Evasion of Network Intrusion Detection System (NIDS)

ABSTRACT

Intrusion Detection is the process of monitoring and analyzing the activity of a network or a computer system in order to detect possible intrusion attacks [1]. The design of a network intrusion detection system (NIDS) is determined by a set of decisions about raw data obtaining, event detection, analysis rules, data storage and response procedures. Focusing on the analysis techniques, artificial intelligence has been widely explored, including approaches based on machine learning, neural networks, evolutionary computation, etc.

Nowadays, Network Intrusion Detection Systems are quickly updated in order to prevent systems against new attacks. This situation has provoked that attackers focus their efforts on new sophisticated evasive techniques when trying to attack a system. Unfortunately, most of these techniques are based on network protocols ambiguities, so NIDS designers must take them into account when updating their tools. In this paper, we present a new approach to improve the task of looking for new evasive techniques.

Signature based Network Intrusion Detection Systems (NIDS) apply a set of rules to identify hostile traffic in network segments. Currently they are so effective detecting known attacks that hackers seek new techniques to go unnoticed. Nowadays NIDS are prepared against most of these evasive techniques, as they are recognized and sorted out. The emergence of new evasive forms may cause NIDS to fail. In this paper we present an innovative functional framework to evade NIDS.

KEYWORDS

Effectiveness, efficiency, Evasion, Intrusion detection, Network security.

RELATED WORK

NIDS evasive techniques were first proposed by Ptacek and Newsham in 1998. In their seminal paper, the authors stressed two main problems in some network protocols when using NIDS. The first is the existence of some ambiguities in the TCP and IP protocols. Those ambiguities allow systems to interpret on their own way how to implement some characteristics of those protocols. For instance, they do not determine what should be done when a packet encapsulates an erroneous checksum field in its TCP header. An evasion succeeds when NIDS ignore packets which are going to be processed on the endpoint systems or vice versa. The second problem presented by Ptacek and Newsham is that some NIDS are vulnerable to Denial of Service (DoS) attacks. An attacker sends several fake hostile packets
to the NIDS provoking it to log all the alerts, in such a way that becomes overloaded. In this scenario, the NIDS may not process all the incoming packets, an the attacker could exploit that situation to perform a real attack over the endpoint. Several tools has been implemented with the aim of generate evasive traffic, thus exploiting the properties exposed above. For example, fragroute intercepts network traffic and modifies the packets before forwarding them to their destination, or ids probe, which generates traffic data from original traces.

Current research in techniques designed to prevent evasions are based mainly on network traffic modification, in order to remove the ambiguities of protocols. Thus, a common interpretation of them is established between the NIDS and the endpoint. Watson et al have proposed a system called Protocol Scrubbing that generates well formed TCP data from traffic. With that, there is only one way to process the information. Handley et al. introduced the concept of traffic normalizers, which are intermediate elements that are located in networks to remove possible ambiguities before being exposed to the NIDS. Because some of the evasive techniques are based on packet fragmentation and reassembly, the state of each connection and the previous packets must be stored and processed, in order to analyze the consistency of connections. That consumes a large quantity of resources, leading into a bottleneck when working with high speed networks. Other solutions which does not require traffic modification have been proposed. Varguese et al. present an idea based on dividing the entire signature of the NIDS into single smaller strings and use a fast path to find matches with them. If a match is found, then packets are given to a slower, more effective path to inspects the packet in a deeper way. Shankar and Paxson proposed a system that informs the NIDS about the network topology and the interpretation policy of the endpoint being monitored. Thus, the NIDS can adapt its configuration taking into account that information. Snort has adopted this technique in the IP processor (frg3) of its last release. Finally, Antichi et al. propose the use of Bloom Filters to look for signature matching over single received packets without the need of reassembly. These systems improve the efficiency of the NIDS and never gives false negatives (they detect all), but increments largely the number of false positives.
Primary, we use KDDCUP 1991 dataset which is standard dataset as input. WEKA tool is open source data mining tool. C4.5 is a classification algorithm that generates a classifier in form of a tree. C4.5 is an algorithm used to generate a decision tree. 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. Because the AdaBoost algorithm uses supervised learning, a set of data has to be labeled for training. This labeled data set should contain both normal samples labeled as \(+1\)" and attack samples labeled as \(-1\)". AdaBoost used for training and testing phase.
Now, we need to find frequent data items, for that purpose Modified Aproiri approach used. Then we are forming the detection rules. Rules are sent to Snort tool. Snort is free and open source light-weight network intrusion detection and prevention system. In evasion, rules are totally changed, so that packets are not detected by NIDS.

