International Journal of Security and Its Applications
Vol. 11, No. 11 (2017), pp.1-12
http://dx.doi.org/10.14257/ijsia.2017.11.11.01

Machine Learning Based Botnet Detection in Software Defined Networks

Farhan Tariq[1]* and Shamim Baig[2]

[1]*Electrical and computer engineering, Centre for Advanced Studies in Engineering, Islamabad, Pakistan*
[2]*Computer Science and Engineering, HITEC University, Taxila, Pakistan*
**farhantariq2@gmail.com, msbaig@case.edu.pk*

Abstract

This paper proposed a flow-based approach to detect botnet by applying machine learning algorithms to software defined networks without reading packet payload. The proposed work uses network flows as input and process it in two windows based modules to extract a statistical feature set to be used for classification. The first module process network flow stream to extract flow traces. The window size of this module is 10 which means a flow trace with 10 flows is considered as a trace of interest and forwarded to the next module for further processing. The second module processes the selected trace and fetches historical flows in last 60-minute window for the source and destination IPs of the trace. The feature set is extracted from selected trace and relevant historical flows. The approach applies supervised decision tree based machine learning algorithm to create a model during a training phase using extracted feature set. This model is then used to classify flow traces during the testing phase. The dataset for experimentation is extracted from publicly available real botnet and normal traces. The experimental findings show that the method is capable to detect unknown botnet. The results show detection rate of 97% for known botnets and 90% for unknown botnets.

Keywords*: botnet, malware, machine learning, NBA, SDN, TSDR, OpenFlow, Opendaylight*

1. Introduction

The Bots, machines infected with bot-malware, compromised to report back to their command and control (referred to as "c2c" hereafter) servers and form a network controlled by the botmaster. This network of bots empowers the master to execute malicious activities remotely and anonymously. The bot machines not only use as a computing resource to launch an attack on other targets rather these machines are also directly affected by malicious activities like data theft and loss. The ransomware is the recent example of such attacks where individual victims should pay to get back one's data. Botnets are comprised of three main components including bots, c2c servers, and bot-master. The c2c servers operate as an intermediate layer between bots and their master. The bots registered with their c2c servers to establish a communication channel. This communication channel is used for heartbeat and commands exchange. The bot-master connect with c2c servers to update instruction set and get visibility of the bot network. Botnet generally has three phases including infection, c2c communication, and attack phase in their life cycle. The infection and attack phase of the botnet is same as in other malware where infection phase used to infect machines and then these machines are used to launch specific attacks. The c2c communication phase classifies a malware into bot-malware. This phase is used to registered newly infected machines with their c2c

Received (June 20, 2017), Review Result (September 22, 2017), Accepted (October 14, 2017)

ISSN: 1738-9976 IJSIA

servers and established a communication channel between bots and their command servers. The mechanism to connect back to c2c servers of bot machines classifies botnets into centralized and distributed botnets. The botnet where every bot machine connects back to central c2c servers is classified as centralized whereas in distributed architecture a bot first search in the local network for other bot machines to form a local network and only one commander locally selected connect back to the c2c servers. The botnet started with a centralized architecture using IRC protocol and gradually evolve to distributed architecture and start using common protocols to hide its communication.

The increasing number of botnet attacks and their evolving nature drive the need for continuous improvement of detection techniques. The detection techniques in botnet literature started with simple signature-based approaches and evolve gradually with time. The current proposals focusing on behaviorally based approaches and mainly targeting network level information. Majority of these techniques use only network header level information to counter packet payload challenges *e.g.* encrypted botnet traffic and. processing complexities. The use of machine learning algorithm to extract botnet behavior patterns from the monitored network is common in these approaches. The network flow based approaches suffer mainly with two issues. First one is flow collection which comes up with many challenges in traditional IP networks [2]. Secondly, the detection techniques that use a diverse dataset with real traffic traces and have a good percentage of unknown traffic in the testing dataset for which the model is not trained, suffer from the relatively high false positive rate.

