Current Trends in Computer Sciences & Applications
DOI: 10.32474/CTCSA.2019.01.000110
ISSN: 2643-6744
Research Article

Detecting Distributed Denial-of-Service DDoS Attacks

Fasidi FO[1] and Adebayo OT[2]*

[1]*Department of Computer Science, Federal University of Technology, Akure, Nigeria*

[2]*Department of Information Technology, Federal University of Technology, Nigeria*

***Corresponding author:** Adebayo OT, Department of Information Technology, Federal University of Technology, Akure, Nigeria

Received: February 04, 2019 **Published:** February 15, 2019

Abstract

Since the number of damage cases resulting from distributed denial-of-service (DDoS) attacks has recently been increasing, the need for agile detection and appropriate response mechanisms against DDoS attacks has also been increasing. The latest DDoS attack has the property of swift propagation speed and various attack patterns. There is therefore a need to create a lighter mechanism to detect and respond to such new and transformed types of attacks with greater speed. In a wireless network system, the security is a main concern for a user.

Introduction

Security of information is of utmost importance to organization striving to survive in a competitive marketplace. Network security has been an issue since computer networks became prevalent, most especially now that internet is changing the face of computing. As dependency on Internet increases on daily basis for business transaction, so also is cyber-attacks by intruder who exploit flaws in Internet architecture, protocol, operating systems and application software to carry out their nefarious activities Such hosts can be compromised within a short time to run arbitrary and potentially malicious attack code transported in a worm or virus or injected through installed backdoors. Distributed denial of service (DDoS) use such poorly secured hosts as attack platform and cause degradation and interruption of Internet services, which result in major financial losses, especially if commercial servers are affected (Duberdorfer, 2004).

Related Works

Brignoli et al. [1] proposed DDoS detection based on traffic self-similarity estimation, this approach is a relatively new approach which is built on the notion that undisturbed network traffic displays fractal like properties. These fractal-like properties are known to degrade in presence of abnormal traffic conditions like DDoS. Detection is possible by observing the changes in the level of self-similarity in the traffic flow at the target of the attack. Existing literature assumes that DDoS traffic lacks the self-similar properties of undisturbed traffic. The researcher shows how existing bot- nets could be used to generate a self-similar traffic flow and thus break such assumptions. Streilien et al,2005. Worked on detection of DoS attacks through the polling of Remote Monitoring (RMON) capable devices. The researchers developed a detection algorithm for simulated flood-based DoS attacks that achieves a high detection rate and low false alarm rate.

Yeonhee Lee [2] focused on a scalability issue of the anomaly detection and introduced a Hadoop based DDoS detection scheme to detect multiple attacks from a huge volume of traffic. Different from other single host-based approaches trying to enhance memory efficiency or to customize process complexity, our method leverages Hadoop to solve the scalability issue by parallel data processing. From experiments, we show that a simple counter-based DDoS attack detection method could be easily implemented in Hadoop and shows its performance gain of using multiple nodes in parallel. It is expected that a signature-based approach could be well suited with Hadoop. However, we need to tackle a problem to develop a real time defense system, because the current Hadoop is oriented to batch processing.

Proposed System Architecture of Intrusion Detection Based on Association Rule

The structure of the proposed architecture for real time detection of Dos instruction detection via association rule mining, it is divided into two phases: learning and testing. The network sniffer processed the tcpdump binary into standard format putting into learning, during the learning phase, duplicate records as well as columns with single same data were expunge from the record so as to reduce operational. Another table Hashmap was created by the classification model to keep track of the count of various likely classmark that can match the current read network traffic, this table will be discarded once the classmark with highest count had been selected. Depicted in Table 1 is the Association rule classifier algorithm (Tables 2-4).

Table 1: Association Rule Mining Classifier Algorithm.

1.Read the first traffic data from table 3.4 into the memory
2.Read the first rule from table 3.2
3.Read the first number table 3.3
4.Check the corresponding attribute(s) of the number read in table 3.4
5.Check if the attribute matches the rule read
6.If it matches, make an entry/increment of the attack-type in the rule read in the hash
map, and go to (3) to read the next number
7.Repeat (4) to (6) again until all the number had been read
8.Read the next rule in (2) and perform (3) to (7) until all the rules had been read.
9.Check the attack entry with highest count in the hash map table,and assign the attack

Table 2: Sample Rules.

Neptune: CONFIDENCE=100.0%, SUPPORT=0.001926388830027008%
private->teardrop: CONFIDENCE=100.0%, SUPPORT=0.012714166278178252%
finger->land: CONFIDENCE=100.0%, SUPPORT=1.2842592200180053E-4%
1032->smurf:CONFIDENCE=100.0%,SUPPORT=4.7445672624345185%
0.12->land: CONFIDENCE=100.0%, SUPPORT=1.2842592200180053E-4%cp, telnet->Neptune: CONFIDENCE= 100.0%, SUPPORT=0.001926388830027008%
tcp, finger->land: CONFIDENCE=100.0%, SUPPORT=1.2842592200180053E-4%
udp, private->teardrop: CONFIDENCE=100.0%, SUPPORT=0.012714166278178252%
udp, SF->teardrop: CONFIDENCE=100.0%, SUPPORT=0.012714166278178252%
icmp,1032-> smurf: CONFIDENCE=100.0%, SUPPORT=4.7445672624345185%
icmp,1480->pod: CONFIDENCE=100.0%, SUPPORT=0.0025685184400360108%
udp,28->teardrop: CONFIDENCE=100.0%, SUPPORT=0.012714166278178252%
tcp,1->land: CONFIDENCE=100.0%, SUPPORT=1.2842592200180053E-4%
icmp,0->smurf: CONFIDENCE=100.0%, SUPPORT=4.7445672624345185%
icmp,1->pod: CONFIDENCE=100.0%, SUPPORT=0.0025685184400360108%
udp,3->teardrop: CONFIDENCE=98.98989898989899%, SUPPORT=0.012585740356176451%
icmp,511->smurf: CONFIDENCE=93.09766132524902%, SUPPORT=4.417081161329928%
icmp,255->smurf: CONFIDENCE=99.70495885664789%, SUPPORT=4.730568836936323%
tcp, 8->land: CONFIDENCE=100.0%, SUPPORT=1.2842592200180053E-4%
icmp,1.00->smurf: CONFIDENCE=99.31517973148549%, SUPPORT=4.712075504168063%
icmp,0.04->pod: CONFIDENCE=80.0%, SUPPORT=0.0020548147520288084%

Table 3: Sampled number combination table.

0,20	0,1,20	1,3,20	2,6,20
1,20	0,2,20	1,4,20	2,7,20
2,20	0,3,20	1,5,20	2,8,20
3,20	0,4,20	1,6,20	2,9,20
4,20	0,5,20	1,7,20	2,10,20
5,20	0,6,20	1,8,20	2,11,20
6,20	0,7,20	1,9,20	2,12,20
7,20	0,8,20	1,10,20	2,13,20
8,20	0,9,20	1,11,20	2,14,20
9,20	0,10,20	1,12,20	2,15,20
10,20	0,11,20	1,13,20	2,16,20
11,20	0,12,20	1,14,20	2,17,20
12,20	0,13,20	1,15,20	2,18,20
13,20	0,14,20	1,16,20	2,19,20
14,20	0,15,20	1,17,20	3,4,20
15,20	0,16,20	1,18,20	3,5,20
16,20	0,17,20	1,19,20	3,6,20
17,20	0,18,20	2,3,20	3,7,20
18,20	0,19,20	2,4,20	3,8,20
	1,2,20	2,5,20	3,9,20

Table 4: Sample Network Traffic Data.

