Proceedings of the APAN – Research Workshop 2016
ISBN 978-4-9905448-6-7

An Analysis of Botnet Attack for SMTP Server using Software Define Network (SDN)

Mohd Zafran Abdul Aziz[1,2], Koji Okamura[3]
[1]Faculty of Electrical Engineering,Universiti Teknologi Mara,40450, Shah Alam, Selangor. Malaysia
[2]Department of Advanced Information Technology, Graduate School of Information Science and Electrical Engineering, Kyushu University, Japan
[3]Research Institute for Information Technology, Kyushu University, Japan
E-Mails: zafran.fke@gmail.com, oka@ec.kyushu-u.ac.jp

***Abstract*— SDN architecture overwhelms traditional network architectures by software abstraction for a centralize control of the entire networks. It provides manageable network infrastructures that consist millions of computing devices and software. In this work, we present multi-domain SDNs architecture with an integration of Spamhaus server. The proposed method allows SDN Controllers to update the Spamhaus server with latest detected spam signatures. It can help to prevent any spam email from entering others SDN domains. We also discussed a method for analyzing SMTP spam frames using a decision tree algorithm. We use Mininet tool to simulate the multi-domain SDNs with the Spamhaus server. The simulation results show that a packet Retransmission Timeout (RTO) between server and client can help to detect the SMTP spam frames.**

***Index Terms*—SDN, Software Define Network, SMTP, Spam, Botnet, SDN Security, OpenFlow, Mininet**

I. INTRODUCTION

SDN is an architecture for multi devices communication in integrated networks. It provides manageable network infrastructures that consist millions of computing devices and software. Due to growing of device connectivity and speeds, tradition networks such as LANs and WANs are no longer capable of optimizing all connectivity (e.g. network routing) and to secure networks from multi-faceted security threats. Traditional firewall and IDS are not capable of preserving a large network such as monitoring all inbound and outbound packets because the internet data is too huge to be monitored. Cloud Computing, Bigdata and IoT create deadly network traffics for the traditional network architecture, which it will cause an obsoleting and soon it will cripple the existing network functionality. SDN is one of a promising architecture that allows huge WANs/MANs to be controlled using a high-level of abstraction. The SDN architecture splits the centralize control of the entire networks (control plane) from an actual network data and routing process (data plane). All network behavior will be programmed in the centralize control using programmatic software such as SDN Application and Controller. The SDN architecture also provides a centralized security control that can help to prevent illegitimate access or network attacks such as DDos.

In this work, we present multi-domain SDNs architecture with an integration of Spamhaus server. The proposed method allows SDN Controllers to update the Spamhaus server with latest detected spam signatures. It can help to prevent any spam email from entering others SDN domains. We also discussed the method for analyzing SMTP spam frames using a decision tree algorithm. We divided this work into six sections. The first Introduction section provides an introduction to SDN and traditional network architecture. It follows the Related Works section that discusses SDN and STMP attack using botnets. After that, we discuss methodology adopted to prevent spam in SMTP protocol in the Methodology section. In the Simulation Setup section, we simulate the proposed method using an actual data in Mininet tool. We present simulation results and discussion using the Mininet in the Results and Discussion section. Finally, we conclude this work and propose a future work in the Conclusion section.

II. RELATED WORKS

This section presents related works:

A. *Software Define Network (SDN)*

SDN is an architecture for multi devices communication in integrated networks. In the initial stage, it allows multiple LANs devices and systems to be integrated into WAN networks. The first SDN began after Java language released by Sun Microsystem, which AT&T Labs Geoplex project used Java to program APIs to implement middleware networking [1]. The Geoplex provided open networking standard for network integrations and communications such as system

managements and provisions, integrated security and system authentication, network monitoring etc. The most prominent functionality of the Geoplex is it allows network IPs to be mapped to one or many system and services [2]. In 2008, research and development for SDN continue by UC Berkeley and Stanford University [3]. By 2011, Open Networking Foundation (ONF) continues to develop OpenFlow for SDN [4]. The ONF provides SND resources (e.g. switch specification) for product manufacturer and software developer to implement SDN using the OpenFlow's standard and protocol [5].