This paper proposes a flow based bot detection technique using machine learning algorithm in software defined networks (referred to as 'SDNs' hereafter) with the aim to address above mention issues with current approaches. The SDNs technology platform decouples control logic from forwarding devices to a logically centralized controller. This centralized control of the network makes flow stats collection much simpler. To counter the high false positives the proposed work uses the intelligence from both real-time flow stream data and historical flow data to extract more enrich feature set. This feature set is then used to feed into the machine learning algorithm where it is compared with a pre-computed model for classification. The proposed approach operates in training and testing modes. The training mode is used to create a model based on labeled dataset whereas testing mode is used to detect botnet in the monitored network.

The contribution of this paper is summarized as below

The work proposes to combine real-time flow trace information with historical context to extract enrich feature set that helps classification to increase detection and reduce false positive rate.

The proposed work address botnet problem in SDNs, The emerging network technology platform.

The proposed approach tested against two real traffic traces. The results show good system performance with relatively high detection and low false positive rate.

The rest of the paper is organized in the following sequence.

The related work is discussed in section II. The section III presents proposed system architecture and discussed each module details. The experimental setup and results evaluation is discussed in section IV. The section V concludes the paper and discussed directions for future growth of the presented work.

2. Related Work

The botnet detection techniques chasing the trends of botnet started with protocol and structure dependent signature-based techniques and moves toward more sophisticated network behavioral-based approaches. The recent approaches in botnet literature targeting

network flow statistical features and applying machine learning algorithms for botnet classifications. Both supervised and unsupervised algorithms are used in these approaches. The most of these proposed approaches use mining based approach for flow collection from already stored location and only a few works with flow stream. The machine learning algorithms are mainly applied on the entire dataset rather than working in some time intervals.

The livadas et al. [9] first introduced the use of machine learning algorithms in flow-based botnet detection techniques. This work is protocol and structure dependent and focuses only on IRC chat traffic to detect botnet and benign IRC chat traffic. The work proposed by bilge [6] analyzed wide-scale NetFlow data for botnet detection. The supervised machine learning algorithms including support vector machine, C4.5, and random forest implemented for the classification process. The approach first identifies clients and servers from monitored netflows and then extract features for identified servers. The flow size based, client access pattern based and temporal behavior based features are extracted. These features fed into classification algorithm for training and testing purposes. The result shows the best performance for the random forest with 70% detection rate and 0.5% false positive rate. The paper also discusses that the approach can detect botnet in real-time. The work proposed by David Zhao [5] investigate time intervals for flow analysis to detect botnet. The flows are processed for said time interval to extract feature vector. This feature vector is then fed into the classification process. The ISOT dataset is used for experiments with the reptree algorithm. The work shows the best result for 300 second time interval with a detection rate of 98.3%. The paper also shows that the approach is capable to detect botnet in real-time.

The selection of dataset is critical in machine learning based approaches. The dataset must have traces of real-world botnet rather than self-created botnet or script generated traces. The number of proposed approaches in literature does not introduce any diversity in testing data. Although these approaches show high detection rate due to dataset biasedness the detection model may perform poorly with the more diverse dataset. The work proposed by E. Beigi [3] uses a real-world publicly available dataset and introduce much diversity by including more than 40% unknown botnet samples in testing data. The approach applies C4.5 decision tree based supervised algorithm for botnet detection. The commonly used features in botnet detection techniques are distributed into the four groups and a greedy algorithm is applied for feature selection. The work shows 75% detection accuracy. The software defined networks are not much explored for botnet detection. The literature only has few proposed flows based techniques. Most of the proposed techniques collect network flows directly from network elements in classical approach. The work proposed by T. Farhan [1] explored OpenFlow counters for botnet detection. The approach centrally collects network flows in form of OpenFlow counters from SDN controller. This method also uses real-world publicly available botnet dataset. The work injects unknown botnet samples in test data for which detection model does not train and achieve 50% diversity. The C4.5 decision tree based supervised classification algorithm is used and experiments show 80% detection rate.

