Flow-Based Approach on Bro Intrusion Detection

Hashem Alaidaros, M. Mahmuddin
School of Computing, College of Arts and Sciences,
Universiti Utara Malaysia, Kedah, Malaysia.
ady@uum.edu.my

***Abstract*—Packet-based or Deep Packet Inspection (DPI) intrusion detection systems (IDSs) face challenges when coping with high volume of traffic. Processing every payload on the wire degrades the performance of intrusion detection. This paper aims to develop a model for reducing the amount of data to be processed by intrusion detection using flow-based approach. We investigated the detection accuracy of this approach via implementation of this model using Bro IDS. Bro was used to generate malicious features from several recent labeled datasets. Then, the model made use the machine learning classification algorithms for attribute evaluation and Bro policy scripts for detecting malicious flows. Based on our experiments, the findings showed that flow-based detection was able to identify the presence of all malicious activities. This verifies the capability of this approach to detect malicious flows with high accuracy. However, this approach generated a significant number of false positive alarms. This indicates that for detection purpose, it is difficult to make a complete behavior of the malicious activities from only limited data and flow-level.**

***Index Terms*—Flow-Based Detection; Bro Intrusion Detection System; Machine Learning; Public Datasets.**

I. INTRODUCTION

The rapid increase of computers, users, and services connected to the Internet has resulted in the growth of Internet traffic volume and bandwidth in wide area networks (WAN). Continuous growth of network traffic presents serious threats against the security protection devices such as network intrusion detection systems (IDSs) [1]. With the increase in network volume and speed, existing network IDSs, which analyses per packet based, face challenges when capturing full payload traffic for malicious inspection, which in turn affects the performance and accuracy of IDS.

This issue motivates us to introduce flow-based IDS approach, which reduces the amount of data to be analyzed by looking at aggregated information of related packets in the form of flow. In this paper, we studied how the flow-based IDS approach can detect certain malicious activities, using open source: Bro IDS [2]. We used Bro as it provides comprehensive data analysis and it is able to deal with large datasets. Bro traffic analysis was applied on several labeled datasets to generate malicious features. From these malicious features, we used machine learning algorithms to generate the most important attributes and rules that will be useful for implementing flow detection method.

For validating detection methods, we used different recent labeled datasets. This paper emphasizes how the false positive rate impacts on the flow-based detection compared with the packet-based detection. However, the goal of this paper is not to develop an algorithm for detection intrusions, but to focus on studying the accuracy of the flow-based detection.

This paper is organized as follows: Section II presents an overview of packet-based and flow-based IDS. Section III presents the design of Bro flow-based detection method. In this section, more details on how detection algorithm is developed will be described. Then, in Section IV, we explain how to evaluate this method on the testbed with the real traffic datasets used in this paper. Finally, while the result and discussion are presented on section V, Section VI presents the conclusion and the future works.

II. OVERVIEW ON PACKET-BASED AND FLOW-BASED

A. Packet-based IDS

In packet-based, also named Deep Packet Inspection (DPI), a detecting engine has to scan and analyze per packet header and payload to determine whether the packet is an intrusion or not. This approach is mostly used by signature-based IDS, which compares what it analyzed to the given list of signatures in the database. With an increasing data volume in the traffic, the challenges of packet-based NIDS increase. This is because vast amount of data requires vast amount of computational performance, particularly complex algorithms.

Moreover, a drop of packets, resources consumption, and missing potential intrusions will occur if the NIDS is not able to let the analysis process to be done [3]. As it is very time consuming, it is hard or even impossible to perform packet-based approach in this environment [1]. Generally, the advantage of using packet-based IDS in an ideal environment is that all common types of known attacks can be detected since packets contain all complete data up to the application layer (layer 7 in OSI).

B. Flow-based IDS

With the issues of packet-based NIDS, researchers had to find alternative approach that receives little amount of data, while not compromising the accuracy. The candidate alternative that attracts the attention of researchers is flow-based NIDS technique. A flow-based IDS does not look at the payload content for inspection and analysis; however, it relies on information and statistics of network flows. Such information includes number of packets and bytes transferred over a particular time and start and end time of a flow.

