Computer Science and Information Technologies
Vol. 5, No. 2, July 2024, pp. 99~111
ISSN: 2722-3221, DOI: 10.11591/csit.v5i2.p99-111

Implementation of automation configuration of enterprise networks as software defined network

Lindo Prasetyo, Ifan Prihandi, Muhammad Rifqi, Rahmat Budiarto
Department of Informatics, Faculty of Computer Science, Mercu Buana University, Jakarta, Indonesia

Article Info

Article history:

Received Jun 27, 2023
Revised Dec 2, 2023
Accepted Jan 10, 2024

Keywords:

Ansible
Cisco ACI
Network configuration
Optimization
Software defined network

ABSTRACT

Software defined network (SDN) is a new computer network configuration concept in which the data plane and control plane are separated. In Cisco system, the SDN concept is implemented in Cisco Application Centric Infrastructure (Cisco ACI), which by default can be configured through the main controller, namely the Application Policy Infrastructure Controller (APIC). Conventional configuration on Cisco ACI creates problems, i.e.: the large number of required configurations causes the increase of time required for configuration and the risk of misconfiguration due to repetitive works. This problem reduces the productivity of network engineers in managing Cisco system. In overcoming these problems, this research work proposes an automation tool for Cisco ACI configuration using Ansible and Python as an SDN implementation for optimizing enterprise network configuration. The SDN is implemented and experimented at PT. NTT Indonesia Technology network, as a case study. The experimental result shows the proposed SDN successfully performs multiple routers configurations accurately and automatically. Observations on manual configuration takes 50 minutes and automatic configuration takes 6 minutes, thus, the proposed SDN achieves 833.33% improvement.

Corresponding Author:

Rahmat Budiarto
Department of Informatics, Faculty of Computer Science, Mercu Buana University
Jakarta 11650, Indonesia
Email: rahmat.budiarto@mercubuana.ac.id

1. INTRODUCTION

Software defined network (SDN) is a new concept for changing networks [1]. It is a network approach with the principle of separating the data plane from the control plane in contrast to conventional networks [2], [3]. It makes the production networks becoming programmable, more flexible and fast in supporting virtualized servers and storage in modern data centers [4], [5]. SDN does this by extracting the control plane functions from forwarding devices such as switches and routers and detaching these functions on the SDN controller [6]. Currently there are many SDN solutions from several vendors such as Cisco Application Centric Infrastructure (ACI), Vmware NSX, Cisco SD-WAN, and Fortinet SD-WAN [7]. The ACI is one of the concepts in SDN by implementing architecture under application requirements. This architecture aims to modify, optimize, and accelerate the application development cycle [8]. Cisco ACI is controlled by an SDN controller known as the Application Policy Infrastructure Controller (APIC) [7].

In configuring the Cisco ACI, technicians only need to access APIC as the main controller, which has the characteristics of using a web user interface (a point and click graphical user interface that can only perform one configuration at a time) and has 2-layer and 3-layer configuration types. When many configurations are required, the characteristics of Cisco ACI may cause new problems such as the time

required for configurations and repetitive works will increase the risk of misconfiguration. This problem reduces the level of effectiveness of network technicians in configuring Cisco ACI.

Many research works have been carried out, such as Siddartha and Praveen [9] that discuss operating system (OS) upgrading progress automation on Cisco SD-WAN controller devices. In this study, Python was used as an automation tool that utilizes the representational state transfer application program interface (REST API) features provided by the Cisco SD-WAN controller device. The experimental results of this study show that the proposed method can speed up the OS upgrading process on Cisco SD-WAN controller devices when compared to manual upgrades. Further related research was carried out by Fauzi et al., [10], which discuss the automation of the enhanced interior gateway routing protocol (EIGRP) routing protocol configuration on Cisco routers. In this study, Ansible was used as a tool to automate the configuration of router devices by utilizing the Secure Shell (SSH) feature provided by Cisco routers. Based on the existing problems and referring to previous related researches, this research will automate the Cisco ACI configuration using Ansible and Python programming language. Ansible will be used to develop the automation tool that removes some repetitive work on servers, while Python is used for implementing the SDN, because it has a very clear, complete, and easy to understand programming code. Therefore, this paper contributes towards the development of automation method for CISCO ACI configurations to assists network administrator in increasing their productivity through SDN implementation.