ALGORITHMS USED:

1. **C4.5 ALGORITHM**

   C4.5 is an algorithm used to generate a decision tree developed by Ross Quinlan. C4.5 is an extension of Quinlan's earlier ID3 algorithm. The decision trees generated by C4.5 can be used for classification, and for this reason, C4.5 is often referred to as a statistical classifier.

   **ALGORITHM**

   C4.5 builds decision trees from a set of training data in the same way as ID3, using the concept of information entropy. The training data is a set \( \mathcal{S} = \{s_1, s_2, \ldots\} \) of already classified samples. Each sample \( s_i = (x_1, x_2, \ldots) \) is a vector where \( x_1, x_2, \ldots \) represent attributes or features of the sample. The training data is augmented with a vector \( \mathcal{C} = (c_1, c_2, \ldots) \) where \( c_1, c_2, \ldots \) represent the class to which each sample belongs.

   At each node of the tree, C4.5 chooses one attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. Its criterion is the normalized information gain (difference in entropy) that results from choosing an attribute for splitting the data. The attribute with the highest normalized information gain is chosen to make the decision. The C4.5 algorithm then recourses on the smaller sub lists.

   This algorithm has a few base cases.

   - All the samples in the list belong to the same class. When this happens, it simply creates a leaf node for the decision tree saying to choose that class.
   - None of the features provide any information gain. In this case, C4.5 creates a decision node higher up the tree using the expected value of the class.
   - Instance of previously-unseen class encountered. Again, C4.5 creates a decision node higher up the tree using the expected value.

2. **ADABOOST ALGORITHM**
• AdaBoost, short for Adaptive Boosting, is a machine learning algorithm, formulated by Yoav Freund and Robert Schapire. It is a meta-algorithm, and can be used in conjunction with many other learning algorithms to improve their performance. AdaBoost is adaptive in the sense that subsequent classifiers built are tweaked in favor of those instances misclassified by previous classifiers. AdaBoost is sensitive to noisy data and outliers. In some problems, however, it can be less susceptible to the overfitting problem than most learning algorithms. The classifiers it uses can be weak (i.e., display a substantial error rate), but as long as their performance is not random (resulting in an error rate of 0.5 for binary classification), they will improve the final model. Even classifiers with an error rate higher than would be expected from a random classifier will be useful, since they will have negative coefficients in the final linear combination of classifiers and hence behave like their inverses.

• AdaBoost generates and calls a new weak classifier in each of a series of rounds \( t = 1, \ldots, T \). For each call, a distribution of weights \( D_t \) is updated that indicates the importance of examples in the data set for the classification. On each round, the weights of each incorrectly classified example are increased, and the weights of each correctly classified example are decreased, so the new classifier focuses on the examples which have so far eluded correct classification.

3. THE APRIORI ALGORITHM:

   Basics:
   • The Apriori Algorithm is an influential algorithm for mining frequent itemsets for Boolean association rules.

   Key Concepts:
   • Frequent Itemsets: The sets of item which has minimum support (denoted by \( \text{Lifor ith-Itemset} \)).
   • Apriori Property: Any subset of frequent itemset must be frequent.
   • Join Operation: To find \( L_k \), a set of candidate k-itemsets is generated by joining \( L_{k-1} \) with itself.

   • The Apriori Algorithm in a Nutshell

   • Find the frequent item sets: the sets of items that have minimum support
• A subset of a frequent item set must also be a frequent item set
• i.e., if \{AB\} is a frequent item set, both \{A\} and \{B\} should be a frequent item set
• Iteratively find frequent item sets with cardinality from 1 to k (k-item set)
• Use the frequent item sets to generate association rules.

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 exposed by Ppacek and Newsham in 1998. 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 to 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. We have tested our framework by using a simple NIDS based on the C4.5 algorithm over the only publicly available datasets that have their records labelled. We have shown the effectiveness and degree of reduction of the complexity to model the behaviour of the NIDS. Taking advantage of this reduction, we have provided evasions over the original C4.5-based NIDS. Concretely, we have found evasions that allow attackers to perform a SYN flooding attack and a port scanning attack to systems in such a way that the NIDS would not detect them.

ACKNOWLEDGMENT

It gives us immense pleasure to express our gratitude to each individual associated directly or indirectly with the successful completion of the report. We would like to express our thanks towards our project Guide Prof. Sarika Ursal for her invaluable co-operation and guidance that she gave us throughout our project report. We would also like to thank our Head of Department, Prof. R. V. Patil for inspiring us and providing us all the lab facilities with the internet, which are required for project work.

REFERENCES:


