Multiclass Botnet Detection and Countermeasures Selection

Farhan Tariq[1] **and Shamim baig**[2]
1 farhantariq2@gmail.com *2* msbaig@myu.edu.pk
Center for Advance Studies in Engineering Muslim Youth University, Islamabad

Summary
The increasing number of botnet attacks incorporating new evasion techniques making it infeasible to completely secure complex computer network system. The botnet infections are likely to be happen, the timely detection and response to these infections helps to stop attackers before any damage is done. The current practice in traditional IP networks require manual intervention to response to any detected malicious infection. This manual response process is more probable to delay and increase the risk of damage. To automate this manual process, this paper proposes to automatically select relevant countermeasures for detected botnet infection. The propose approach uses the concept of flow trace to detect botnet behavior patterns from current and historical network activity. The approach uses the multiclass machine learning based approach to detect and classify the botnet activity into IRC, HTTP, and P2P botnet. This classification helps to calculate the risk score of the detected botnet infection. The relevant countermeasures selected from available pool based on risk score of detected infection.

Keywords:
botnet, detection, mitigation, countermeasure, malware, Multiclass machine learning, NBA, SDN, TSDR, OpenFlow, Opendaylight, flows.

1. Introduction

The growing number of botnet attacks and evasion advancement posing threat to inter-connected computer systems. The advancing idea of malware presenting new assault vectors. This changing threat landscape turning into the greatest test to current detection methodologies. The signature-based botnet detection methods are not commonsense to adapt up to a quickly changing impression of botnet attacks. The encrypted command and control (C2C) correspondence of botnets additionally a bottleneck for signature-based methodologies. This shifted the researcher's interest into behavior-based botnet detection methods. The behavioral based proposals target both host level and network level information but the operational challenges of detection methods that rely on host level information resulted in a high number of network level proposals. To counter encrypted C2C communication traffic the network level proposals start focusing on session level traffic only.

The bot-malware with administration ability and command and control component, become one of the most powerful tools for malicious activities. A machine infected with bot-malware is perceived as bot machine. The bot machines attempt to enlist with their C&C servers. This enrollment built up a network of bot machines and their C&C servers. The network of bot machines controlled by bot-master is called botnet. The botnet is for the most part sorted into centralized and decentralized botnet on premise of their underline command and control correspondence mechanism. The IRC and HTTP botnet fall under the category centralized botnet, as these register back to their main C&C servers. The P2P botnet is decentralized as these bot machines first search comparable bot machines to be available and select a commander from them locally. The selected bot machine serves as a C&C server for all local bots. This local C&C server at that point associates back to central C&C servers for command and control communications. The botnet life span categories into recruitment, C&C communication, and attack phase. The recruitment phase is when a machine was infected with a bot-malware either directly or update from a previous malicious code. The C2C communication phase started as soon as these infected machines try to associates back to their C2C server for enrollment. In the attack phase bot machines actively perform actions received from C2C server for attack.

The botnet detection methods mainly relay on C2C communication phase as the network communication of bot machines with their C2C server in this phase have unique characteristics. This stage likewise assists with recognizing both bot machine and associated C&C server. These botnet detection methods encounter with flow collection challenges and require manual response process due to segregated control. The introduced delays during the response part due to the dependency of manual human intervention making it a difficult job to protect against malicious actors. The mean time to respond to detected botnet infection is critical to prevent any damage. The botnet literature lacking integration of detection and response process.

The work in this paper, proposed an automated countermeasures selection approach for detected botnet infections as an effort to reduced delays introduced due to a completely manual response process. We use multiclass

Manuscript received May 5, 2024
Manuscript revised May 20, 2024
https://doi.org/10.22937/IJCSNS.2024.24.5.23

botnet detection approach as proposed in [1] for botnet detection. This detection method also detects botnet class as compare to similar previous proposals [4] [5]. In general countermeasure strategy or reaction, frameworks have a much wider scope as discussed in [2]. But this work focuses only on a reaction method of detected botnet infections at network level considering the SDN dynamic programmable capability. The work proposed in [3] also exploits SDN dynamic network programmability to propose an automatic countermeasure selection against identified vulnerabilities in monitored virtual machines using the attack graph model approach.

