Journal of Theoretical and Applied Information Technology
30th November 2013. Vol. 57 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

# MALWARE DECTION USING IP FLOW LEVEL ATTRIBUTES

[1]AHMED ABDALLA, HAITHAM A. JAMIL, HAMZA AWAD HAMZA IBRAHIM, SULAIMAN MOHD NOR

[1]Faculty of Electrical Engineering, Universiti Teknologi Malaysia, Malaysia

E-mail: ahmedflaminko@gmail.com, haithamjamil@gmail.com, hamysra76@hotmail.com, s_mohdnor@utm.my

## ABSTRACT

Although the task of malware detection in network traffic had been done successfully through Deep Packet Inspection (DPI) in the last two decades, this approach is becoming less efficient due to the continuous increasing of network traffic volumes and speeds and concerns on user's privacy. The recent alternative approach is the flow-based detection which has the ability to inspect high speed and backbone network traffic because it significantly aggregates and reduces the inspected data. However, the capability of this approach to detect packet-based attacks such as viruses and trojans is questionable because of the absence of the actual data at the payload level. In this paper we proof through experiments the ability to detect network flows that contain malicious packets that had been previously marked as malicious by Snort using only flow level attributes using several Machine Learning (ML) classifiers. We created our dataset from captured traces of a subnet of our university's network. The detection accuracy is found to be 75% True Positive (TP) with almost zero False Negative which we consider as a verification of the capability of flow-based approach to detect malware. This finding is encouraging for future researches where it can be combined with more traditional detection methods to form more powerful NIDSs.

**Keywords:** *Network Intrusion Detection System (NIDS), Flow level network traffic inspection, Snort, Malware Detection, Machine Learning, NetFlow*

## 1. INTRODUCTION

Malware and intrusion detection in network traffic is a process that is crucial in any network today. With the ever increasing rate of attacks, malware and intrusion detection has become more significant. One of the most common and efficient worldwide acceptable Network Intrusion Detection System (NIDS) is Snort [1]. It is an open source and software based packet inspection system that can run on any general purpose computer. The inspection is based on matching every packet contents against a predefined set of rules that look for certain signature patterns in the packet header or packet payload to identify a malware. Although this approach is found to be very efficient as long as the rule set used is updated and contains the most recent version, at many times it is not feasible or not applicable at all. One of these situations is the case of malware detection in high speed networks when signature-based NIDS, such as Snort, cannot cope to inspect every passing by packet and run thousands of rules concurrently. Another case is when deep packet inspection is not desirable because of privacy concerns or not possible because of encryption.

An alternative approach is the flow level inspection [2-4]. In this approach the inspection is done for flow records created on the basis of connection level that may contain multiple of packets that represent a whole session between two communicating parties. In flow-level inspection, a (unidirectional) flow is defined as all packets that have the same 5-tuples in a certain period of time. These 5-tuples are; source and destination IP addresses, source and destination port number, and transport layer protocol. This approach significantly decreases the amount of inspected data. Moreover, most of core routers today are equipped with platforms that can be enabled and configured to create flow statistics records for all the traffic that forwarded in form of NetFlow [5]. Authors of [3] calculated the ratio between packets exported by NetFlow (containing the flow records) and the packets on their experimental network to be on average equal to 0.1% of the network load measured in bytes with the overhead due to NetFlow to be on average 0.2%. NetFlow protocol is originally innovated for network traffic monitoring and classification, but it has been found useful in flow-based NIDS.

The evolvement of NetFlow flow-based inspection is a feasible and cost effective solution for high speed networks and in cases of encrypted traffic. However, a question may arise here about the capability and the efficiency of this approach to detect malware. Although the malware exist in the payload, the question is will the flow features exhibit the presence of malware? In this paper we are working to verify that flow-based NIDS has the capability to highlight flows that contain packet payload attacks. In order to achieve this objective, this paper tries to answer these two questions; (i) is it possible to use only basic NetFlow attributes to detect IP flows that contain packet(s) with payload attack(s) currently detectable by the Snort NIDS? If so, how effective is it? (ii) what are the most significant attributes that can be used in flow-based malware detection? To answer these questions we create flow-based dataset and used several Machine Learning (ML) algorithms to build flow level classifiers. We regard the set of events alerted by Snort as representing the most complete available knowledge. The function of ML is to determine how best to reproduce the alerts at the flow level that approximate a packet signature. The outcome of these algorithms to reproduce alerts is found to be approximately 75% TP and almost zero FP.

The rest of our paper is organized as follows. We discuss related work in Section 2. In section 3 we describe the characteristics of packet level and flow level network attacks and justify the capability of flow level inspection to detect malware using ML algorithms. In section 4 we present our overall experiment framework and describe the environment and experiment setup. In section 5 we provide a comprehensive analysis of the alerts produced by Snort on the captured traces. Section 6 describes the preprocessing criteria to create and label the flow-based datasets from the captured traces and Snort alerts. In section 7 we present our basic and derivative features for our classifiers and justify our selection. We reviewed selected ML algorithms and performance evaluation methodology in section 8. We present various aspects of classification results in section 9 before we present our conclusion and suggestions for future work in section 10.