A flow can be defined as a unidirectional data stream between two computer systems, where all transmitted packets of this stream share the following 5-tubles: IP source and destination address, source and destination port number and protocol type [4]. Nowadays, routers are equipped with the ability to be configured to generate flow statistics records in the form of Netflow [4].

Flow-based NIDS consists of the following components: an exporter (flow aggregator), a collector, and an analyzer. Flow aggregator creates flow records by the accounting

traffic statistics from aggregating relevant packets that share certain flow keys. The collector's task is to retrieve the flows, and then store and organize them in suitable format for NIDS for further analysis. The analyzer (detection engine) or NIDS then accesses the collector to analyze and process the flows for suspicious detection. It then sends the decision to the reporter, and commands are sent to other device, such as firewall to filter or block the malicious traffic.

NIDS in flow-based analyzes small amount of data (flow records). Thus, computational process appeared in packet-based is not a primary concern. For example, based on our experiments, the flow-based detection CPU consumption was decreased by 30% compared to packet-based detection in identical environment. However, it seems that the overhead and computational resources consumption in the detection analysis disappeared, but on the cost of flow exporting process. This is because the exporter, or Netflow device, such as router offloads and absorbs the task from NIDS itself. Several works attempt to improve the performance of flow aggregation process, such as [5] and [6].

Since flow records contain aggregated data up to transport layer, this issue encourages researchers to enhance flow-based detection accuracy and reduce the false negatives. Researchers have achieved promising results to detect attacks (such as DoS, worms, SSH etc.) by focusing on this approach only [7] [8] [9]. The author of [10] presents a detailed overview of these attacks. This paper differs from the other works in that we utilized the existing recent labeled dataset for deriving Bro detection scripts. In addition, we employed Bro for traffic investigation and analysis since it has potential to produce efficient performance [11]. For more details on the performance of the packet-based and flow-based IDS, refer to [12].

III. DESIGN

In this section, we present the design and implementation of the flow-based detection (see Figure 1) using Bro IDS [2] to study the detection accuracy impacts on flow-based detection. Bro is a Unix-based open-source network intrusion detection system that monitors, analyze, and inspects all traffic to detect suspicious activity even in high-speed network. Researchers prefer using Bro as it provides more flexibility in defining policy and scripting rules, and it comes with a large set of pre-built functions. Thus, we can put Bro in novel ways by extending Bro script and writing own code to satisfy our environment. In addition, Bro is an excellent choice for feature extraction [13].

In Bro, real-time highly structured log files (can be used for digital forensic analysis or later research analysis) are broken down by protocols, and alerts are written in plain text ASCII to take further actions. Basically, we analyze the outbound traffic coming from the internal network to detect possible intrusive activities performed by local machines. The output of this processing is the list of machines IP addresses, which performs malicious activities. The monitored and local hosts have been configured in Bro.

As shown in Figure 1(a), all incoming packets are passed to packet-based detection. While the Bro in packet-based listens directly from network interface, the Bro in flow-based listens from the collector to receive the flow records as input (see Figure 1b). For the flow detection, we build Bro detection scripts using statistical analysis of the traffic within a flow. Before we present how we implement these bro script detections for both the flow and packet detection, discussion on the datasets used in this paper is presented in the next subsection.

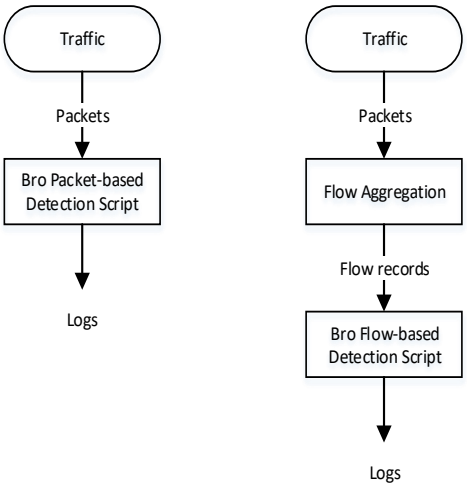

Figure 1: Illustration of (a) Packet-based and (b) Flow-based detection.