The proposed flow based botnet detection research in literature investigate flows dataset either entirely or in some flow intervals. These approaches either operate on current network activity as defined by flow interval or operate only on entire historical flows. The methods proposed for traditional IP network also suffer from flow collection due to inherent limitations of traditional IP network. The proposed method in this paper analyzes both the current network activity and its historical context for botnet detection. The method is implemented in SDNs to address flow collection issues of traditional IP networks.

3. Proposed Detection Model

This work proposed a flow based generic botnet detection method in software defined networks. The work proposes to use intelligence from both real-time flow stream and historical context of a flow trace to extract a more meaningful feature set to help classification process. The approach applies C4.5 a decision tree based supervised machine learning algorithm for botnet classification. The proposed approach operates in two modes the training mode and test mode. The training mode is used to train the C4.5 algorithm to recognize botnet behavior patterns with help of labeled dataset. A decision tree model is created based on labeled training dataset during this mode. The testing mode uses the pre-computed model to classify flow traces. The botnet traces are collected from publicly available CTU-13 botnet dataset and ISOT botnet dataset. For normal traffic representation, the normal flow traces of ISOT dataset is used for training and testing purposes. The following are steps for training and testing phase

Network flow stream collection and processing for trace detection

Historical flow collection for selected trace source and destination IP addresses

Feature extraction from each selected flow trace and historical flows

C4.5 decision tree based supervised machine learning algorithm applied on extracted features to classify selected flow traces.