Figures 1 and 2 show a general SDN architecture and its stacks. In SDN topology, all network nodes or devices are controlled using a control plane. The architecture splits the control plane from actual network data and routing process (data plane). The infrastructure layer communicates with SDN Controller using Control Data Plane (CDP) API (e.g. OpenFlow). All nodes or routers in the SDN network will use the CDP API for all control plane communication. The control layer consists of SDN Control Software or Controller, which extract information from the infrastructure layer such as a list of all devices in the SDN network and its states. It does not provide the entire information of all connected devices, but it provides an abstract view of the SDN network and topology. The application layer uses information from the control layer for a network abstraction administrative such as network analytics; network, system and topology managements etc. [6,7].

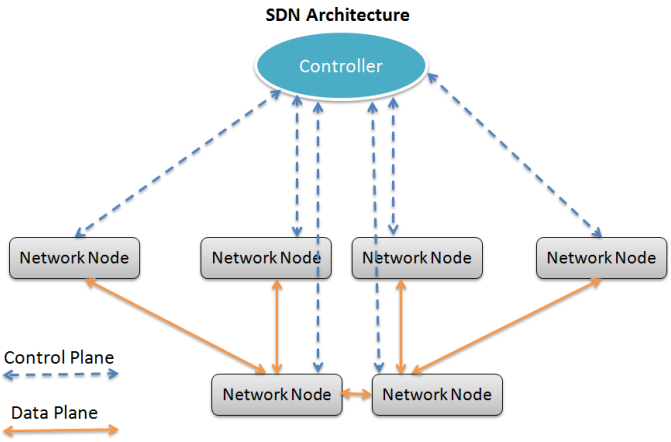

Fig. 1. SDN architecture [8].

Many SDN runs over a virtualized architecture, which the application and control layers may execute in various devices that including a virtual machine in cloud computing [10,11]. This allows application and control layers to be distributed on various computing platforms, which it will increase flexibility, mobility and computing power using the virtualized architecture, system and devices [12–14].

In this work, we will not discuss the advantage of SDN in distributed systems, but we want to assess a network security through SDN. The next subsection will discuss further the network security and threats in the SDN.

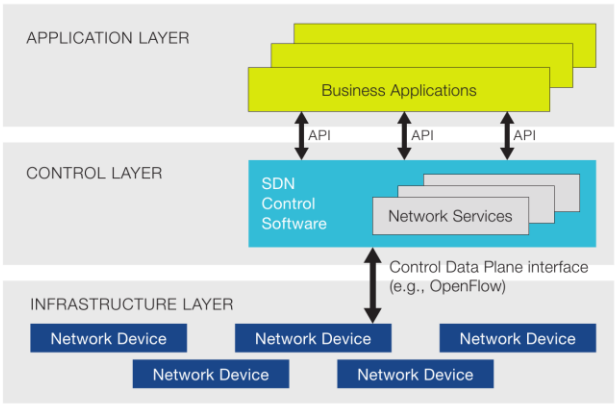

Fig. 2. SDN's stacks [9].

B. *Network Security by SDN*

Distributed systems such as cloud computing and Internet of Things (IoT) are not the main factors for organizations to migrate theirs network infrastructure into SDN, another main reason is a network security that offered by the SDN [15,16]. The SDN allows an abstraction of network security that provides a central authority in a network, which previously hard to be done by traditional distributed networking systems and infrastructures [4,5]. There are also new security problems introduces by an implementation the SDN in network infrastructure, but we are not going to discuss in this publication and one may refer to [16–19] for further examinations regarding these security problems. The following paragraphs will discuss security threats and its countermeasures using SDN.

N. Hoque et al. [20] discuss tools use by attackers and network administrators in SDN. Major attacks on SDN are Dos and DDos [21] that mounted by botnets [22]. Most botnets will try to prevent access to computing resources in the SDN by draining computing capability of the target computing system. An attacker(s) frequently used SYN-Flooding Attack [23], which sends a flood of TCP/SYN packets (by zombie machines) and leave the 3-ways TCP handshake protocol hang-up without ACK packets. This attack applied to all application protocols that are used TCP based connections such as SMTP, FTP, HTTP, DNS etc. Traditional network security systems and infrastructures rely on Intrusion Detection System (IDS) and firewall to protect LAN, WAN from the internet. It might work well for a small and manageable network such as LAN, but not for multi-WANs in a large organization (or a join of multiple organizations) in distance geographical locations. Furthermore, applying SDN for the entire internet is far away than a current topic, which requires, at least a successful implementation of SDN for multi-WANs. We skipped this part, but we want to narrow down our discussion that to improve an efficiency for botnet attack detections on SMTP protocol. The next paragraph will explore the existing methods in preventing the botnet attacks on SMTP protocol.

The most common way to detect botnet attacks are using a signature-based of known attacks [24], and a real-time detection of network anomalies [24,25] using IDS. Both

methods used congestion control and drop packet to block DDos attacks, which called Pushback method [21]. The signature-based requires others systems to provide the signature of known attacks, which can be derived from the integrate the multi-domain SDNs with Spamhaus server. S. Seeber et al [33] proposed to use the existing database (spam signatures) to secure SDN domain. We propose to integrate the Spamhaus server with multi-domain SDNs, which allow SDN

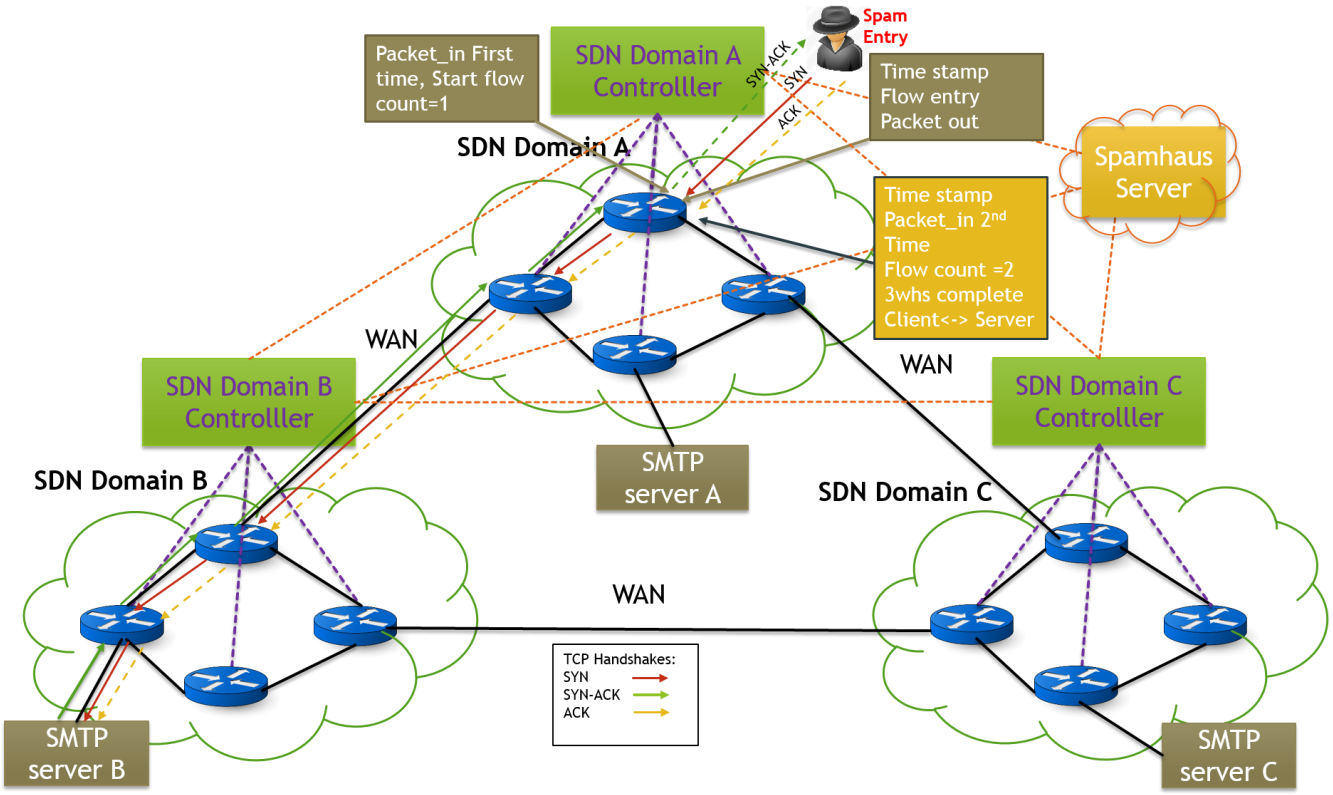

Fig. 3. Integrated Spamhaus in multi-domain SDNs.

real-time detection from a shared database. Routers within the same LANs/WANs may share or distribute attack signatures, for examples a list of blacklisted source and destination IPs, payloads, Time-to-Live (TTL) [26] etc. Another method to detect potential attacks is using a network traffic classification. It can help to identify packets send by botnets at local and enterprise networks [27]. This method may be integrated into the real-time detection method.

