotherWord then  
5: currWordCnt ←count  
6: otherWordCnt ←count  
7: if currWordCnt <= otherWordCnt then  
8: delete currWord from initWordDic  
9: end if  
10: end if  
11: end for  
12: return initWordDic  
13: end procedure

1: procedure GETPHRASES(syscalltraces,initWordDic)  
2: for all traces do  
3: counter=0  
4: wordPos←0  
5: for words in trace do  
6: word = +nextcounterwords  
7: if phrase in phraseDictionary then  
8: increment count of Phrase  
9: else  
10: add phrase to phraseDictionary  
11: end if  
12: counter = counter + 1  
13: if counter > MAXW then  
14: exit  
15: end if  
16: end for  
17: wordPos=wordPos + 1
18: end for
19: calculate Rarity-Index of all Phrases
20: return phraseDictionary
21: end procedure

VI. CONCLUSION

During this work different Intrusion were studied. The objective of this study was to improve performance of intrusion detection and at the same time try to reduce false alarms and training time present in traditional approaches. Using semantic theory with modified definition of word and controlled generation of phrases reduces heavy training overhead. Adding the rarity theory solves problem of incomplete database. The decision using two parameters will likely solve the problem of FPR. Public dataset will help to understand effectiveness of our strategy.

ACKNOWLEDGMENT

The authors would like to acknowledge the support from Assistant. Prof. Khatavkar V.K., COEP, Pune.

REFERENCES