Frequent Element Pattern Matching To Evade Deep Packet Inspection in NIDS

Pallavi Dhade, T.J.Parvat

Abstract*— To detect hostile traffic in network segments or packets , Signature based Network Intrusion Detection Systems (NIDS) uses a set of rules which are so effective in detecting anomalous behavior like known attacks that hackers look for new techniques to go unobserved. Some of the techniques involves, in the manipulations of obscurities of network protocol. At the present, the detection techniques are developed against most of these elusive and equivocal techniques by means of identifying and recognizing. The appearance of new elusive forms may possibly effect NIDS to be unsuccessful. This paper presents an innovative functional framework to perform modeling over NIDS. Main, NIDS demonstrated precisely through Apriori algorithm. At this point, the paper consists of watching for circumventions on models are simpler and easier than directly trying to understand the behavior of NIDS. We present a proof of concept showing how to perform deep packet inspection in NIDS using two publicly available datasets. This framework can be used for analyzing ,Modeling and detecting the commercial NIDS after elusion.

***Index Terms*—Apriori Algorithm, Deep packet inspection, Network Intrusion Detection systems, frequent elements matching, High speed network**

I. INTRODUCTION

Computer security is defined as the protection of computing systems against threats to confidentiality, integrity, and availability [2]. The goal of Confidentiality (or secrecy) is that information is disclosed only according to policy, integrity means that information is not destroyed or corrupted and that the system performs correctly, availability means that system services are available when they are needed. Computing system refers to computers, computer networks, and the information they handle. Security threats come from different sources such as natural forces, accidents, failure of services (such as power) and people known as intruders. The categories of intruders are the external intruders who are unauthorized users of the system they attack, and internal intruders, who have permission to access the system with some restrictions.

The traditional prevention techniques such as user authentication, data encryption, avoiding programming errors and firewalls are used as the first line of defense for computer security. To manage the huge amount of personal, public and critical data we always choose the Information Technology systems, which become a critical component in organizations Identifying such a anomalous behavior and hostile actions, for protecting those systems, which is one of the most important goal in security. Intrusion Detection Systems (IDS) are software or hardware tools that automatically examine, check and observe events that take place in a computer or a network, looking for indication of intrusion [1].

Manuscript Received on July 08, 2012.

Pallavi Dhade, Department of Computer Engineering, Sinhgad Institute of Technology, University of Pune, Pune, India.

Prof. T. J. Parvat, Department of Computer Engineering, Sinhgad Institute of Technology, University of Pune, Pune, India

Network Intrusion Detection Systems (NIDS) just analyze network traffic captured on the network segment. NIDS may look for either anomalous activity (anomaly they are installed. NIDS may seek for either anomalous activity (anomaly based NIDS) or known hostile patterns (signature based NIDS) on the network. Firewalls they do not normally block packets, but aware about the intrusion alarm. This situation focuses on elusions over the signatures of these systems. There are some problems in network protocols that create state where endpoint systems process the packets generates a different demonstration of data in the NIDS and in the end system. If the representation of the data in the NIDS and in the end systems are different then the evasion is successful. Sometimes it is not possible to detect the attacks. Thus, a possible appearance of new elusive techniques would be critical for systems that are supposed to be secure. This is the inspiration and motivation of the work, in which a new approach to watch for elusions over NIDS, giving a proof of concept showing how to perform deep packet inspection in NIDS using two publicly available datasets. This framework can be used for analyzing and Modeling and detecting the malicious behavior in the commercial NIDS.

The main methodology to model the NIDS behavior by means of Genetic programming. The goal of framework is to look for new evasive techniques by analyzing NIDS behavior. Genetic programming is a branch of genetic algorithms. The main difference between genetic programming and genetic algorithms is the representation of the solution. Genetic programming creates computer programs in the lisp or scheme computer languages as the solution. Genetic algorithms create a string of numbers that represent the solution. Genetic programming, uses four steps to solve problems:

1) Generate an initial population of random compositions of the functions and terminals of the problem (computer programs). 2) Execute each program in the population and assign it a fitness value according to how well it solves the problem. 3) Create a new population of computer programs. i)Copy the best existing programs ii)Create new computer programs by mutation. iii)Create new computer programs by crossover 4) The best computer program that appeared in any generation, the best-so-far solution, is designated as the result of genetic programming . In last work, GP was used to model simple NIDS with great accuracy, using publicly available dataset. This paper mainly deals with the elusion over the NIDS and the efficiency of modeling the NIDS with Apriori algorithm using available sets for commercial use.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

II. STATE OF ART

There are some ambiguities in network protocol, which permit different systems to implement them in a different way. An elusion is successful when DPI is not paying attention to packets which are going to be processed on the endpoints system and vice versa. For example, if the TCP or IP packet contains some erroneous field, that protocol does not have an idea what to do with those packets. TCP protocols either ignore or accept or reject those packets. As shown in Figure 1, an elusion could successful if the NIDS implementation of the TCP protocol differs from the endpoint system implementation[1].

