International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 6 (2018) pp. 4059-4066


# Detection and Analysis of Network Intrusions using Data Mining Approaches

**M. Naga Surya Lakshmi [1]* and Dr Y Radhika[2]**

[1] *Research Scholar, Department of Computer Science and Engineering, GIT, GITAM University, Visakhapatnam, Andhra Pradesh, India.*

[2]*Professor, Department of Computer Science and Engineering, GIT, GITAM University, Visakhapatnam, Andhra Pradesh, India.*

** Corresponding author*

**Abstract:**

The ultimate role of an intrusion detection system is to identify network threats or attacks in contrast to computing systems. The intrusion detection system (IDS) is one of the essential network protection device or software for guarding computing systems and it is proficient to identify and to examine network traffic data packets. Snort is free open-source software used as a network protection tool. Though, the Snort tool can detect only acknowledged attacks. In order to detect advanced network attacks, this research paper is developed based on advanced snort rules; k-SVM classification method is used for detection of network attacks. In this paper, the KDDCUP'99 dataset is used for the experimental study. The main goal of this research paper is to detect fraudulent network traffic. The main phases of research are data preparation, including the cleaning process, classification of the dataset, feature extraction, proposed snort rules, detection of attacks. The proposed system has produced effective detection rates.

**Keywords:** Anomaly, Kernel, misuse, Signature, Snort, Support Vector Machine

## INTRODUCTION

Roesch Martin has developed open-source IDS called Snort in the year 1999. Snort is mostly used to for detecting signature based attacks. Snort has a vast online community. Mostly snort is deployed at the router for the detection of Network Intrusions or Host-based intrusions. Snort detects attacks based on the rules written in the prescribed format and syntax. The specifications of snort rules indicating the bit/byte patterns of network traffic such as HTTP traffic and TCP streams. For many years, snort has developed a variety of rules for detecting a diversified class of network traffic and various types of attacks. For example, Snort has different rules for detecting attacks occur during the streaming, e-mail traffic, web browsing, Denial-of-service attacks and other types of network exploits.

Snort is a multi variant packet investigation tool, and it can detect attacks by using Sniffer_mode, Network_Intrusion_Detection_System_mode and Packet_Logger_mode The Operational modes of snort are configured using command line arguments. If no command line switches are given, snort automatically tries to go into NIDS mode and it tries to look for snort configuration file stored at "/snort/etc". Snort works almost like TCPDUMP and it decodes network data packets and dumps them to "stdout". For displaying sharply shaped traffic, in this paper filtering interface like BPF is used. The major benefit of snort rules is they are flexible and simple to modify when compared with other commercial NIDS. The architecture of snort is shown in figure 1.1.

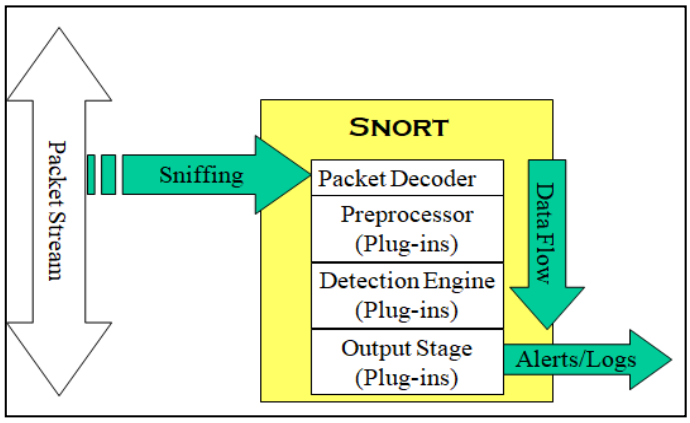


**Figure 1.1:** Snort Architecture

*Packet_Decoder:* It Separates data from a network packet and converts into TCPDUMP format file and it sends extracted packet to the preprocessor module.

*Preprocessor:* The role of the preprocessor module is to generate an alert if any attack was found in the data packet.

*Detection_Engine:* This module is fundamentally used to discover an intrusion movement which exists in the packet by using snort rules. If the intrusion is found then it executes the suitable signature and if not snort drops the packet. The time taken is relative to a number of rules or signatures defined.

*Logging_and_Alerting System:* The role of this module is to generate an alert to the administrator and

*Output_Modules:* This component used to keep the output generated by an alerting and logging system.

Snort rules are simple, lightweight, flexible and powerful. There are many simple guidelines are helpful in writing more powerful rules in snort environment to protect the network from attacks. In general, many Snort rules are described only in one line. By adding Backslash symbol at the end of the line, multiple lines can be added in snort rules. Basically, snort rule consist of two parts. One is "rule_header" and another one is "rule_options". The rule's action, protocol value, IP addresses and port numbers of both source and destination and some other fields are included in the rule_header section. Information about the inspecting part of the packet and alert messages are included in the rule_options section. A sample snort rule is illustrated in figure 1.1

```
alert tcp $HOME_NET 27374 -> $EXTERNAL_NET any (msg:"BACKDOOR ATTACK";
window: 54540; tag: host, 54540, bytes, dst;)
```

**Figure 1.1:** An example of Snort rule

## LITERATURE STUDY

Performance of the snort tool on real-time data has been compared with the SSENet-2011 dataset and proven that the scope of the working of snort is limited and Vasudevan, A.R et.al [1] used advanced methodologies to increase the performance of the snort. Snort-based detection and investigated is proposed by Claude Turner, et.al [2]. Five different snort rule versions are tested on the network traffic. Around 88% of projected rules are failed to provide proper protection in means of security. To enhance security standards, there is much scope to write more complex snort rules. Anna L. Buczak [3] focused on various machine learning techniques and Data mining approaches used to detect misuse and anomaly attacks in real time cyber traffic.