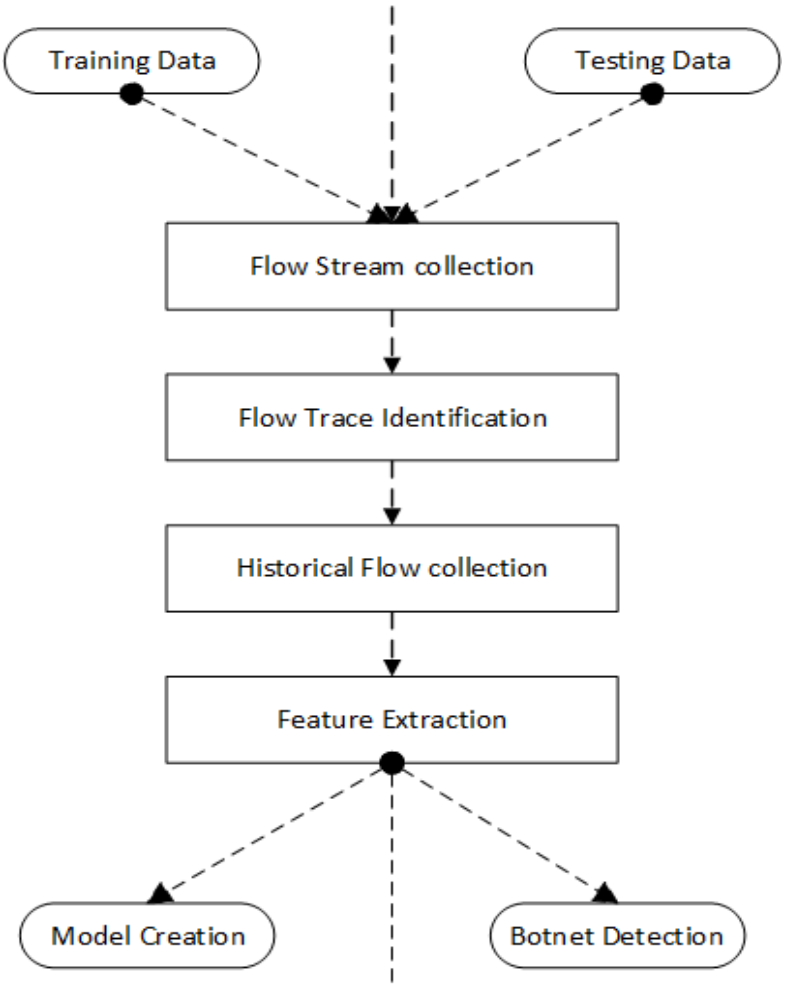

Figure 1. Steps in Training and Testing Mode

3.1. Architectural Overview

The architecture of the proposed system is shown in Figure 2. The system consists of flow stream collection module, flow trace detection module, historical flow collection module, feature extraction module, and flow classification module. The flow stream collection module is responsible to collect a stream of flows from SDN controller. This module processes the flow stream data to convert it into small batches and forward batches to trace detection module. The trace detection module performs batch processing to extract flow traces. This module process flows traces until 10 flows are counted for the same flow trace. A flow trace with 10 flows is considered as a trace of interest and forwarded to feature extraction module for further processing. The feature extraction module processes the detected trace for feature extraction and request for historical flow data for the source and destination IP of the said trace. The historical flow collection module processes this request and fetch all the flows of the requested source and destination IP addresses for last 60-minute duration. The collected batch of flows is then sent back to feature extraction module as a request response. The feature extraction module process both batches collected from trace detection module and historical flow collection module to extract a global feature SET equipped with statistical intelligence from the current network activity and the historical context for the same. The global feature set fed into the decision tree based supervised classification module. The classification module operates in training and testing mode. During the training mode, the global feature set of each detected trace is provided with a pre-assigned label. This help classification module to create a decision tree model. This model is then used in testing to classify flow traces.

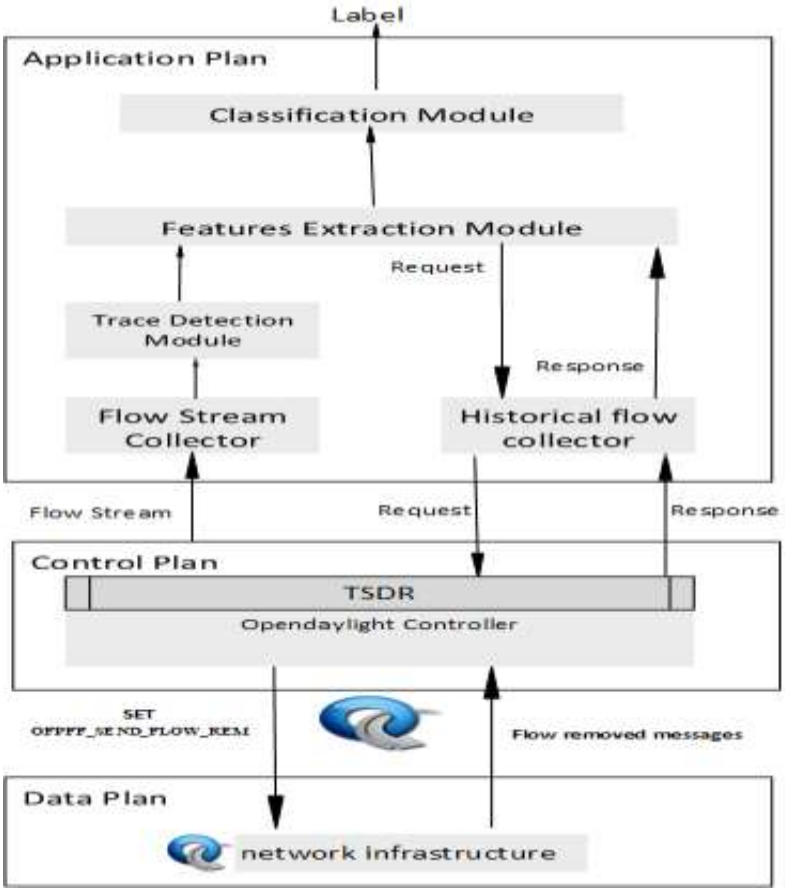

Figure 2. System Architectural View

3.2. Flow Stream Collection

This module is responsible to collect a stream of network flows centrally from SDN controller (opendaylight). The time series data repository (TSDR) feature of opendaylight is used to collect this stream. TSDR makes flow stats collection abstract for user applications. The collected stream is processed to form small batches of flows. The time window to form a batch is configurable. The stream of batches forwarded to trace detection module.