tcp	telnet	S0	0	0	0	2	1	0.5	1	0.5	1	1	2	1	0	1	1	1	0.5
tcp	telnet	S0	0	0	0	3	2	0.67	1	0.67	0.67	2	3	1	0	0.5	0.67	1	0.67
tcp	telnet	S0	0	0	0	4	3	0.75	1	0.75	0.5	3	4	1	0	0.33	0.5	1	0.75
tcp	telnet	S0	0	0	0	5	4	0.8	1	0.8	0.4	4	5	1	0	0.25	0.4	1	0.8
tcp	telnet	S0	0	0	0	6	5	0.83	1	0.83	0.33	5	6	1	0	0.2	0.33	1	0.83
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	196	51	0.26	0.02	0.26	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	197	52	0.26	0.02	0.26	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	198	53	0.27	0.02	0.27	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	199	54	0.27	0.02	0.27	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	200	55	0.28	0.01	0.28	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	201	56	0.28	0.01	0.28	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	202	57	0.28	0.01	0.28	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	203	58	0.29	0.01	0.29	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	204	59	0.29	0.01	0.29	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	205	60	0.29	0.01	0.29	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	206	61	0.3	0.01	0.3	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	207	62	0.3	0.01	0.3	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	208	63	0.3	0.01	0.3	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	209	64	0.31	0.01	0.31	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	210	65	0.31	0.01	0.31	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	211	66	0.31	0.01	0.31	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	212	67	0.32	0.01	0.32	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	213	68	0.32	0.01	0.32	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	214	69	0.32	0.01	0.32	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	215	70	0.33	0.01	0.33	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	216	71	0.33	0.01	0.33	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	217	72	0.33	0.01	0.33	0	0	0
icmp	ecr_i	SF	1032	0	0	509	509	0	0	1	0	218	73	0.33	0.01	0.33	0	0	0
icmp	ecr_i	SF	1032	0	0	2	2	0	0	1	0	219	74	0.34	0.01	0.34	0	0	0
icmp	ecr_i	SF	1032	0	0	317	317	0	0	1	0	220	75	0.34	0.01	0.34	0	0	0
icmp	ecr_i	SF	1032	0	0	318	318	0	0	1	0	221	76	0.34	0.01	0.34	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	222	77	0.35	0.01	0.35	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	223	78	0.35	0.01	0.35	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	224	79	0.35	0.01	0.35	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	225	80	0.36	0.01	0.36	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	226	81	0.36	0.01	0.36	0	0	0
icmp	ecr_i	SF	1032	0	0	511	511	0	0	1	0	227	82	0.36	0.01	0.36	0	0	0
udp	private	SF	28	0	3	93	93	0	0	1	0	165	93	0.56	0.02	0.56	0	0	0
udp	private	SF	28	0	3	94	94	0	0	1	0	166	94	0.57	0.02	0.57	0	0	0
udp	private	SF	28	0	3	95	95	0	0	1	0	167	95	0.57	0.02	0.57	0	0	0
udp	private	SF	28	0	3	96	96	0	0	1	0	168	96	0.57	0.02	0.57	0	0	0
udp	private	SF	28	0	3	97	97	0	0	1	0	169	97	0.57	0.02	0.57	0	0	0
udp	private	SF	28	0	3	98	98	0	0	1	0	170	98	0.58	0.02	0.58	0	0	0
udp	private	SF	28	0	3	99	99	0	0	1	0	171	99	0.58	0.02	0.58	0	0	0
tcp	finger	S0	0	1	0	1	1	1	1	1	0	1	8	1	0	1	0.38	1	0.12

System Implementation

This chapter presents implementation of Association rule classifier model, documentation of the designed system and the user interfaces. The software and hardware requirement needed for the system and also the testing of the system for verification and validation of functions, as well as the result [3-10].

Interface Design

Start Page: This page is the first page that is seen when the application is executed. The option button enables the use of already generated rules to be reused for classification of another file. If the check box is unselected, the option of selecting a folder containing already generated rule is enabled. Furthermore, select the source file and then the new file to classify. The page is as shown below (Figure 1).

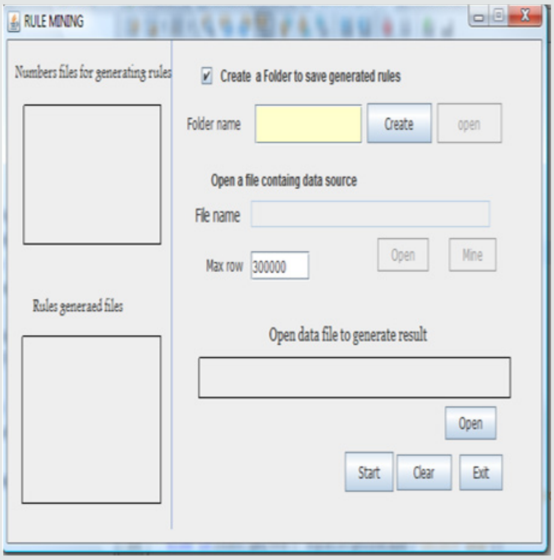

Figure 1: Start page.

Creating Folder: The first step is to create a folder where the rules will be saved, and also the numbers for generating the rules will be saved. This operation is allowed only if the check box on the form is checked, if there is an error with the folder creation process, an error message will be displayed. The open button for the file name is enabled if the folder is created successfully. The interface for the creation of folder is shown below (Figure 2).

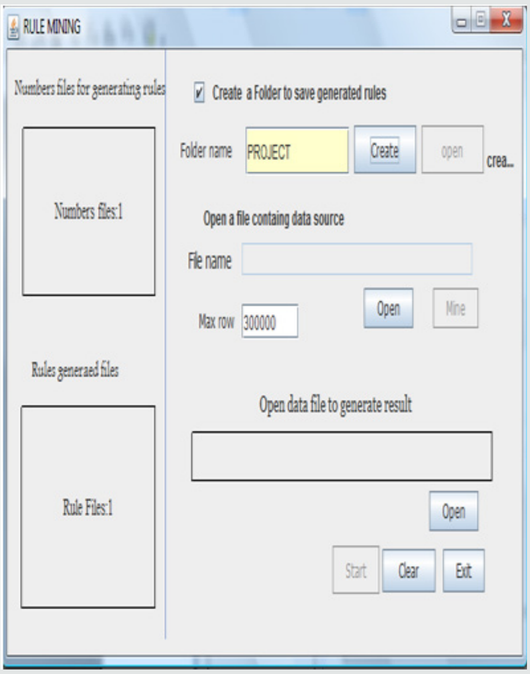

Figure 2: Setting folder name.

Source File: The source of data for generating the rule is the next requirement. The open button closes to the Mine button enable you to specify the file containing the data from which the rules will be generated. Before the rules are generated from the file, the size of the file is calculated to obtain the number of combination(arrangement) required to generate the rule. The selected source file is seen below (Figure 3).

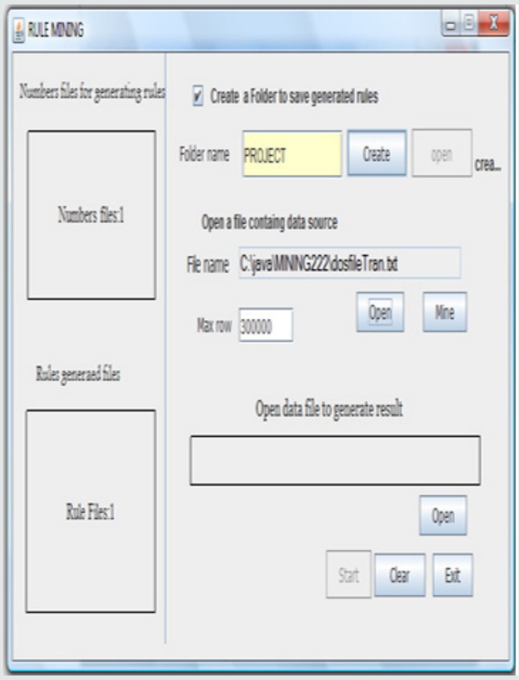

Figure 3: Locating source file.

a) Mining File: The selected source file is mined to extract the rules needed for classification. Depending on the size of the file, it could take a while to complete. On completion, a message dialog is displayed, as shown below (Figure 4).