A Hybrid innovative detection scheme has been planned by Syed Rizvi, et.al [4], they developed honeypot based detection scheme in a virtual environment which produced reduction rate inefficiency of hybrid signature based snort system. EC logic encoding techniques have been considered by Mohsen Rouachedab, Hassen Sallay [5], but it has a limitation in handling network threats. The "Extended Kalman Filter" based approach to detect false injected data was used by Bo Sun et.al [6]. The performance of this system produced better accuracy in WSNs.

A framework using IDS-C, IDS-Cr, IDS-M and Genetic Algorithm used by Hicham Toumi, et.al [7] for signature database comparison using mobile agents.

Supeno. Djanali, et.al [8], conducted experiments by capturing around 5000 request and data preprocessing used in the experiments exaggerated quality and amount of generated rules. A Signature database created using honeypot log data and new IDS rules are generated. The IDS designed by Zibusiso Dewa and Leandros A. Maglaras [9], generated low FP percentage, but when compared with dataset size, the FP rate should be moderate and they used C4.5 for pruning. The planned system has recorded less accuracy. The IDS suggested by Shengyi Pan et. al [10] has only correctly classified 90.4% of tested instances and the average detection accuracy was 73.43%.

G.V. Nadiammai, M. Hemalatha [11] has recommended EDADT algorithm, which has shown a higher false alarm rate. S.M. Hussein, et.al [12] performed an experimental study on the dataset using Bayes Net, Naïve Bayes and J48 methods. They used Percent Correct Classification accuracy strategy; it was shown relatively low accuracy. Time taken for building model using naïve Bayes is higher in contrast to model using J48. L´ıdio Mauro Lima de Campos, et.al [13] suggested Bayesian network model based on classifier for detection and produced better accuracy with a small dataset. To improve the performance of their proposed IDS, the decision tree classifiers are used. N. Khamphakdee, et.al [14] developed network traffic converter using association rules, which converts network data into ARFF format done for the limited dataset.

Aymen. Abid, et.al [15] has developed density-based outlier detection mechanism using DBSCAN approach. Performance is executed on a real-life Intel Berkeley database and used in WSNs to detect performance evaluations like False alarm rate and Accuracy. Specific numbers of test case have taken into consideration for every iterative activity. Adeeb Alhomoud et.al [16] conducted an experimental study on both snort and Suricata. Both tools implemented on various platforms like Linux, FreeBSD and ESXi and results are compared. In windows, related operating systems snort has shown better results compared with Suricata. Naila Belhadj Aissaa and Mohamed Guerroumia [17], they developed an intrusion detection system using Maximum Likelihood approach, which used to reduce the threshold values of the attributes and has shown very high False alarm rate.

Security of the mobile agents itself is an obstacle for intrusion detection. Intrusion detection using mobile agents developed by Saidi [18] and they captured flooding attacks like DoS and DDoS attacks in a cloud environment.

Avrim L. Bluma and Pat Langley [19] extracted features of attributes selected using machine learning algorithms and input data is mostly focused on web content and a huge amount of low quality of information has been used for intrusion detection.

For the detection of advanced network threats, a hybrid approach using feature selection and integrated approach were developed by Huan Liu et.al [20]. S. Das [21] has suggested hybrid algorithm BBHFS and it is used to get the better performance of the learning methods and an ID3 classification approach used for dataset classification which is a comparatively low-performance method with support vector machine. Ron Kohavi and George H. John [22] developed a wrapper method, and it is used for feature extraction. It requires more search space and a best-first search approach with complex operators seems to be less accurate. Eric. P. Xing, et.al [23] has designed classification model, which is applied on molecular biology dataset and the hidden Markov method used. The proposed system produced only the features of attributes and attribute significance are not considered into account.

## PROPOSED SYSTEM

### Architecture

The Proposed Intrusion Detection System has been divided into six phases or modules and activity of each and phase is shown in Figure 3.1. The role of each module is described as follows.