3.3. Flow Trace Detection

This module is responsible to assemble a batch of flows of two network endpoints. A flow trace is an ordered sequence of flows between two network endpoints [7]. This module uses this key < Source IP, Destination IP, Destination port, Protocol > to identify flows of two endpoints for the same application. The flows with the same key are grouped together to form a batch. Each flow of a batch is counted and when a batch reaches the count of 10 flows that batch is exported to the feature extraction module. Any subsequent flow of the exported batch is recognized as a new trace and go through the batch assembly process. The count of 10 flows is selected on bases of flow interval research in the literature which varies between a range of 10 to 60 flows.

3.4. Historical Context

The historical context for the detected flow trace is extracted from all the flows generated from trace source address or pointed to trace destination addresses in last 60-minute duration. The historical flow collection module is responsible to pull these flows centrally from the SDN controller for said duration. The source and destination address of the trace of interest are provided by feature extraction module. These addresses used as matching criteria to fetch flows from the controller. The first batch of flows pulled from the controller with a filter on source address to match with trace source. The second batch of flows pulled with a filter on destination address to match with trace destination. These flows are then forwarded to feature extraction module for further processing.

3.5. Feature Extraction Module

The proposed feature extraction module works with two time-based separated batches of flows. The goal of this module is to extract statistical features from current network activity and historical context of the same. To achieve this two batch of flows are collected, the first batch of 10 flows collected from trace detection module and the second batch is collected from historical flow collection module. The first batch is processed as current network activity and features extracted from it as shown in Table 1. These features are estimated using following equations where Bytes represent total bytes, Packets represent total packets, and duration represents the total time of a flow of interest. The equation IV is used to calculate flow interarrival time. This is the time between two consecutive flows in each flow trace. The mean, variance, minimum and maximum values of these extracted features are then calculated over a complete trace of flows.

$$\text{Bytes per second} = \frac{Bytes}{Duration} \tag{1}$$

$$Packets\ per\ second = \frac{Packets}{Duration} \tag{2}$$

$$Bytes\ per\ packet = \frac{Bytes}{Packets} \tag{3}$$

$$Flow\ interarrival\ time = \frac{Start\ time\ of\ flow(n)}{Start\ time\ of\ flow\ (n-1)} \tag{4}$$

Table 1. Extracted Features - Current Network Activity

Feature	Description
Duration {Min, Max, Mean}	Flow duration
Bytes {Min, Max, Mean}	Total bytes received
Packets {Min, Max, Mean}	Total packets received
BPS {Min, Max, Mean}	Bytes per second
PPS {Min, Max, Mean}	Packets per second
BPP {Min, Max, Mean}	Bytes per packet
Fit {Min, Max, Mean}	Flow interarrival time

The second batch of flows is processed as historical context to extract scaler features from the network activity of source and destination IP address of a trace in last 60-minute duration. The extracted features are shown in Table 2.

Table 2. Extracted Features - Historical Context

Feature	Description
Count of DIP	The unique count of DIP for the SIP
Count of DP-SIP	The Unique count of DP for the SIP
Count of flows-SIP	Total number of flows from SIP
Count of packets-IP	Total number of packets for SIP
Count of Bytes-SIP	Total Number of bytes for the SIP
Count of SIP	The Unique count of SIP for DIP
Count of DP-DIP	The unique count of DP for DIP
Count of flows-DIP	Total number of flows to DIP

3.6. Supervised Classification

The classification module uses C4.5 decision tree based supervised classification algorithm. The input to the classification module is global feature vector generated by feature extraction module. The module operates in two different modes. The system initially runs in the training mode and the global feature vectors with pre-assign labeled to each fed into the classification module. The C4.5 algorithm learns with help of provided labeled and create a decision tree model. This model is then used in a testing mode to detect botnet. Each input of feature vector during testing mode traverse through decision tree model and get assign a label.