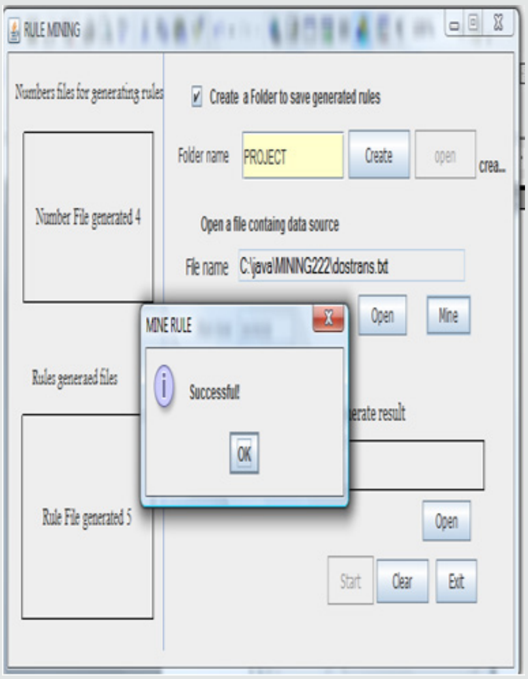

Figure 4: Mining source file.

b) File Classification: After obtaining all the rules from the source file, the open button for selecting a data file to classify is enabled. A file can be classified based on the rules generated. The selected file can be obtained using the open button, as shown below (Figure 5).

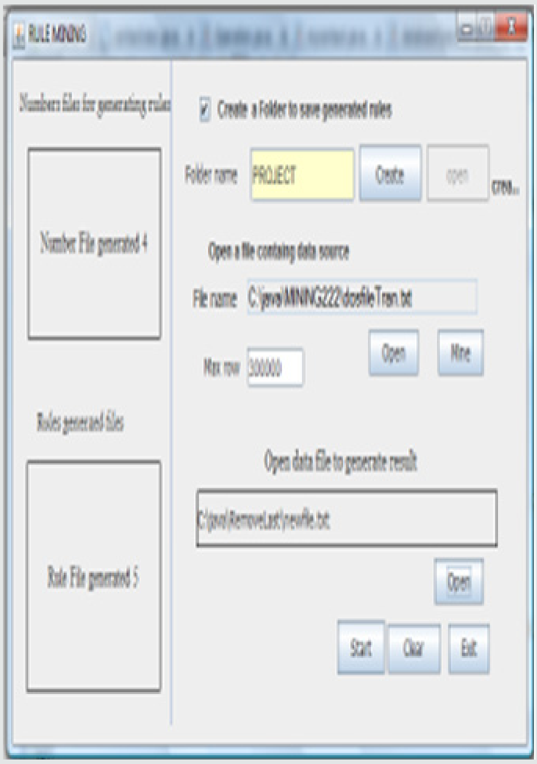

Figure 5: Locating file to classify.

c) Generate Result: Click on the start button to begin the generation of output or result from the selected file to classify, based on the rules generated. The output or result is saved in the folder called result within the folder specified above. Depending on the size of the file to classify, the output might take a while. On completion of the classification, a dialog appears to signify the completion of the classification. This is shown in Figure 6.

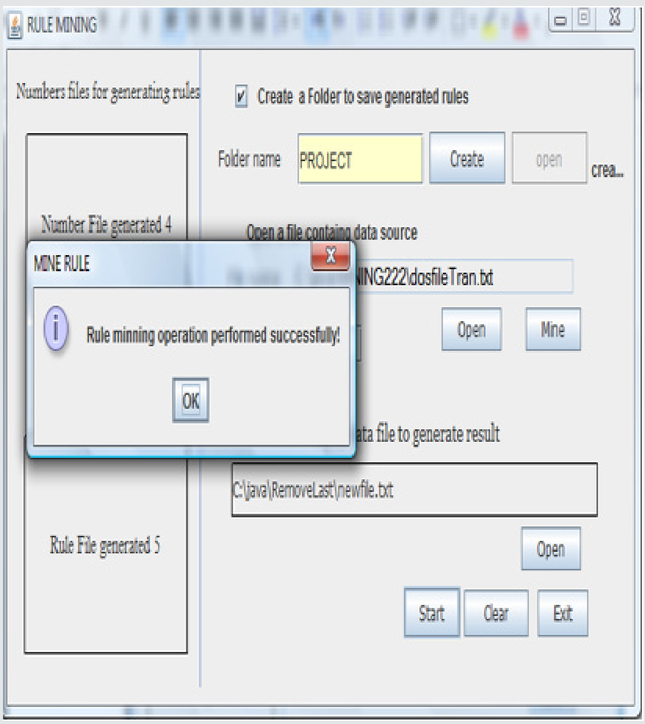

Figure 6: Generating result file from selected file based on rules obtained.

d) Exit Program: Click the Exit button to exit or terminate the application (Figure 7).

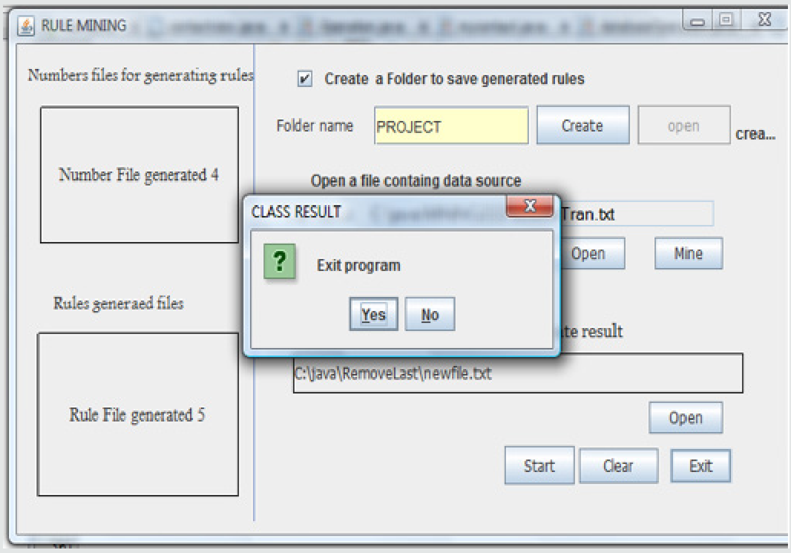

Figure 7: Exit program.

Experimental Setup and Results

The training dataset consisted of 37,079 records, among which there are 99(.266%) teardrop, 36,944 (99.66%) smurf, 20(0.05%) pod, 15(0.04%) neptume and 1(0.026%) land connections. The training dataset use for testing is made up of 400 records out of which there are 98(24.5%) teardrop, 266(66.5%) smurf, 2 (5%) pod, 15(3.75%) neptume and 1 (0.025%) land while the test dataset is made up of 300 records out of there are 40 (13.3%) Pod, 107 (35.6%) smurf, 9(3%) teardrop, 43(14.3%) neptune (14.3%), 9(3%) land, 33(11%) apache2, 21(7%) normal, 25(8.3%) mailbomb and 8(2.6%) snmpgetattack [11-20].

The test and training data are not from the same probability distribution. In each connection are 20 attributes out of 41 attributes describing different features of the connection (excluding the label attribute)

The experiment with association rule classification are divided into two major phases, in the first phase, rules were generated for each, (and combinations of attribute) of the traffic network dataset using the two association rule indices; Confidence and support, in the second phase, the rule generated in the first phase were then pruned to remove irrelevant rules so as to improve the classification process, the pruning process include;

i. Removal of all rules with confidence less than 50%

ii. Removal of all duplicate rules

iii. All identical rules pointing to difference attacks were also removed

iv. All one attribute rules were not considered for classification

Both the initial rules generate, and pruned rules were then used to classify the training set as well as testing data set.

Results

Tables 5-14 shows the confusion matrix obtained Association rule mining with 20 attributes.