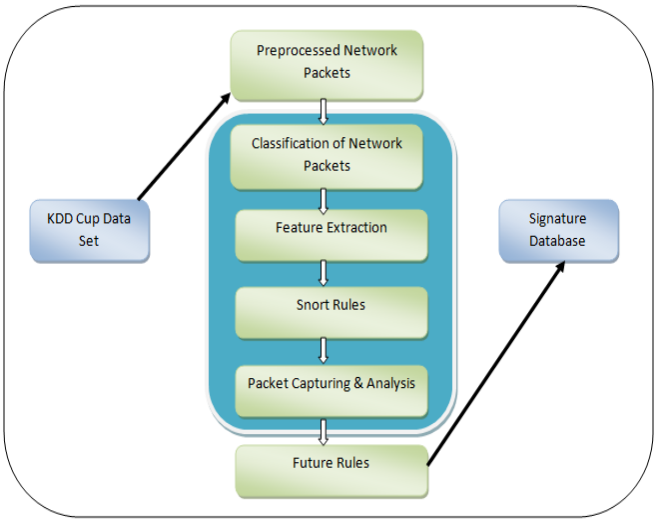


**Figure 3.1:** Proposed Architecture of an IDS

### *Preprocessing Network Packet:*

In order to remove inconsistencies, handling missing values and removing noise from the data set, the dataset is undergone preprocessing phase by using the WEKA tool [18, 22, and 36] and eliminated fewer frequency attributes from the dataset. The Pseudo code for removing attributes having less frequency [24] is shown in Figure 3.2.

```
public boolean input(Instance instance) {
  if (getInputFormat() == null) {
    throw new IllegalStateException("No i/p instance format defined"); }
  if (m_NewBatch) {resetQueue(); m_NewBatch = false;}
  if (m_removeFilter != null) { m_removeFilter.input(instance);
    Instance proc_val = m_removeFilter.output();
    proc_val.setDataset(getOutputFormat());
    copyValues(proc_val, false, instance.dataset(), getOutputFormat());
    push(proc_val); return true;
  } bufferInput(instance);return false; }
public boolean batchFinished() throws Exception {
  if (getInputFormat() == null) {
    throw new IllegalStateException("No input instance format defined"); }
  if (m_removeFilter == null) {Instances toFilter = getInputFormat();
    int[] attsToDelete = new int[toFilter.numAttributes()]; int numToDelete = 0;
    for(int i = 0; i < toFilter.numAttributes(); i++) {
  if (i==toFilter.classIndex()) continue;
  AttributeStats stats = toFilter.attributeStats(i);
  if (stats.missCount == toFilter.numInstances()) { attsToDelete[numToDelete++] = i;
  } else if (stats.distinctCount < 2) { attsToDelete[numToDelete++] = i;  }
   else if (toFilter.attribute(i).isNominal()) {
    double variancePercent = (double) stats.distinctCount
      / (double)(stats.totalCount - stats.missCount) * 100.0;
    if (variancePercent > m_maxVariancePercentage) {
      attsToDelete[numToDelete++] = i; } } }
```

**Figure 3.2:** The Pseudo Code for Preprocessing

### *Classification of Network Packets:*

The network packets have been classified by using kernel-based support vector machines approach. The output of this module is generated in two different files in order to predict the attacks. First, output file consists of anomaly data and a second output file

Consist of misuse packet information given by the kddcup99 dataset [19, 31]. A total of 5910 records are classified. The anomaly classification Pseudo code is shown in figure 3.3 and The Pseudo code for the misuse classification [24] is shown in figure 3.4.

```
Step1:anomaly<-read.csv("/Dataset_Anomaly.csv", na.strings=c(".", "NA", "", "?"),
    strip.white=TRUE, encoding="UTF-8")
Step2:aRow<-nrow(anomaly)
Step3:aCol<-ncol(anomaly)
Step4:sub<-sample(1:aRow,floor(0.66*aRow))
Step5:anomalyTrainingSet<- anomaly[sub,]
Step6:anomalyTestSet<- anomaly[-sub,]
Step7:anomalyClassifier<- ksvm(AttackType~.,data=anomalyTrainingSet,
    type = 'C-svc', kernel = 'rbfdot')
Step8:anomalyPrediction<-predict(anomalyClassifier, anomalyTestSet[,-aCol])
Step9:confusionMatrix(anomalyPrediction,anomalyTestSet[,aCol] )
```

**Figure 3.3:** The Pseudo Code for Anomaly Classification

```
Step1:misuse<-read.csv("/Dataset_Misuse.csv", na.strings=c(".", "NA", "", "?"),
    strip.white=TRUE, encoding="UTF-8")
Step2:mRow<-nrow(misuse)
Step3:mCol<-ncol(misuse)
Step4:sub<-sample(1:mRow,floor(0.66*mRow))
Step5:misuseTrainingSet<- misuse[sub,]
Step6:misuseTestSet<- misuse[-sub,]
Step7:misuseClassifier<- ksvm(AttackType~.,data=misuseTrainingSet,
    type = 'C-svc', kernel = 'rbfdot', kpar=list(sigma=0.000015))
Step8:misusePrediction<-predict(misuseClassifier, misuseTestSet[,-mCol])
Step9:confusionMatrix(misusePrediction,misuseTestSet[,mCol] )
```

**Figure 3.4:** The Pseudo Code for Misuse Classification

### *Feature Extraction:*

*Methods for Feature Selection:*

The required attributes for the proposed system has been derived by using the feature extraction methods proposed by Blum and Avrim.L [19]. Feature extraction is categorized into three major types named filter method, wrapper_method, and hybrid_method. *Filter_method:* This Filter method [20] uses an outer learning procedure to calculate the performance of selected features. *Wrapper_method:* The wrapper method [21] —Wrap around the learning algorithm. It uses one predetermined classifier to evaluate features or feature subsets. Wrapper algorithm is used to perform search through the space of possible features and evaluate each subset by executing a model on each subset. Each feature subset is evaluated on the basis of classification, performance and the best feature is selected. This method is computationally expensive than the filter method [22]. *Hybrid_method:* The hybrid method [23] which combines wrapper and filter approach to achieve the best possible performance with a particular learning algorithm.