In this work, we used Round-Trip Time (RTT) and Retransmission Timeout (RTO) to detect an anomaly in SMTP traffic, which similar to works done by [27–32]. We enhance the existing detection methods using a new decision tree algorithm for improving detection efficiency. Second, we integrated Spamhaus [33] into SDN for a detection botnet controller list (BCL) among SDN domains. The Spamhaus server will serve all SDN Domain Controllers with latest botnet controller list (BCL). We discuss the proposed solutions in the Methodology section.

III. METHODOLOGY

This section will present the problem statements and proposed solutions. Based on latest literature as aforementioned for botnet focusing on smtp protocol detection in SDN, RTT and RTO are used for anomaly detection in SMTP traffic. However, the aforementioned literature did not Controllers to update the Spamhaus with latest botnet controller list (BCL). This will mitigate any botnet attack on smtp server from entering others SDN domains because all SDN domains will have the latest latest botnet controller list (BCL) from the Spamhaus server.

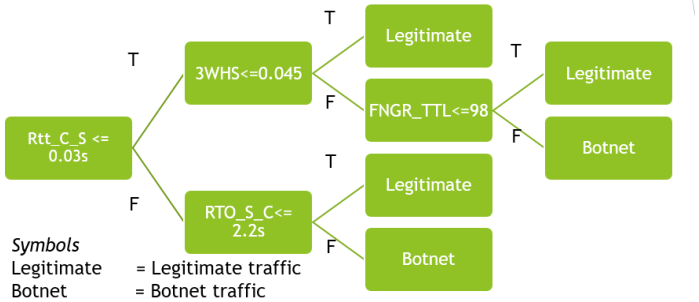

Fig. 4. Decision tree

Figure 3 show the proposed method for the Spamhaus implementation in multi-domain SDNs. For an example, a bulk botnet attack SMTP server were executed by botnets in Domain A. Controller SDN in the Domain A will verify all SMTP frames using information from the Domain A Controller. The Domain A will have latest botnet controller list

(BCL) because the Domain A Controller is connected to the Spamhaus server. At the same time, SDN Controller in the Domain A will begin to learn and detect anomaly traffic in the Domain A. The SDN Controller will use the existing algorithms and the proposed decision tree algorithm to analyze the SMTP frames as shown in Figures 4 and 5. The SDN controller Domain A will all blocked traffic based on algorithm decision tree and this information is forwarded and will update the Spamhaus server. This will enable botnet controller list (BCL) sharing between multi-domain SDNs.

Module 0
If dst port==25, Then
  Forward to controller
  Packet_in Flow count
  Go to module 1
Else drop the packet

Module 1 (RTT Client-Server)
If rtt client <-> server between two switch t>= 0.0087s, Then
  Go to module 2
Else
  Go to module 3

Module 2 (3 ways handshake flow count and time)
If flow count packet_in == 2, same src ip, same dist ip, time arrival for 2nd flow <= 0.087 between client <-> server, Then
  Install the flow in flow table, forward the next packet
Else
  Go to module 4

Module 3 (RTO Server-Client)
If RTO from server < 2.2 second, Then
  Install the flow in the flow table and forward the next packet
Else
  Blacklist the ip source send information to Spamhaus

Module 4 (TTL)
If ip TTL<= 96, Then
  Install the flow in the flow table and forward the next packet
Else
  Blacklist the ip source send information to Spamhaus

Fig. 5. Decision tree algorithm

IV. Simulation Setup

This section discussed the simulation setup using Mininet [34]. It allows one to create a virtual network and its components. The Mininet being used by OpenFlow for SDN simulation [35]. Figure 3 shows the overview architecture of simulation setup for this work. The simulation used the internet traffic dataset from University New Brunswick (UNB), Canada [36]. The same dataset was used by E. B. Beigi et al. [32] for botnet detection in their publication. Figures 6 and 7 show the simulation of the dataset using Mininet.

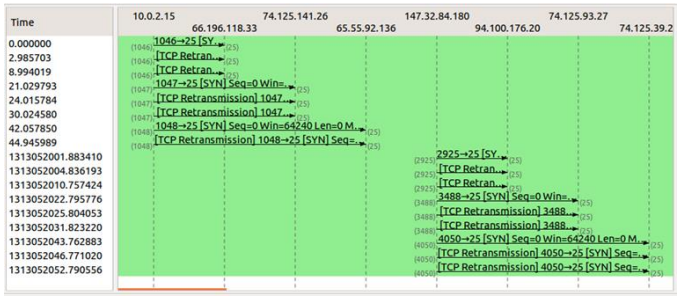

Fig. 6. A flow graph of SYN flood

Figures 8 and 9 show two traffics from seven traffic datasheets that were tested in the simulation. Figures 10 and Table 1 show the summary of max RTT and RTO for seven traffic datasheets. These results can be used to identify botnet smtp attack packets in a network. Refer to the decision tree in Fig. 4, any packet does not satisfy the decision tree is dropped from the SDN domain.

Refer to the Botnet training and testing columns, any packet RTO between server and client greater than 2.2 seconds (a baseline from botnet training), the packet must be dropped.

The RTO and RTO2 (2nd time runs of the RTO) provided significant results for a botnet detection. The 3WHS is expected to be less or equal to 0.045 second, which provides an unimportant timing for a botnet detection.

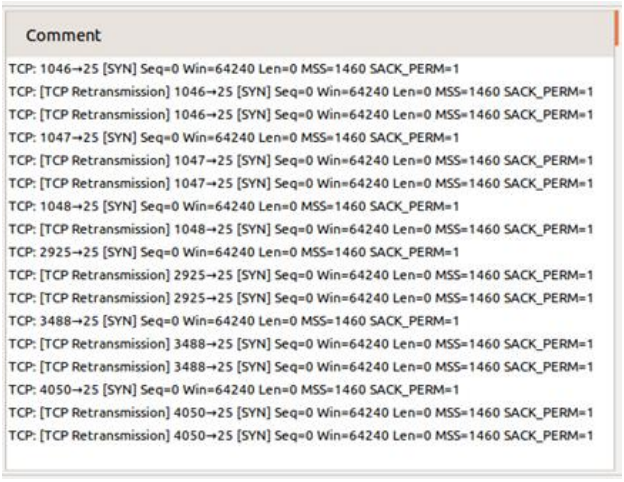

Fig. 7. A flow graph botnet for SYN flood (comment)

V. Results & Discussion

Refer to the Jun-12 until Jun-16 columns, the RTT between client and server must be less or equal to 0.03 second. The RTO and RTO2 are less than zero second, which provides an insignificant timing for a botnet detection. The 3WHS is expected to be less or equal to 0.045 second, which also provides an unimportant timing for a botnet detection. Refer to Figure 11 and Table 2, the TTL for botnet training and testing are equal to 128.