Table 5: Confusion matrix obtained from one attribute combination from test dataset (unprune rules).

	Known Attacks						Unknown Attacks			
	Pod	Smurf	Teardrop	Neptune	Land	Normal	Appache	Mail bomb	Snmpget	Unknown
Pod (40)	0	40(100%)	0	0	0	0	0	0	0	0
Smurf (107)	0	107(100%)	0	0	0	0	0	0	0	0
Teardrop (9)	0	9(100%)	0	0	0	0	0	0	0	0
Neptume (44)	0	0	0	0	44(100%)	0	0	0	0	0
Land (9)	0	0	0	0	9(100%)	0	0	0	0	0
Normal (22)	0	3(13.6%)	0	3 (13.6%)	0	16(72%)	0	0	0	0
Appache (34)	0	0	0	2(6%)	32(94%)	0		0	0	0
Mailbomb (26)	0	26(100%)	0	0	0	0	0	0	0	0
Snmpget(8)	0	8(100%)	0		0	0	0	0	0	0

Table 6: Confusion matrix obtained from one attribute combination from test dataset (prune rules).

	Pod	Smurf	Teardrop	Neptune	Land	Normal	Appache	Mail bomb	Snmpget	Unknown
Pod (40)	4(10%)	0	34(85%)	0	0	0	0	0	0	2(5%)
Smurf (107)	0	103(96.3%)	0	0	0	0	0	0	0	4(3.7%)
Teardrop (9)	0	9(100%)	0	0	0	0	0	0	0	0
Neptume (44)	1(2.2%)	0	3(6.8%)	9(20.5%)	1(2.2%)5	0	0	1(2.2%)	0	29(65.9%)
Land (9)	0	0	1(11.1%)	1(11.1%)	7(77.7%)	0	0	0	0	0
Normal (22)	0	0	0	2 (9%)	0	20(91%)	0	0	0	0
Appache (34)	0	0	1(2.9%)	0	3(8.82%)	0		0	0	29(85.2%)
Mailbomb (26)	0	0	14(53.8%)	0	0	0	0	0	0	12(46.1%)
Snmpget(8)	0	8(100%)	0		0	0	0	0	0	0

Table 7: Confusion matrix obtained from one and two attribute combination from test dataset (unprune rules).

	Pod	Smurf	Teardrop	Neptune	Land	Normal	Appache	Mail bomb	Snmpget	Unknown
Pod (40)	0	0	0	0	40(10%)	0	0	0	0	0
Smurf (107)	0	107(100%)	0	0	0	0	0	0	0	0
Teardrop (9)	0	0	7(88%)	0	2(12%)	0	0	0	0	0
Neptume (44)	0	0	0	0	44(10%)	0	0	0	0	0
Land (9)	0	0	0	0	9(100%)	0	0	0	0	0
Normal (22)	0	3(13.6%)	0	3 (13.6%)	0	16(72%)	0	0	0	0
Appache (34)	0	0	0	0	34(10%)	0	0	0	0	0
Mailbomb (26)	0	0	0	0	26(10%)	0	0	0	0	0
Snmpget(8)	0	8(100%)	0		0	0	0	0	0	0

Table 8: Confusion matrix obtained from one and two attribute combination from test dataset (prune rules).

	Pod	Smurf	Teardrop	Neptune	Land	Normal	Appache	Mail bomb	Snmpget	Unknown
Pod (40)	0	40(100%)	0	0	0	0	0	0	0	0
Smurf (107)	0	107(100%)	0	0	0	0	0	0	0	0
Teardrop (9)	0	2(12%)	7(88%)	0	0	0	0	0	0	0
Neptume (44)	0	0	0	44(100%)	0	0	0	0	0	0
Land (9)	0	0	0	0	9(100%)	0	0	0	0	0
Normal (22)	0	0	0	2 (9%)	0	20(91%)	0	0	0	0
Appache (34)	0	1(2.9%)	0	1(2.9%)	0	0	0	0	0	32(94.1%)

Table 9: Confusion matrix obtained from one, two and three attribute combination from test dataset (unprune rules).

	Pod	Smurf	Teardrop	Neptune	Land	Normal	Appache	Mail bomb	Snmpget	Unknown
Pod (40)	0	40(100%)	0	0	0	0	0	0	0	0
Smurf (107)	0	107(100%)	0	0	0	0	0	0	0	0
Teardrop (9)	0	0	9(100%)	0	0	0	0	0	0	0
Neptume (44)	0	0	0	0	44(100%)	0	0	0	0	0
Land (9)	0	0	0	0	9(100%)	0	0	0	0	0
Normal (22)	0	2(9%)	0	3 (13.6%)	0	17(77%)	0	0	0	0
Appache (34)	0	0	0	0	34(10%)	0	0	0	0	0
Mailbomb (26)	0	0	0	0	26(10%)	0	0	0	0	0
Snmpget(8)	0	8(100%)	0		0	0	0	0	0	0

Table 10: Confusion matrix obtained from one, two and three attribute combination from test dataset (prune rules).

	Pod	Smurf	Teardrop	Neptune	Land	Normal	Appache	Mail bomb	Snmpget	Unknown
Pod (40)	38(95%)	2(5%)	0	0	0	0	0	0	0	0
Smurf (107)	0	107(100%)	0	0	0	0	0	0	0	0
Teardrop (9)	0	0	9(100%)	0	0	0	0	0	0	0
Neptume (44)	0	0	0	44(100%)	0	0	0	0	0	0
Land (9)	0	0	0	0	9(100%)	0	0	0	0	0
Normal (22)	0	0	0	2 (9%)	0	20(91%)	0	0	0	0
Appache (34)	0	1(2.9%)	0	1(2.9%)	0	0	0	0	0	32(94.1%)
Mailbomb (26)	0	10(138.5%)	0	0	0	0	0	0	0	16(61.5%)
Snmpget(8)	0	3(37.5%)	0		0	0	0	0	0	5(62.5%)

Table 11: Confusion matrix obtained from one, two, three and four attribute combination from test dataset (unprune rules).

	Pod	Smurf	Teardrop	Neptune	Land	Normal	Appache	Mail bomb	Snmpget	Unknown
Pod (40)	0	40(10%)	0	0	0	0	0	0	0	0
Smurf (107)	0	107(100%)	0	0	0	0	0	0	0	0
Teardrop (9)	0	0	9(100%)	0	0	0	0	0	0	0
Neptume (44)	0	0	0	10(22.7%)	34(77.8%)	0	0	0	0	0
Land (9)	0	0	0	0	9(100%)	0	0	0	0	0
Normal (22)	0	2(9)	0	3 (13.6%)	0	17(77%)	0	0	0	0
Appache (34)	0	0	0	0	34(100%)	0	0	0	0	0
Mailbomb (26)	0	0	0	0	26(100%)	0	0	0	0	0
Snmpget(8)	0	0	8(100%)		0	0	0	0	0	0

Table 12: Confusion matrix obtained from one, two, three and four attribute combination from test dataset (prune rules).

	Pod	Smurf	Teardrop	Neptune	Land	Normal	Appache	Mail bomb	Snmpget	Unknown
Pod (40)	38(95%)	2(5%)	0	0	0	0	0	0	0	0
Smurf (107)	0	107(100%)	0	0	0	0	0	0	0	0
Teardrop (9)	0	0	9(100%)	0	0	0	0	0	0	0
Neptume (44)	0	0	0	44(100%)	0	0	0	0	0	0
Land (9)	0	0	0	0	9(100%)	0	0	0	0	0
Normal (22)	0	0	0	2 (9%)	0	20(91%)	0	0	0	0
Appache (34)	0	1(2.9%)	0	1(2.9%)	0	0	0	0	0	32(94.1%)
Mailbomb (26)	0	10(138.5%)	0	0	0	0	0	0	0	16(61.5%)
Snmpget(8)	0	3(37.5%)	0		0	0	0	0	0	5(62.5%)

Table 13: Confusion matrix obtained from one, two, three, four and five attribute combination from test dataset (unprune rules).

	Pod	Smurf	Teardrop	Neptune	Land	Normal	Appache	Mail bomb	Snmpget	Unknown
Pod (40)	40(100%)	0	0	0	0	0	0	0	0	0
Smurf (107)	0	107(100%)	0	0	0	0	0	0	0	0
Teardrop (9)	0	0	9(100%)	0	0	0	0	0	0	0
Neptume (44)	0	0	0	10(22.7%)	34(77.8%)	0	0	0	0	0
Land (9)	0	0	0	0	9(100%)	0	0	0	0	0
Normal (22)	0	2(9%)	0	3 (13.6%)	0	17(77%)	0	0	0	0
Appache (34)	0	0	0	0	34(100%)	0	0	0	0	0
Mailbomb (26)	0	0	0	0	26(100%)	0	0	0	0	0
Snmpget(8)	0	8(100%)	0		0	0	0	0	0	0

Table 14: Confusion matrix obtained from one, two, three, four and five attribute combination from test dataset (prune rules).

	Pod	Smurf	Teardrop	Neptune	Land	Normal	Appache	Mail bomb	Snmpget	Unknown
Pod (40)	38(95%)	2(5%)	0	0	0	0	0	0	0	0
Smurf (107)	0	107(100%)	0	0	0	0	0	0	0	0
Teardrop (9)	0	0	9(100%)	0	0	0	0	0	0	0
Neptume (44)	0	0	0	14(100%)	0	0	0	0	0	0
Land (9)	0	0	0	0	9(100%)	0	0	0	0	0
Normal (22)	0	0	0	2 (9%)	0	20(91%)	0	0	0	0
Appache (34)	0	1(2.9%)	0	1(2.9%)	0	0	0	0	0	32(94.1%)
Mailbomb (26)	0	10(138.5%)	0	0	0	0	0	0	0	16(61.5%)
Snmpget(8)	0	3(37.5%)	0		0	0	0	0	0	5(62.5%)

Discussion

The results in Tables 15-19, were obtained from classification of training data set with raw unprune rule set, from the tables, the degree of accuracy of classification of smurf attack ranges between 99.6% to 100%. Pod attacks classification could not be classified correct by the classification model, about 95% pod attacks were classified as smurf attacks, while the rest were classified as Pod. 98% of teardrop and 100% of land attacks were also correctly classify, while less than 20% of Neptune attacks were classified correctly using rules based on 0ne, two and three combinational attributes, rules based on four and five combinational attributes has better performance of classification of Neptune attacks (65%) than one – three attributes combination. Table 20 shows the summary of all the attacks correctly classified in Tables 15-19.

Table 15: Confusion matrix obtained from one attribute combination from training dataset (unprune rules).

Predicted as Actual	Neptune	Smurf	Pod	Teardrop	Land
Neptune (16)	**14 (87.5%)**	0.00%	0.00%	0.00%	2 (12.5%)
Smurf (264)	0 (0.00) %	**264 (100%)**	0(0.00%)	0(0.00%)	0(0.00%)
Pod (20)	0(0.00) %	20(100%)	**0(0.00) %**	0(0.00) %	0(0.00) %
Teardrop (99)	0(0.00) %	99(100%)	0(0.00) %	**0(0.00) %**	0(0.00) %
Land (1)	0(0.00) %	0(0.00) %	0(0.00) %	0(0.00) %	**1(100%)**

Table 16: Confusion matrix obtained from one and two attribute combination from training dataset (unprune rules).

Predicted as Actual	Neptune	Smurf	Pod	Teardrop	Land
Neptune (16)	**12 (75. %)**	0.00%	0.00%	0(0.00%)	4(25%)
Smurf (264)	0(0.00) %	**263 (99.62%)**	0(0.00 %)	1(0.37%)	0(0.00%)
Pod (20)	0(0.00) %	19(95%)	**(0.0) %**	0(0.00) %	1(5.00) %
Teardrop (99)	0(0.00) %	99(100%)	0(0.00) %	**0(0.00) %**	0(0.00) %
Land (1)	0(0.00) %	0(0.00) %	0(0.00) %	0(0.00) %	**1(100%)**

Table 17: Confusion matrix obtained from one, two and three attribute combination from training dataset (unprune rules).

Predicted as Actual	Neptune	Smurf	Pod	Teardrop	Land
Neptune (16)	0 (0.00%)	0.00%	0.00%	0(0.00%)	16(100%)
Smurf (264)	0(0.00) %	264 (100%)	0(0.00%)	0(0.00%)	0(0.00%)
Pod (20)	0(0.00) %	19(95%)	1(5.00) %	0(0.00) %	0(0.00) %
Teardrop (99)	0(0.00) %	0(0.00%)	0(0.00) %	99(100) %	0(0.00) %
Land (1)	0(0.00) %	0(0.00) %	0(0.00) %	0(0.00) %	1(100%)

Table 18: Confusion matrix obtained from one, two, three and four attributes combination from training dataset (unprune rules).

Predicted as Actual	Neptune	Smurf	Pod	Teardrop	Land
Neptune (16)	14 (87.5%)	0.00%	0.00%	0(0.00%)	2(12.5%)
Smurf (264)	0(0.00) %	264 (100%)	0(0.00%)	0(0.00%)	0(0.00%)
Pod (20)	0(0.00) %	19(95%)	1(5.00) %	0(0.00) %	0(0.00) %
Teardrop (99)	0(0.00) %	0(0.00%)	0(0.00) %	99(100) %	0(0.00) %
Land (1)	0(0.00) %	0(0.00) %	0(0.00) %	0(0.00) %	1(100%)

Table 19: Confusion matrix obtained from one, two, three, four and five attributes combination from training dataset (unprune rules).

Predicted as Actual	Neptune	Smurf	Pod	Teardrop	Land
Neptune (16)	14 (87.5%)	0.00%	0.00%	0(0.00%)	2(12.5%)
Smurf (264)	0(0.00) %	264 (100%)	0(0.00%)	0(0.00%)	0(0.00%)
Pod (20)	0(0.00) %	19(95%)	1(5.00) %	0(0.00) %	0(0.00) %
Teardrop (99)	0(0.00) %	0(0.00%)	0(0.00) %	99(100) %	0(0.00) %
Land (1)	0(0.00) %	0(0.00) %	0(0.00) %	0(0.00) %	1(100%)

Table 20: Percentages of Correctly Classified Attacks in Table 15-19.

	Neptune	Smurf	Pod	Teardrop	Land
Table 15	87.5%	100%	0%	0%	1000%
Table 16	75%	99.6%	0%	0%	100%
Table 17	87.5%.5%	100%	5%	99%	100%
Table 18	87.5%	100%	5%	100%	100%
Table 19	87.5%	100%	5%	100%	100%

Table 21: Confusion matrix obtained from one attribute combination from training dataset (prune rules).

Predicted as Actual	Neptune	Smurf	Pod	Teardrop	Land
Neptune (16)	**11(68.75%)**	0.00%	0.00%	1(6.25%	3(18.75%)
Smurf (264)	0(0.00) %	**263 (99.62%)**	0(0.00%)	1(0.37%)	0(0.00%)
Pod (20)	0(0.00) %	0(0.00%)	**18(90) %**	1(5.00) %	1(5.00) %
Teardrop (99)	0(0.00) %	0(0.00%)	0(0.00) %	**99(100) %**	0(0.00) %
Land (1)	0(0.00) %	0(0.00) %	0(0.00) %	0(0.00) %	**1(100%)**

Table 22: Confusion matrix obtained from one and attribute combination from training dataset (prune rules).

