IMPACT: International Journal of Research in Engineering & Technology (IMPACT: IJRET)
ISSN(E): 2321-8843; ISSN(P): 2347-4599
Vol. 2, Issue 3, Mar 2014, 129-140

TO FIND NEW EVASION TECHNIQUES ON NETWORK INTRUSION DETECTION SYSTEM

RUTUJA R. PATIL[1] & P. R. DEVALE[2]

[1]Research Scholar, Department of Information Technology, Bharati Vidyapeeth Deemed University, College of Engineering, Pune, Maharashtra, India

[2]Professor, Department of Information Technology, Bharati Vidyapeeth Deemed University, Pune, Maharashtra, India

ABSTRACT

These days, Signature based Network Intrusion Detection Systems (NIDS), which apply a set of rules to identify hostile traffic in network segments are quickly updated in order to prevent systems against new attacks. The objective of an attacker is to find out new evasion techniques to stay unseen. Unfortunately, majority of the existing techniques are based on the ambiguities of the network protocols. As a result of the emergence of the new evasion techniques, NIDS system may fail to give the correct results. The central idea of our paper is to develop a network based intrusion detection system based on Apriori algorithm and other approaches for attack detection and test the input thus produced by the Apriori algorithm with the well known snort intrusion detection system, once candidate sets for detecting different attacks are generated. These candidates in turn will be passed as inputs to the snort intrusion detection system for detecting different attacks.

KEYWORDS: NIDS, Evasion, Apriori Algorithm, AdaBoost Algorithm, Snort

INTRODUCTION

OVERVIEW OF NETWORK INTRUSION DETECTION SYSTEM

Information Technologies have become a critical component of global economy in the last few years. Their protection against hostile actions determines how fast information society and communications will evolve. Security measures are normally classified as: preventive, detective, corrective and recovery. The most convenient are the first, but their cost is the highest and they do not assure to disable the risk totally, so it is preferable to distribute resources over all the techniques. Thus, a called *Perimeter Defense* should be performed, in which different protection barriers (preventive, detective, corrective and recovery) must be placed into the IT systems. There is an increasing public demand to develop systems that can guard against different attacks that are attempted by hackers.

One security system which falls into this category is the Intrusion Detection System (IDS). Intrusion Detection Systems are software or hardware tools that automatically scan and monitor events that take place in a computer or a network, looking for evidence of intrusion [1] And Network Intrusion Detection Systems (NIDS) just analyze network traffic captured on the network segment where they are installed. These systems can be broadly classified into two major categories depending on the analysis techniques of IDS these are mainly:

- **Anomaly-Based IDS**

Anomaly-based IDS works on a performance baseline based on normal network traffic evaluations.

It samples current network traffic activity to this baseline in order to detect whether or not it is within baseline parameters. Data mining techniques can be used for intrusion detection efficiently.

- **Signature-Based IDS**

Network traffic is examined for preconfigured and predetermined attack patterns known as signatures. It is widely available, it uses known patterns as it is easy to implement but they cannot detect attacks for which it has no signature and they are also prone to false positives since they are commonly based on regular expressions and string matching. Since they are based on pattern match, signatures usually don't work that great against attacks with self-modifying behavior. Signature based NIDS are effective at detecting attacks they are prepared for (they may fail to detect zero-day attacks until their signatures become updated). This situation causes attackers to focus their efforts in finding evasions over the signatures of these systems.

An evasion can be defined as any technique that modifies a detectable attack into any other form in order to avoid being detected. The overall idea is to perform some changes to cause that the NIDS does not process the entire attack packet, remaining so undetected. NIDS normally are, in conjunction with firewalls, one of the first objectives to deal with when someone is trying to attack a system. That implies that attackers try to develop sophisticated techniques to avoid being detected. In general, NIDS do not give real time information about what is happening, but they log alerts. Human security auditors are then who have to analyze those alerts searching for hostile activity. If the NIDS gives erroneous information, auditor can be distracted and would not be able to focus their efforts in the real attack.

Currently, proposed evasive techniques are based in ambiguities present in transport and network layer protocols (mainly TCP and IP) [1]. Those ambiguities provoke that different systems interpret the protocols in a different way. An attacker attempting to evade NIDS detection can modify the transmitted packets in such a way that lead into a situation where a system has different information than another one. When using NIDS, an evasion can appear if the NIDS and the monitored endpoint interpret protocols in a different way, so the information processed is different in both systems.