The countermeasure selection method proposed in this work uses the severity score computation approach to select appropriate countermeasures against detected botnet infections. The proposed method uses a subset of components of countermeasure strategy including detection system, countermeasure knowledge, atomic countermeasure options, and list of actions and select actions as defined in [2]. The proposed work selects a pool of countermeasures considering all three botnet types and list down potential action items against each. The cost and effectiveness of each actionable item are computed based on subject matter expertise. These selected countermeasures, related actionable items, the effectiveness, and the cost of each countermeasure helps to formulate the countermeasure matrix. This matrix helps to select the most appropriate countermeasures based on a calculated severity score of a detected botnet infection.

The rest of the paper is as follows: Section 2 discussed the background and related work. Multiclass botnet detection method overview is provided in section 3. The proposed countermeasure selection method detail is discussed in section 4. The section 5 finally conclude the paper.

2. Related Work:

To the best of our knowledge there is no proposals to automate response function of detected botnet infection. There are few proposals that try to automate countermeasure selection for intrusion detection system and identified vulnerabilities or try to automate response process in general using SDNs dynamic programmability.

The botnet literature has a rich collection of detection proposals starting from signature-based approaches towards more complex machine learning based behavior approaches, but lacking proposals for integrated response to these detected botnet infections. The software defined networks (SDNs) with centralized visibility and dynamic control not only simplified flow collection but also provide an opportunity to automate and integrate response task with detection techniques. The work proposed in [3] first detect vulnerabilities in Virtual machines method in cloud infrastructure and then select and apply appropriate countermeasure to virtually patch the detected vulnerability using dynamic programmability feature of SDNs. The work uses attack graph methodology for attack mapping and correlation for attack analysis and then select countermeasures based on calculated attack score.

The work proposed by A. Roy [2] present an attack countermeasure tree structure that helps to access attacks and related countermeasures together in a tree. The goal of countermeasures selection method is to minimize the operational cost and maximize the benefits of implementing selected countermeasures. To achieve the goal, they work proposed a few target capacities dependent on greedy, branching and bound strategies. The work proposed in [15] apply genetic algorithm to select optimum countermeasures using Bayesian attack graph methodology.

The work proposed by Moshtapha Chakir in [7], proposed a real-time risk assessment method for intrusion detection systems. The work apply pattern matching algorithm to classify IDS alerts into different classes and then apply risk assessment based on individual risk score of events. The alerts are prioritized base of risk assessment results. The SDN4S [6] proposed automated countermeasures based on SDNs dynamic programmability. The work proposed incident specific playbooks that include action and network-based countermeasures that applied automatically when a related alert is received. The goal of the work is to minimize the delays between detection and response process.

3. Multiclass botnet detection

In this paper, we apply multiclass machine learning based botnet detection approach as proposed in [1] to detect botnet and its class as well. The approach divides botnets into three classes including IRC, HTTP, and P2P based on its underline command and control communication network architecture. The approach collects network flows centrally from SDNs controller and forms bag of flows (BoFs) for same flows between two end points and call it as flow trace, Any BoFs having 10 or more flows marked as flow trace of interest. The flow traces of interest are shipped to feature computation module where historical flows of the detected flow trace fetched using time series data repository (TSDR) plugin of OpenDayLight SDN controller. Feature vector is then computed from real-time BoFs and batch of historical flows of the same trace. These features SET from current and historical network footprint then form a feature vector that is used by multiple one-class supervised machine learning algorithms to detect botnet and its class. The figure 1 present the flow diagram of botnet detection method.