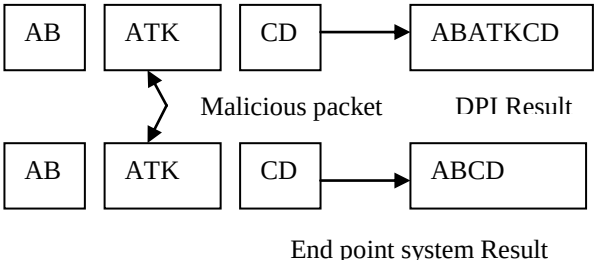

Fig 1: Elusion example

In this example, the DPI preprocessor accepts the packet containing a malicious field, while the endpoint does not, so the final structure after the preprocessing phase will be different. Several tools have been implemented to exploit the properties exposed above. Examples are Fragroute [4], which intercepts network traffic and modifies the packets before forwarding them to their destination, and idsprobe [7], which is a tool that generates traffic data from original traces. Many techniques have been designed to prevent evasions. Most of them are based on network traffic modification, to remove the ambiguities and establish a common understanding of the protocols for DPI and endpoints. Watson et. al [5] propose a system called Protocol Scrubbing that generates well formed

TCP data from traffic. Our goal is to first model the NIDS then perform the evasion. AdaBoost is the algorithm for constructing a ”strong” classifier as linear combination,

$$ht(x) = \sum_{t=1}^{T} \alpha t\, ht(x)$$

of “simple” “weak” classifiers ht(x).

III. TOOLS USED FOR EVADING PURPOSE

There are different tools used for performing the evasion over Deep packet inspection in NIDS. In this there are two steps , the first is the identification in the form of data labeling ,data testing and creation of decision tree. Here the weka tool is used for the purpose of creation of the decision tree. There are mainly tools used weka is mainly responsible for the creation of the decision tree. Here we can identify that how many parameters used for the creation of the decision tree. For this the KDDcup99 is firstly pass to the weka tool, which consists of signatures of all the viruses and attack which mainly occurs in all protocol. So , by identifying the different parameter the decision tree can be created , but in this the focus is on the ICMP , where mainly the signatures of the ICMP are identified, by using it we can detect the viruses and attack. Now ,the job of Adaboost algorithm comes which perform the data labeling and data testing phases. Here we get that the data is normal or abnormal means malicious or non-malicious. Now the second part of our system starts where we first provide an ICMP text file to our system, then we apply the Apriori algorithm for generation of the rules, here for this the support and confidence is needed to provide. Then our normal NIDS for example(Snort) starts and it perform the detection. Now some of the signatures are identified by the Snort, but there are some ambiguities in that where we can perform the evasion. Means here the deep packet inspections is evaded by our system. For this we first close the Snort. The we are going to perform evasion. Here we again generate the new rules for the evasion. After evasion we are again start Snort, so the snort is fail to detect some of the signatures over which we have provided evasion. Here the evasion is successful. But now the work of our system starts, where though it is not identified by the Snort, but our system should be able to detect those signatures. Thus , the ultimate aim of our system. Some of the algorithms and tools which are used as :

A. C4.5 Algorithm

Systems that construct classifiers are one of the commonly used tools in data mining. Such systems take as input a collection of cases, each belonging to one of a small number of classes and described by its values for a fixed set of attributes, and output a classifier that can accurately predict the class to which a new case belongs. These notes describe C4.5, a descendant of CLS and ID3 . Like CLS and ID3, C4.5 generates classifiers expressed as decision trees, but it can also construct classifiers in more comprehensible ruleset form. We will outline the algorithms employed in C4.5, highlight some changes in its successor See5/C5.0, and conclude with a couple of open research issues.

Decision trees

Given a set S of cases, C4.5 first grows an initial tree using the divide-and-conquer algorithm
as follows:

• If all the cases in S belong to the same class or S is small, the tree is a leaf labeled with the most frequent class in S.

• Otherwise, choose a test based on a single attribute with two or more outcomes. Make this test the root of the tree with one branch for each outcome of the test, partition S into corresponding subsets S1, S2, . . . according to the outcome for each case, and apply the same procedure recursively to each subset.

There are usually many tests that could be chosen in this last step. C4.5 uses two heuristiccriteria t rank possible tests: information gain, which minimizes the total entropy of the subsets {Si } (but is heavily biased towards tests with numerous outcomes), and the default gain ratio that divides information gain by the information provided by the test outcomes. Attributes can be either numeric or nominal and this determines the format of the test outcomes. For a numeric attribute A they are $\{A \le h, A > h\}$ where the threshold h is found by sorting S on the values of A and choosing the split between successive values that maximizes the criterion above. An attribute A with discrete values has by default one outcome for each value, but an option allows the values to be grouped into two or more subsets with one outcome for each subset.

The initial tree is then pruned to avoid overfitting. The pruning algorithm is based on a pessimistic estimate of the error rate associated with a set of N cases, E of which do not belong to the most frequent class. Instead of E/N, C4.5 determines the upper limit of the binomial probability when E events have been observed in N trials, using a user-specified confidence whose default value is 0.25. Pruning is carried out from the leaves to the root. The estimated error at a leaf with N cases and E errors is N times the pessimistic error rate as above. For a subtree, C4.5 adds the estimated errors of the branches and compares this to the estimated error if the subtree is replaced by a leaf; if the latter is no higher than the former, the subtree is pruned. Similarly, C4.5 checks the estimated error if the subtree is replaced by one of its branches and when this appears beneficial the tree is modified ccordingly. The pruning process is completed in one pass through the tree. C4.5's tree-construction algorithm differs in several respects from CART [17], for instance:

Tests in CART are always binary, but C4.5 allows two or more outcomes.

• CART uses the Gini diversity index to rank tests, whereas C4.5 uses information-based criteria.

• CART prunes trees using a cost-complexity model whose parameters are estimated by cross-validation; C4.5 uses a single-pass algorithm derived from binomial confidence limits.

• This brief discussion has not mentioned what happens when some of a case's values are unknown. CART looks for surrogate tests that approximate the outcomes when the tested attribute has an unknown value, but C4.5 apportions the case probabilistically among the outcomes.

Ruleset classifiers

Complex decision trees can be difficult to understand, for instance because information about one class is usually distributed throughout the tree. C4.5 introduced an alternative formalism consisting of a list of rules of the form "if A and B and C and ... then class X", where rules for each class are grouped together. A case is classified by finding the first rule whose conditions are satisfied by the case; if no rule is satisfied, the case is assigned to a default class. C4.5 rulesets are formed from the initial (unpruned) decision tree. Each path from the root of the tree to a leaf becomes a prototype rule whose conditions are the outcomes along the path andwhose class is the label of the leaf. This rule is then simplified by determining the effect of discarding each condition in turn. Dropping a condition may increase the number N of cases covered by the rule, and also the number E of cases that do not belong to the class nominated by the rule, and may lower the pessimistic error rate determined as above. A hill-climbing algorithm is used to drop onditions until the lowest pessimistic error rate is found.

To complete the process, a subset of simplified rules is selected for each class in turn. These class subsets are ordered to minimize the error on the training cases and a default class is chosen. The final ruleset usually has far fewer rules than the number of leaves on the pruned decision tree. The principal disadvantage of C4.5's rulesets is the amount of CPU time and memory that they require. In one experiment, samples ranging from 10,000 to 100,000 cases were drawn from a large dataset. For decision trees, moving from 10 to 100K cases increased CPU time on a PC from 1.4 to 61 s, a factor of 44. The time required for rulesets, however, increased from 32 to 9,715 s, a factor of 300

B. The Apriori algorithm

One of the most popular data mining approaches is to find frequent itemsets from a transaction dataset and derive association rules. Finding frequent itemsets (itemsets with frequency larger than or equal to a user specified minimum support) is not trivial because of its combinatorial explosion. Once frequent itemsets are obtained, it is straightforward to generate association rules with confidence larger than or equal to a user specified minimum confidence. Apriori is a seminal algorithm for finding frequent itemsets using candidate generation [17]. It is characterized as a level-wise omplete search algorithm using anti-monotonicity of itemsets, "if an itemset is not frequent, any of its superset is ever frequent". By convention, Apriori assumes that items within a transaction or itemset are sorted in lexicographic order.

Let the set of frequent itemsets of size k be Fk and their candidates be Ck . Apriori first scans the database and searches for frequent itemsets of size 1 by accumulating the count for each item and collecting those that satisfy the minimum support requirement. It then iterates on the following three steps and extracts all the frequent itemsets.

1. Generate Ck+1, candidates of frequent itemsets of size k +1, from the frequent itemsets of size k.
2. Scan the database and calculate the support of each candidate of frequent itemsets.
3. Add those itemsets that satisfies the minimum support requirement to Fk+1.

Function apriori-gen in line 3 generates Ck+1 from Fk in the following two step process:

1. Join step: Generate RK+1, the initial candidates of frequent itemsets of size k + 1 by taking the union of the two frequent itemsets of size k, Pk and Qk that have the first k−1 elements in common.

$$R_{K+1} = P_k \cup Q_k = \{item_1, \ldots, item_{k-1}, item_k, item_{k'}\}$$
$$P_k = \{item_1, item_2, \ldots, item_{k-1}, item_k\}$$
$$Q_k = \{item_1, item_2, \ldots, item_{k-1}, item_{k'}\}$$

where, i teml < i tem2 < • • • < i temk < i temk_ .

2. Prune step: Check if all the itemsets of size k in Rk+1 are frequent and generate Ck+1 by removing those that do not pass this requirement from Rk+1. This is because any subset of size k of Ck+1 that is not frequent cannot be a subset of a frequent itemset of size k + 1. Function subset in line 5 finds all the candidates of the frequent itemsets included in transaction t. Apriori, then, calculates frequency only for those candidates generated this way by scanning the database. It is evident that Apriori scans the database at most kmax+1 times when the maximum size of frequent itemsets is set at kmax. The Apriori achieves good performance by reducing the size of candidate sets However, in situations with very many frequent itemsets, large itemsets, or very low minimum support, it still suffers from the cost of generating a huge number of candidate setsand scanning the database repeatedly to check a large set of candidate itemsets. In fact, it is necessary to generate 2100 candidate itemsets to obtain frequent itemsets of size 100.

The impact of the algorithm

Many of the pattern finding algorithms such as decision tree[17], classification rules and clustering techniques that are frequently used in data mining have been developed in machine learning research community. Frequent pattern and association rule mining is one of the few exceptions to this tradition. The introduction of this technique boosted data mining research and its impact is tremendous. The algorithm is quite simple and easy to implement. Experimenting with Apriori-like algorithm is the first thing that data miners try to do.

Current and further research

Since Apriori algorithm was first introduced and as experience was accumulated, there have been many attempts to devise more efficient algorithms of frequent itemset mining. Many of them share the same idea with Apriori in that they generate candidates. These include hash-based technique, partitioning, sampling and using vertical data format. Hash-based technique can reduce the size of candidate itemsets. Each itemset is hashed into a corresponding bucket by using an appropriate hash function. Since a bucket can contain different itemsets, if its count is less than a minimum support, these itemsets in the bucket can be removed from the candidate sets. A partitioning can be used to divide the entire mining problem into n smaller problems. The dataset is divided into n non-overlapping partitions such that each partition fits into main memory and each partition is mined separately. Since any itemset that is potentially frequent with respect to the entire dataset must occur as a frequent itemset in at least one of the partitions, all the frequent itemsets found thisway are candidates, which can be checked by accessing the entire dataset only nce. Sampling is simply to mine a random sampled small subset of the entire data. Since there is no guarantee that we can find all the frequent itemsets, normal practice is to use a lower support threshold. Trade off has to be made between accuracy and efficiency. Apriori uses a horizontal data format, i.e. frequent itemsets are associated with each transaction[17].