Researching in an evasive techniques is, along with the discovery and detection of new kind of attacks, the principal tool to improve the effectiveness of the NIDS. Currently, the fast adaptation of the NIDS against new attacks provokes that attackers try to perform evasive techniques (more stealthy and hard to detect) instead of directly exploiting those new attacks. Thus, a security administrator is not aware to have been evaded until posterior forensic analysis of the compromised system, when probably the damage has been done. That is the main motivation of our work, whose primordial objective is to discover new forms of NIDS evasive techniques.

IDEA OF PROJECT

In this paper work we focus on misuse detection. In these types of techniques generally attack signatures are collected and stored in a database in the same way as virus protection software does in order to detect the related attacks. Signature based NIDS are effective at detecting attacks for what they are prepared. Firewalls do not normally block packets, but make aware about the intrusion alarm. This situation causes attackers to focus their efforts in finding evasions over the signatures of these systems. The overall idea of intruder is to perform some changes to cause evasions that the NIDS does not process the entire attack packet, which remains undetected. An evasion succeeds if the processing of the packets generates a different representation of the raw data in the NIDS and in the end systems. Data contained in TCP

segments can encapsulate some attacks, but if the NIDS processes those segments are processed differently from the endpoint, it will not be able to detect those attacks.

The aim of this paper is to look for new evasive techniques by analyzing NIDS behavior. In this method first we build NIDS using C4.5 algorithm. Publicly available dataset KDD-99 is given to it. AdaBoost ALGORITHM for supervised learning where labeling of dataset is done as normal or attack is applied. Modified Apriori algorithm generates rules which are checked on snort for evasion. We use other methods like Genetic Algorithm to compare our results.

ARCHITECTURE OF THE SYSTEM

In this method, publicly available dataset KDD- 99 which contain information about attacks is used. It is given as input to C4.5 algorithm using Weka tool. Weka tool is implementation of various classifying and clustering algorithm. C4.5 algorithm gives output as a tree.

After applying adaboost algorithm on output of C4 5, that contains steps like data labeling, training, testing, where Data labeling will contain identification normal and attack packets. +1 meaning attack packet and -1 meaning normal packet. Training phase will contain initialization of parameters. Testing phase will contain real identification attack packets and classifying each detected attack under its category (Such as Dos attack, probe attack, U2R attack, R2Lattack). Detection result and false alarm rate will then get displayed.

Modified apriori algorithm, which contain process of creation of rules for detecting attacks is then used. After creating these rules we pass these rules to snort. Snort is an open source IDS. Now this method will detect the packets in the network. Also it evades the packets by changing the rules. Detection output will get stored in a text file. The workflow is depicted in the following block diagram.

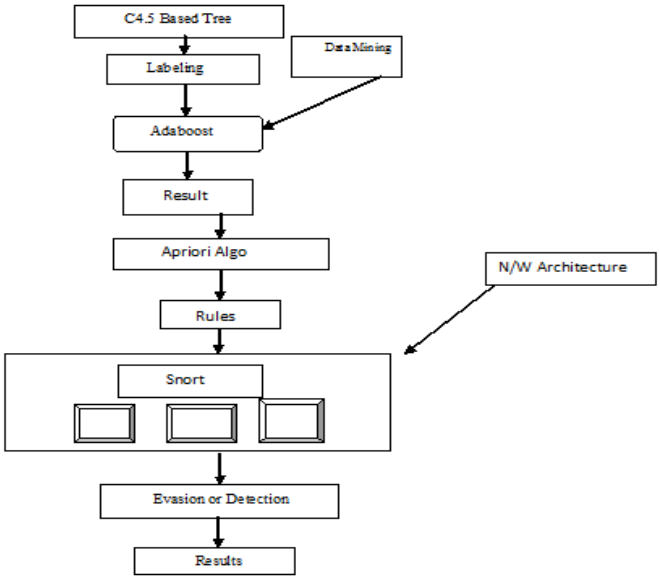

Figure 1: Architecture of NIDS System

INTRUSION DETECTION DATA

Attack types fall into four main categories namely

- **Probing:** Surveillance and other probing

- **DoS:** Denial of service
- **U2S:** Unauthorized access to local super user (root) privileges and
- **R2L:** Unauthorized access from a remote machine.

Probing

Probing is a class of attacks where an attacker scans a network to gather information or find known vulnerabilities. An attacker with a map of machines and services that are available on a network can use the information to look for exploits. There are different types of probes: some of them abuse the computer's legitimate features; some of them use social engineering techniques. This class of attacks is the most commonly heard and requires very little technical expertise.

Denial of Service Attacks

Denial of Service (DoS) is a class of attacks where an attacker makes some computing or memory resource too busy or too full to handle legitimate requests, thus denying legitimate users access to a machine. There are different ways to launch DoS attacks: by abusing the computers legitimate features; by targeting the implementations bugs; or by exploiting the system's misconfigurations. DoS attacks are classified based on the services that an attacker renders unavailable to legitimate users.

User to Root Attacks

User to root (U2R) exploits are a class of attacks where an attacker starts out with access to a normal user account on the system and is able to exploit vulnerability to gain root access to the system. Most common exploits in this class of attacks are regular buffer overflows, which are caused by regular programming mistakes and environment assumptions.

Remote to User Attacks

A remote to user (R2L) attack is a class of attacks where an attacker sends packets to a machine over a network, then exploits machine's vulnerability to illegally gain local access as a user. There are different types of R2U attacks; the most common attack in this class is done using social engineering.

INTRUSION DETECTION DATASETS

KDD Cup'99 Data Set

The data set used to perform the experiment is taken from KDD Cup '99[8][9], which is widely accepted as a benchmark dataset and referred by many researchers. "10% of KDD Cup'99" from KDD Cup '99 data set was chosen to evaluate rules and testing data sets to detect intrusion. The entire KDD Cup '99 data set contains 41 features. Connections are labeled as normal or attacks fall into 4 main categories.

- **DOS:** Denial Of Service
- **Probe:** E.g. Port scanning
- **U2R:** Unauthorized access to root privileges
- **R2L:** Unauthorized remote login to machine.

In this dataset there are 3 groups of features: Basic, content based, time based features.

- Training set Consists 5 million connections
- 10% training set - 494,021 connections
- Test set have - 311,029 connections
- Test data has attack types that are not present in the training data. Problem is more realistic
- Train set contains 22 attack types
- Test data contains additional 17 new attack types that belong to one of four main categories.

Weka

Weka is a collection of machine learning algorithms for data mining tasks. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes. WEKA consists of Explorer, Experimenter, Knowledge flow, Simple Command Line Interface, Java interface.

ALGORITHMS USED

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.

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 $S=s_1,s_2,..$ of already classified samples. Each sample $s_i=x_1,x_2,..$ is a vector where $x_1,x_2,..$ represent attributes or features of the sample. The training data is augmented with a vector $C=c_1,c_2,..$ Where $c_1,c_2,..$ 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.

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,.., 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.

THE APRIORI ALGORITHM

Basics

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

Key Concepts

Frequent Item Sets: The sets of item which has minimum support.

Apriori Property: Any subset of frequent itemset must be frequent.

Join Operation: To find Lk, a set of candidate k-itemsets is generated by joining Lk-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.

LITERATURE REVIEW

With respect to our paper topic “Network Intrusion detection Evading System” we have gone through following documentation of the work done in this field in the past. Some of the highlighted work is presented in the subsection below.

INSERTION, EVASION AND DENIAL OF SERVICE: ELUDING NETWORK INTRUSION DETECTION SYSTEM

The concept of evasion was first proposed by Ptacek and Newsham [2]. In this seminal paper, the authors highlighted the existence of some ambiguities in the TCP and IP protocols, which allow different systems to implement them in a different way. An evasion succeeds when NIDS ignore packets which are going to be processed on the endpoints or vice versa. For example, TCP does not specify what should be done with TCP packets containing an erroneous checksum field. Implementations of the TCP protocol can ignore, accept or reject those packets. As shown in Figure 1, an evasion could succeed if the NIDS implementation of the TCP protocol differs from the endpoint implementation.