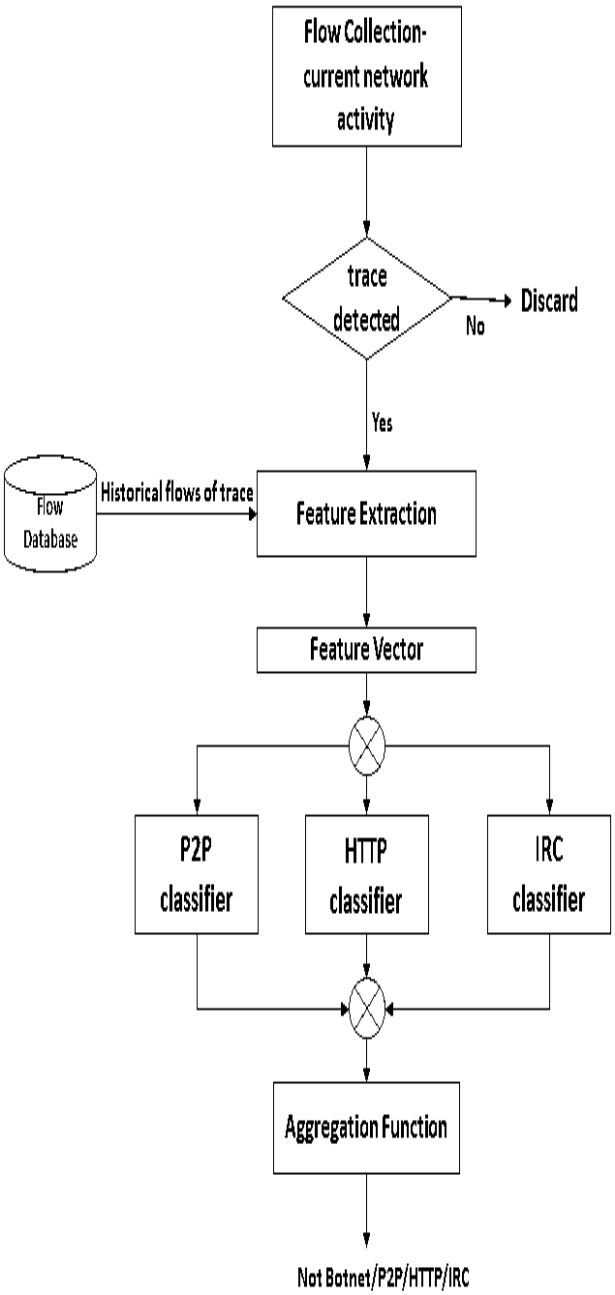

Figure 1: Flow diagram of botnet detection [1]

4. Proposed countermeasure selection method

The multiclass botnet detection model detects botnet infections and ships the alerts to the proposed countermeasure selection method where each alert contains detected trace of interest <Source IP, Destination IP, Destination Port, Protocol>, detection confidence, and detected botnet class (IRC, HTTP, or P2P). The countermeasure selection method processed these alerts and suggests relevant countermeasures. Figure 1 shows the workflow of the proposed countermeasure selection method. The proposed work maintains a pool of countermeasures in form of the matrix comprises of available countermeasures, corresponding actionable items, effectiveness, and cost of each countermeasure. The detected alerts are processed computation. The computed severity score is then used to select a subset of countermeasures from a pool of available countermeasures. The proposed method maintains three files to keep track of user-defined false positives, currently through an alert analysis function for severity score active alerts, and potential C&C servers reported by the detection model. The detail of the countermeasure matrix, Alert Preprocessing, Alert analysis, and countermeasure selection functions are provided in the subsequent sections.

4.1 Countermeasure Matrix

This section discussed the countermeasures selected to form a countermeasure matrix as shown in Table 1. The selection of countermeasures in this work based on subject matter expertise with an effort to maintain the generality but this selection may not cover the complete list of possible countermeasures against botnet infection. The countermeasure matrix is formulated to define possible actionable items against the corresponding countermeasure, the effectiveness of a countermeasure, and the cost of a countermeasure in terms of impact on the network if the countermeasure is applied. The effectiveness and cost values are key factors during the countermeasure selection decision. For the scope of this work, we assign static values to the effectiveness and cost of each countermeasure.