Predicted as Actual	Neptune	Smurf	Pod	Teardrop	Land
Neptune (16)	**14 (93.75.%)**	0.00%	0.00%	0(0.00%)	1(6.25%)
Smurf (264)	0(0.00) %	**264 (100%)**	0(0.00%)	0(0.00%)	0(0.00%)
Pod (20)	0(0.00) %	0(100%)	**20(100) %**	0(0.00) %	0(0.00) %
Teardrop (99)	0(0.00) %	0(0%)	0(0.00) %	**0(100) %**	0(0.00) %
Land (1)	0(0.00) %	0(0.00) %	0(0.00) %	0(0.00) %	**1(100%)**

Table 23: Confusion matrix obtained from one, two and three attribute combination from training dataset (prune rules).

Predicted as Actual	Neptune	Smurf	Pod	Teardrop	Land
Neptune (16)	**15 (93.75%)**	0.00%	0.00%	0(0.00%)	1(6.25%)
Smurf (264)	0(0.00) %	**264 (100%)**	0(0.00%)	0(0.00%)	0(0.00%)
Pod (20)	0(0.00) %	0(0.00%)	**20(100%)**	0(0.00) %	0(0.00) %
Teardrop (99)	0(0.00) %	0(0.00%)	0(0.00) %	**99(100) %**	0(0.00) %
Land (1)	0(0.00) %	0(0.00) %	0(0.00) %	0(0.00) %	**1(100%)**

Table 24: Confusion matrix obtained from one, two, three and four attributes combination from training dataset (prune rules).

Predicted as Actual	Neptune	Smurf	Pod	Teardrop	Land
Neptune (16)	**16 (100. %)**	0.00%	0.00%	0(0.00%)	2(12.5%)
Smurf (264)	0(0.00) %	**264 (100%)**	0(0.00%)	0(0.00%)	0(0.00%)
Pod (20)	0(0.00) %	19(95%)	**20(100) %**	0(0.00) %	0(0.00) %
Teardrop (99)	0(0.00) %	0(0.00%)	0(0.00) %	**99(100) %**	0(0.00) %
Land (1)	0(0.00) %	0(0.00) %	0(0.00) %	0(0.00) %	**1(100%)**

Table 25: Confusion matrix obtained from one, two, three, four and five attributes combination from training dataset (prune rules).

Predicted as Actual	Neptune	Smurf	Pod	Teardrop	Land
Neptune (16)	**16 (100%)**	0.00%	0.00%	0(0.00%)	0(0.00%)
Smurf (264)	0(0.00) %	**264 (100%)**	0(0.00%)	0(0.00%)	0(0.00%)
Pod (20)	0(0.00) %	0(0.00%)	**20(100) %**	0(0.00) %	0(0.00) %
Teardrop (99)	0(0.00) %	0(0.00%)	0(0.00) %	**99(100) %**	0(0.00) %
Land (1)	0(0.00) %	0(0.00) %	0(0.00) %	0(0.00) %	**1(100%)**

Table 26: Percentages of Correctly Classified Attacks in Table 21-25 (prune rule).

	Neptune	Smurf	Pod	Teardrop	Land
Table 21	**87.5%**	100%	90%	100%	1000%
Table 22	75%	**99.6%**	100%	100%	100%
Table 23	87.5%.5%	100%	**100%**	100%	100%
Table 24	87.5%	100%	100%	**100%**	100%
Table 25	87.5%	100%	100%	100%	**100%**

The results in Tables 21-25, were obtained from classification of training data set with pruned rule set, the pruned rule set gives a better results than the unpruned data set. All the attacks types except Neptune and pod were correctly classified (100%) for all the rules categories, pod was 90% correctly classifier with single attributes rules and 100% correctly classified with other four categories of rules. Neptune recorded 100% correct classification for 4 and 5 attributes combinational rules, 93% correct classification for 2 and 3 attributes combinational rules and 69% correctly classified for one attributes rules. Table 26 shows the summary of all the attacks correctly classified in Tables 21-25.

Implementation with Test Data

The association rule classifier was tested with test data that did not belong to the same network with the training dataset, there are three (3) (Appache, Mail bomb, Snmpget attacks in the test data that were not present in the training set. (Figures 1-7) shows the confusion matrix table obtained from the association rule classification of the test data

Discussion

Table 27: Confusion matrix obtained from one attribute combination from test dataset (unprune rules).

	Known Attacks						Unknown Attacks			
	Pod	Smurf	Teardrop	Neptune	Land	Normal	Appache	Mail bomb	Snmpget	Unknown
Pod (40)	0	40(100%)	0	0	0	0	0	0	0	0
Smurf (107)	0	107(100%)	0	0	0	0	0	0	0	0
Teardrop (9)	0	9(100%)	0	0	0	0	0	0	0	0
Neptume (44)	0	0	0	0	44(100%)	0	0	0	0	0
Land (9)	0	0	0	0	9(100%)	0	0	0	0	0
Normal (22)	0	3(13.6%)	0	3 (13.6%)	0	16(72%)	0	0	0	0
Appache (34)	0	0	0	2(6%)	32(94%)	0		0	0	0
Mailbomb (26)	0	26(100%)	0	0	0	0	0	0	0	0
Snmpget(8)	0	8(100%)	0		0	0	0	0	0	0

Table 28: Confusion matrix obtained from one and two attribute combination from test dataset (unprune rules).

	Pod	Smurf	Teardrop	Neptune	Land	Normal	Appache	Mail bomb	Snmpget	Unknown
Pod (40)	0	0	0	0	40(10%)	0	0	0	0	0
Smurf (107)	0	107(100%)	0	0	0	0	0	0	0	0
Teardrop (9)	0	0	7(88%)	0	2(12%)	0	0	0	0	0
Neptume (44)	0	0	0	0	44(10%)	0	0	0	0	0
Land (9)	0	0	0	0	9(100%)	0	0	0	0	0
Normal (22)	0	3(13.6%)	0	3 (13.6%)	0	16(72%)	0	0	0	0
Appache (34)	0	0	0	0	34(10%)	0	0	0	0	0
Mailbomb (26)	0	0	0	0	26(10%)	0	0	0	0	0
Snmpget(8)	0	8(100%)	0		0	0	0	0	0	0

Table 29: Confusion matrix obtained from one, two and three attribute combination from test dataset (unprune rules).

	Pod	Smurf	Teardrop	Neptune	Land	Normal	Appache	Mail bomb	Snmpget	Unknown
Pod (40)	0	40(100%)	0	0	0	0	0	0	0	0
Smurf (107)	0	107(100%)	0	0	0	0	0	0	0	0
Teardrop (9)	0	0	9(100%)	0	0	0	0	0	0	0
Neptume (44)	0	0	0	0	44(100%)	0	0	0	0	0
Land (9)	0	0	0	0	9(100%)	0	0	0	0	0
Normal (22)	0	2(9%)	0	3 (13.6%)	0	17(77%)	0	0	0	0
Appache (34)	0	0	0	0	34(10%)	0	0	0	0	0
Mailbomb (26)	0	0	0	0	26(10%)	0	0	0	0	0
Snmpget(8)	0	8(100%)	0		0	0	0	0	0	0

Table 30: Confusion matrix obtained from one, two, three and four attribute combination from test dataset (unprune rules).

	Pod	Smurf	Teardrop	Neptune	Land	Normal	Appache	Mail bomb	Snmpget	Unknown
Pod (40)	0	40(10%)	0	0	0	0	0	0	0	0
Smurf (107)	0	107(100%)	0	0	0	0	0	0	0	0
Teardrop (9)	0	0	9(100%)	0	0	0	0	0	0	0
Neptume (44)	0	0	0	10(22.7%)	34(77.8%)	0	0	0	0	0
Land (9)	0	0	0	0	9(100%)	0	0	0	0	0
Normal (22)	0	2(9)	0	3 (13.6%)	0	17(77%)	0	0	0	0
Appache (34)	0	0	0	0	34(100%)	0	0	0	0	0
Mailbomb (26)	0	0	0	0	26(100%)	0	0	0	0	0
Snmpget(8)	0	0	8(100%)		0	0	0	0	0	0

Table 31: Confusion matrix obtained from one, two, three, four and five attribute combination from test dataset (unprune rules).