*Attribute evaluators:*

Attribute evaluator is mostly used for describing ranking all the features based on the metric. A variety of attribute evaluators exist in WEKA. Correlation-based feature subset extraction *"cfs_subset_eval"* [25, 29] method was used in the proposed system. This method evaluates the significance of a subset of attributes by considering the entity predictive capability of each feature by considering the degree of redundancy between them. Subsets of features that are highly associated with the class while having low inter-correlation are preferred. The minimum number of instances required is one. Using a locally predictive method it identifies and predicts attributes and it adds attributes with the highest correlation with the class in an iterative fashion [26]. In the proposed approach of attribute evaluation missing value is treated as a separate value. The proposed dataset is built using nominal classes and numeric classes of data and attributes are non unary attributes. The required minimum number of instances must be more than one. The selected attributes for anomaly dataset using *WEKA* are six attributes and their order of existence is shown in the Table 1. The attributes derived for handling misuse attacks are shown in the Table 2.

**Table 1:** Feature derived for anomaly attacks using WEKA

| Feature Derived | Attribute Name | Feature Derived | Attribute Name |
|---|---|---|---|
| 12 | Logged-in | 33 | Dst-host-srv-count |
| 28 | Srv-rerror-rate | 36 | Dst-host-same-src-port-rate |
| 32 | Dst-host-count | 39 | Dst-host-srv-serror-rate |

**Table 2:** The Features derived for misuse attacks using WEKA

| Feature Derived | Attribute Name | Feature Derived | Attribute Name |
|---|---|---|---|
| 2 | Protocol-type | 28 | Srv-rerror-rate |
| 3 | Service | 32 | Dst-host-count |
| 5 | Src-bytes | 33 | Dst-host-srv-count |
| 6 | Dst-bytes | 35 | Dst-host-diff-srv-rate |
| 17 | Num-file-creations | 36 | Dst-host-same-src-port-rate |
| 23 | Count | 39 | Dst-host-srv-serror- rate |
| 24 | Srv-count | | |

***Search Methods:***

The Search methods mainly used to search all probable set of features and to resolve the best feature set, the *best_first* search method is used. This approach looks for the attribute space of subset by using the greedy technique. Attributes can be identified in two ways, either by using a "*forward_selection"* or "*backward_elimination"* [27, 30] approach. In some cases, both forward selection and backward elimination techniques can be combined as a hybrid approach to reduce time complexity and to minimize the number of iterative steps.

Forward selection, can start with an empty set and adds a new attribute for each iteration whereas backward elimination starts with a full set and eliminates one unnecessary attribute at a time. Features derived using weka environment is shown in the Table 3. The features derived using WEKA is not much suitable for the proposed system; in order to handle more real-time attacks, the feature of an attribute set is derived manually after studying and analyzing the behavior patterns of each attack. These features have produced better results when compared with the existing mechanism in the intrusion detection system. The derived feature is shown in Table 4.

**Table 3:** The Features derived for Individual attack using WEKA

| Attack Name | Features Derived | Attack Name | Features Derived |
|---|---|---|---|
| Back | 5,10,24,28,37,40 | Perl | 14,16,17,18,33,34,40 |
| Buffer Overflow | 1,5,13,14,17,31,32,33 | Phf | 5,10,14,19,28 |
| Ftp_Write | 6,9,16,17,19,32,33,37 | Pod | 2,4,5,8 |
| Guess_Passwd | 5,10,11,31,38,39,40 | Portsweep | 5,6,29,34,40 |
| Imap | 1,5,33,38 | Rootkit | 16,33,34 |
| Ipsweep | 5,6,12,37 | Satan | 5,30,31,34,37,40 |
| Land | 4,7,25,26,29,31,32,33,38,39 | Smurf | 2,5,24,31,32,38 |
| Loadmodule | 1,5,13,14,17,29,32,33,35 | Spy | 1,15,34,38,39 |
| Multihop | 6,13,17,33,34,36 | Teardrop | 5,6,8,24,31,34,37,40 |
| Neptune | 6,13,17,33,34,36 | Warezclient | 3,23,33,39 |
| Nmap | 6,25,37 | Warezmaster | 5,6,33,34 |

**Table 4:** Manually derived Features for Individual attack based on the behavior pattern

| Attack Name | Features Derived | Attack Name | Features Derived |
|---|---|---|---|
| Back | 2, 3, 4,5, 6, 10, 24, 29 | Perl | 2,3,4,12,33,35 |
| Buffer Overflow | 2,3, 4, 5, 6, 12, 34 | Phf | 2,3,4,5,32,34 |
| Ftp_Write | 2,3,4,5,6,12 | Pod | 2,3,4,5,8,32,35 |
| Guess_Passwd | 2,3,4,5,6,10,11 | Portsweep | 2,3,4,32,33 |
| Imap | 2,3,4,23,34 | Rootkit | 2,3,4,12,32,35 |
| Ipsweep | 2, 3,4,5,23,32 | Satan | 2,3,4,5,32,36 |
| Land | 2,3,4,7,24,39 | Smurf | 2,3,4,5,32,35 |
| Load Module | 2,3,4,32,36 | Spy | 2,3,4,35 |
| Multihop | 2,3,4,5,6,33,36 | Teardrop | 2,3,4,5,6,32,34 |
| Neptune | 2,3,4,25,26 | Warezclient | 2,3,4,5,6,10,32,39 |
| Nmap | 2,3,4,5,29,31 | Warezmaster | 2,3,4,32,33 |

**Proposed Snort Rules:**

Existing snort rules are not adequate to detect an advanced or a new form of attacks. There is a need for a new signature database for detecting advanced attacks. The proposed system is equipped with advanced signature rules. Based on the occurrence and the pattern of each attack new set of snort rules have been derived. *Back Attack* [28, 35] can occur as a Denial of Service Attack to block the web server, the attacker creates requests with multiple URL's embeds with more front slashes. In response to the request, the server will try to process each and every request, in turn, which causes to slow down the server process and it becomes unable to process other genuine requests from the clients. To handle such situation, snort rule written for the back attack by considering the number of sources and destination bytes are being

transferred during the transmission and based on the protocol *TCP*, the flag values are also taken into consideration to prevent unnecessary blockage of the server resource.