## 2. RELATED WORK

A lot of work had been done to build flow-based NIDSs and measure their efficiency to detect network attacks. The two most famous options for analyzing network traffic are the misuse approach and anomaly approach. Anomaly-based detection is done on the basis of monitoring network operation and flagging any deviation from network normal behavior as an indicator of an attack as reported in [6-13]. Misuse-based detection is done through comparing and matching network traffic event against a predefined set of database of signatures of known attacks as reported in [14-17].

Since we are questioning the capability and the efficiency of flow-level NIDS to detect malware in network traffic that could be detected through deep packet inspection, we use a different approach from all the above work. Rather than alarming a certain network event depending on its abnormality score from the baseline operation, we trained our ML classifiers to find rules that search flow features so as to reproduce alerts for flows which are correlated to the alerts at the packet level.

ML algorithms have been used heavily for network traffic classification. Several approaches has been evolved in this field including unsupervised learning of application classes via clustering of flow features and derivation of heuristics for packet-based identification [18]; semi-supervised learning from marked flow data [19], and supervised learning from flow features [20, 21].

## 3. NIDS: PACKET OR FLOW LEVEL?

Many previous works compared the two approaches of packet inspection and flow inspection on NIDS [2, 22, 23]. One of the major differences between them is the nature of the detectable attacks in each approach. Packet level inspection is ideal to detect malware attack which is defined as any malicious or unauthorized code or code parameter that can cause the attacked system to function undesirably. On the other hand, flow level inspection is ideal to monitor network traffic performance and detect any deviations from baseline operation that could be caused by network traffic attacks. These traffic attacks are often not dedicated to a specific machine but rather to a whole network such as DoS, scans, botnets. Some attacks may have these two characteristics (e.g. a malware that spreads through network machines causing network performance degradation). Hence, such types are detectable by both approaches. Figure 1 illustrates this concept. In this paper we are trying to proof that most of packet level detectable attacks are actually located in the shaded common area as illustrated in figure 1.

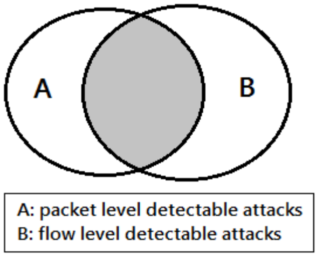
A
B
A: packet level detectable attacks
B: flow level detectable attacks

*Figure 1: Common Attacks That Are Detectable By Both Packet Level And Flow Level Inspection*

We claim that flow level inspection can also detect almost all malware attack types. Besides the fact that some of flow record information is also packet features, we rely on the assumption that there are some hidden correlations between packet payload and its flow record even though there may be no recognizable correlations implied in the original packet. This can be related to the attacker that tends to use certain elementary information when creating malware (e.g., malware traffic very often uses constant destination port, although that may not be specified in the signature).

We used ML algorithms because of their essential advantage to learn to characterize flows according to all obvious and hidden features included in the original packet-level signature.

## 4. GNERAL FRAMEWORK AND SETTINGS

The general framework of our experiment is shown in figure 2. This framework can be viewed as two sequential stages. The first stage aims to produce labeled NetFlow-based dataset from captured network traces. The labeling process depends on the alarms produced by Snort inspection on these captured traces for making the corresponding NetFlow records as either normal or malicious. A flow record is marked as malicious if it at least contains one packet that matched one of the Snort activated rules and sets an alarm accordingly.

The second stage is dedicated to use this dataset to train and test several ML classifiers to reproduce these alarms depending only on basic selected NetFlow features and finding the most significant attributes that affect the classification process.

Our captured traces were mirrored from a residential college network which is a subset of the university's campus network. The traces were captured using tcpdump tool and logged in pcap format. The whole captured traffic size was 1.3 G bytes which approximated to 4 million packets for a time window of 13 minutes. These pcap traces produced 466K unidirectional IP flows.

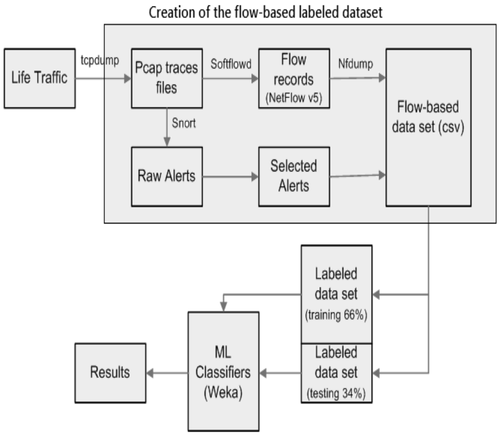
Creation of the flow-based labeled dataset
Life Traffic
tcpdump
Pcap traces files
Softflowd
Flow records (NetFlow v5)
Nfdump
Flow-based data set (csv)
Snort
Raw Alerts
Selected Alerts
Labeled data set (training 66%)
Labeled data set (testing 34%)
ML Classifiers (Weka)
Results

*Figure 2: General Experiment Framework*

To produce alerts, we used Snort engine version 2.5.9.3 and Snort rule set version 2.5.9.0. Since we are interested only in packet payload alerts, we set the configurations to enable Snort engine and disable both Snort decoder and Snort preprocessor. We include all rule files that are concerned for malware or network traffic attacks.