	Pod	Smurf	Teardrop	Neptune	Land	Normal	Appache	Mail bomb	Snmpget	Unknown
Pod (40)	40(100%)	0	0	0	0	0	0	0	0	0
Smurf (107)	0	107(100%)	0	0	0	0	0	0	0	0
Teardrop (9)	0	0	9(100%)	0	0	0	0	0	0	0
Neptume (44)	0	0	0	10(22.7%)	34(77.8%)	0	0	0	0	0
Land (9)	0	0	0	0	9(100%)	0	0	0	0	0
Normal (22)	0	2(9%)	0	3 (13.6%)	0	17(77%)	0	0	0	0
Appache (34)	0	0	0	0	34(100%)	0	0	0	0	0
Mailbomb (26)	0	0	0	0	26(100%)	0	0	0	0	0
Snmpget(8)	0	8(100%)	0		0	0	0	0	0	0

The results in Tables 27-31 were obtained from classification of test data set with the raw unpruned rules. From the tables, Pod attacks were classified Teardrop and smurf attacks. Smurf and Teardrop attacks were 100% and 88% classified correctly respectively, all Neptume attacks were classified as Land attacks, all Land attacks were correctly classified, between 77% and 90% of Normal traffic were classified correctly. 94% and 6% of Appache attack were classified as Land and Neptume attack respectively. Smnpget attacks were classified as smurf and Teardrop attacks.

Table 32: Confusion matrix obtained from one attribute combination from test dataset (prune rules).

	Pod	Smurf	Teardrop	Neptune	Land	Normal	Appache	Mail bomb	Snmpget	Unknown
Pod (40)	4(10%)	0	34(85%)	0	0	0	0	0	0	2(5%)
Smurf (107)	0	103(96.3%)	0	0	0	0	0	0	0	4(3.7%)
Teardrop (9)	0	9(100%)	0	0	0	0	0	0	0	0
Neptume (44)	1(2.2%)	0	3(6.8%)	9(20.5%)	1(2.2%)5	0	0	1(2.2%)	0	29(65.9%)
Land (9)	0	0	1(11.1%)	1(11.1%)	7(77.7%)	0	0	0	0	0
Normal (22)	0	0	0	2 (9%)	0	20(91%)	0	0	0	0
Appache (34)	0	0	1(2.9%)	0	3(8.82%)	0		0	0	29(85.2%)
Mailbomb (26)	0	0	14(53.8%)	0	0	0	0	0	0	12(46.1%)
Snmpget(8)	0	8(100%)	0		0	0	0	0	0	0

Table 33: Confusion matrix obtained from one and two attribute combination from test dataset (prune rules).

	Pod	Smurf	Teardrop	Neptune	Land	Normal	Appache	Mail bomb	Snmpget	Unknown
Pod (40)	0	40(100%)	0	0	0	0	0	0	0	0
Smurf (107)	0	107(100%)	0	0	0	0	0	0	0	0
Teardrop (9)	0	2(12%)	7(88%)	0	0	0	0	0	0	0
Neptume (44)	0	0	0	44(100%)	0	0	0	0	0	0
Land (9)	0	0	0	0	9(100%)	0	0	0	0	0
Normal (22)	0	0	0	2 (9%)	0	20(91%)	0	0	0	0
Appache (34)	0	1(2.9%)	0	1(2.9%)	0	0	0	0	0	32(94.1%)

Table 34: Confusion matrix obtained from one, two and three attribute combination from test dataset (prune rules).

	Pod	Smurf	Teardrop	Neptune	Land	Normal	Appache	Mail bomb	Snmpget	Unknown
Pod (40)	38(95%)	2(5%)	0	0	0	0	0	0	0	0
Smurf (107)	0	107(100%)	0	0	0	0	0	0	0	0
Teardrop (9)	0	0	9(100%)	0	0	0	0	0	0	0
Neptume (44)	0	0	0	44(100%)	0	0	0	0	0	0
Land (9)	0	0	0	0	9(100%)	0	0	0	0	0
Normal (22)	0	0	0	2 (9%)	0	20(91%)	0	0	0	0
Appache (34)	0	1(2.9%)	0	1(2.9%)	0	0	0	0	0	32(94.1%)
Mailbomb (26)	0	10(138.5%)	0	0	0	0	0	0	0	16(61.5%)
Snmpget(8)	0	3(37.5%)	0		0	0	0	0	0	5(62.5%)

Table 35: Confusion matrix obtained from one, two, three and four attribute combination from test dataset (prune rules).

	Pod	Smurf	Teardrop	Neptune	Land	Normal	Appache	Mail bomb	Snmpget	Unknown
Pod (40)	38(95%)	2(5%)	0	0	0	0	0	0	0	0
Smurf (107)	0	107(100%)	0	0	0	0	0	0	0	0
Teardrop (9)	0	0	9(100%)	0	0	0	0	0	0	0
Neptume (44)	0	0	0	44(100%)	0	0	0	0	0	0
Land (9)	0	0	0	0	9(100%)	0	0	0	0	0
Normal (22)	0	0	0	2 (9%)	0	20(91%)	0	0	0	0
Appache (34)	0	1(2.9%)	0	1(2.9%)	0	0	0	0	0	32(94.1%)
Mailbomb (26)	0	10(138.5%)	0	0	0	0	0	0	0	16(61.5%)
Snmpget(8)	0	3(37.5%)	0		0	0	0	0	0	5(62.5%)

Citation: Fasidi FO, Adebayo OT. Detecting Distributed Denial-of-Service DDoS Attacks. Curr Tr Comp Sci & App 1(2)- 2019. CTCSA. MS.ID.000110. DOI: 10.32474/CTCSA.2019.01.000110.

Table 36: Confusion matrix obtained from one, two, three, four and five attribute combination from test dataset (prune rules).

	Pod	Smurf	Teardrop	Neptune	Land	Normal	Appache	Mail bomb	Snmpget	Unknown
Pod (40)	38(95%)	2(5%)	0	0	0	0	0	0	0	0
Smurf (107)	0	107(100%)	0	0	0	0	0	0	0	0
Teardrop (9)	0	0	9(100%)	0	0	0	0	0	0	0
Neptume (44)	0	0	0	14(100%)	0	0	0	0	0	0
Land (9)	0	0	0	0	9(100%)	0	0	0	0	0
Normal (22)	0	0	0	2 (9%)	0	20(91%)	0	0	0	0
Appache (34)	0	1(2.9%)	0	1(2.9%)	0	0	0	0	0	32(94.1%)
Mailbomb (26)	0	10(138.5%)	0	0	0	0	0	0	0	16(61.5%)
Snmpget(8)	0	3(37.5%)	0		0	0	0	0	0	5(62.5%)

The results in Tables 32-36 were obtained from classification of test data set with the raw pruned rules. 3,4, and 5 attributes rules classified Pod, Smurf Teardrop, Neptume and Land attacks correctly. Appache, mailbomb and snmpget attacks were classified as either Unknown, Smurf or Teardrop attacks. Table 37 shows the summary of all the correctly classified attacks. All the attacks present in the test dataset which were not used for training of the Association rule classifier were classified as attacks with unprune rule, the pruned rule classified these attacks as unknown attacks. Tables 38 & 39 below shows how they were classified [21-46].

Table 37: Summary Correctly Classsified Attacks from the Test Dataset.

	Pod	Smurf	Teardrop	Neptune	Land	Normal	Appache	Mail bomb	Snmpget
Table 32	0	96%	0	22.50%	77.70%	91%	0	0	0
Table 33	0	100%	88%	100%	100%	91%	0	0	0
Table 34	95%	100%	100%	100%	100%	91%	0	0	0
Table 35	95%	100%	100%	100%	100%	91%	0	0	0
Table 36	95%	100%	100%	100%	100%	91%	0	0	0

Table 38: Classification of Attacks not Present in the Test Data (unprune Rule).