The request count has been restricted in order to prevent more URL requests from the attacker and spoofing rate is also bounded to 0.8 or 1. Another DOS attack is the *land* attack, generally, the Land attack appears when an attacker tries to sends a spoofed SYN packet with the same address for the source and destination. To prevent the spoofing in source and destination addresses, the SYN packet count is restricted by setting the *srv_count* value to two. *Neptune or SYN Flood* [24,33], is a DOS attack which causes to overflow the data structure used to store the information about each and every new connection request received by the server. This data structure is finite in size and every half-open TCP connection causes server to add a record to the data structure. Intentionally generated too many half-open connections causes an overflow of data structure and data structure are unable to accept any new request until the data structure is freed. The Proposed snort rules are shown in Table 5.

**Table 5:** Proposed Snort Rules for each type of attack

| Attack Name | Rule |
|---|---|
| Back | protocol=tcp, service=http, flag=SF/SH/RSTR, src_byte=54540/54060, dst_byte=7300/8314,hot=2,srv_count=13,same_srv_rate=1/0.8, |
| Buffer Overflow | protocol=tcp, service=ftp/ftp_data,telnet, flag=SF/RSTO, src_bytes=6247,dst_bytes=70529,loggin_in=1,dst_host_same_srv_rate=1, |
| Ftp_Write | protocol=tcp, service=FTP/FTP_DATA/login, flag=SF,src_bytes=676,dst_bytes=39445,logged_in=1, |
| Guess_Passwd | protocol=tcp, service=telnet, flag=SF/RSTO, src_byte=125or126, dst_byte=179,hot=1, num_failed_login=1, |
| Imap | protocol=tcp, service= imap4, flag=SF/SH, count=4, dst_host_same_srv_rate=1, dst_host_srv_count<=1, |
| Ipsweep | protocol=icmp, service=eco_i/private, flag= SF/RSTO/REJ, src_byte= 8, count=1, dst_host_count=71, |
| Land | protocol=tcp, service=finger/telnet, flag=SO, land=1, srv_count=2, dst_host_srv_serror_rate=0.58/0.12, |
| Load Module | protocol=tcp, service=ftp/ftp_data,telnet, flag=SF, dst_host_count= 6, dst_host_same_src_port_rate=1/0.25, |
| Multihop | protocol= tcp, service=telnet/ftp_data, flag= SF,src_byte= 1412, dst_byte=988002, dst_host_srv_count=3, dst_host_same_src_port_rate= 1 |
| Neptune | protocol=tcp, service=private/smtp/telnet, flag=SO,serror_rate= 1/0.94, srv_serror_rate=1/0.93, |
| Nmap | protocol=tcp/udp/ICMP, service=private/nntp/telnet, flag=SF/SH, src_byte=207,same_srv_rate=1/0.5, srv_diff_host_rate=1, |
| Perl | protocol=tcp, service=telnet, flag=SF, logged_in=1, dst_host_srv_count =2, dst_host_diff_srv_rate=0.07, |
| Phf | protocol=tcp, service=telnet, flag=SF, src_bytes=51,dst_host_count= 255, dst_host_same_srv_rate=1, |
| Pod | protocol=ICMP, service=ecr_i/tim_i, flag=SF, src_byte=1480, wrong_fragment=1, dst_host_count=255, dst_host_diff_srv_rate=1, |
| Portsweep | protocol=TCP, service=Private/ftp/telnet,flag=S0/RSTR, dst_host_count=255, dst_host_srv_count=2, |
| Rootkit | protocol=tcp/UDP, service=telnet/ftp_data, flag=SF, logged_in=1, dst_host_count=255, dst_host_diff_srv_rate=0, |
| Satan | protocol=tcp, service=private/telnet, flag=SF/REJ, src_byte=54, dst_host_count=255, dst_host_same_src_port_rate=0, |
| Smurf | protocol=ICMP, service=ecr_i, flag=SF,src_byte=1032, dst_host_count=255, dst_host_diff_srv_rate=0, |
| Spy | protocol=TCP,Service=Telnet,Flag=SF, dst_host_diff_srv_rate=0.02 |
| Teardrop | protocol=UDP, protocol=tcp,service=telnet,flag=SF, src_byte=237, dst_bytes=1540,dst_host_count=255, dst_host_same_srv_rate=0.18, |
| Warezclient | protocol=tcp, service=ftp/ftp_data, flag=SF/RSTO, src_byte>980, dst_byte>1208, hot>10, dst_host_count=255, dst_host_srv_serror_rate>0.02, |
| Warezmaster | protocol=tcp, service=ftp/ftp_data, flag=SF, dst_host_count=218, dst_host_srv_count>10, |

## RESULT ANALYSIS

Proposed Snort rules have been tested on real-time traffic and simulated data set and attacks detected on the daily basis for a week. The files have been downloaded from the internal network. Attacks detected on a daily basis are shown in Table 6.

Snort has recognized a total of 210 attacks on day one from a total of 918 network packet, among them 884 are TCP packets and 34 are UDP. Total time taken for detecting TCP packets is 0.0037 milliseconds and time taken for detecting UDP packets is 0.0001 milliseconds.