We made two variants of our experiment according to the number of rules activated in each variant. In the first variant, the number of rules activated was 3269. These rules are activated by default in the preferred setting of Sourcefire, the creator of Snort. These rules are claimed to have always 100% TP alert rate. In the second variant, we activated all rules in the included rule files and had 13120 rules which are four times the default.

## 5. ALERTS ANALYSIS

In the experiment for the first variant (with 3269 activated rules), Snort took four minutes to process all traces and produced 1111 alerts that originate from eleven distinct Snort signature IDs and six class types. Table 1 shows the results of these alerts for the given rules and classtypes.

The "classtype" keyword is used to categorize a rule as detecting an attack that is part of a more general type of attack class. Snort provides a default set of attack classes that are used by the default set of rules it provides. Defining classifications for rules provides a way to better organize the event data Snort produces. More information about different class types are available in [24].

*Table 1: First Variant Snort Alerts Analysis*

| | Signature ID | Rule File | Classtype | alerts |
|---|---|---|---|---|
| 1 | 3476 | WEB-IIS | Web-application | 210 |
| 2 | 14764 | WEB-ACTIVEX | Attempted-user | 1 |
| 3 | 15306 | FILE-IDENTIFY | Misc-activity | 3 |
| 4 | 15474 | BAD-TRAFFIC | Attempted-dos | 4 |
| 5 | 15709 | FILE-PDF | Attempted-user | 1 |
| 6 | 15912 | BAD-TRAFFIC | Attempted-dos | 9 |
| 7 | 18611 | WEB-MISC | Attempted-admin | 1 |
| 8 | 19187 | BAD-TRAFFIC | Attempted-user | 359 |
| 9 | 19653 | WEB-PHP | Web-application | 15 |
| 10 | 21355 | BAD-TRAFFIC | Attempted-recon | 504 |
| 11 | 23391 | BACKDOOR | Trojan-activity | 4 |
| **Total number of alerts** | | | | **1111** |

Figure 3 shows the distribution of these alerts over Snort class types, transport layer protocol and over application layer protocol.

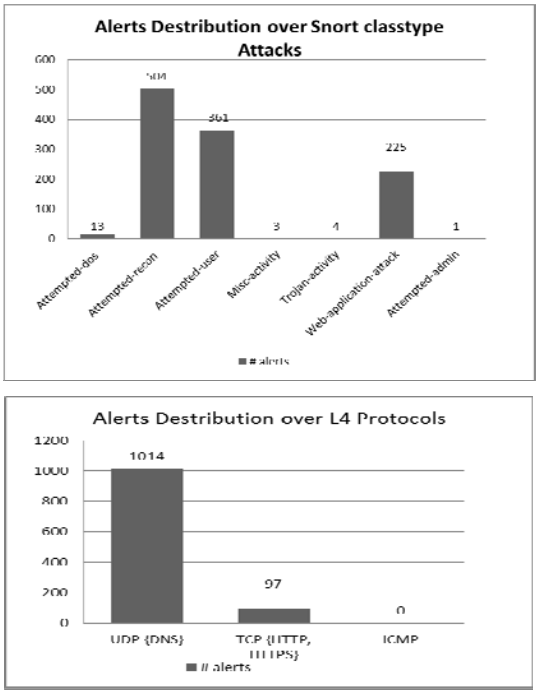

*Figure 3: Alerts Distribution Over Snort Class Types Attack And Over Layer 4 Protocols For Dataset1*

In the experiment for the second variant (with 13120 activated rules), Snort took more than 20 minutes to process all traces and produced 12480 alerts that originate from 55 distinct Snort signature IDs and 13 class types. Table 2 shows the results of these alerts for the given rules and classtypes.