2. THEORITICAL BACKGROUND

2.1. Cisco ACI

Cisco ACI is a data center architecture designed to meet the requirements of today's traditional networks, and to meet the emerging demands of new computing trends and business factors deployed in networks and is based on the Cisco Nexus 9504 and Cisco Nexus C9336'FX2 equipment that allows one to Connect to MiCC components with a speed up to 100 Gbps and more. SDN has garnered a lot of attention in the networking industry over the past few years because it promises to be a more agile and programmable network infrastructure. Cisco ACI not only addresses the challenges of network agility and programmability that software-based overlay networks are trying to address, but also provides solutions to new challenges that SDN technology cannot currently address. The main controller application on SDN, namely APIC, is responsible for all tasks that enable traffic transportation, which includes fabric activation, switch firmware management, and configuration of network policy installations [11], [12].

2.2. Ansible

Ansible is an open-source automation tool for managing and configuring computers based on the yet another markup language (YAML) language. Red hat and Ansible are developed by the open-source community. Ansible is designed to handle complex infrastructure rather than a single case with the advantages of being easier in the setup process, easy to manage, low cost of up to 100 nodes using Ansible Tower or free using ansible web executable (AWX) Cloud. The creation of the Ansible scripts is easy for System Administrators and operators to understand due to the use of YAML Configuration Files, which is administrator oriented. The configuration is easy to understand and and to distribute remotely using the SSH so that server setup does not require additional commands, and server setup process is faster [13], [14].

Ansible is executed using a script called playbook and consists of modules. Each module represents a logical command with customizable arguments and execution parameters. Modules are Python scripts that run on the target machine. Ansible has a domain specific language (DSL) for describing modules in playbooks. The DSL allows the use of variables for example the set_fact module can assign values to variables. Values in other variables can look like {{var_name}} which will be replaced with var_name values during module execution [15].

2.3. WSL

Windows subsystem Linux (WSL) was released in 2016 as a feature of Windows 10 that allows the distinction of running Linux distributions through a kernel compatible with a Linux interface [16]. The WSL allows Linux developers to run Linux environments on Windows, including most of the command line tools, utilities, and applications directly in windows, without any modifications, and without virtual machine (VM) overhead [17]. WSL can run executable and linkable format (ELF) binaries natively. The subsystem provides a kernel interface and makes it possible to run unmodified ELF64 executables [18].

2.4. YAML

YAML is very smart, human friendly and ideal for data serialization for all programming languages [19]. YAML is a format for data serialization with a computer data structure storage process for use later.

However, unlike formats that serialize data structures to a stream of raw binary data (e.g., MP3 audio and JPEG images), YAML serializes data structures to plain text [20]. YAML is widely used in data exchange, serialization, and configuration of applications or environments and basically a superset of JSON with a lightweight syntax that is optimized for human readability and editing, and has a type system consisting of scalars (numbers, strings, and booleans) and collections (lists and maps) [21].

2.5. REST API

Representational state transfer (REST) is an architecture used to design services that are consumed across multiple platforms and environments to support interoperability and the World Wide Web (WWW). The application programming interface of the REST (REST API) is broadly part of the microservices design. Research efforts were made to extend the REST architecture to support distributed systems [22], [23]. The REST API is designed for web services that focus on system resources such as transfers and requests for data using HTTP with GET, POST, PUT, or DELETE commands which are used for application development because they can be used by many programming languages and many platforms [24], [25].

3. METHOD

This study creates an automation tool for Cisco ACI configuration and implement the tool as an SDN. Experiments are carried out to measure the performance of the proposed tool whether it is as expected and to compare the time required for manual configuration and automatic configuration using the Cisco ACI configuration automation tool. Manually configuring Cisco ACI has problems, including consumed time for configuration and configuration errors (human error). Thus, automation through SDN implementation may reduce the problems. The steps in this study is depicted in Figure 1.

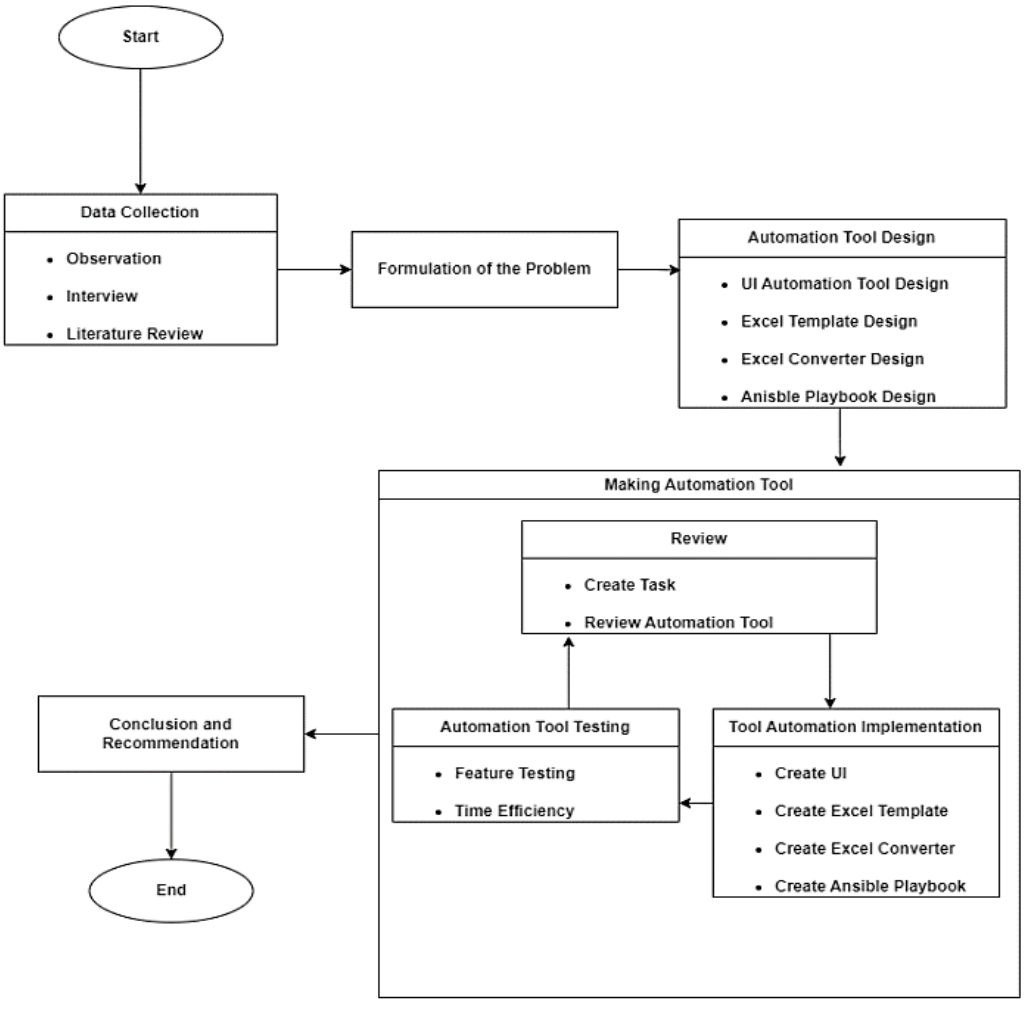

Figure 1. Research steps

3.1. Data collection

At the data collection stage, two methods were used, namely,

a) Literature study

Literature study was conducted to collect data in the form of information or theories related to this research, which were obtained through books, journals and the Internet include: Cisco ACI, Ansible, Python, WSL application, YAML, Microsoft Excel, CLI and REST API.

b) Observation

Data collection is carried out by making direct observations on the production network at PT. NTT Indonesia Technology consisting of Cisco ACI devices and interviews with network engineers.

3.2. Formulation of the problem

At the problem formulation stage, we carry out a requirement analysis to understand the existing system in order to develop its information system. At this stage an analysis of the running system is carried out at PT. NTT Indonesia Technology in the production network configuration. The current business flow is described as shown in Figure 2.

a) The network technician prepares a list of configurations that will be entered into the Cisco ACI device in the form of an Excel file.
b) The network technician opens a browser application to open the Cisco ACI device web to configure.
c) Network Technicians configure Cisco ACI devices via the web based on the list of configurations contained in the Excel file.
d) Network technician validates the configuration that has been entered into the Cisco ACI device with a list of configurations contained in the Excel file.

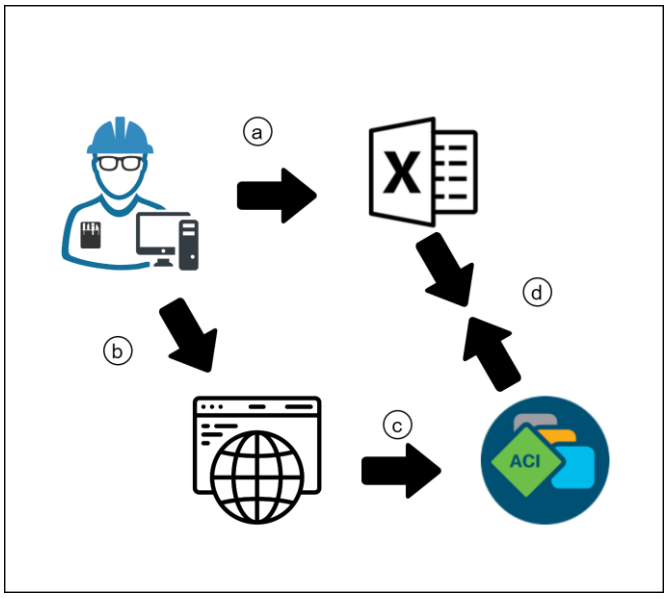

Figure 2. Current business process flow

3.3. Automation tool design

At this stage, the design of a network configuration automation tool is carried out. The automation tool is designed by mimicking the existing real production network of PT. NTT Indonesia Technology. The design of existing business processes in the automation tool is illustrated in Figure 3.

Figure 3 shows the planning of business processes in the Cisco ACI configuration automation tool, namely, the network technician or user must create a configuration list in the form of an Excel file and then the user only needs to run the Cisco ACI configuration automation tool. The created Excel file must be converted from Excel format to YAML form because only the YAML data form can be used by Ansible. After that, the user can automatically configure according to the configuration selected on the menu. Furthermore, the Cisco ACI device will send results in the form of a REST API response that the user can see in the terminal display in the Automation tool.

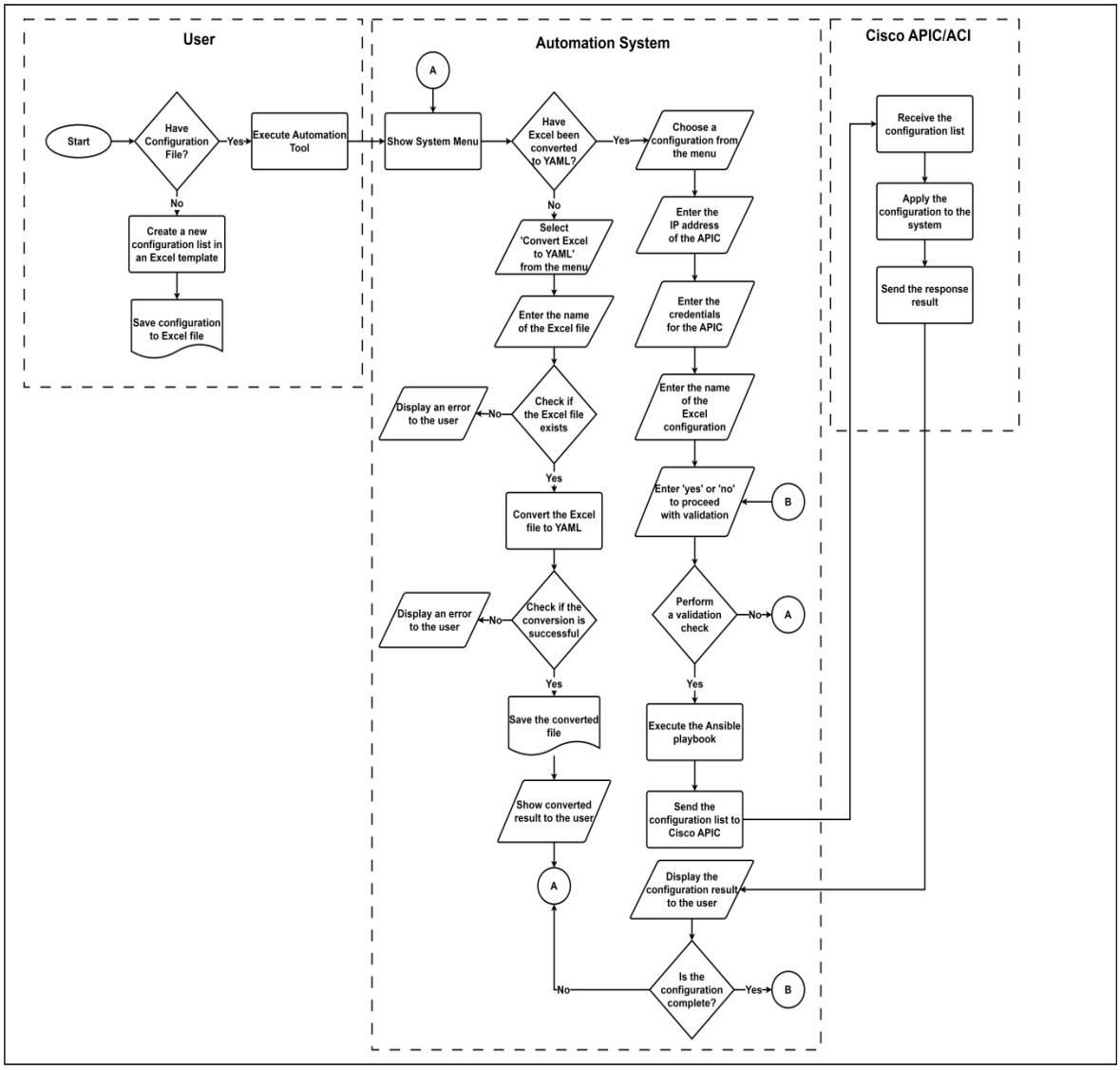

Figure 3. Proposed business process

3.5. Automation tool implementation

At this stage, the creation of a network configuration automation tool is carried out at PT. NTT Indonesia Technology in accordance with the results of the system design. The tool is made by creating a script on Ansible related to network configuration on Cisco ACI.

3.6. Testing and evaluation

At this stage testing of network configuration automation tools is conducted by testing all system functionality. Testing is carried out by ensuring the configuration entered the Cisco ACI system matches the data based on the Excel file. At this stage, an evaluation is carried out regarding the results of the research that has been carried out by comparing the manual and automatic network configuration processes.

4. RESULTS AND DISCUSSION

4.1. Result

The experiment results are presented along with steps of implementation. The first part is preparing configuration parameters as data input. The output of this process is configuration table in Excel format. Second part is automation tool implementation result, i.e.: the configuration time measurement and configuration accuracy.

4.1.1. Data preparation

Before implementing the proposed automation tool, the first thing that must be considered is to prepare the data to be used. The data is in the form of an Excel template file containing the configuration to be automated. The steps regarding how to fill in the Excel template file are as follows.

- Prepare files that are in the "input_data" folder found on the GitHub that was created.
- The Excel template has several sheets; each sheet which includes the configuration will begin with the word "SEC_" as shown in Figure 4.

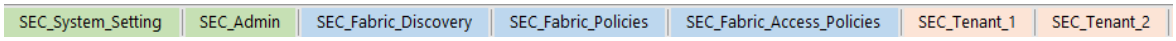

Figure 4. Configuration section

- In the "index" sheet there is information regarding the list of configurations that can be automated and show the location of the configuration sheet. The "index" sheet can help the user to find the location of the configuration sheet. For example, when SNMP Policy configuration is to be performed, the user must enter the configuration on the “SEC_FABRIC_POLICIES” sheet.
- Furthermore, the configuration sheet will have the format shown in Figure 5. It can be seen in the information section starting with "#". Each of these sentences can be interpreted only as information to help the user to find the configuration to be entered. Figure 5 shows the tables for Fabric Setup and Fabric Membership Pod configurations. There are "key_start" and "key_end" which are delimiters for the configuration table, so it must be ensured that all configurations are carried out between the "key_start" and "key_end" rows.