**Table 6:** Attacks detected on daily basis for one week

| DAY | IP Protocol Types | Count | No. of Packets | Rate (ms) | Attacks Detected |
|---|---|---|---|---|---|
| DAY1 | UDP | 884 | 918 | 0.0037 | 210 |
| | TCP | 34 | | 0.0001 | |
| DAY2 | UDP | 1079 | 1739 | 0.0044 | 239 |
| | TCP | 660 | | 0.0027 | |
| DAY3 | UDP | 1233 | 1286 | 0.0128 | 189 |
| | TCP | 53 | | 0.0006 | |
| DAY4 | UDP | 1619 | 1896 | 0.0034 | 197 |
| | TCP | 277 | | 0.0006 | |
| DAY5 | UDP | 1821 | 1950 | 0.0035 | 213 |
| | TCP | 129 | | 0.0002 | |
| DAY6 | UDP | 1713 | 1742 | 0.0071 | 235 |
| | TCP | 29 | | 0.0001 | |
| DAY7 | UDP | 2184 | 3161 | 0.0076 | 359 |
| | TCP | 977 | | 0.0034 | |

The rate of detection is observed higher when a number of packets received. The proposed system achieved the high detection rate when compared with the existing systems.

Snort is executed in NIDS mode and the sample captured packet during the detection phase is shown in Figure 4.1.

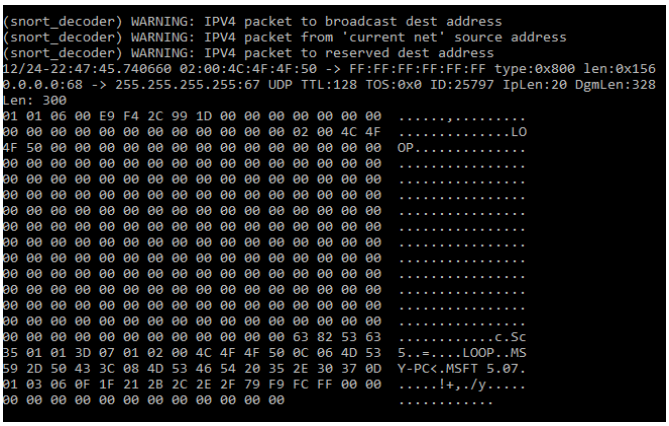


**Figure 4.1:** Sample Packet Capturing Using Snort

The total number of packets detected and analyzed using "Wireshark" tool are shown in Table 6 and the graphical representation of the day wise count of packets is shown in Figure 4.2. Proposed snort rules are tested on real-time traffic and various security violations have created in order to test the efficiency of the proposed system, such as TCP packet flooding [17, 34] was created by using third party websites like *Team viewer* and *Ammyy admin*. These websites will cause to generate more TCP packets during communication and the proposed rules are tested on these packets to capture and to detect network threads.

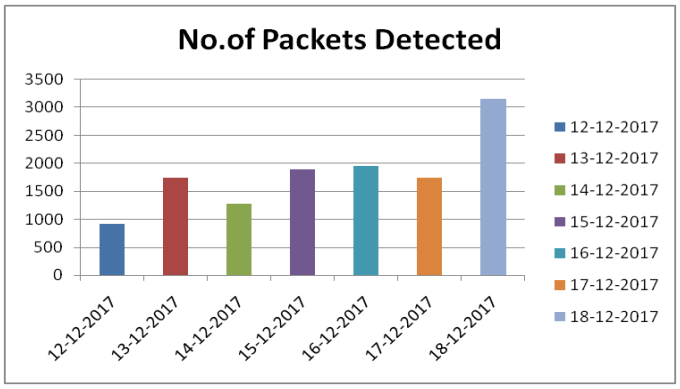


**Figure 4.2:** The Number of packets Captured Using Snort

Each and every packet captured was analyzed using the "Wireshark" tool, and to differentiate the packet's wire shark color codes have been used. This coloring code will differentiate each packet based on the threat level, protocol value and other fields which causes various misuse attacks in the network. Sample TCP packet detection is shown in Figure 4.3. The SEQ/ ACK flag values are analyzed by taking the proposed rules into consideration;

It was alerted with a warning the message under the severity level field. The generated SEQ field is group sequence value. The encapsulated packet was analyzed and it generated black color code, which indicates a Bad TCP segment. The threat was detected in Frame 6 which has a frame length of 66 bytes and all the data in that frame is captured for analysis.

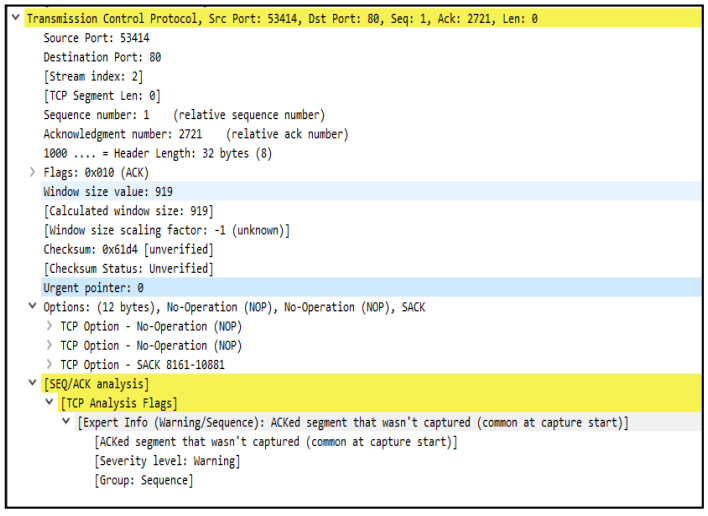


**Figure 4.3:** Detecting Captured TCP Packet using Wire Shark

The total number of attacks detected using proposed snort rules are shown in Figure 4.4. Snort rules are tested for a period of one week and inclination in performance has been observed.