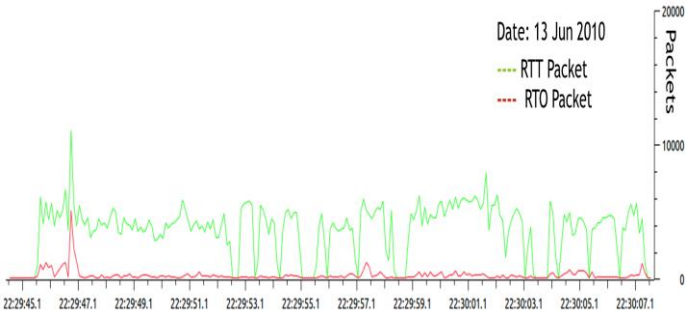

Fig. 8. A total of SMTP packets per second on 13 Jun 2010

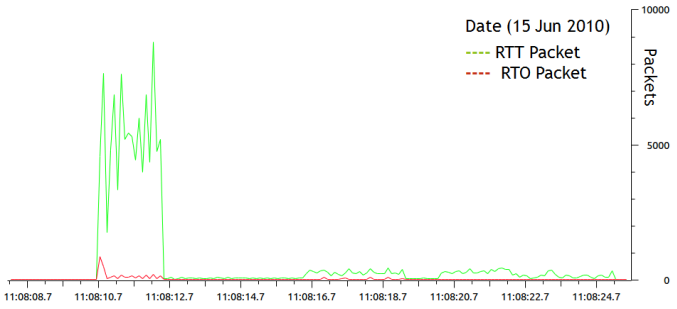

Fig. 9. A total of SMTP packets per second on 15 Jun 2010

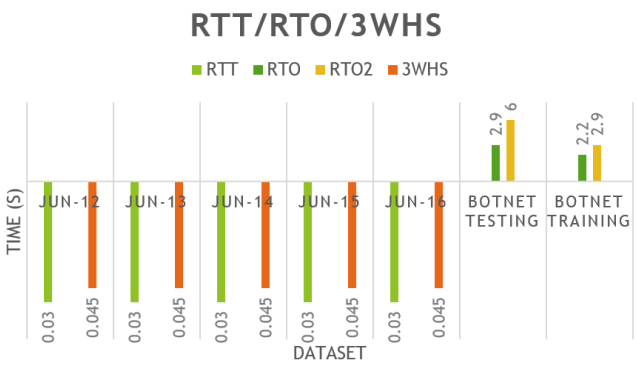

Fig. 10. A summary of max RTT and RTO for seven traffic datasheets

Table 1
A summary of max RTT and RTO for seven traffic datasheets

DATASET	RTT (s)	RTO (s)	RTO2 (s)	3WHS (s)
Jun-12	0.03	0	0	0.045
Jun-13	0.03	0	0	0.045
Jun-14	0.03	0	0	0.045
Jun-15	0.03	0	0	0.045
Jun-16	0.03	0	0	0.045
Botnet Testing	0	**2.9**	**6**	0
Botnet Training	0	2.2	2.9	0

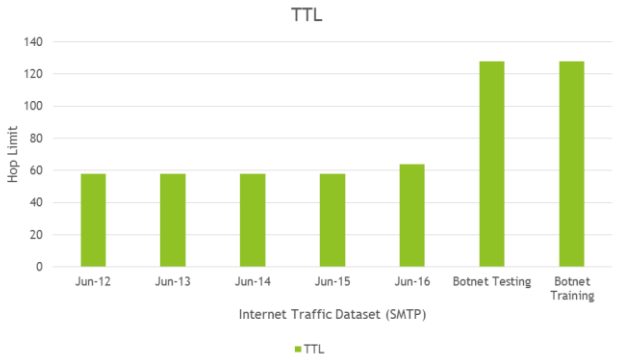

Fig. 11. A graph of average TTL for packet for seven traffic datasheets

Table 2
A table of average TTL for packet for seven traffic datasheets

DATASET	TTL
Jun-12	58
Jun-13	58
Jun-14	58
Jun-15	58
Jun-16	64
Botnet Testing	**128**
Botnet Training	128

VI. Conclusion

We have presented multi-domain SDNs with Spamhaus server. The proposed method allows SDN Controllers to update the Spamhaus server with latest botnet controller list (BCL) and it will help to prevent any botnet attack on smptp server from entering others SDN domains. We also discussed the method for analyzing SMTP traffics flow using decision tree algorithm. The method utilized a packet RTO between server and client to detect the SMTP traffic flow. We plan to implement the multi-domain SDNs with Spamhaus server as a future work. We hope the future experiment will provide a solution for securing the multi-domain SDNs from botnet attack to smtp server.

References

[1] G. Vanecek, GeoPlex: Universal Service Platform for IP Network-based Services, 1997. http://www.cerias.purdue.edu/news_and_events/events/security_seminar/details/index/56218-noZCKZKV1F3c-372-hq15nTbsPk31bQ8W.

[2] N.V. Michah Lerner, George Vanecek, Middleware Networks: Concept, Design and Deployment of Internet Infrastructure, Kluwer Academic Publishers Norwell, 2000.

[3] S. Shenker, Gentle Introduction to Software-Defined Networking, 2012. https://www.youtube.com/watch?feature=player_detailpage&v=eXsCQdshMr4&t=168.

[4] Open Networking Foundation, Software-Defined Networking: The New Norm for Networks, 2012.

[5] O.N. Foundation, OpenFlow, (2016). https://www.opennetworking.org/sdn-resources/openflow/57-sdn-resources/onf-specifications/openflow?layout=blog (accessed January 29, 2016).