4.2 Alert Preprocessing

The preprocessing function process each detected alert against user-defined false positives. The user-defined false positives are a set of flow traces that are marked as false positive by the user. These flow traces are added and deleted manually by the user. A reported flow trace is discarded during alert preprocessing if its entry is found in user-defined false positives set. The reported flow trace that gets passed from the false-positive check, processed further to check if its entry exists in active-alerts. The alert preprocessing function maintains an entry of a flow trace with a timestamp that is forwarded to the alert analysis function. The lifetime of a flow trace in active-alerts is 60-minute. All reported flow traces forwarded to alert analyzer function if their entries do not exist in any of the user-defined false positive and active alert list.

4.3 Alert Analysis

The forwarded alerts from the preprocessing stage processed through alert analysis function to compute related severity scores. This computation is based on detection confidence, Local IP weight, and local and public reputation score of the remote IP of reported alert. The detection confidence is shipped with each reported alert from the detection model. The weight of local IPs of the monitored network is maintained manually in a file. The alert analysis

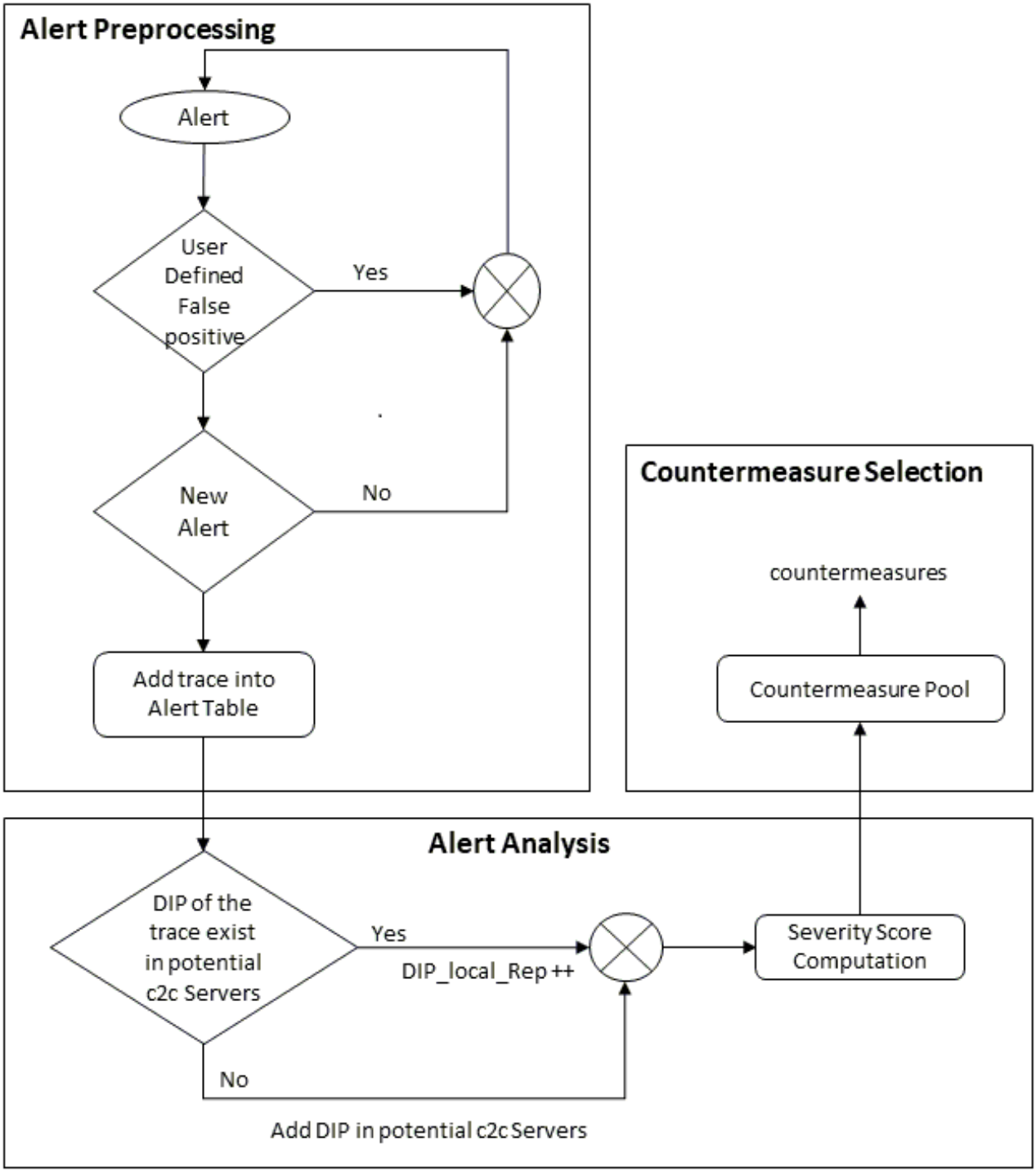

Figure 2: Flow diagram of Countermeasure selection method

function maintains a weighted list of remote IPs extracted from each processed alert to form a set of potential C&C servers. This set of remote IPs with related reputation scores used as a local threat intelligence source during severity score computation.

Table 1: Countermeasure matrix

Sr. No.	Counter measure	Actionable Items	Effectiveness	Cost
1	Alert	forward alert to SOC	1	1
2	Deep Packet Inspection	host_tracker API call, identify the closest Switch, Add rules for traffic redirection from DPI path	2	2
3	Block DIP for SIP	host_tracker API call, add a rule to block DIP for SIP on the directly connected switch	3	3
4	Block DIP for VLAN of SIP	host_tracker API call, add a rule to block DIP on the Directly connected switch	4	4
5	Block DIP on network	Block DIP on the edge switch	4	5
6	Block Internet Access of SIP	host_tracker API call, add a rule to block Internet Access of SIP	3	4