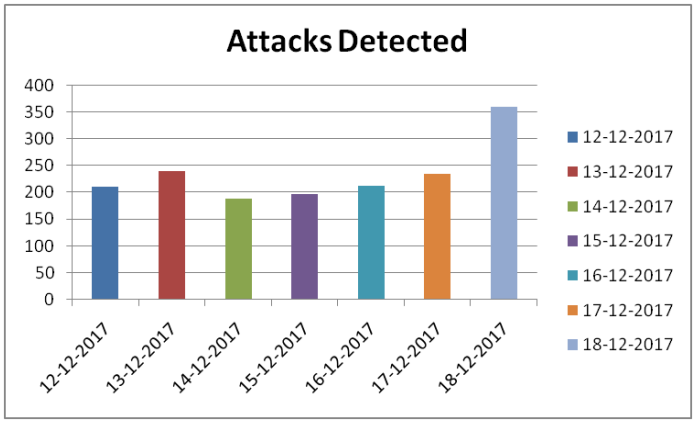


**Figure 4.4:** The Number of Attacks Detected using proposed snort rules

## CONCLUSION & FUTURE WORK

The intrusion detection systems are very efficient for monitoring and detecting network traffic data packets. This research paper has proven that alerts are generated when there is a deviation in the behavioral patterns of the packets. The patterns are matched and compared with the proposed snort rules signature base. The proposed system was methodically tested and compared with existing snort rules, the proposed rules proved to be more accurate and efficient. In future work, advanced data mining techniques and machine learning techniques used for detecting new suspicious attacks on a huge amount of data.

## REFERENCES

[1] Vasudevan, A.R., E. Harshini, and S. Selvakumar (2011). SSENet-2011: a Network Intrusion Detection System Dataset and its Comparison with KDD CUP 99 Dataset. IEEE, 978-1-4577-1088-9

[2] *Claude Turner et al.* (2016). A Rule Status Monitoring Algorithm for Rule-Based Intrusion Detection and Prevention Systems, Complex Adaptive Systems, Conference Organized by Missouri University of Science and Technology 2016 - Los Angeles, CA, Procedia Computer Science 95 ( 2016 ) 361 – 368, 1877-0509, doi: 10.1016/j.procs.2016.09.346

[3] Anna L. Buczak. (2015). A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection, 10.1109/COMST.2015.2494502, IEEE Communications Surveys & Tutorials, 1553-877X (c)

[4] *Syed Rizvi et al* (2015). Advocating for Hybrid Intrusion Detection Prevention System and Framework Improvement, Complex Adaptive Systems, Conference Organized by Missouri University of Science and Technology 2016 - Los Angeles, CA, Procedia Computer Science 95 ( 2016 ) 369 – 374, doi: 10.1016/j.procs.2016.09.347

[5] Mohsen Rouachedab, Hassen Sallay (2012). An Efficient Formal Framework for Intrusion Detection Systems, The 2nd International Symposium on Frontiers in Ambient and Mobile Systems(FAMS), Procedia Computer Science10, 968–975, 1877-0509, doi: 10.1016 / j.procs. 2012. 06.132

[6] SUN *et al.*: Anomaly Detection Based Secure In-Network Aggregation for Wireless Sensor Networks, IEEE SYSTEMS JOURNAL, VOL. 7, NO. 1, March 2013, 13-25, 1932-8184, Digital Object Identifier 10.1109/JSYST.2012.2223531

[7] Hicham Toumi *et al.*, Cooperative Intrusion Detection System Framework Using Mobile Agents For Cloud Computing, Journal of Theoretical and Applied Information Technology 10th December 2014. Vol.70 No.1, 1992-8645, 76-84

[8] Supeno Djanali *et al.*, Coro: Graph-Based Automatic Intrusion Detection System Signature Generator For Evoting Protection, Journal of Theoretical and Applied Information Technology 30th November 2015. Vol.81. No.3, 1992-8645,535-546

[9] Zibusiso Dewa and Leandros A. Maglaras, Data Mining and Intrusion Detection Systems, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 7, No. 1, 2016,62-76

[10] PAN *et al.*:, Developing a Hybrid Intrusion Detection System Using Data Mining for Power Systems, IEEE TRANSACTIONS ON SMART GRID, 1949-3053 _c 2015 IEEE, Digital Object Identifier 10.1109/TSG.2015.2409775,1-9

[11] G.V. Nadiammai, M. Hemalatha, Effective approach toward Intrusion Detection System using data mining techniques, Egyptian Informatics Journal (2014) 15, 37–50, 1110-8665 _ 2013 Production and hosting by Elsevier B.V. on behalf of Faculty of Computers and Information, Cairo University. http://dx.doi.org/10.1016/j.eij.2013.10.003

[12] Safwan Mawlood Hussein *et al.*:, Evaluation Effectiveness of Hybrid IDS Using Snort with Naïve Bayes to Detect Attacks, 978-1-4673-0734-5/12/$31.00 ©2012 IEEE, 256-260