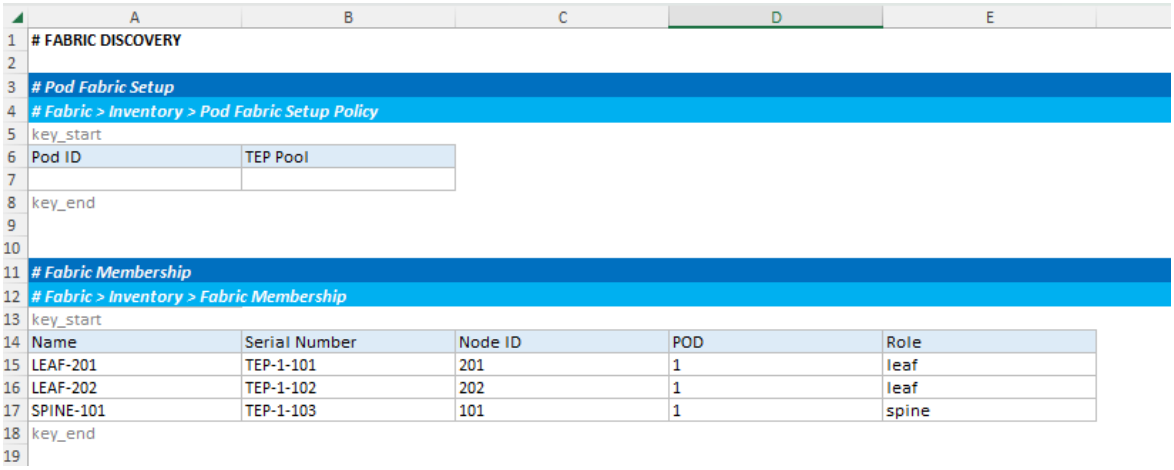

Figure 5. Configuration sheet

- Next, in the available table, we will find several configuration tables that are optional or drop down. So, it must be ensured that the user only selects the available options, as can be seen in Figure 6.

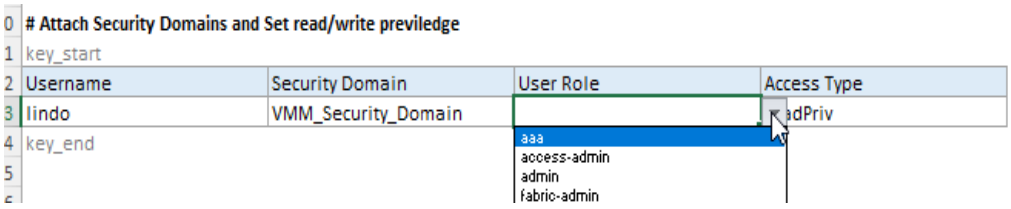

Figure 6. Configuration table

4.1.2. Automation tool implementation result

At this stage, the procedure for using the Cisco ACI automation configuration tool using Ansible and Python will be explained. In running the automation tool, system requirement is needed, i.e.: the WSL software. The following are the steps to run the Cisco ACI configuration automation tool.

– The first step that needs to be taken is to download the automation tool from Github. The file is in the ZIP version, which can be seen in Figure 7.