A. Data Collection

The most challenging validating detection method is the lack of standard public datasets [14]. DARPA 1999, which is believed to be the most standard public trace was criticized due to its age and inability to reflect real-world traffic by [15]. This urges us to find a variety of newer datasets to assess and verify the accuracy of our detection algorithms. Only small numbers of datasets are publicly available. Although we did not receive responses from many public datasets creators, we managed to get datasets presented in Table 1, which are made public for research community and will be used in this paper.

The information Security and Object Technology (ISOT) dataset, generated by University of Victoria on 2011 has a combination of malicious and normal datasets [16]. The other datasets are generated from Czech Technical University (CTU) in different scenarios and published in 2013. We selected four individual scenarios: CTU-50 [17], CTU-51 [18], CTU-52 [19], and CTU-53 [20] from CTU datasets. Explanations of ISOT datasets and CTU scenarios by their authors are found on [21] and [22] respectively.

These datasets consist of real traffic in PCAP format. However, these datasets are labeled, i.e. IP addresses of malicious and non-malicious hosts are known. Labeling this traffic helps us to validate the accuracy of the detection methods [23]. These datasets have been used in several researches [21], [24]. Table 2 shows the statistical details of each dataset.

However, these existing recent public datasets are limited to certain type of attacks. Based on the literature [11], flow-based detection gives promising results when detecting botnets activities that perform repetitive traffic patterns (e.g. when a bot infected machines connects frequently to Command & Control (C&C) server to receive commands, when bot spammer sends many emails to SMTP servers, or when "keep-alive" sent from time to time for IRC botnet).

With these points in mind, and rather than reviewing this work on general botnet detection, we selected the following bot-related malicious types to be considered in this work: Spam bots, IRC bots, and P2P bots. For more details on the characteristics of these malicious activities, refer to [25]. Spam activities are found in ISOT and CTU-50 datasets. For IRC-bot, it was performed on CTU-51 and CTU-52 datasets.

Finally, P2P bot activities are generated on both ISOT and CTU-53 datasets.

For the non-malicious traffic, unfortunately, all the previous datasets mentioned in Table 1 (except ISOT dataset) do not contain full-payload background traffic for privacy reasons. For this, we used a one-day complete payload trace captured at Alfaisal University, Prince Sultan College Jeddah (PSCJ), Information Technology Center, at the main gateway link that connects hundreds of hosts with an educational network to the Internet. The size of this trace is 2 GB and contains eight millions packets, corresponding to 478K flows. We named this trace as "PSCJ" and it contains a variety of network activities. Similar to non-malicious ISOT, PSCJ dataset involves everyday activity usage such as HTTP web behavior, popular sharing file packets, IRC traffic, emails, and streaming media. However, PSCJ trace is combined and injected along with each of the datasets listed in Table 1.

Table 1
Total number of packets, and flows on datasets

Datasets	Total # Packets	Total # Exported Flows
ISOT	157 millions	5.2 millions
CTU-50	2.1 millions	159,704
CTU-51	66.3 millions	31.7 millions
CTU-52	3.9 millions	1.7 millions
CTU-53	351,537	11,117

Table 2
Datasets traffic statistic

Datasets	Average bytes per flow	Average packets per flow	Average flow Duration (sec.)
ISOT	25,196	44	11.32
CTU-50	10,438	21	8.94
CTU-51	2,161	2	0.06
CTU-52	2,311	2	0.04
CTU-53	46,063	57	223.27

B. Bro Detection Scripts Derivation

For developing Bro detection scripts, we utilized the existing labeled datasets and we used Bro along with machine learning to collect and extract malicious flow and packet features from these datasets. Figure 2 illustrates the steps for deriving Bro detection scripts for each individual malicious type. In this paper, we applied several labeled datasets (as shown the Figure 2) to build detection scripts on each malicious activity.

a. Bro Analysis

Bro performs packet and flow analysis against the given dataset. Non-malicious and malicious dataset are combined and replayed. Malicious dataset selection depends on the type of malicious activity type. In this case, CTU-50, CTU-52, and CTU-53 datasets are used for spam, IRC-bot, and P2P-bot respectively. For non-malicious traffic, PSCJ trace was used. For each malicious type, the combined dataset was converted into network flows and then, the flows were used as input for Bro.