*Table 2: Second Variant Snort Alerts Analysis*

| | Signature ID | Classtype | Alerts |
|---|---|---|---|
| 1 | 254 | bad-unknown | 2965 |
| 2 | 895 | trojan-activity | 18 |
| 3 | 1045 | web-application-attack | 4 |
| 4 | 1054 | web-application-attack | 11 |
| 5 | 1079 | web-application-activity | 13 |
| 6 | 1384 | denial-of-service | 4166 |
| 7 | 1388 | misc-attack | 41 |
| 8 | 1560 | web-application-activity | 2 |
| 9 | 1715 | web-application-activity | 1 |
| 10 | 1917 | network-scan | 2962 |
| 11 | 2381 | attempted-admin | 60 |
| 12 | 2441 | web-application-attack | 2 |
| 13 | 2707 | attempted-admin | 16 |
| 14 | 3550 | attempted-user | 2 |
| 15 | 3679 | attempted-user | 16 |
| 16 | 4135 | attempted-dos | 3 |
| 17 | 6690 | attempted-user | 1 |
| 18 | 7567 | successful-recon-limited | 10 |
| 19 | 8734 | successful-recon-limited | 157 |
| 20 | 10997 | misc-attack | 92 |
| 21 | 11257 | attempted-user | 22 |
| 22 | 11263 | attempted-dos | 389 |
| 23 | 11968 | protocol-command-decode | 3 |
| 23 | 12007 | protocol-command-decode | 5 |
| 25 | 12073 | protocol-command-decode | 17 |
| 26 | 12074 | protocol-command-decode | 2 |
| 27 | 12181 | protocol-command-decode | 47 |
| 28 | 13476 | web-application-attack | 57 |
| 29 | 15147 | attempted-user | 4 |
| 30 | 15167 | trojan-activity | 37 |
| 31 | 15168 | trojan-activity | 8 |
| 32 | 15699 | attempted-user | 2 |
| 33 | 15912 | attempted-dos | 16 |
| 34 | 16079 | web-application-attack | 3 |
| 35 | 16301 | attempted-user | 7 |
| 36 | 16482 | attempted-dos | 1 |
| 37 | 17294 | attempted-dos | 19 |
| 38 | 17410 | attempted-user | 52 |
| 39 | 17487 | attempted-dos | 2 |
| 40 | 17567 | attempted-admin | 11 |
| 41 | 17579 | attempted-user | 16 |
| 42 | 17750 | attempted-dos | 5 |
| 43 | 18611 | attempted-admin | 2 |
| 44 | 19187 | attempted-user | 747 |
| 45 | 19894 | attempted-user | 42 |
| 46 | 19996 | trojan-activity | 1 |
| 47 | 20242 | attempted-admin | 2 |
| 48 | 20278 | attempted-user | 4 |
| 49 | 21669 | attempted-dos | 129 |
| 50 | 21817 | attempted-dos | 65 |
| 51 | 23041 | trojan-activity | 2 |
| 52 | 23246 | trojan-activity | 2965 |
| 53 | 23408 | attempted-dos | 18 |
| 54 | 23861 | attempted-user | 4 |
| 55 | 26554 | trojan-activity | 11 |
| **Total number of alerts** | | | **12480** |

Figure 4 shows the distribution of these alerts over Snort alerts types, transport layer protocol and over application layer protocol.

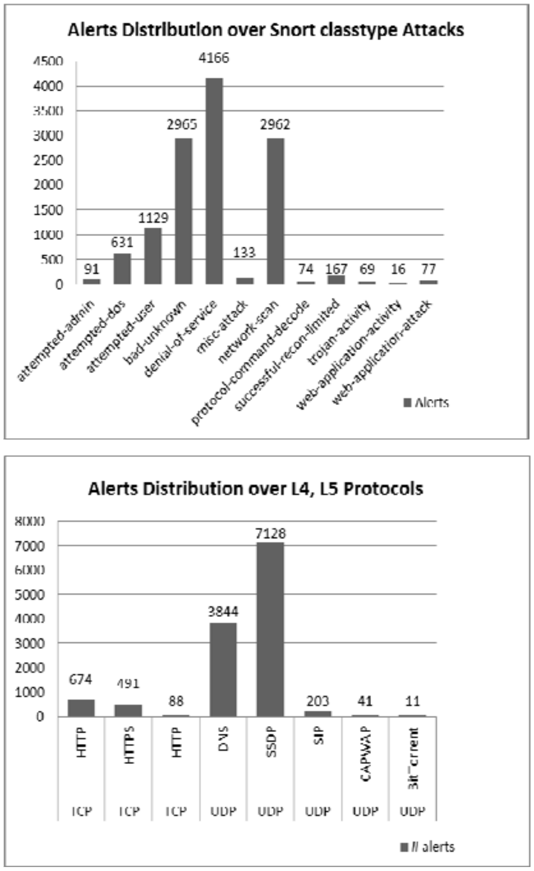

*Figure 4: Alerts Distribution Over Snort Class Types Attack And Over Layer 4 Protocols For Dataset2*

## 6. DATASET CREATION AND LABELING

In order to get the corresponding flows of our captured traces we used softflowd [25] as a software flow exporter. Softflowd is a flow-based network traffic analyzer capable of Cisco NetFlow data export. Softflowd semi-statefully tracks traffic flows recorded by listening on a network interface or by reading a packet capture file. These flows may be reported via NetFlow to a collecting host or summarized within softflowd itself. Softflowd supports data export using versions 1, 5 or 9 of the NetFlow protocol.