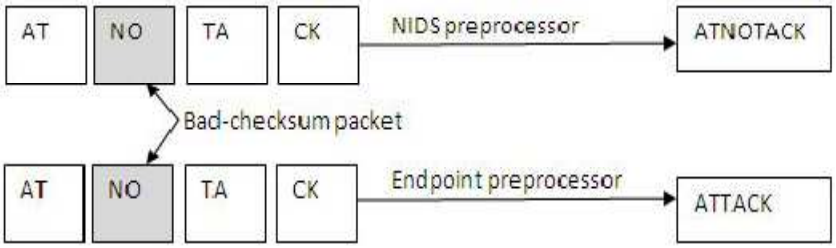

Figure 2: Example of Evasion in this Example, the NIDS Preprocessor Accepts the Packet Containing a Bad Checksum Field, While the Endpoint Does Not, So the Final Structure after the Preprocessing Phase will be Different

FRAGROUTE

Fragroute intercepts, modifies, and rewrites egress traffic destined for a specified host, implementing most of the attacks (D. Son (2002) Fragroute [Online]. http://www monkey.org/~dugsong/fragroute/) [4]. It features a simple ruleset language to delay, duplicate, drop, fragment, overlap, print, reorder, segment, source-route, or otherwise monkey with all outbound packets destined for a target host, with minimal support for randomized or probabilistic behavior. This tool was written in good faith to aid in the testing of network intrusion detection systems, firewalls, and basic TCP/IP stack behavior.

A TOOL FOR OFFLINE AND LIVE TESTING OF EVASION RESILIENCE IN NETWORK INTRUSION DETECTION SYSTEMS

In this paper [5] a framework is created for testing the degree to which network intrusion detection systems (NIDS) detect and handle evasion attacks. This prototype system, idsprobe, takes as input a packet trace and from it constructs a configurable set of variant traces that introduce different forms of ambiguities that can lead to evasions. Test harness then uses these variant traces in either an offline configuration, in which the NIDS under test reads traffic from the traces directly, or a live setup, in which author employs replay technology to feed traffic over a physical network past a NIDS reading directly from a network interface, and to potentially live victim machines. Summary reports of the differences in NIDS output tell the analyst to what degree the NIDS's results vary, reflecting sensitivities to (and possible detections of) different evasions

PROTOCOL SCRUBBING

This paper [6] present design and implementation of protocol scrubbers which are active interposed mechanism for transparently removing packets from protocol layers in real-time. The contribution of this work is: the identification of

transport scrubbing as a mechanism that enables passive NID to operate correctly, the design and implementation of high performance half-duplex TCP/IP scrubber, and the creation of TCP/IP stack fingerprint scrubber. The transport scrubber converts ambiguous network flows in well-behaved flows that are interpreted identically at all downstream endpoints. The fingerprint scrubber removes clues about the identity of an end host's operation system to successfully and completely block the known scans.

DETECTING EVASION ATTACKS AT HIGH SPEEDS WITHOUT REASSEMBLY

This paper [7] suggests breaking with signature using an approach called Split-Detect. Author focus on the simplest form of signature, an exact string match, and start by splitting the signature into pieces. By doing so the attacker is either forced to include at least one piece completely in a packet, or to display potentially abnormal behavior (e.g., several small TCP fragments or out-of-order packets) that cause the attacker's flow to be diverted to a slow path. Author proved that under certain assumptions this scheme can detect all byte-string evasions. It also show using real traces that the processing and storage requirements of this scheme can be 10% of that required by a conventional IPS, allowing reasonable cost implementations at 20 Gbps. While the changes required by Split-Detect may be a barrier to adoption, this paper exposes the assumptions that must be changed to avoid normalization and reassembly in the fast path

ACTIVE MAPPING: RESISTING NIDS EVASION WITHOUT ALTERING TRAFFIC

A critical problem faced by a Network Intrusion Detection System (NIDS) is that of ambiguity. The NIDS cannot always determine what traffic reaches a given host nor how that host will interpret the traffic, and attackers may exploit this ambiguity to avoid detection or cause misleading alarms. This paper [8] present a lightweight solution, Active Mapping, which eliminates TCP/IP-based ambiguity in a NIDS' analysis with minimal runtime cost. Active Mapping efficiently builds profiles of the network topology and the TCP/IP policies of hosts on the network; a NIDS may then use the host profiles to disambiguate the interpretation of the network traffic on a per-host basis. Active Mapping avoids the semantic and performance problems of traffic normalization, in which traffic streams are modified to remove ambiguities. Author developed a prototype implementation of Active Mapping and modified a NIDS to use the Active Mapping-generated profile database in our tests. Author found wide variation across operating systems' TCP/IP stack policies in real-world tests (about 6,700 hosts), underscoring the need for this sort of disambiguation.