[6] S.H. Park, B. Lee, J. You, J. Shin, T. Kim, S. Yang, RAON: Recursive abstraction of OpenFlow networks, Proc. - 2014 3rd Eur. Work. Software-Defined Networks, EWSDN 2014. (2014) 115–116. doi:10.1109/EWSDN.2014.29.

[7] V.K. Gurbani, M. Scharf, T. V. Lakshman, V. Hilt, E. Marocco, Abstracting network state in Software Defined Networks (SDN) for rendezvous services, IEEE Int. Conf. Commun. (2012) 6627–6632. doi:10.1109/ICC.2012.6364858.

[8] Mouli, Why SDN Concepts Need To Extend Into The Wan, (2016). http://www.aryaka.com/blog/why-sdn-concepts-need-to-extend-into-the-wan/ (accessed January 31, 2016).

[9] SDxCentral, Inside SDN Architecture, (2016). https://www.sdxcentral.com/resources/sdn/inside-sdn-architecture/ (accessed January 31, 2016).

[10] S. Azodolmolky, SDN-based cloud computing networking, in: Transparent Opt. Networks, 2013: pp. 2–5. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6602678.

[11] R. Jain, S. Paul, Network virtualization and software defined networking for cloud computing: A survey, IEEE Commun. Mag. 51 (2013) 24–31. doi:10.1109/MCOM.2013.6658648.

[12] A. Dixit, F. Hao, S. Mukherjee, Towards an elastic distributed sdn controller, in: Proc. …, 2013: pp. 7–12. doi:10.1145/2491185.2491193.

[13] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, et al., Onos, in: Proc. Third Work. Hot Top. Softw. Defin. Netw. - HotSDN '14, 2014: pp. 1–6. doi:10.1145/2620728.2620744.

[14] R. Beverly, K. Sollins, Exploiting Transport-Level Characteristics of Spam, (2008). http://18.7.29.232/handle/1721.1/40287.

[15] S. Scott-Hayward, G. O'Callaghan, S. Sezer, Sdn Security: A Survey, 2013 IEEE SDN Futur. Networks Serv. (2013) 1–7. doi:10.1109/SDN4FNS.2013.6702553.

[16] R. Kl, P. Smith, OpenFlow: A Security Analysis, in: 21st IEEE Int. Conf. Netw. Protoc., 2013. doi:10.1109/ICNP.2013.6733671.

[17] R.L. Smeliansky, SDN for network security, Sci. Technol. Conf. (Modern Netw. Technol. (MoNeTeC), 2014 First Int. (2014) 1–5. doi:10.1109/MoNeTeC.2014.6995602.

[18] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, et al., Heuristic Approaches to the Controller Placement Problem in Large Scale SDN Networks, IEEE Trans. Netw. Serv. Manag. 12 (2015) 4–17. doi:10.1109/TNSM.2015.2402432.

[19] Z.Y. and F.B. Wang Shuling, Li Jihan, Research on SDN Architecture and Security, Telecommun. Sci. 29 (2013) 117–122.

[20] N. Hoque, M.H. Bhuyan, R.C. Baishya, D.K. Bhattacharyya, J.K. Kalita, Journal of Network and Computer Applications Network attacks : Taxonomy , tools and systems, 40 (2014) 307–324.

[21] J. Ioannidis, S.M. Bellovin, Implementing Pushback: Router-Based Defense Against DDoS Attacks, 2014. doi:10.1007/s13398-014-0173-7.2.

[22] S. Lim, J. Ha, H. Kim, Y. Kim, S. Yang, A SDN-oriented DDoS blocking scheme for botnet-based attacks, 2014 Sixth Int. Conf. Ubiquitous Futur. Networks. (2014) 63–68. doi:10.1109/ICUFN.2014.6876752.

[23] R.K. Sahu, N.S. Chaudhari, A performance analysis of network under SYN-flooding attack, IFIP Int. Conf. Wirel. Opt. Commun. Networks,