We used softflowd-0.9.8 to read our captured packet traces and to export version 5 NetFlow records. To collect and process these NetFlow data, we use Nfdump tools [26] version 1.5.8. The captured traces produced a total number of 466K unidirectional flows in NetFlow v5 format. Figure 5 shows the distribution of these flows over transport layer protocols.

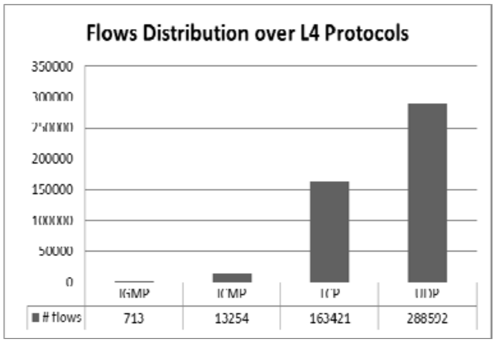

*Figure 5: Flow Distribution Of Captured Traces Over Transport Layer Protocols*

As had been mentioned in the previous section, we need concurrent packet and flow traces so that the alerts that Snort made on packets can be associated with its corresponding flow record for the same connection. In other words, if Snort has raised an alert on a packet at time *t* then we search for the flow with the same IP 5-tuples, flow start time $Ts$, and flow end time $Te$ such that $Ts \leq t \leq Te$ and marked that flow as malicious. One packet may produce multiple Snort alerts, and one flow will often correspond to multiple packets, which means that individual flows can be associated with many Snort alerts.

A Visual Basic user program is written to associate Snort alerts with its corresponding flow record. Since we have two variants of alert sets, we also made two variants of datasets; dataset1 and dataset2.

In dataset1, we used the 1111 alerts to mark their corresponding flow records as malicious and mark the rest which have no match alert as benign. The labeling process produced 1063 malicious flow out of the total 466K flow. All resulted flows resulted from the captured traces are included in the dataset. The percentage of malicious flows is only 0.23%.

In dataset2, 12480 alerts are used for the labeling process which produced 7844 malicious flows. In this variant of dataset we tried to increase the ratio of malicious flows in order to get a more realistic classification results. Hence, all flows that have protocol or service number which had never raised a Snort alarm is removed from the labeled dataset. The remaining 221K flows consist only of TCP, UDP protocols and service of (53, 80, 443, 1900, 5060, 5247, 6881, and 8080) HTTP, HTTPS, and DNS. Hence the percentage of malicious flows now has increase to 3.5%. Figure 6 shows the distribution of malicious flows over transport and application layer protocols.

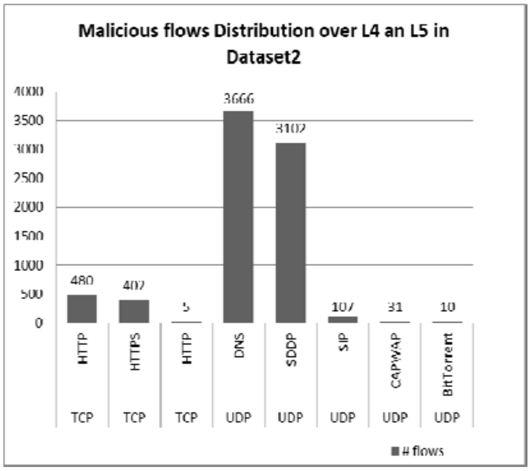

*Figure 6: Malicious Flows Distribution Over Transport And Application Layer In Dataset2*

## 7. FEATURE SELECTION

Since NetFlow is originally designed for network performance monitoring, not all its record fields are of benefit in NIDS. There are some fields that are not used (pad1, pad2) and there are others that have no benefit in intrusion detection (e.g., nexthop, autonomous system number of the source, etc.). Out of the 20 fields that exist in v5 NetFlow record, only eleven are found to be useful in NIDS. We decided to neglect three additional attributes from these eleven, which are the source and destination IP addresses and TOS. Our decision not to use IP addresses is due to privacy concerns that may sometimes prevent using this information and also because we want our classification to be network independent. We choose to remove TOS from dataset attributes because this field is often set to zero and not used in IP protocol. Hence, we chose to use only six basic NetFlow fields for our dataset. These fields are; flow start and end time, protocol, source and destination port, number of packets, number of bytes, accumulative OR of TCP flags. We chose to derive a flow duration attribute to replace the two basic features flow start time and flow end time. We derived three additional attributes that had been used frequently in flow-based NIDS which are; average packet length (in bytes), average bit rate, average packet rate. Table 3 shows the final ten attributes of our datasets and Table 4 shows the derived attributes.