[13] L´ıdio Mauro Lima de Campos *et al.*:, Network Intrusion Detection System Using Data Mining, EANN 2012, CCIS 311, pp. 104–113, 2012. Springer-Verlag Berlin Heidelberg 2012

[14] Nattawat Khamphakdee *et al.*:, Network Traffic Data to ARFF Converter for Association Rules Technique of Data Mining, 2014 IEEE Conference on Open Systems (ICOS), October 26-28, 2014, Subang, Malaysia, 978-1-4799-6367-6/14/$31.00©2014 IEEE, 89-93

[15] Aymen Abid *et al.*:, Outlier detection for wireless sensor networks using density-based clustering approach, *IET Wireless. Sens. Syst.*, 2017, Vol. 7 Iss. 4, pp. 83-90, The Institution of Engineering and Technology 2017, ISSN 2043-6386

[16] Adeeb Alhomoud et al., Performance Evaluation Study of Intrusion Detection Systems, The 2nd International Conference on Ambient Systems, Networks and Technologies, (ANT), Procedia Computer Science 5 (2011) 173–180, 1877–0509 © 2011 Published by Elsevier Ltd. Selection and/or peer-review under responsibility of Prof. Elhadi Shakshuki and Prof. Muhammad Younas. doi:10.1016/j.procs.2011.07.024

[17] Naila Belhadj Aisa, Mohamed Guerroumia, Semi-Supervised Statistical Approach for Network Anomaly Detection, The 6th International Symposium on Frontiers in Ambient and Mobile Systems (FAMS 2016), Procedia Computer Science 83 ( 2016 ) 1090 – 1095, 1877-0509 © 2016 The Authors. Published by Elsevier B.V. This is an open-access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/). Peer-review under responsibility of the Conference Program Chairs doi: 10.1016/j.procs.2016.04.228

[18] A. Saidi *et al.*:, The functional of A Mobile Agent System to Enhance DoS and DDoS Detection in Cloud, International Journal of Applied Engineering Research ISSN 0973-4562 Volume 11, Number 6 (2016) pp 4615-4617

[19]. Blum, Avrim L. & Pat Langley (1997). Selection of relevant features and examples in machine learning. Artificial Intelligence, 97(1-2), 245–271

[20]. Liu, H. & Yu, L. (2005).Towards integrating feature selection algorithms for classification and clustering. IEEE Transactions on Knowledge and Data Engineering, 17(4), 491-502

[21]. Das, S. (2001). Filters, Wrappers and a Boosting-Based Hybrid for Feature Selection. Proc. 18thInt'l Conf. Machine Learning, 74-81

[22]. R. Kohavi and G.H. John (1997).Wrappers for Feature Subset Selection. Artificial Intelligence.97 (1-2), 273-324

[23]. Xing, E. et al. (2001). Feature Selection for High-Dimensional Genomic Microarray Data. Proc. 15th Int'l Conf. Machine Learning, 601-608

[24] M. Naga Surya Lakshmi, DR Y Radhika, Effective Approach For Intrusion Detection Using Ksvm And R, Journal of Theoretical and Applied Information Technology 15th September 2017. Vol.95. No.17, ISSN: 1992-8645

[25] Mohammadreza Ektefa, Sara Memar, Fatimah Sidi and Lilly Suriani Affendey, "Intrusion detection using data mining techniques", IEEE, 2010.

[26] Matthew V. Mahoney, "Network traffic anomaly detection based on packet bytes", ACM, 2003.

[27] Matthew V. Mahoney and Philip K. Chan, "Learning nonstationary models of normal network traffic for detecting novel attacks", ACM, 2002.

[28] Matthew V. Mahoney and Philip K. Chan, "Learning Rules for Anomaly Detection of Hostile Network Traffic", Florida Institute of Technology, Melbourne, FL 32901.

[29] G. V. Nadiammai and M. Hemalatha, "Handling intrusion detection system using snort based statistical algorithm and semi-supervised approach", Research Journal of Applied Sciences, Engineering and Technology 6(16): 2914-2922, 2013.

[30] Matthew V. Mahoney and Philip K. Chan, "PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic, Florida Institute of Technology", Melbourne, FL 32901.

[31] M. Ali. Aydin, A. Halim Zaim and K. Gokhan Celyan, "A hybrid intrusion detection system design for computer network security", Computer and Electrical Engineering 35(2009) 517-526.

[32] Qingqing Zhang, Hongbian Yang, kai Li and Qian Zhang, "Research on the intrusion detection technology with hybrid model", 2nd Conference on environmental science and information application technology, IEEE, 2010.

[33] Summary Thaseen and Aswani Kumar, "Intrusion detection model using fusion of PCA and optimized SVM", IEEE, 2014.

[34] Divya and Surendra Lakra, "SNORT: A Hybrid intrusion detection system using artificial intelligence with a snort", International journal computer technology & application, Vol 4(3), 466-470, 2013.

[35] Vinod Kumar and Dr. Om Prakash Sangwan, "Signature-based intrusion detection system using SNORT", International Journal of computer application & information technology, 2012.

[36] Natttawat Khamphakdee, Nunnapus Benjamas and Saiyan Saiyod, "Improving intrusion detection system based on snort rules for network probe attack detection", International conference on information and communication technology, IEEE, 2014.

[37] Kapil Wankhade, Sadia Patka, and Ravinndra Thool, "An efficient approach for intrusion detection using data mining methods", IEEE, 2013.