In addition, non-relevant packets and flows within the datasets were eliminated using filtering scripts to limit the amount of packet processing of these datasets. These filtering scripts are based on known infected and normal machines IP addresses, port numbers, and protocols, depending on the malicious activities selection. The output of this step is the logs (such as weird.log, notice.log, connection.log, smtp.log, netflow.log etc.,) generated by both the Bro flow and packet analysis. These files contain details regarding unusual activities that shed some insights on the behavior of the malicious traffic. The features extracted from Bro logs files are presented in Table 3. These logs (in CSV format) were stored in MySQL database for the next step.

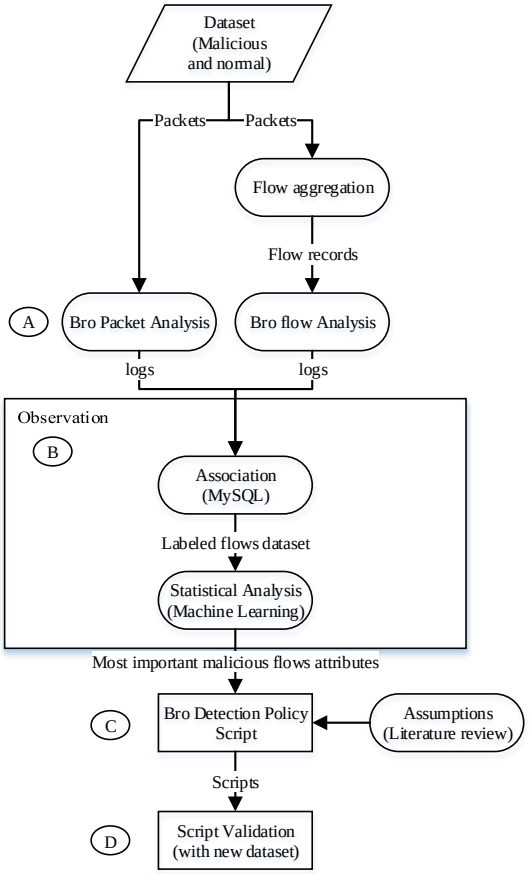

Figure 2: Overall Bro detection process.

Table 3
Features of flow and packet generated from Bro logs

Type	Features	Description
Shared features	src_ip	IP address of source host
	dest_ip	IP address of destination host
	src_p	Source port number
	dest_p	Destination port number
	Proto	Protocol
Flow-based features	Octs	Total octets or bytes per flow
	Pkts	Total packets per flow
	T_s	Flow time start (for the first packet in a flow)
	T_f	Flow time finish (for the last packet in a flow)
	Flag	TCP Flags
Packet-based features	*pkt_len*	Packet length
	pyd_len	payload length
	pyd_cnt	Payload content
	pkt_time	Time stamp
Derived flow features	*duration*	Flow duration (duration = $T_f - T_s$)
	Bpp	Average bytes per packet in flow (Octs/Pkts)
	Bps	Average bytes per second (Octs/duration)
	Pps	Average packets per second (Pkts/duration)
	tot_flows	Total number of flows sent by a host
	Tot_pkts	Total number of packets sent by a host
	Tot_octs	Total number of bytes sent by a host
	Tot_dur	Total number of all flows durations by a host

b. Observation

In this step, we observed, parsed and mined the data (based on finding from the previous step) in the logs that indicate a certain level of abnormality and then identified the malicious flow and packet features to be used in the next step. In this step, firstly, the association from Bro packet analysis logs to Bro flow analysis logs was made. This can be done by

matching the 5-tuple and timestamps of both Bro analysis logs using MySQL. When a match is found, the flow is labelled.

Since Bro packet logs are generated from full packet inspection, it is rich of information. Hence, we assume that Bro packet-based logs make the benchmark for the alert decision. With this condition in mind and to reduce the number of flows for further analysis, the decision to determine a malicious flow needs to be made. In this case, a decision that a flow is malicious is based on the decision derived from the corresponding packet generated by Bro packet analysis. In other word, flows that are alerted by Bro packet analysis are identified as malicious, hence they are extracted and labeled herewith.

After this association is performed, the flows labeled as datasets mixed with malicious and non-malicious labels for each malicious type was generated. Secondly, statistical analysis on these flow datasets was then taken place as shown in Figure 2. For this purpose, we took the advantage of using an open-source toolkit, WEKA data mining package that has a collection of popular machine learning algorithms. These algorithms were used for learning the flow characteristics from the datasets (in CSV format) based on the hidden features trained by both malicious and non-malicious traffic.