	Pod	Smurf	Teardrop	Neptune	Land
Appache			1(2.9%)	1(2.9%)	34((100%)
mailbomb					26(100%)
Snmpget			8(100%)		

Table 39: Classification of Attacks not Present in the Test Data (prune Rule).

	Pod	Smurf	Teardrop	Neptune	Land	Unknown
Appache		2(6%)				32(94.1%)
mailbomb		10(38.5%)				16(61.5%)
Snmpget		3(62.5%)				5(62.5%)

Conclusion

The need for effective and efficient security on our system cannot be over-emphasized. This position is strengthened by the degree of human dependency on computer systems and the electronic superhighway (Internet) which grows in size and complexity on daily basis for business transactions, source of information or research. Association Rule methods of improving intrusion detection systems based on machine learning techniques were described and implemented on Intel Duo-core, CPU 2.88GHz, 1024MB RAM using Java programming language.

The work is motivated by increasing growth in importance of intrusion detection to the emerging information society. The research work provided background detail on intrusion detection techniques, and briefly described intrusion detection systems. In this research, an Association rule-based algorithm, was newly developed for mining known known-patterns. The results of the developed tools are satisfactory though it can be improved upon. These tools will go a long way in alleviating the problems of security of data by detecting security breaches on computer system.

References

1. Brignoli, Delio (2008) DDoS detection based on traffic self-similarity. University of Canterbury. Computer Science and Software Engineering.
2. Yeonhee Lee, Youngseok Lee (2011) Detecting DDoS Attacks with Hadoop.

3. Adetunmbi AO, Zhiwei Shi, Zhongzhi Shi, Adewale Olumide S (2006) Network Anomalous Intrusion Detection using Fuzzy-Bayes. IFIP International Federation for Information Processing, Intelligent Information Processing III, (Boston: Springer) 228: 525 -530.
4. Agrawal R, Sikrant R (1994) Fast Algorithms for mining association rules. In Proceedings of the 20th VLDB Conference, Santiago, Chile.
5. Aha D (1997) Lazy learning. In Artificial Intelligence. Kluwer Academic Publishers 11: 7-10.
6. Aha D, Kibler D, Albert M (1991) Instance-based learning algorithms. Machine Learning 6(1): 37-66.
7. Anderson D, Frivoid T, Valdes A (1995) Detecting Unusual Program Behaviour using the Statistical Component of the Next-generation Intrusion Detection Expert System (NIDES). Computer Science Laboratory SRI-CSL 95-06.
8. Bellovin SM (1993) Packets Found on an Internet. Computer Communications Review 23(3): 26-31.
9. Beverly R (2003) MS-SQL Slammer/Sapphire Traffic Analysis. MIT LCS.
10. Cohen WW (1995) Fast Effective Rule Induction. In Machine Learning: the 12th International Conference, Lake Tahoe, CA.
11. Cohen W (1995) Fast effective rule induction. In Machine Learning: Proceedings of the Twelfth International Conference.
12. Daelemans W, Van den Bosch, A Zavrel J (1999) Forgetting exceptions is harmful in language learning. In Machine Learning 34(1-3): 11-41.
13. Daelemans W, Zavrel J, Van Der Sloot K, Van Den Bosch A (2005) TiMBL: Tilburg Memory-Based Learning - Reference Guide. Induction of Linguistic Knowledge.
14. Deraison R (2003) Nessus.
15. Farmer D Venema W (1993) Improving the Security of Your Site by Breaking Into it.
16. Fayyad (1996a) Mining Scientific Data. Communications of the ACM 39(11): 51-57.
17. Fayyad (1996b) From Data Mining to Knowledge Discovery: An Overview. In Advances in Knowledge Discovery and Data Mining. Fayyad U, Piatesky-Shappiro G, Smyth P, Uthurusamy R (Eds.). AAAI/MIT Press, Cambridge, MA.
18. Fayyad, Gregory Piatetsky Shapiro and Padhraic Smyth (1996) The KDD Process of Extracting Useful Knowledge from Volumes of Data. Communications of the ACM 39(11): 27-34.
19. Floyd S, Paxson V (2001) Difficulties in Simulating the Internet. IEEE/ ACM Transactions on Networking.
20. Fyodor (1998) Remote OS detection via TCP/IP Stack Finger Printing.
21. Fyodor (2002) NMAP Idle Scan Technique (Simplified).
22. Fyodor (2003) NMAP.
23. Hall M, Smith L (1996) Practical feature subset selection for machine learning. In Proceedings of the Australian Computer Science Conference (University of Western Australia). University of Waikato.
24. Hendrickx I (2005) Local Classification and Global Estimation. Koninklijke drukkerij Broese & Peereboom.
25. Ilgun (1995) State Transition Analysis: A Rule-based Intrusion Detection Approach. IEEE Transactions on Software Engineering 21(3): 181-199.
26. Jonsson E, Olovsson T (1997) A Quantitative Model of Security Intrusion Process Based on Atacker Behaviour. IEEE Transactions on Software Engineering 23(4).
27. Kendall K (1999) A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems. Masters Thesis, Massachusetts Institute of Technology.
28. Kumar S, Spafford EH (1995) A Software Architecture to Support Misuse Intrusion Detection. In Proceedings of the 18th National Information Security Conference 1995: 194-204.
29. Lunt (1992) A Real-time Intrusion Detection Expert System (IDES). A Final Technical Report Technical Report. Computer science Laboratory, SRI International, Menlo Park, California, February 1992.
30. Lunt T (1993) Detecting Intruders in Computer Systems. In Proceedings of the 1993 Conference on Auditing and Computer Technology 1993.
31. Mitchell T (1997a) Does machine learning really work? In AI Magazine, p. 11-20.
32. Mitchell T (1997b) Machine Learning. McGraw-Hill International Editions.
33. Moore Paxson V, Savage S, Shannon C, Staniford S, Weaver N (2003) The Spread of the Sapphire/Slammer Worm. CAIDA, ICSI, Silicone Defense, UC Berkeley EECS and UC San Diego CSE.
34. Pietraszek T (2004) Using adaptive alert classification to reduce false positives in intrusion detection. In Recent Advances in Intrusion Detection 3224: 102-124.
35. Sanfilippo S (2003) HPING.
36. SANS (2000) An Analysis of the “Shaft” Distributed Denial of Service Tool. SANS Institute.
37. Spafford H (1998) The Internet Worm Program: An Analysis. Purdue Technical Report CSDTR-823.
38. Standler Ronald B (2001) Computer Crime.
39. Staniford S, Paxson V, Weaver N (2002) How to Own the Internet in your Spare Time. Proc. 11th USENIX Security Symposium.
40. Van Den Bosch A, Daelemans W (1998) Do not forget full memory in memory-based learning of word pronunciation. In NeMLaP3/CoNLL 98: 195-204.
41. Van Den Bosch A, Krahmer E, Swerts M (2001) Detecting problematic turns in human machine interactions: Rule-induction versus memory-based learning approaches. In Meeting of the Association for Computational Linguistics (2nd edn)., pp. 499-506.
42. Weiser Kevin (2002) Valve Launches Largest DDoS in History.
43. Wang Y, Wang, J, Miner A (2004) Anomaly intrusion detection system using one class SVM. 5th Annual IEEE Information Assurance Workshop, West Point, New York.
44. Warren RH, Johnson JA, Huang GH (2004) Application of Rough Sets to Environmental Engineering Models. JF Peters (Eds.).: Transaction on Rough Sets. LNCS Springer verlag Berlin Heidelberg 3100: 356-374.
45. William W Streilein, David J Fried, Robert K Cunningham (2005) Detecting Flood-based Denial-of-Service Attacks with SNMP/RMON.
46. Van Ryan, Jane S (1998) Center for Information Security Technolog Releases CMDS Version.

DOI: 10.32474/CTCSA.2019.01.000110