*Table 3: Final Ten Attributes Selected For Datasets*

| | Attribute | Basic | Description |
|---|---|---|---|
| 1 | Pro | Basic | IP protocol type |
| 2 | Srp | Basic | TCP/UDP source port number or equivalent |
| 3 | Dsp | Basic | TCP/UDP destination port number or equivalent |
| 4 | Pkt | Basic | Number of Packets in the flow |
| 5 | Byt | Basic | Number of Layer 3 bytes in the packets of the flow |
| 6 | Flg | Basic | Cumulative OR of TCP flags |
| 7 | Dur | Derived | Flow duration |
| 8 | Bps | Derived | Average byte rate |
| 9 | pps | Derived | Average packet rate |
| 10 | Bpp | Derived | Average packet length in bytes |

*Table 4: Derived Attributes*

| | Attribute | Attributes derived from | Derivation Formula |
|---|---|---|---|
| 1 | Dur | Ts, Te<br>(flow start tim and end time) | Te - Ts |
| 2 | Bps | Byt, Dur | Byt ÷ Dur |
| 3 | pps | Pkt, Dur | Pkt ÷ Dur |
| 4 | Bpp | Byt, Pkt | Byt ÷ Pkt |

## 8. MACHINE LEARNING ALGORITHMS

We used eight data mining machine learning algorithms in WEKA program (Waikato Environment for Knowledge Analysis) [27] to evaluate the efficiency of flow-based IP flow features to reproduce Snort malware alerts. The selected ML algorithms are from three different classifiers types; bayes, rules and trees. The machine learning algorithm selected from bayes classifiers is Bayes Net [28]. Algorithms selected from rules classifiers are Decision Table, JRip, PART [28], whilst those selected from trees classifiers are J48 decision tree [29], Random Forest [30], Random Tree [31] and REPTree [32]. We split our dataset into two parts, training set (66%) and testing set (34%) to be used in all these classifiers.

Any classifier algorithm has the possibility of single detection for either attack or normal which can be represented by four different outcomes as shown in table 5. The evaluating parameters used to measure the detection efficiency are True Positive (TP) rate and False Positive (FP) rate.

*Table 5: Single Detection Possibilities*

| | Predicted Normal | Predicted Attack |
|---|---|---|
| Actual Normal | TN | FP |
| Actual Attack | FN | TP |

TP rate, (or detection rate), is defined as the proportion of detected attacks over all the attacks. It is calculated according to the formula:

TP rate (Detection rate) = TP/ (TP+FN)*100%

False Positive rate FP: FP rate is the percentage of normal data which is falsely classified as attacks. It is calculated according to the formula:

FP rate = FP/ (FP + TN)*100%

## 9. RESULTS AND DISCUSSION

### 9.1. Full Set Attributes Classification Results

First we used dataset1 which compose of 466K flow records from which there are 1063 are marked as malicious. Figure 7 shows the complete ten features classification results for the selected eight algorithms.

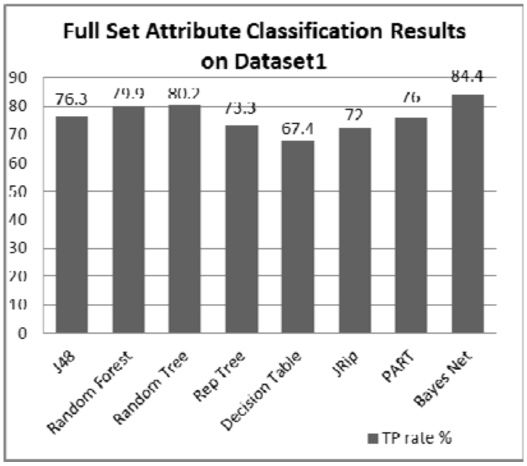

*Figure 7: Full Set Attribute ML Classification Results On Dataset1*

As shown in figure 6, most of the classifiers result in TP rate greater than 75%. Bayes Net had the highest TP rate (0.84%), but it also has the highest FP rate. In general, it is observed that the FP rate is very small and is almost zero for all the classifiers. This can be attributed to the low ratio of malicious flows (0.23%) compared to the benign flows. This is the reason why we increase the ratio of malicious flows in dataset2.

Dataset2 is used to validate our first results and to see if we can obtain the same results. Dataset2 composes of 221K flow records from which there are 7844 flows which are marked as malicious. Figure 8 shows the complete ten features classification results for the selected eight algorithms.