The main goal for using these algorithms is to classify the features and to find out the most important malicious flow attributes that provide maximum detection accuracy. Further, the corresponding rules (from the classification process) of these significant attributes were considered in the next step to improve the accuracy of the Bro scripts. Table 4 shows the features selected as input of WEKA for the purpose of attribute evaluator to generate the most important attributes in each malicious type.

Table 4
Attribute that is used for classification

Attribute	Spam-bot	IRC-bot	P2P-bot
Flow duration	√	√	√
Protocol	√	√	√
Source port			√
Destination port	√	√	√
# Packets per flow	√	√	√
# bytes per flow	√	√	√
# Bytes per packet	√	√	√
# Bytes per second	√	√	√

These feature selections should be relevant to the behavior of the malicious type. For example, in the IRC-bot, number of packets per flow is much related to the Ping-Pong (keep-alive) communication used in regular IRC channel. Ignoring the unrelated features will avoid noisy attributes that have negative effects on the classification accuracy.

For generating the most significant attributes used in Bro scripts, we used Wrapper subset evaluation that creates all possible subsets from our feature vectors, with the best first search method. Then, every subset is classified with full training set by each classification machine learning algorithms (listed in Table 5). Based on the accuracy results generated from this algorithm, the most effective features were generated.

c. Bro Detection Scripting

Rules of the most important features derived from the previous step, and rules inspired from the literature [26] [27] were converted into Bro script syntax. Bro detection script mainly has two phases. In the first phase, flows that are not matched with the rules mentioned above were discarded before entering the next phase. This step helps to reduce the amount of flows to be processed. In the second phase, further analyses were performed on the flows that match with those rules. The main goal of the second phase is to achieve better detection rate with low false positive rate.

Table 5
Attribute selection setting and classification selection

Type		Description
Attribute evaluator		Wrapper
Search method		Best first
Attribute selection mode		Full training
Number of folds		5
Classification algorithm	Tree	J48 (C4.5)
		Random Tree
		RepTree
		BFTree
	Rule	JRip
		PART
		DTNB

Basically, the second phase was implemented in Bro script to record the number of flows of a certain host IP address, within a specified time interval. If the number of flows exceeds a threshold, the host is considered as malicious. For detecting these repetitive pattern malicious activities in flow detection, we added scripts that track the malicious IP addresses at every certain period of time to make sure that the malicious activities occur frequently. For example, a keep-alive message (Ping-Pong) is exchanged in regular interval by the command and control (C&C) server to check whether the client host is alive.

The threshold assigned in Bro is not static. Depending on the malicious activity type, thresholds were updated and calculated periodically (e.g. scheduled of 50 seconds) based on the total number of potential concurrent connections associated with a host and the total number of all hosts participating within the predefined duration of time. Two points were considered in this step. Firstly, to avoid false negative alerts, the detection scope should not be narrow, but this will be on the account on false positive alert. Secondly, since flow aggregator may produce several flow records with the same connection (e.g. when downloading big file), analysing these records repeatedly in the flow-based detection may waste resources. However, we solve this issue by discarding the existing flows.

d. Validation

Once the detection script is implemented, it is important to validate it and determine the possible false negatives and positive alerts. To do so, we ran the Bro policy script obtained from the previous step into the flow-based detection model in Figure 1 (b). Here, we input new datasets that carries same malicious activities besides the background traffic. The logs were then inspected to see whether the known IP addresses of the infected machines (that produce malicious flows) are detected or not.

In this step, we validated spam and P2P-bot detection by ISOT dataset while in IRC-bot we used CTU-51 dataset. For all malicious types and non-malicious traffic in ISOT dataset, we combined these datasets with PSCJ trace as additional background dataset. Next section presents the experimental testbed where the validation took place. The output of this stage is discussed on Section V.