Using vertical data format is to use a different format in which transaction IDs (TIDs) are associated with each itemset. With this format, mining can be performed by taking the intersection of TIDs. The support count is simply the length of the TID set for the itemset. There is no need to scan the database because TID set carries the complete information required for computing support. The most outstanding improvement over Apriori would be a method called FP-growth (frequent pattern growth) that succeeded in eliminating candidate generation . It adopts a divide and conquer strategy by (1) compressing the database representing frequent items into a structure called FP-tree (frequent pattern tree) that retains all the essential information and (2) dividing the compressed database into a set of conditional databases, each associated with one frequent itemset and mining each one separately. It scans the database only twice. In the first scan, all the frequent items and their support counts (frequencies) are derived and they are sorted in the order of descending support count in each transaction. In the second scan, items in each transaction are merged into a prefix tree and items (nodes) that appear in common in different transactions are counted. Each node is associated with an item and its count. Nodes with the same label are linked by a pointer called node-link. Since items are sorted in the descending order of frequency, nodes closer to the root of the prefix tree are shared by more transactions, thus resulting in a very compact representation that stores all the necessary information. Pattern growth algorithm works on FP-tree by choosing an item in the order of increasing frequency and extracting frequent itemsets that contain the chosen item by recursively calling itself on the conditional FP-tree. FP-growth is an order of magnitude faster than the original Apriori algorithm.

There are several other dimensions regarding the extensions of frequent pattern mining. The major ones include the followings:

(1) incorporating taxonomy in items : Use of taxonomymakes it possible to extract frequent itemsets that are expressed by higher concepts even when use of the base level concepts produces only infrequent itemsets.

(2) incremental mining: In this setting, it is assumed that the database is not stationary and a new instance of transaction keeps added. The algorithm in updates the frequent itemsets without restarting from scratch.

(3) using numeric valuable for item: When the item corresponds to a continuous numeric value, current frequent itemset mining algorithm is not applicable unless the values are discretized. A method of subspace clustering can be used to obtain an optimalvalue interval for each item in each itemset .

(4) using other measures than frequency, such as information gain or $\chi 2$ value: These measures are useful in finding discriminative patterns but unfortunately do not satisfy anti- monotonicity property. However, these measures have a nice property of being convex with respect to their arguments and it is possible to estimate their upperbound for supersets of a pattern and thus prune unpromising patterns efficiently. AprioriSMP uses this principle .

(5) using richer expressions than itemset: Many algorithms have been proposed for sequences, tree and graphs to enable mining from more complex data structure .

(6) closed itemsets: A frequent itemset is closed if it is not included in any other frequent itemsets. Thus, once the closed itemsets are found, all the frequent itemsets can be derived from them. LCM is the most efficient algorithm to find the closed itemsets .

C. AdaBoost Alorithm

Ensemble learning deals with methods which employ multiple learners to solve a problem. The generalization ability of an ensemble is usually significantly better than that of a single learner, so ensemble methods are very attractive. The AdaBoost algorithm proposed by Yoav Freund and Robert Schapire is one of the most important ensemble methods, since it has solid theoretical foundation, very accurate prediction, great simplicity (Schapire said it needs only “just 10 lines of code”), and wide and successful applications. Let X denote the instance space and Y the set of class labels. Assume Y = {−1,+1}. Given a weak or base learning algorithm and a training set {(x1, y1), (x2, y2), . . . , (xm, ym)} where xi ∈ X and yi ∈ Y (i = 1, . . . ,m), the AdaBoost algorithm works as follows. First, it assigns equal weights to all the training examples (xi , yi)(i ∈ {1, . . . ,m}). Denote the distribution of the eights at the t-th learning round as Dt .

From the training set and Dt the algorithm generates a weak or base learner ht : X → Y by calling the base learning algorithm. Then, it uses the training examples to test ht , and the weights of the incorrectly classified examples will be increased. Thus, an updated weight distribution Dt+1 is obtained. From the training set and Dt+1 AdaBoost generates another weak learner by calling the base learning algorithm again. Such a process is repeated for T rounds, and the final model is derived byweighted majority voting of the T weak learners,where theweights of the learners are determined during the training process. In practice, the base learning algorithm may be a learning algorithm which can use weighted training examples directly; otherwise the weights can be exploited by sampling the training examples according to the weight distribution Dt .

In order to deal with multi-class problems, Freund and Schapire presented the Ada- Boost.M1 algorithm which requires that the weak learners are strong enough even on hard distributions generated during the AdaBoost process. Another popular multi-class version of AdaBoost is AdaBoost.MH which works by decomposing multi-class task to a series of binary tasks. AdaBoost algorithms for dealing with regression problems have lso been studied. Since many variants of AdaBoost have been developed during the past decade,Boosting has become the most important "family" of ensemble methods.

Impact of the algorithm

AdaBoost is one of the most important ensemble methods, so it is not strange that its high impact can be observed here and there. In this short article we only briefly introduce two issues, one theoretical and the other applied. In 1988, Kearns and Valiant posed an interesting question, i.e., whether a weak learning algorithm that performs just slightly better than random guess could be "boosted" into an arbitrarily accurate strong learning algorithm. In other words, whether two complexity classes, weakly learnable and strongly learnable problems, are equal. Schapire found that the answer to the question is "yes", and the proof he gave is a construction, which is the first Boosting algorithm. So, it is evident that AdaBoost was born with theoretical significance.

AdaBoost has given rise to abundant research on theoretical aspects of ensemble methods, which can be easily found in machine learning and statistics literature. It is worth mentioning that for their AdaBoost paper , Schapire and Freund won the Godel Prize, which is one of the most prestigious awards in theoretical computer science, in the year of 2003. AdaBoost and its variants have been applied to diverse domains with great success. For example, Viola and Jones combined AdaBoost with a cascade process for face detection. They regarded rectangular features as weak learners, and by using AdaBoost to weight the weak learners, they got very intuitive features for face detection. In order to get high accuracy as well as high efficiency, they used a cascade process (which is beyond the scope of this article). As the result, they reported a very strong face detector: On a 466MHz machine, face detection on a 384 × 288 image cost only 0.067 seconds, which is 15 times faster than state-of-the-art face detectors at that time but with comparable accuracy. This face detector has been recognized as one of the most exciting breakthroughs in computer vision (in particular, face detection) during the past decade. It is not strange that "Boosting" has become a buzzword in computer vision and many other application areas.

IV. WORKFLOW WITH FRAMEWORK

This section we present an overall description of the proposed framework. Figure 2 shows a graphical description. The main objective is to look for new evasion techniques on a given NIDS. After elusion, main goal to detect those changes. We use weka tool to obtain a model that classifies as similar as possible to the NIDS. Due to the use of a simple syntax, the Adaboost algorithm has a simpler semantics. Looking for evasive techniques over the model is easier than over the NIDS. If evasions succeed over the model, and given that this model may have a quite similar behavior than then original NIDS, it is likely that the evasions will also succeed over the NIDS. But now the work our system starts that to detect those elusion. Our framework is composed of a set of tasks described in the following sections.

A. Generate the datasets

The Adaboost modeling process at issue requires a labeled dataset. This dataset must represent as well as possible real traffic. Due to the necessity of generating different traffic profiles, a controlled environment is required. Generated traffic should include normal (simple web requests, remote connections, web navigation, etc) and intrusive (malicious) traffic. Traffic is processed by means of data mining techniques to extract the most significant features. It also needs to be labeled in order to identify it as normal or hostile. Obtained traffic should be exposed to the NIDS, which analyzes the dataset looking for intrusive actions. Output given by the NIDS is appended to its corresponding processed frame[2]. Thus, the obtained dataset is composed of registers with the form:

$$R1,R2,R3,...,RN,L,O$$

Where each *Ri* is the field *i* of the trace (for example, the source port, the flag bits, the amount of data exchanged, etc.), *L* is the label which indicates the nature of data (normal or attack) and *O* is the output given by the NIDS (normal or intrusion). The overall dataset is then divided into smaller sets, one being the training subset and the remainder the testing subsets.

B. NIDS Modeling

As we have mentioned, in our framework Adaboost algorithm is used to model the behavior of NIDS. First, values for some parameters are established. This process can be made manually or automatically. This technique consists of performing the Adaboost modeling phase several times, by using different combination of parameters. The entire dataset is randomly divided into 10 subsets, called folds. Each training phase is performed with one fold, using the remainder to test the evolved model. When all folds have been used to train a different model, it is taken as final combination of parameters the one that gives the best results in test.

The principal advantage of using this technique is that we explore several combinations of parameter values so we can assure that we are using an optimum values for them, as the training phase is performed with all the different subsets (folds) of the entire dataset, so it does not depends on an initial selection, but in the complete dataset. We are searching classification models, an optimum fitness function can be the classification error, that is, the rate that indicates how many traces has been correctly classified, not taking into account whether those traces are positives or negatives. Once the parameters are fixed, we obtain the NIDS models by training them with the entire training subset (which has to be considerably bigger than the remainder, used for testing). Then, we perform the test of the obtained models using the testing set. Results must be stored to be processed afterwards. Because the Adaboost search is heuristic, it is appropriate to perform the training phase several times, using different random seeds, taking the results for the best individual (the one that has produced the best test results) and the average of the individuals. Using different random seeds covers a bigger searching space. A manual optimization of the model is then performed. The tree model obtained has normally redundant branches or nodes, so performing a pruning phase could be interesting to improve the efficiency of the model. Although the improvement of the efficiency is not a primary objective to be satisfied, the prune of the tree largely simplifies models semantics, which is in fact the core of this framework. The output of this phase must be a model easy to understand and interpret, whose behavior must be as similar as possible to the NIDS.

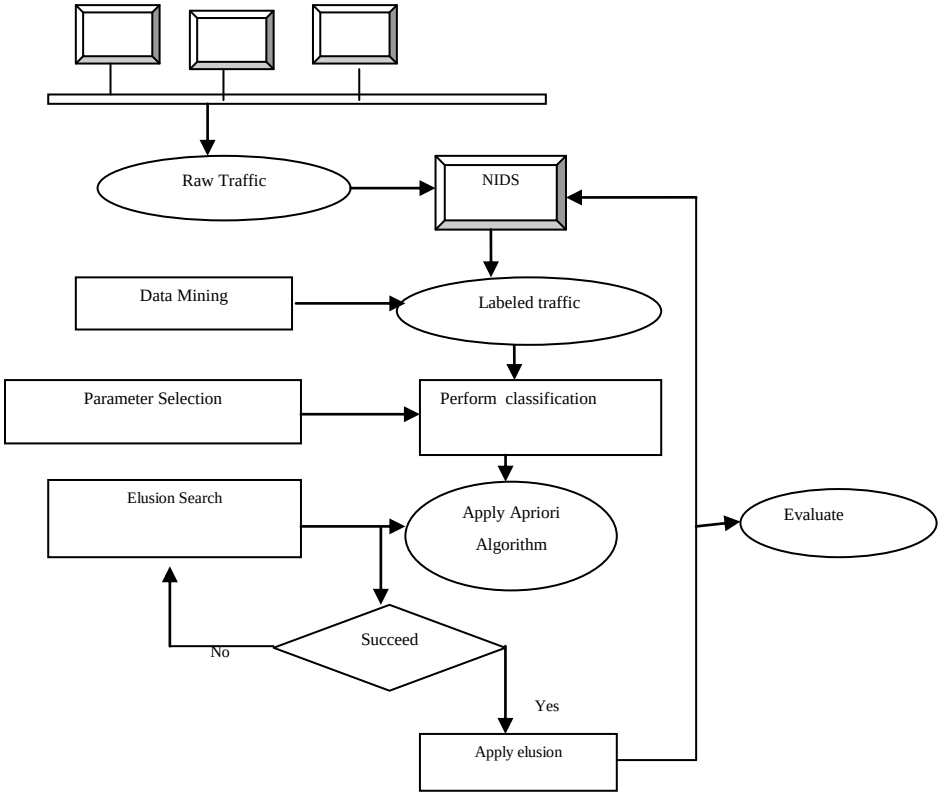

Fig 2:Functional Framework

c. Analysis and design of evasive techniques

Once the model is obtained, it is analyzed in order to discover some points of the internal structure of the NIDS, thus conceiving an idea of its behavior. Mainly, the Model indicates which are the fields that the NIDS takes into account to classify traces. This information is used to perform a brute force modification of those fields. The idea is to automate the process by changing the value of the fields that are present in the model, generating new modified traces. Before changing the value, it should be assured that traces with the new value remain being attacks and still coherent with the protocols. For that purpose, a set of rules must be established and fulfilled, indicating which variables can be changed and which values can be set to them. New valid values are given for those fields in hostile traces which were previously detected by the NIDS (true positives), establishing a new dataset composed by old and new (modified) traces. Then, the NIDS is applied to those new modified traces. New false negatives would indicate that the evasions performed have been successful. The process is repeated for each field that appears in the model, and also multiple simultaneous changes (to more than one field at the same time) can be done.

D . Specific proof concept

The two main objectives of the proof of concept presented are first, to find evasions over the NIDS analyzing the corresponding model. For that purpose, we have created a basic NIDS based on the C4.5 algorithm [2,4]. This algorithm is a supervised learning classifier whose output is a tree. A simplification of the framework has been made to fulfill our goals. Instead of creating a specific dataset, in this work we have opted to use the only two publicly available datasets that are labeled (as normal or intrusive).

Results showed that with an accuracy of 96%, the behavior of a self-built NIDS can be modeled by reducing its complexity. In this work, we improve the study by using a different dataset, the KDD-99 derived from raw traffic captured during MIT/LL 1998 evaluation . The use of an extra dataset corroborates that the accuracy of using Apriori algorithm to model a NIDS is not limited to one scenario and attack , but also to another that uses several kind of attacks. It is obvious that these datasets are both quite old, taking into account the fast growth of the complexity in the Information Technologies. However, they have been widely used in the literature and they provide a huge set of labeled traces. Thus, it is useful for providing insight into the problem at issue and to analyze if the idea is sound.

After obtaining models for each dataset, we are challenged to find real evasions over the original C4.5 based NIDS. We look for evasions by modifying the value of one or more fields of the traces and exposing them to the original NIDS. We must choose fields and values in such a way that the traces remain coherent with protocols, being still attacks (for example, if we change the bit of some TCP flag in a port scanning attack, we are not evading the NIDS and attacking the endpoint, but transforming the malicious trace into a normal one). For that purpose, need is to analyze the nature of the attacks we are working with. It is also needed that traces to be modified are true positives. An evasion is considered successful if, after the modification of the trace, the NIDS does not detect it as an intrusion. And then the main aim of our system is to perform detection over elusion or change.

IV. EXPERIMENTAL WORK

Figure 3 shows a scheme of the modeling phase. At first the datasets are prepared. The KDD provides both normal and Port scanning traffic, captured in various days at different hours. We use five different raw traffic files, processing them in order to take just TCP traffic. Thus, we establish five datasets containing labeled traces from both malicious and normal nature. These traces are composed of the fields (Ri) of the TCP header. In the case of the KDD dataset, we have taken 10% of the original traces, preprocessed them in order to make the output binary (i.e. normal or intrusion) and normalizing the non-numerical fields. We use the weka tool [8] to obtain the C4.5 based NIDS (step 1 in Figure 3). For that purpose, we randomly choose a subset of each dataset to perform the training phase, testing over the remainder. This testing phase provides, for each trace, the output given by the NIDS, i.e. if it has properly classified the trace or not. This information is appended to each trace, obtaining the final dataset (step 2 in the Figure 3). We perform another division of the dataset, in this case to obtain two new different subsets, one to be used in the training phase and another one to test the individuals (step 3 in the Figure 3). Table 1 shows the performance of the NIDS created for both the NSL dataset and the KDD. As can be observed, in the case of the NSL, NIDS are tending to classify the traces as

Table I
Performance Of Self Built Ids Using C4.5

	Detection Rate	False Alarm Rate
KDD	82.23%	0.1%
NSL	99%	65%

intrusive, so its detection rate and its false alarm rate are both very high. However, the NIDS built with the KDD has lower rates, which indicates that it is more likely to classify the traces as normal. So, given that the NIDS which are going to be modeled are very different in nature, the first goal of our proof of concept, which was to prove the feasibility of using Apriori algorithm to model NIDS goes a step further.