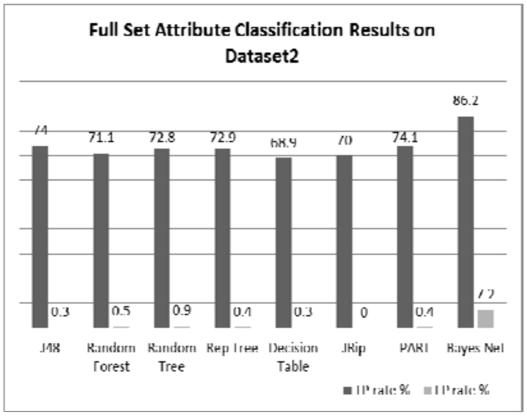

*Figure 8: Full Set Attribute ML Classification Results On Dataset2*

It is remarkable that the TP rate of most of the classifiers is slightly less than 75%. This is because in labeling the malicious flows of dataset2, we depended on all Snort rules where most of these rules have reliability less than 100% in producing alerts. The FP rate is also greater than that resulted from dataset1. That is because the classifier job in dataset2 became more challenging. The removal of flows with port numbers that had never set a Snort alarm in the packet traces causes TN rate to decrease and in turn increase the TP rate. Since we think the results of dataset2 are more realistic and the degradation of accuracy is not that much, we made the following analysis on the basis of dataset2.

### 9.2. Minimum Effective Attributes

Here we searched for the minimum flow attribute set that gives the maximum TP rate for each classifier and does not increase FP rate. Table 5 shows the minimum effective attributes and the maximum TP achieved for each classifier.

*Table 5: Minimum Effective Attributes*

| ML Classifier<br>Attribute | | J<br>4<br>8 | R<br>M<br>F | R<br>M<br>T | R<br>P<br>T | D<br>N<br>T | J<br>R<br>P | P<br>R<br>T | B<br>S<br>N |
|---|---|---|---|---|---|---|---|---|---|
| Dur | | | | | | | | | ☑ |
| Pro | | | | | | | | | ☑ |
| Srp | | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ |
| Dsp | | ☑ | | | | | ☑ | ☑ | ☑ |
| Pkt | | | | | | | | | ☑ |
| Byt | | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ |
| bps | | ☑ | ☑ | | ☑ | | ☑ | ☑ | ☑ |
| pps | | | | ☑ | | | | | ☑ |
| Bpp | | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ |
| Flg | A | ☑ | | ☑ | | | ☑ | | ☑ |
| | S | ☑ | ☑ | ☑ | | | ☑ | ☑ | ☑ |
| | F | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ |
| | R | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ |
| **Max TP achieved** | | **74.1** | **75.2** | **76.1** | **74.8** | **69.9** | **72.3** | **75.3** | **85.2** |

### 9.3. Most Significant Attributes

Here we searched for the key attributes that give each classifier its major classification efficiency. We consider the common minimum effective attributes in each classifier which are source port (Sp), number of bytes (Byt), and average packet size (Bpp). Figure 9 shows the classifiers results using only these three attributes and all possible two combinations of them.

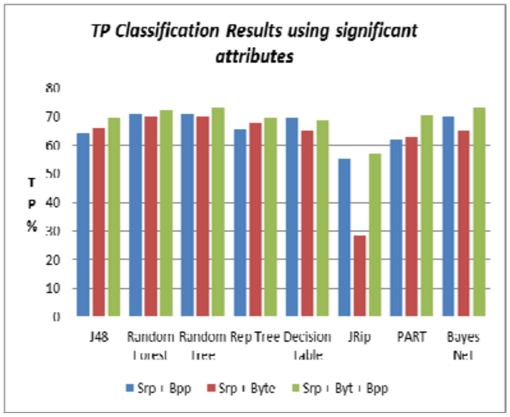

*Figure 9: Most Three Significant Classification Results On Dataset2*

Finally, we assess the individual impact of each attribute on the classification results. Figure 10 shows these results.

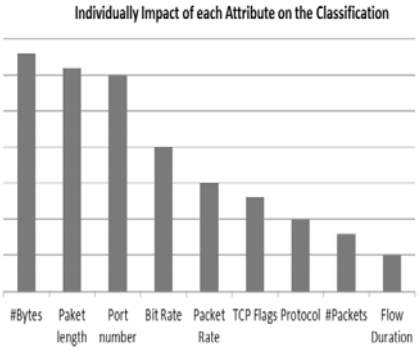

*Figure 10: Individual Impact Of Each Attribute On Classification*

## 10. CONCLUSION AND FUTURE WORK

From the results obtained we have shown that malware detection can be done based on the flow level attributes with very good detection performance (approximately 0.75 detection rate). Service port number, average packet length and number of bytes are the most significant attributes for detecting snort malware when using flow level NIDS.

Our work is not seeking to replace packet level with flow level. It is better viewed as a step towards an integrated, complementary approach of top-down inspection that leverages flow level approach characteristics from the wider view with less computational resources in high speed networks. It is an initial stage of intrusion detection then forwards the highlighted flows for the deep packet inspection to specify the exact malicious packet and the type of attack. In this way we can obtain the large volume real-time detection complementing conventional NIDS approach.

## REFERENCES:

[1] M. Roesch, "Snort-lightweight intrusion detection for networks," in *Proceedings of the 13th USENIX conference on System administration*, 1999, pp. 229-238.