REVERSE ENGINEERING OF NETWORK SIGNATURES

This paper [9] describes a reverse engineering process and a reverse engineering tool that are used to analyze the way signatures are matched by network-based intrusion detection systems. The reverse engineering process involves the dynamic analysis of the sensor binary when it is stimulated with legitimate and malicious input. The analysis results are then used to guide the selection of appropriate evasion techniques from a set of alternatives. The results of the analysis are used to either generate variations of attacks that evade detection or produce non-malicious traffic that over-stimulates the sensor. This shows that security through obscurity does not work. That is, keeping the signatures secret does not necessarily increase the resistance of a system to evasion and over-stimulation attacks.

SNORT – LIGHTWEIGHT INTRUSION DETECTION FOR NETWORKS

Snort fills an important ``ecological niche" in the realm of network security: a cross-platform, lightweight network intrusion detection tool that can be deployed to monitor small TCP/IP networks and detect a wide variety of suspicious

network traffic as well as outright attacks. It can provide administrators with enough data to make informed decisions on the proper course of action in the face of suspicious activity. Snort can also be deployed rapidly to fill potential holes in a network's security coverage, such as when a new attack emerges and commercial security vendors are slow to release new attack recognition signatures. This paper [10] discusses the background of Snort and its rules-based traffic collection engine, as well as new and different applications where it can be very useful as a part of an integrated network security infrastructure.

EVOLVING HIGH-SPEED, EASY-TO-UNDERSTAND NETWORK INTRUSION DETECTION RULES WITH GENETIC PROGRAMMING

An ever-present problem in intrusion detection technology is how to construct the patterns of (good, bad or anomalous) behavior upon which an engine have to make decisions regarding the nature of the activity observed in a system. This has traditionally been one of the central areas of research in the field, and most of the solutions proposed so far have relied in one way or another upon some form of data mining–with the exception, of course, of human-constructed patterns. In this paper [11], we explore the use of Genetic Programming (GP) for such a purpose. Here author shows that GP can offer at least two advantages over other classical mechanisms: it can produce very lightweight detection rules (something of extreme importance for high speed networks or resource-constrained applications) and the simplicity of the patterns generated allows to easily understanding the semantics of the underlying attack.

MODELING INTRUSION DETECTION SYSTEMS USING LINEAR GENETIC PROGRAMMING APPROACH

This paper [12] investigates the suitability of linear genetic programming (LGP) technique to model efficient intrusion detection systems, while comparing its performance with artificial neural networks and support vector machines. Due to increasing incidents of cyber attacks and, building effective intrusion detection systems (IDSs) are essential for protecting information systems security, and yet it remains an elusive goal and a great challenge. We also investigate key feature identification for building efficient and effective IDSs. Through a variety of comparative experiments, it is found that, with appropriately chosen population size, program size, crossover rate and mutation rate, linear genetic programs could outperform support vector machines and neural networks in terms of detection accuracy. Using key features gives notable performance in terms of detection accuracies. However the difference in accuracy tends to be small in a few cases.

MODELING INTRUSION DETECTION SYSTEM USING HYBRID INTELLIGENT SYSTEMS

This paper [13] presents two hybrid approaches for modeling IDS. Decision trees (DT) and support vector machines (SVM) are combined as a hierarchical hybrid intelligent system model (DT–SVM) and an ensemble approach combining the base classifiers. The hybrid intrusion detection model combines the individual base classifiers and other hybrid machine learning paradigms to maximize detection accuracy and minimize computational complexity. Empirical results illustrate that the proposed hybrid systems provide more accurate intrusion detection systems.