The models are created by first evolving them using a training subset (step 4 in Figure 3), using the remaining subsets to test whether the obtained models have a good performance with different traffic from the one used to evolve them (step 5 in Figure 3). Critical component in C4.5 is that it performs a heuristic search. Accordingly seven different seeds have been used over each training subset, thus obtaining seven different evolved individuals. Then, a Testing process is performed with each individual. In the following section the best and average result for each model is shown. Each individual represents one different NIDS model, and because they must be as simple as possible, a maximum depth of 4 is established

The models are created by first evolving them using a training subset (step 4 in Figure 3), using the remaining subsets to test whether the obtained models have a good performance with different traffic from the one used to evolve them (step 5 in Figure 3). Critical component in C4.5 is that it performs a heuristic search. Accordingly seven different seeds have been used over each training subset, thus obtaining seven different evolved individuals. Then, a Testing process is performed with each individual. In the following section the best and average result for each model is shown. Each individual represents one different NIDS model, and because they must be as simple as possible, a maximum depth of 4 is established.

As it was previously stated, one of the goals of this proof of concept is to corroborate that this is a good paradigm to be used when modeling the NIDS. In order to compare with some other t echniques, we have obtained models using two different techniques. Concretely, we have used the Naïve Bayes approach, which is a specific Bayesian classifier which assumes strong independence among

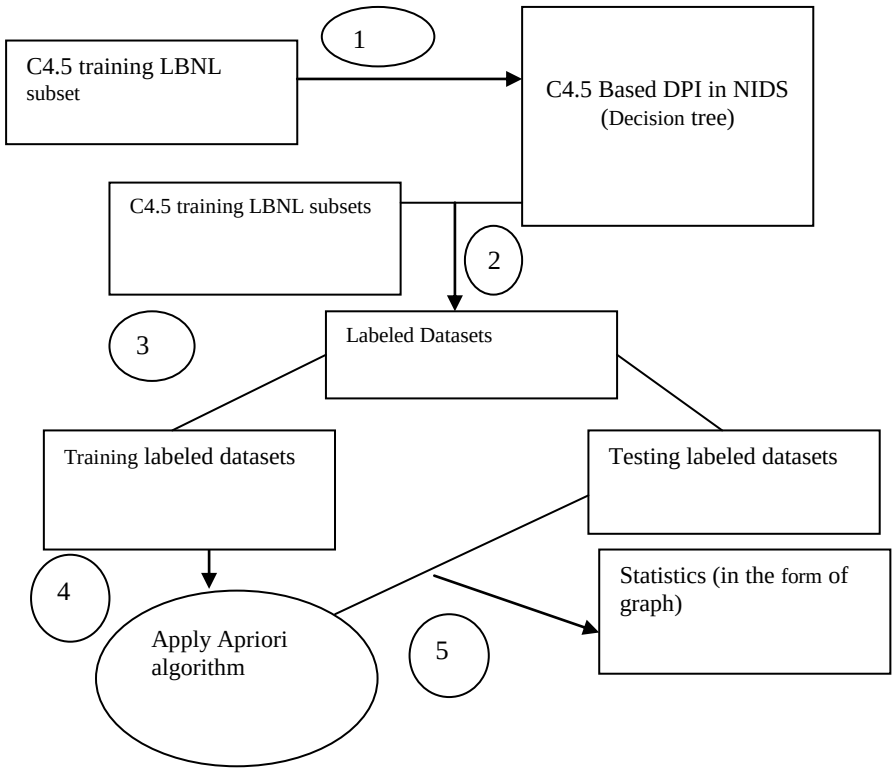

Fig 3. Detailed designed for experimental work

fields [7] and whose output is not a tree, but a probabilistic model. The second method used is the C4.5 algorithm, which is the one used to create the NIDS under study, but limiting its maximal tree depth to 4. It is obvious that the C4.5 algorithm will reach better results if its maximal depth would not be limited to 4, because it is the algorithm used to obtain the original NIDS. However, this limitation of the maximal depth is needed to assure that the complexity of the models to be compared is similar.

In order to evade these models, we are interested in changing traces corresponding to true positives. We analyze the models manually looking for any field that, when changed, will make the NIDS to fail in the detection. It is possible that there is no possible change that causes the evasion of the NIDS. In this case, we should repeat modeling process (by changing some field or the fitness function) in order to obtain another model over which we would look for new evasive methods. Now here the evasion is done and the existing NIDS fail to detect the malicious behavior. So again by identifying the fields and the parameter where the changes has already been done, we can perform the better detection with this system. As by generating rules by using Apriori algorithm where by providing support and confidence we identify the parameters which are responsible for evasion. Then the values of those parameters changed and again the detection has been performed which detects the attack which are ignored by the original NIDS after evasion. In this way we improve the detection rate and accuracy.

V.CONCLUSION

Currently, NIDS are prepared to detect a huge variety of attacks. Some of them, like Snort, take into account the possibility of being evaded with the techniques. However, they are not prepared to new evasive forms that can appear. In this paper we present a new framework to look for evasions over a given NIDS. The core of the framework is to model the NIDS using Adaboost Algorithm obtain an easier to understand individual which works as similar as possible to the NIDS. This model allows the understanding of how the NIDS classifies network data. Once this model is obtained, we can look for some way of evading the NIDS detection by changing some of the fields of the packets. The final aim of using our framework is not to break the detection of the NIDS, but to analyze NIDS robustness with high detection rate accuracy.

ACKNOWLEDGMENT

It is a pleasure for me to present this paper where guidance plays an invaluable key and provides concrete platform for completion of the paper.

I would also like to express my sincere thanks to my internal guide Prof. Mr. T. J. Parvat. Department of Computer Engineering , for his unfaltering encouragement and constant scrutiny without which I wouldn't have looked deeper into my work and realized both our shortcomings and our feats. This work would not have been possible without him.

REFERENCES

1. Xu Kefu, Guo Li, Tan Jianlong, Liu Ping,"Traffic aware frequent element matching algorithm for Deep Packet Inspetion",International Conference on Network Security,wireless communication & Trusted Computing, 2010 .
   Sergio Pastrana Agustin Orfila Arturo Ribagorda,"A functional framework to evade NIDS", Hawaii International conference on System Sciences , 2011 .
2. J. R. Koza, "Genetic Programming: On the Programming of Computers", International conference on security sciences, 2010

3. S. Pastrana, A. Orfila, and A. Ribagorda, “Modeling NIDS evasion with Genetic Programming”, on the Proceedings of The 2010 International Conference on Security and Management, SAM 2010
L. Juan, C. Kreibich, C.-H. Lin, and V. Paxson, "A Tool for
4. Offline and Live Testing of Evasion Resilience in Network Intrusion Detection Systems,",5th international conference on Detection of Intrusions and Malware, and Vulnerability,2008
M. Hall, E. Frank, G. Holmes, B. Pfahringer, P.Reutemann, H.
5. Witten, "The WEKA Data Mining Software: An Update", An extensive empirical study of feature selection metrics for text classification , 2009
6. Po-Ching Lin; Ying-Dar Lin; Tsern-Huei Lee; Yuan-Cheng Lai; , "Using String Matching for Deep Packet Inspection," *Computer* , vol.41,no.4,pp.23-28,April2008
7. Kun Huang; Dafang Zhang; , "A Byte-Filtered String Matching Algorithm for Fast Deep Packet Inspection," . The 9th International Conference for Computer science 2008
8. Randy Smith, Cristian Estan, Somesh Jha, Shijin Kong. Deflating the big bang:fast and scalable deep packet inspection with extended finite automata. SIGCOMM 2008, pages:207-218
9. Anees Shaikh, Jennifer Rexford, and Kang Shin. Load-sensitive routing of long-lived IP flows. In Proceedings of the ACM SIGCOMM. Aug.1999, pages: 215-226.
10. Kocak, T.; Kaya, I.; , "Low-power bloom filter architecture for deep packet inspection," *Communications Letters, IEEE* , vol.10, no.3, pp. 210- 212, Mar 2006 doi: 10.1109/LCOMM.2006.1603387
11. Po-Ching Lin; Ying-Dar Lin; Tsern-Huei Lee; Yuan-Cheng Lai; , "Using String Matching for Deep Packet Inspection," *Computer* , vol.41, no.4, pp.23-28, April 2008 doi: 10.1109/MC.2008.138
12. Kun Huang; Dafang Zhang; , "A Byte-Filtered String Matching Algorithm for Fast Deep Packet Inspection," *Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for* , vol., no., pp.2073-2078, 18-21 Nov. 2008 doi: 10.1109/ICYCS.2008.26
13. Jieyan Fan; Dapeng Wu; Kejie Lu; Nucci, A.; , "NIS04-3: Design of Bloom Filter Array for Network Anomaly Detection," *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE* , vol., no., pp.1-5, Nov. 27 2006-Dec. 1 2006 doi: 10.1109/GLOCOM.2006.281
14. Jia Ni; Chuang Lin; Zhen Chen; Ungsunan, P.; , "A Fast Multi-pattern Matching Algorithm for Deep Packet Inspection on a Network Processor," *Parallel Processing, 2007. ICPP 2007. International Conference on* , vol., no., pp.16, 10-14 Sept. 2007 doi: 10.1109/ICPP.2007.7
15. A. Broder and M. Mitzenmacher, “Network applications of bloom filters: a survey,” *Internet Mathematics*, vol. 1, no. 4, pp. 485-509, July 2003.
16. XindongWu · Vipin Kumar · J. Ross Quinlan · Joydeep Ghosh · Qiang Yang · Hiroshi Motoda · Geoffrey J. McLachlan · Angus Ng · Bing Liu · Philip S. Yu · Zhi-Hua Zhou · Michael Steinbach · David J. Hand · Dan Steinberg SURVEY PAPER Top 10 algorithms in data mining, Published online: 4 December 2007© Springer-Verlag London Limited 2007

AUTHORS PROFILE

Pallavi Dhade, Assistant Professor, M.E.(Computer Engineering pursuing),, Department of Computer Engineering, Sinhgad Institute of Technology, Lonavala, Pune, Maharashtra state, India, research area Network security

Prof. T.J.Parvat, Professor, M.E, P.hd(ng pursuing),, Department of Computer Engineering, Sinhgad Institute of Technology, Lonavala, Pune, Maharashtra state, India ,research area network secuity