4. Evaluation

To evaluate the proposed method the real network traces for botnet and normal traffic are used to perform experiments. These traces are collected from two publicly available well-known botnet sources. The botnet traces are collected from [4] and for normal traffic representation traces are collected from [8]. The data is collected in the form of packet capture "pcap" files. The six pcap files of different botnet families including (Neris, Rbot, Virut, Menti, Murlou, and Sougo) are collected from [4] and one pcap files for normal traffic is collected from [8]. These files are used to simulate botnet and normal traffic in the experimental setup. This simulation is divided into two phases. The phase-1 for the training mode and phase-2 for the testing mode. The distribution of botnet traffic for training and testing purposes is shown in Table 3 and Table 4 respectively.

Table 3. Botnet Traces in Training Data

BOTNET	TYPE	PERCENTAGE OF FLOWS
Rbot	IRC	9.87%
Menti	IRC	4.3%
Virut	HTTP	13.5%

As depicted in Table 4 the traffic for testing purposes has 50% representation of unknown bot families for which the decision tree model is not trained. This is to test if the proposed model can detect unknown botnet and to reduce dataset biasedness.

Table 4. Botnet Traces in Testing Data

BOTNET	TYPE	PERCENTAGE OF FLOWS
Neris	IRC	6.4%
Rbot	IRC	2.77%
Virut	HTTP	9.67%
Menti	IRC	1.68%
Murlo	IRC	2.96%
Soguo	HTTP	3.85%

4.1. Experimental Setup

The experimental setup is based on SDN controller. There is a number of opensource SDN controller available but opendaylight widely capture market footprint. This is a java based opensource controller and has a good documentation set publicly available. The time series data repository (tsdr) feature of opendaylight is also one of the reasons to select it for the experimental setup. The tsdr feature of opendaylight makes flow state collection abstract for user applications. The custom application only requires communicating with tsdr for the collection of either flow stream or historical time series of flow stats. The prototype of the proposed system is developed in java. To simulate the dataset for evaluation, a network emulator tool called mininet is used to create Virtualized network

infrastructure. This infrastructure is then used to simulate the dataset for training and testing purposes.

4.2. Results

To evaluate the detection capabilities of proposed botnet detection method, experiments are performed using labeled ground truth data and testing data. The results are compiled in two steps. The first step concludes the result at the individual trace of interest granularity as shown in Table 5. This Table helps to analyze system performance for individual botnet family sample included in the testing dataset. Table 5 list three known botnet family samples and three unknown botnet family samples. The botnet samples including Rbot, Virut, and Menti are considered known botnet as detection model is trained for these whereas Neris, Murlo, and Soguo are considered as unknown botnet. The average detection rate of the proposed method for a known botnet is 97.1% and for an unknown botnet is 90.4%.

Table 5. Result Summary for Individual Bot Samples

Type	TOI	Correctly classified	Incorrectly classified	Detection
Normal	378	360	18	95.2%
Neris	50	48	2	96.0%
Rbot	19	18	1	94.7%
Virut	58	55	3	94.8%
Menti	7	7	0	100%
Murlo	17	15	2	88.2%
Soguo	27	24	3	88.9%

The second step concludes results to analyze overall system performance. The confusion matrix as shown in Table 6 is formed to summarized results. This matrix helps to calculate overall system accuracy using following equation.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (5)$$

The overall accuracy of the proposed method is 94.8% which is higher than the approaches proposed in [1] and [3] that use similar diversity in their testing dataset.