[2] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An overview of IP flow-based intrusion detection," *Communications Surveys & Tutorials, IEEE,* vol. 12, pp. 343-356, 2010.

[3] A. Sperotto and A. Pras, "Flow-based intrusion detection," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, 2011, pp. 958-963.

[4] J. A. Copeland III, "Flow-based detection of network intrusions," ed: Google Patents, 2007.

[5] Cisco. (2013, 15/7/2013). *Cisco IOS NetFlow*. Available: http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html

[6] Z. Li, Y. Gao, and Y. Chen, "Towards a high-speed router-based anomaly/intrusion detection system," ed, 2005.

[7] Y. Gao, Z. Li, and Y. Chen, "A dos resilient flow-level intrusion detection approach for high-speed networks," in *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on*, 2006, pp. 39-39.

[8] A. Lakhina, M. Crovella, and C. Diot, "Characterization of network-wide anomalies in traffic flows," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, 2004, pp. 201-206.

[9] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," in *ACM SIGCOMM Computer Communication Review*, 2005, pp. 217-228.

[10] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, and N. Taft, *Structural analysis of network traffic flows* vol. 32: ACM, 2004.

[11] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," in *ACM SIGCOMM Computer Communication Review*, 2004, pp. 219-230.

[12] A. Wagner and B. Plattner, "Entropy based worm and anomaly detection in fast IP networks," in *Enabling Technologies: Infrastructure for Collaborative Enterprise, 2005. 14th IEEE International Workshops on*, 2005, pp. 172-177.

[13] M. P. Stoecklin, J.-Y. Le Boudec, and A. Kind, "A two-layered anomaly detection technique based on multi-modal flow behavior models," in *Passive and Active Network Measurement*, ed: Springer, 2008, pp. 212-221.

[14] M.-S. Kim, H.-J. Kong, S.-C. Hong, S.-H. Chung, and J. W. Hong, "A flow-based method for abnormal network traffic detection," in *Network Operations and Management Symposium, 2004. NOMS 2004. IEEE/IFIP*, 2004, pp. 599-612.

[15] C. Gates, J. J. McNutt, J. B. Kadane, and M. I. Kellner, "Scan detection on very large networks using logistic regression modeling," in *Computers and Communications, 2006. ISCC'06. Proceedings. 11th IEEE Symposium on*, 2006, pp. 402-408.

[16] F. Dressler, W. Jaegers, and R. German, "Flow-based worm detection using correlated honeypot logs," in *Communication in Distributed Systems (KiVS), 2007 ITG-GI Conference*, 2007, pp. 1-6.

[17] A. Karasaridis, B. Rexroad, and D. Hoeflin, "Wide-scale botnet detection and characterization," in *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, 2007.

[18] L. Bernaille, R. Teixeira, and K. Salamatian, "Early application identification," in *Proceedings of the 2006 ACM CoNEXT conference*, 2006, p. 6.

[19] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, "Offline/realtime traffic classification using semi-supervised learning," *Performance Evaluation*, vol. 64, pp. 1194-1213, 2007.

[20] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *ACM SIGMETRICS Performance Evaluation Review*, 2005, pp. 50-60.

[21] H. Jiang, A. W. Moore, Z. Ge, S. Jin, and J. Wang, "Lightweight application classification for network management," in *Proceedings of the 2007 SIGCOMM workshop on Internet network management*, 2007, pp. 299-304.

[22] H. M. Alaidaros, M. Mahmuddin, and A. Al Mazari, "From Packet-based Towards Hybrid Packet-based and Flow-based Monitoring for Efficient Intrusion Detection: An overview," 2012.

[23] G. Schaffrath and B. Stiller, "Conceptual integration of flow-based and packet-based network intrusion detection," in *Resilient Networks and Services*, ed: Springer, 2008, pp. 190-194.

[24] S. Incorporation. (2013). *SNORT Users Manual 2.9.5*. Available: http://manual.snort.org/

[25] Softflowd. (2013). *Softflowd*. Available: http://www.mindrot.org/projects/softflowd/

[26] (2013). *Nfdump*. Available: http://nfdump.sourceforge.net/

[27] S. R. Garner, "Weka: The waikato environment for knowledge analysis," in *Proceedings of the New Zealand computer science research students conference*, 1995, pp. 57-64.

[28] S. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Frontiers in Artificial Intelligence and Applications,* vol. 160, p. 3, 2007.

[29] J. R. Quinlan, *C4. 5: programs for machine learning* vol. 1: Morgan kaufmann, 1993.

[30] L. Breiman, "Random forests," *Machine learning,* vol. 45, pp. 5-32, 2001.

[31] D. Aldous, "The continuum random tree. I," *The Annals of Probability,* pp. 1-28, 1991.

[32] J. Park, H.-R. Tyan, and C.-C. Kuo, "Internet traffic classification for scalable qos provision," in *Multimedia and Expo, 2006 IEEE International Conference on*, 2006, pp. 1221-1224.