IV. TESTBED EXPERIMENT

We ran Bro flow-based detection on testbed depicted in Figure 3. The experimental environment run on two machines interconnected through a Gigabit switch. Both machines are running 12.04 Linux-based Ubuntu Desktop 64-bit with Intel i7 3.1 GHz with 32 GB of RAM. Both machines NICs support Gbps. To get more reliable experimental results, detection method should receive the PCAP traffic in a way that represents the real network environment instead of reading the PCAP file in offline way. For this purpose, the first machine was used for the traffic generation which replays real (previously captured) network traffic datasets on the wire using tcpreplay v4.0.5 [28] and sent it to the second machine.

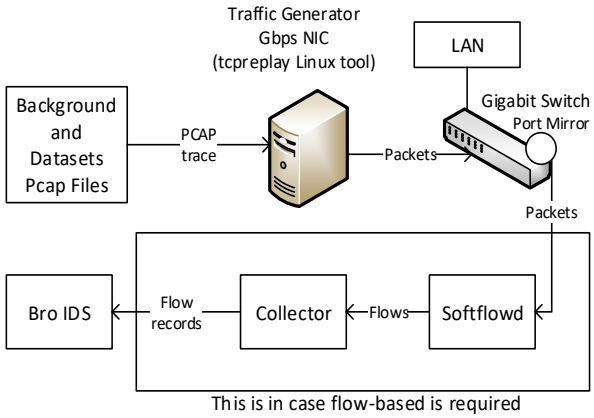

Figure 3: Testbed schematically diagram for proposed mechanism.

The super feature of tcpreplay tool is the different traffic speeds that can be adjusted and controlled when injecting to the network interface. The replaying trace speed can be either in the original captured speed or by multiplying the original capture speed by a value (e.g. we reply traffic three times as fast as it was originally captured). The switch support Giga bit speed with port mirror enabled it to forward all traffic to the analyzing machine.

The second machine was installed with Softflowd v0.9.9 [29] and Bro 2.3. Softflowd was used as a flow aggregator with default parameters to generate flow records from the dataset packets received and to be exported to the collector. Softflowd is also capable of Cisco netflow export format. In addition, to verify that Softflowd receive all packets from tcpreplay machine, we used Softflowctl program to track Softflowd process for statistical measurements. Bro provides command-line interface (CLI) and is used for malicious detection using policy script obtained from the previous section. In addition, Bro is configured to collect the flow records by reading the flows from a UDP socket in the localhost in a way to be suitable for Bro analysis.

V. RESULT AND DISCUSSION

To check the correctness of labeling decisions in datasets, they were analyzed and verified manually using Bro logs. For example, after extracting and analyzing the flows initiated from the internal hosts who were labeled as spammers in the ISOT dataset, we observed that these machines exhibited unusual usage comparing to other machines. These machines engaged in large number of SMTP connections (7,699 flows) within short time of period, by sending packets (with randomly source email addresses and advertising words in the email's subject) to many external servers on port 25. However, only one machine (172.16.0.2) in ISOT dataset seem to have either incomplete traffic or been wrongly labelled as spam. All other labelled machines as infected machines on the other datasets were found to be correctly labeled.

Table 6
Best three important attribute

Classification algorithm	Spam-bot	IRC-bot	P2P-bot
J48	dest_p duration pkts	dest_p pkts octs	src_p dest_p duration
Random Tree	dest_p duration pkts	dest_p pkts duration	src_p dest_p pkts
Rep Tree	dest_p duration pkts	dest_p pkts octs	src_p dest_p duration
BF Tree	dest_p duration pkts	dest_p pkts duration	src_p dest_p duration
PART	dest_p duration pkts	dest_p pkts duration	orig_p dest_p duration
Jrib	dest_p duration pkts	dest_p octs duration	orig_p dest_p pkts
DTNB	dest_p duration pkts	Pkts octs pps	orig_p dest_p pps

For the most significant attributes used in Bro scripts, Table 6 lists the best three important attributes generated by different classification algorithms (that listed in Table 5) for each malicious type. We found that these attributes were very useful in our Bro detection script. For example, for Spam detection, it was obvious that destination port is among these attributes as port 25 on the SMPT destination server is a good sign of this malicious type. It was also observed that packets and bytes per flow in IRC-bot were the most significant attributes since their values are constant (when e.g. for each time keep-alive is exchanged).