7	Isolate SIP	host_tracker API call, add a rule to Isolate SIP on the directly connected switch	4	5

computation. The destination IP of newly processed alert added into the potential C&C servers list with reputation score and timestamp if it does not already exist there. The lifetime of this newly added destination IP is 60-minute which gets re-initialized if the same destination IP and with different flow trace is reported during the lifespan. The alert analysis function assigns a reputation score of "0" to newly added entry in potential C&C servers list. This reputation score gets incremented by 0.1 on a scale of 0-1 if the same destination IP and different flow trace key is reported within 60-minutes. The alert analysis function also queries for the public reputation of the remote IP of the processed alert.

The severity score of each processed alert is computed using the following equation

$$\text{Severity Score} = \frac{\alpha+\beta+\gamma+\delta}{4}$$

Where
α represent the detection confidence of reported alert
β is the static weight of Source IP added manually for all local IPs
γ is the local reputation score of remote IP
δ is the reputation score of remote IP queries from a publicly available threat intelligence source
The mapping of resulted severity score to different severity levels is defined in Table 2. These levels are used during the countermeasure selection process.

Table 2: Severity Score to severity level mapping

Severity Score	Severity Level
Severity Score<0.5	low
0.5<Severity Score<0.75	Medium
0.75<Severity Score<0.90	High
0.90<Severity Score	Critical

4.4 Countermeasure selection

The countermeasure selection method is designed to select appropriate countermeasures against detected botnet infection from the available pool of countermeasures. The goal of this selection is to maximize the effect and minimize the impact on the underline network of the computer system. To achieve this, a countermeasure selection matrix is formulated based on alert severity level as computed in the previous section and detected the type of botnet of the alert. This matrix map countermeasure to a different combination

of severity level and botnet type which helps to select appropriate countermeasures for the processed alert. Table 3 shows the mapping between countermeasures and pairs of severity level and botnet type.