A FAST APRIORI IMPLEMENTATION

The efficiency of frequent item set mining algorithms is determined mainly by three factors: the way candidates are generated, the data structure that is used and the implementation details. Most papers focus on the first factor,

some describe the underlying data structures, but implementation details are almost always neglected. This paper shows that the effect of implementation can be more important than the selection of the algorithm. Ideas that seem to be quite promising may turn out to be ineffective if we descend to the implementation level. Author theoretically and experimentally analyzes APRIORI which is the most established algorithm for frequent itemset mining. Several implementations of the algorithm have been put forward in the last decade. Although they are implementations of the very same algorithm, they display large differences in running time and memory need. This paper [14] describes an implementation of APRIORI. Author analyzes, theoretically and experimentally, the principal data structure of our solution. This data structure is the main factor in the efficiency of implementation. Author also presents a simple modification of APRIORI that appears to be faster than the original algorithm.

CONCLUSIONS

This paper provides a new method that efficiently improves the task of finding out new forms of evasion by analysing NIDS behaviour thus allowing system administrators to be warned before the attackers could exploit them. The aim of evasion is not to break the NIDS system but to understand and learn different ways of evasion of system and make system more robust. Here in this paper we present a proof of concept showing how to perform detection and evasion in NIDS using publicly available datasets KDD-99.

REFERENCES

1. R. Bace and P. Mell, "NIST Special Publication on Intrusion Detection Systems", 800-31, 2001.
2. T. H. Ptacek and T. N. Newsham, "Insertion, evasion and denial of service: Eluding network intrusion detection," Technical report, 1998.
3. S. Pastrana, A. Orfila, A. Ribagorda, “A Functional Framework to Evade Network IDS”, IEEE xplore, System Sciences (HICSS), 2011 44th Hawaii International Conference.
4. D. Son. (2002) Fragroute. [Online] http://www monkey.org/~dugsong/fragroute/
5. L. Juan, C. Kreibich, C.-H. Lin, and V. Paxson, "A Tool for Offline and Live Testing of Evasion Resilience in Network Intrusion Detection Systems," in DIMVA '08: Proceedings of the 5th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Paris, France, 2008, pp. 267-278.
6. D. Watson, M. Smart, R. G. Malan, and F. Jahanian, "Protocol scrubbing: network security through transparent flow modification," IEEE/ACM Transactions on Networking, vol. 12, pp. 261--273, 2004.
7. G. Varghese, J. A. Fingerhut, and F. Bonomi, "Detecting evasion attacks at high speeds without reassembly," in SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications, Pisa, Italy, 2006, pp. 327--338.
8. U. Shankar and V. Paxson, "Active Mapping: Resisting NIDS Evasion without Altering Traffic," in SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy, Washington, DC, USA, 2003, p. 44.
9. D. Mutz, C. Kruegel, W. Robertson, G. Vigna, and R. A. Kemmerer "Reverse Engineering of Network Signatures", in Proceedings of the AusCERT Asia Pacific Information Technology Security Conference, Gold, 2005.

10. M. Roesch, "Snort - Lightweight Intrusion Detection for Networks," in LISA '99: Proceedings of the 13th USENIX conference on System administration, Seattle, Washington, 1999, pp. 229-238.
11. A. Orfila, A. Ribagorda, “Evolving High-Speed, Easy-to-Understand Network Intrusion Detection Rules with Genetic Programming”, Springer-Verlag Berlin Heidelberg, 2009.
12. S. Mukkamala, A. Sung, and A. Abrham, "Modeling intrusion detection systems using linear genetic programming approach," in IEA/AIE'2004: Proceedings of the 17th international conference on Innovations in applied artificial intelligence, Ottawa, 2004, pp. 633-642.
13. S. Peddabachigaria, A. Abraham, “Modeling intrusion detection system using hybrid intelligent systems”, Journal of Network and Computer Applications.
14. Ferenc Bodon, “A fast APRIORI implementation”, Informatics Laboratory, Computer and Automation Research Institute.
15. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten, "The WEKA Data Mining Software: An Update", in SIGKDD Explorations, Volume 11, Issue 1, 2009.
16. Pallavi Dhade, T.J. Parvat, “To Evade Deep Packet Inspection in NIDS Using Frequent Element Pattern Matching”, IJEIT, Volume 2, Issue 1, July 2012.