For Bro script validation, the notions presented in Table 7 were used in this section to measure the common metrics: true positive and true negative rate and precision. When using new labelled datasets in our experiment, Bro flow-based detection scripts marked all known infected machines IP addresses in these datasets as malicious (FN = zero). This mean that Bro script is able to detect all the infected machines that generate malicious flows with 100% detection accuracy or True Positive Rate (TPR) as calculated in:

$$TPR = \frac{TP}{TP + FN} = \frac{number\ of\ correct\ detected\ as\ malicious}{total\ number\ of\ actual\ malicious\ machine} \quad (1)$$

For false positive rate and precision, we used the following formulas:

$$FPR = \frac{FP}{FP + TN} = \frac{number\ of\ falsely\ detected\ as\ malicious}{total\ number\ of\ actual\ malicious\ machine} \quad (2)$$

$$Precision = \frac{TP}{TP + FP} = \frac{number\ of\ correct\ detected\ as\ malicious}{total\ number\ of\ detected\ as\ malicious} \quad (3)$$

Table 8 shows the FPR and precision for each malicious type. It shows that flow-based detection suffers from false positive alerts generation. This is because of the similarity between malicious and non-malicious recorded data. Such data include flow duration, port number, and number of packets per flow. In addition, unlike packet-based detection, incomplete data (no payload data) of the flow-based analysis also play important role for producing these false alarms. In other words, payloads provide significant role for identifying non-malicious traffic. To test the false positive in packet-based, we ran IRC-bot dataset and no false positive alerts was generated which indicates that the accuracy level when payload is inspected.

Table 7
Metric Notions

	Actual Non-Malicious	Actual Malicious
Detected as malicious	False Positive (FP)	True Positive (TP)
Detected as non-malicious	True Negative (TN)	False Negative (FN)

Table 8
False positive rate (FPR) and precision

Type	Spam-bot	IRC-bot	P2P-bot
FPR[#]	0.05	0.25	0.20
Precision[$]	0.66	0.66	0.42

[#]the value is 0, if there is no FP [$]value is 1, if there is no FP

VI. CONCLUSIONS

In this paper, we showed how Bro is powerful and useful tool for data analysis and feature extraction on the recent labelled dataset. We ran Bro flow-based detection against several labelled datasets to detect malicious activities. Based on our experiments, no false negatives was reported. This indicates that Bro detection implementation along with attributes selection obtained from machine learning promise high detection rate for flow-based detection.

However, false positive alerts were generated from Bro detection which degrades the accuracy of flow-based detection. It is also concluded that since only flow data is available in flow-based detection, it is hard to make complete potential behaviour about the malicious activities found in the datasets. Since packet-based detection does not report any false positive alerts, it deserves higher score in accuracy and gives us a hint in putting this packet Bro detection as the second layer in the flow-based detection for future work.

ACKNOWLEDGMENT

This work has been partially funded and supported by Research Centre, Al-Faisal University, Prince Sultan College Jeddah (PSCJ).

REFERENCES

[1] Koch, R. 2011. Towards Next-Generation Intrusion Detection. In 3rd International Conference on Cyber Conflict (ICCC) IEEE. 1-18.

[2] The Bro Network Security Monitor. [Online]. From: www.bro.org [Accessed on 7 June 2013].

[3] Papadogiannakis, A., Polychronakis, M., & Markatos, E. 2010. Improving the Accuracy of Network Intrusion Detection Systems under Load Using Selective Packet Discarding. In Proceedings of the Third European Workshop on System Security. pp. 15-21.

[4] Claise, B. 2008. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information.

[5] Hofstede, R., Celeda, P., Trammell, B., Drago, I., Sadre, R., Sperotto, A., & Pras, A. 2014. Flow Monitoring Explained: From Packet Capture To Data Analysis With Netflow And IPFIX. Communications Surveys & Tutorials, IEEE. 16(4): 2037-2064.

[6] Krmicek V. 2012. Towards Extended NetFlow/IPFIX. In 4th Workshop on the Usage of Netflow/IPFIX in Network Management IETF

[7] Hellemons, L. 2012. Flow-based Detection of SSH Intrusion Attempts. 16th Twente Student Conference on IT. 100(1.5): 100.