Table 3: Countermeasure selection matrix

Botnet Type	Severity Level	Countermeasure
IRC	low	Alert
HTTP	low	Deep Packet Inspection, Alert
P2P	low	Deep Packet Inspection, Alert
IRC	Medium	Deep Packet Inspection
HTTP	Medium	Block DIP for SIP, Alert
P2P	Medium	Block DIP for VLAN of SIP, Alert
IRC	High	Block DIP for SIP, Alert
HTTP	High	Block DIP for VLAN of SIP, Alert
P2P	High	Block DIP on the network, Alert
IRC	Critical	Block DIP on the network, Alert
HTTP	Critical	Block Internet Access of SIP, Block DIP on the network, Alert
P2P	Critical	Isolate SIP

4. Concluding Remarks

In this paper, we presented an automated countermeasure selection method for detected botnet infections. The multiclass machine learning based botnet detection approach as proposed in [1] is used as a detection module in this work. The detection module output botnet detection and also identify the class of detected botnet as IRC, HTTP, or P2P. The proposed countermeasure selection approach uses these detections as input and apply risk-based severity score computation. A countermeasure matrix is formulated having countermeasures and their respective cost and effectiveness. The computed severity score divided into four severity level including low, medium, high, and critical. These severity level and detected botnet class helps to select optimum countermeasure from available pool.

References

[1] F. Tariq and S. Baig, “Multiclass Machine Learning Based Botnet Detection in Software Defined Networks,” IJCSNS, vol. 19, no. 3, p. 150, 2019.

[2] Nespoli, Pantaleone, et al. "Optimal countermeasures selection against cyber attacks: A comprehensive survey on reaction frameworks." IEEE Communications Surveys & Tutorials 20.2 (2017): 1361-1396.

[3] Chung, Chun-Jen, et al. "NICE: Network intrusion detection and countermeasure selection in virtual network systems." IEEE transactions on dependable and secure computing 10.4 (2013): 198-211.

[4] F. Tariq and S. Baig, “Botnet classification using centralized collection of network flow counters in software defined networks,” Int. J. Comput. Sci. Inf. Secur., vol. 14, no. 8, p.

[5] F. Tariq and S. Baig, “Machine learning based botnet detection in software defined networks,” Int. J. Secur. Its Appl., vol. 11, no. 11, pp. 1–11, 2017.

[6] Koulouris, Theofrastos, M. Casassa Mont, and Simon Arnell. "SDN4S: Software defined networking for security." Hewlett Packard Labs, Palo Alto, CA, USA, Tech. Rep (2017).

[7] Chakir, El Mostapha, Mohamed Moughit, and Youness Idrissi Khamlichi. "A real-time risk assessment model for intrusion detection systems." 2017 International Symposium on Networks, Computers and Communications (ISNCC). IEEE, 2017.

[8] Chakir, El Mostapha, Mohamed Moughit, and Youness Idrissi Khamlichi. "A real-time risk assessment model for intrusion detection systems." 2017 International Symposium on Networks, Computers and Communications (ISNCC). IEEE, 2017.

[9] N. Poolsappasit, R. Dewri, and I. Ray, “Dynamic security risk management using bayesian attack graphs,” IEEE Trans. Dependable and Secure Computing, vol. 9, no. 1, pp. 61–74, Feb. 2012.

[10] Open Networking Fundation, “Software-defined networking: The new norm for networks,” ONF White Paper, Apr. 2012.

[11] Wang, Lingyu, Anyi Liu, and Sushil Jajodia. "Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts." Computer communications 29.15 (2006): 2917-2933.

[12] Cichonski, Paul, et al. "Computer security incident handling guide." NIST Special Publication 800.61 (2012): 1-147.

[13] Don, Moira West-Brown, et al. "Handbook for computer security incident response teams (CSIRTs)." (1998).
[14] Wagner, Neal, et al. "Towards automated cyber decision support: A case study on network segmentation for security." 2016 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, 2016.
[15] Poolsappasit, Nayot, Rinku Dewri, and Indrajit Ray. "Dynamic security risk management using bayesian attack graphs." IEEE Transactions on Dependable and Secure Computing 9.1 (2011): 61-74.
[16] Modi, Ajay, and A. Doupé. "Automated Confidence Score Measurement of Threat Indicators." (2017).

Farhan Tariq *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.*

Dr. M Shamim Baig *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, Principal 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/ Head of Department of Computer Science at Muslim Youth University Islamabad. He has published more than 40 Int'l Journal/ conference papers. He has been Chair IEEE education activities & Keynote/ invited speaker at multiple Seminars*