Table 6. Confusion Matrix

		Actual	
		Botnet	Normal
Detected	Botnet	167	18
	Normal	11	360

5. Conclusion

The proposed work detect botnet in software defined network in near real-time. The delay is approximately 10 consecutive flows counted from the first flow of a botnet trace. To increase detection rate and reduce false positives, the method uses a rich feature set extracted from current network activity of a trace and historical context of the same. The works show promising results with a detection rate of 97% for known and 90% for an unknown botnet. The detection rate is much higher as compared to other approaches with the same level of diversity in testing data. The proposed system shows the accuracy of 94.8%. The system uses TSDR feature of opendaylight to support the concept of stats plan in SDNs. Separating statistics from control plan not only reduce computation load on the controller but also provide a centralized statistics visibility. In context with stats plan, the future extension of this work is to reflect its computational results back to the stats plan so that other application may use this information.

References

[1] F. Tariq and S. BaigBotnet, "classification using centralized collection of network flow counters in software defined networks", International Journal of Computer Science and Information Security, vol. 14, no. 8, **(2016)**, pp. 1075.
[2] A. S. da Silva, C. C. Machado, R. V. Bisol, L. Z. Granville and A. Schaeffer-Filho, "Identification and selection of flow features for accurate traffic classification in sdn", In Network Computing and Applications (NCA), 2015 IEEE 14th International Symposium on, IEEE, **(2015)**, pp. 134-141.
[3] E. B. Beigi, H. H. Jazi, N. Stakhanova and A. A. Ghorbani, Towards effective feature selection in machine learning-based botnet detection approaches. In Communications and Network Security (CNS), 2014 IEEE Conference on IEEE, **(2014)**, pp. 247-255.
[4] S. Garcia, M. Grill, J. Stiborek and A. Zunino, "An empirical comparison of botnet detection methods", computers & security, vol. 45, **(2014)**, pp. 100-123.
[5] D. Zhao, I. Traore, B. Sayed, L. Wu and S. Saad, A. Ghorbani and D. Garant, "Botnet detection based on traffic behaviour analysis and flow intervals", Computers & Security, vol. 39, **(2013)**, pp. 2-16.
[6] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda and C. Kruegel, "Disclosure: detecting botnet command and control servers through large-scale NetFlow analysis", In Proceedings of the 28th Annual Computer Security Applications Conference, ACM, **(2012)**, pp. 129-138.
[7] F. Tegeler, X. Fu, G. Vigna and C. KruegelBotfinder, "Finding bots in network traffic without deep packet inspection", In Proceedings of the 8th international conference on Emerging networking experiments and technologies, ACM, **(2012)**, pp. 349-360.
[8] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu and P. Hakimian, "Detecting P2P botnets through network behavior analysis and machine learning", In Privacy, Security and Trust (PST), 2011 Ninth Annual International Conference on, IEEE, **(2011)**, pp. 174-180.
[9] C. Livadas, R. Walsh, D. Lapsley and W. T. Strayer, "Using machine learning techniques to identify botnet traffic", In Local Computer Networks, Proceedings 2006 31st IEEE Conference on, IEEE, **(2006)**, pp. 967-974.

Authors

Farhan Tariq, he received his Master of Computer Engineering degree with first class honors from CASE Pakistan in 2011. He is currently working towards a Ph.D. degree at Center for Advanced Studies in Engineering. His research interests include network monitoring and security. Specifically, network behavioral monitoring to detect the presence of malicious call-backs.

M Shamim Baig, he is Ph.D. in Computer Science from George Washington University Washington DC, USA; MS in Industrial Electronic from Cranfield Institute of Technology UK. He has more than 40 years of Academic, Research & Engineering Management experience in the field of Supercomputing, Digital System Design, Networking & Information Security. He has been Air Vice Marshal in Pakistan Air Force, Principle Scientific Officer at A.Q. Khan Research Labs & Director General / Dean "Centre of Excellence for Cyber Security" at National University of Science & technology Islamabad. He is currently a Professor/ Director Advanced Studies & Research at Center for Advanced Studies in Engineering Islamabad. He has published more than35 Int'l Journal/ conference papers. He has been Chair IEEE education activities & Keynote/ invited speaker at multiple Seminars.