[8] Vykopal, J., Drašar, M., & Winter, P. 2013. Flow-based Brute-force Attack Detection. Advances in IT Early Warning. Garching near Muenchen: Fraunhofer Research Institution AISEC. 41-51.

[9] Zhao, D., Traore, I., Sayed, B., Lu, W., Saad, S., Ghorbani, A., & Garant, D. 2013. Botnet Detection based on Traffic Behavior Analysis and Flow Intervals. Computers & Security.

[10] Sperotto, A., Schaffrath, G., Sadre, R., Morariu, C., Pras, A., & Stiller, B. 2010. An Overview of IP Flow-based Intrusion Detection. Communications Surveys & Tutorials, IEEE, 12(3): 343-356.

[11] De Ocampo, F. B., Del Castillo, T. M., & Gomez, M. A. 2013. Automated Signature Creator for a Signature Based Intrusion Detection system (PANCAKES). In The Second International Conference on Cyber Security, Cyber Peacefare and Digital Forensic (CyberSec2013).198-205.

[12] Alaidaros, H., Mahmuddin, M., & Al-Mazari, A. 2011. An Overview of Flow-Based and Packet-Based Intrusion Detection Performance in High Speed Networks. In Proceedings of the International Arab Conference on Information Technology (ACIT 2011).

[13] Ruthven, P. B. Contextual Profiling of Homogeneous User Groups for Masquerade Detection. 2014. Gjøvik University College (GUC), Master thesis.

[14] Sommer, R., & Paxson, V. 2010. Outside the Closed World: On using Machine Learning for Network Intrusion Detection. In Security and Privacy (SP) IEEE Symposium. 305-316.

[15] Shiravi, A., Shiravi, H., Tavallaee, M., & Ghorbani, A. A. 2012. Toward Developing A Systematic Approach to Generate Benchmark Datasets for Intrusion Detection. Computers & Security, 31(3): 357-374.

[16] ISOT Botnet Data. [Online]. From: http://www.uvic.ca/engineering/ece/isot/datasets. [Accessed on 5 June 2014].

[17] CTU-50 MCFP Dataset. [Online]. From: mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-50. [Accessed on 6 July 2014].

[18] CTU-51 MCFP Dataset. [Online]. From: mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-51. [Accessed on 2 July 2014].

[19] CTU-52 MCFP Dataset. [Online]. Frpm: mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-52 [Accessed on 2 July 2014].

[20] CTU-53 MCFP Dataset. [Online]. From: mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-53. [Accessed on 1 July 2014].

[21] Saad, S., Traore, I., Ghorbani, A., Sayed, B., Zhao, D., Lu, W. & Hakimian, P. 2011. Detecting P2P botnets through network behavior analysis and machine learning. In Privacy, Security and Trust (PST), 2011 Ninth Annual International Conference. 174-180.

[22] Garcia, S., Grill, M., Stiborek, J., & Zunino, A. 2014. An Empirical Comparison of Botnet Detection Methods. Computers & Security, 45: 100-123.

[23] M. Stevanovic and J. M. Pedersen. 2013. Machine Learning for Identifying Botnet Network Traffic. Networking and Security Section, Department of Electronic Systems, Aalborg University, Technical Report.

[24] Morgan, J. 2015. Streaming Network Traffic Analysis Using Active Learning. Ms Thesis, Dalhousie University Halifax, Canada.

[25] Zhu, Z., Lu, G., Chen, Y., Fu, Z. J., Roberts, P., & Han, K. 2008. Botnet research survey. In Computer Software and Applications. COMPSAC'08. 32nd Annual IEEE International. 967-972).

[26] Sperotto, A., Vliek, G., Sadre, R., & Pras, A. 2009. Detecting spam at the network level. Springer Berlin Heidelberg. 208-216.

[27] Wurzinger, P., Bilge, L., Holz, T., Goebel, J., Kruegel, C., & Kirda, E. 2009. Automatically generating models for botnet detection. In Computer Security–ESORICS 2009. Springer Berlin Heidelberg. 232-249

[28] Tcpreplay. [Online]. From: http://tcpreplay.synfin.net. [Accessed on 1 June 2013].

[29] Softflowd. [Online]. From: http://www.mindrot.org/projects /Softflowd. [Accessed on 5 April 2014].