WOCN. (2012) 2–4. doi:10.1109/WOCN.2012.6335561.
[24] M.-S.K.M.-S. Kim, H.-J.K.H.-J. Kong, S.-C.H.S.-C. Hong, S.-H.C.S.-H. Chung, J.W. Hong, A flow-based method for abnormal network traffic detection, 2004 IEEE/IFIP Netw. Oper. Manag. Symp. (IEEE Cat. No.04CH37507). 1 (2004) 1–14. doi:10.1109/NOMS.2004.1317747.
[25] P. Nevlud, M. Bures, L. Kapicak, J. Zdralek, Anomaly-based Network Intrusion Detection Methods Keywords Detection of Network Anomalies, (2013) 468–474. doi:10.15598/aeee.v11i6.877.
[26] B. Xiao, W. Chen, Y. He, E.H.M. Sha, An active detecting method against SYN flooding attack, Proc. Int. Conf. Parallel Distrib. Syst. - ICPADS. 1 (2005) 709–715. doi:10.1109/ICPADS.2005.67.
[27] H. Chen, C. Mao, S. Tseng, An Approach for Detecting a Flooding Attack Based on Entropy Measurement of Multiple E-Mail Protocols, 18 (2015) 79–88. doi:10.6180/jase.2015.18.1.10.
[28] C. Schafer, Detection of Compromised Email Accounts Used by a Spam Botnet with Country Counting and Theoretical Geographical Travelling Speed Extracted from Metadata, 2014 IEEE Int. Symp. Softw. Reliab. Eng. Work. (2014) 329–334. doi:10.1109/ISSREW.2014.32.
[29] H. Luo, B. Fang, X. Yun, Anomaly detection in SMTP traffic, Proc. - Third Int. Conf. onInformation Technol. New Gener. ITNG 2006. 2006 (2006) 408–413. doi:10.1109/ITNG.2006.34.
[30] G. Kakavelakis, J. Young, Auto-learning of SMTP TCP Transport-Layer Features for Spam and Abusive Message Detection., Lisa. (2011). http://static.usenix.org/events/lisa11/tech/slides/beverly.pdf.
[31] T. Sochor, Overview of e-mail SPAM Elimination and its Efficiency, Res. Challenges Inf. Sci. (RCIS), 2014 IEEE Eighth Int. Conf. (2014) 1 – 11.
[32] E. Biglar Beigi, H. Hadian Jazi, N. Stakhanova, A.A. Ghorbani, Towards effective feature selection in machine learning-based botnet detection approaches, 2014 IEEE Conf. Commun. Netw. Secur. (2014) 247–255. doi:10.1109/CNS.2014.6997492.
[33] S. Seeber, L. Stiemert, Towards an SDN-Enabled IDS Environment, in: Commun. Netw. Secur., 2015: pp. 751–752.
[34] Mininet Team, Mininet, (2016). http://mininet.org/ (accessed February 22, 2016).
[35] M. Gupta, J. Sommers, P. Barford, Fast, accurate simulation for SDN prototyping, Proc. Second ACM SIGCOMM Work. Hot Top. Softw. Defin. Netw. - HotSDN '13. (2013) 31. doi:10.1145/2491185.2491202.
[36] U.N. Brunswick, CTU-Malware-Capture-Botnet-1, (2015). http://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-1/ (accessed May 20, 2012).

Mohd Zafran Abdul Aziz has received his first Bachelor Degree (B. Eng of Electrical and Computer Science) from Kumamoto University ,Japan on March 2001 and obtained his Master Degree (MSc of Engineering) from Tokyo University Of Technology ,Japan on March 2008. He also has 6 years in industrial as project engineer in several multinational company focus on industrial automation and instrument engineer. Currently on study leave as lecturer from Computer Department of Universiti Teknologi MARA, Shah Alam ,Selangor , Malaysia .

He is currently a PhD candidate and belong to Department of Advanced Information Technology, Graduate School of Information Science and Electrical Engineering, Kyushu University, Japan.

Koji Okamura is a Professor at Research Institute for Information Technology, Kyushu University and Director of Cybersecurity Centre Kyushu University, Japan. He received B.S and M.S. Degree in Computer Science and Communication Engineering and Ph.D. in Graduate School of Information Science and Electrical Engineering from Kyushu University, Japan in 1988,1990 and 1998,respectively.

He has been a researcher of MITSUBISHI Electronics Corporation Japan for several years and has been a Research Associate at the Graduate School of Information Science ,Nara Institute of Science and Technology ,Japan and Computer Centre, Kobe University , Japan. He's area of interest is Future Internet and Next Generation Internet, Multimedia Communication and Processing, Multicast/IPV6/QoS , Human Communication over Internet and Active Network. He is a member of WIDE, ITRC , GENKAI , HIJK project and Key person of Core University Program on Next Generation Internet between Korea and Japan sponsored by JSPS/KOSEF.