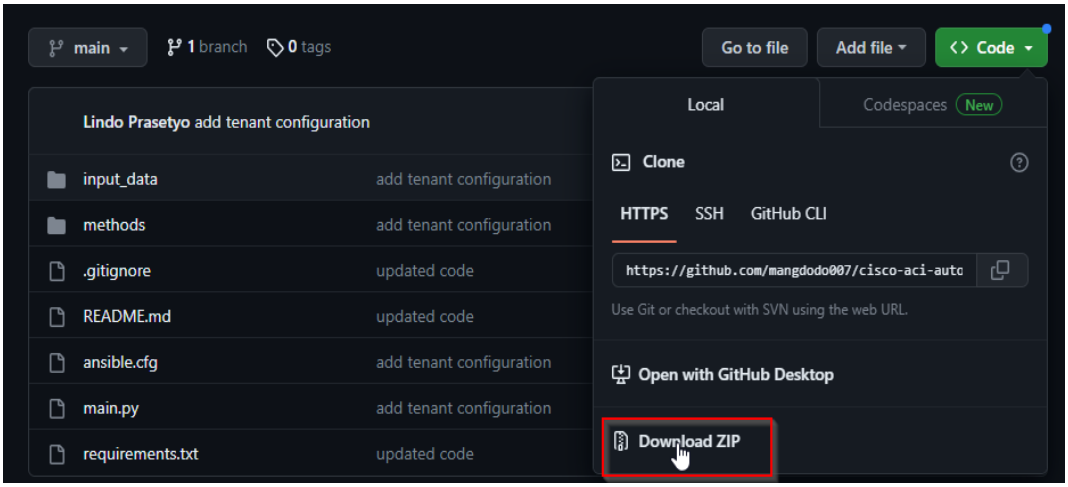

Figure 7. Download the Cisco ACI automation configuration tool

– After downloading the Cisco ACI automation configuration tool, extract the file and move it to the /home/user WSL folder that has been installed.
– Next, open the template_empty.xlsx file in the cisco-aci-automation-TA-lindo/input_data/ folder. The layout of the configuration file can be seen in Figure 8.

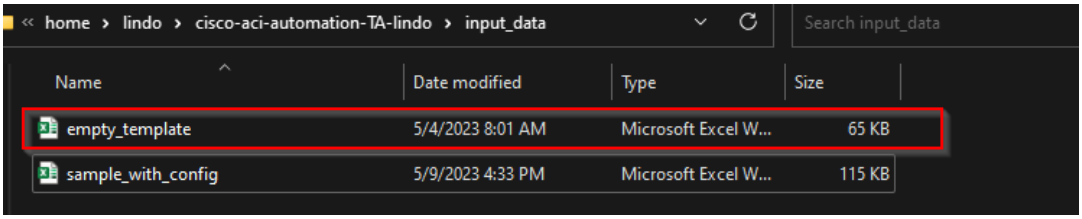

Figure 8. Opening the configuration Excel file

– Next, the Cisco ACI configuration file needs to be entered into an excel file with the file name “empty_template.xlsx. Each section will be labeled on the sheet with the name "SEC_". For example, on the second sheet there is a sheet name SEC_System_Setting, which means that the sheet is the Cisco ACI configuration used to set the system. The appearance of the configuration file name can be seen in Figure 9.

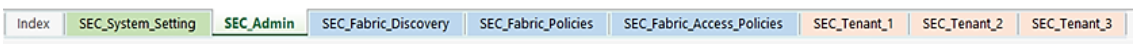

Figure 9. Configuration file name

– After all the configurations have been entered, then the "empty_template.xlsx" file can be saved and closed.
– Then open the Ubuntu terminal or WSL software via the start menu in Windows.

– The terminal will display the directory that will be used. In this research, we move the directory with the command "cd cisco-aci-automation-TA-Lindo/" so that the directory will be moved to "/cisco-aci-automation-TA-lindo/" as shown in Figure 10.

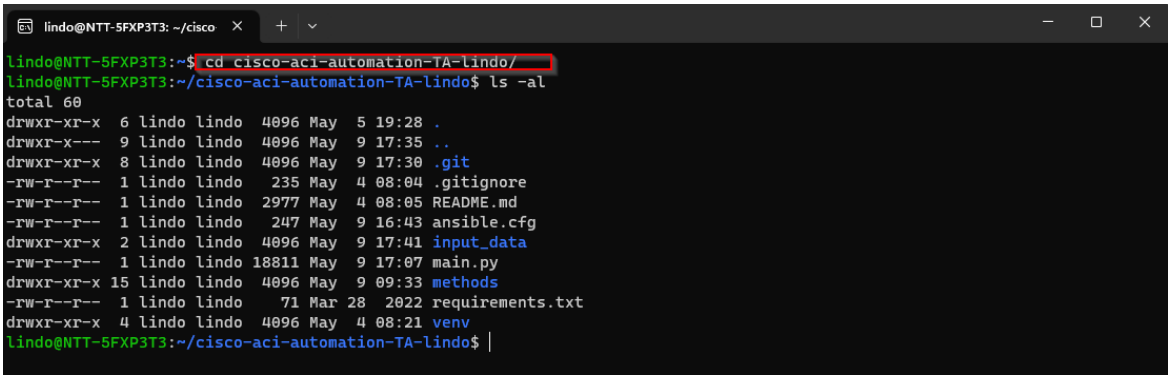

Figure 10. Change directory

– After moving directories, install all the required libraries in the "requirements.txt" file using the "pip install -r requirements.txt" command. The "requirements.txt" file contains the library information needed by the automation tool. After all the requirements have been installed, then execute the application using "python3 main.py" command as shown in Figure 11.
– Before configuring the Cisco ACI device, the device that will run the automation tool can access the APIC from Cisco ACI and know the username and password for the authentication process to the APIC device.
– The next step is to convert Excel input into YAML by selecting menu 21 or convert Excel to YAML on the menu display in Figure 11. Ansible uses the contents of the file that has been converted to YAML format to enter the configuration automatically into the Cisco ACI device.
– After the Excel file has been successfully converted into YAML format, then it can be configured automatically to the Cisco ACI/APIC device by selecting the desired configuration menu. For example, if menu 1 is selected, it will configure the system settings. The configuration process can also be carried out simultaneously by entering a "," or "-" sign, for example entering the number "1.13" to run the system settings and domains configuration. As well as entering the numbers "1-18" to run the system setting configuration up to the EPG.

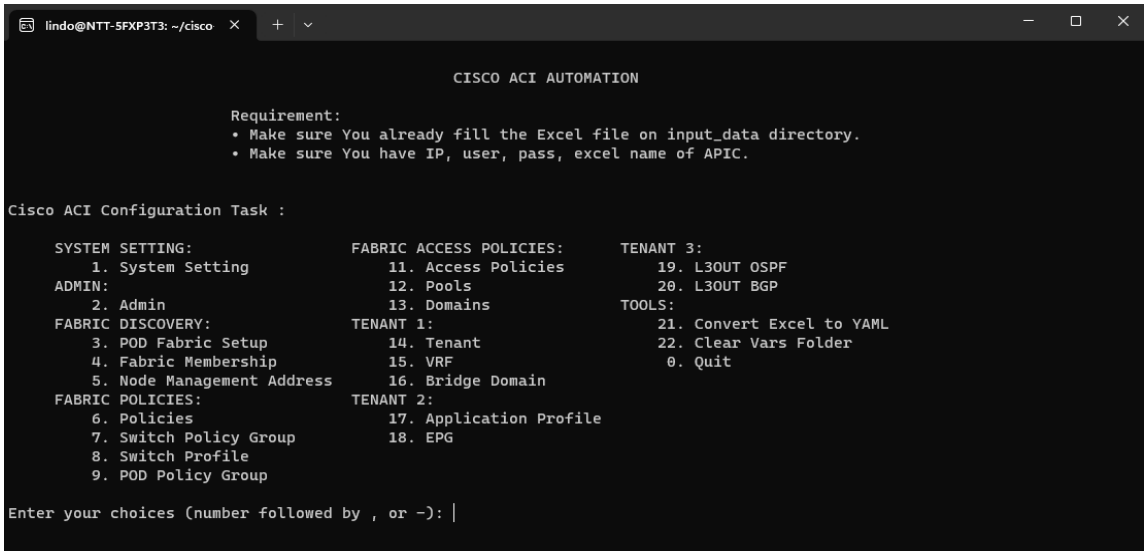

Figure 11. Configuration initial view

– If we have entered the selected menu, then enter information regarding the IP addresses or domain of the Cisco ACI/APIC device, then enter the username and password, and enter the name of the Excel file that was previously prepared in step number 5, without the extension then press enter. The process can be seen in Figure 12. Then the user confirms to enter the configuration by entering "y" to continue and entering "n" to return to the main menu. Then hit enter.

Figure 12. Account information input process

– After confirming the automation tool, configuration will be automatically carried out on the device according to the selected device and will display a recap of information from the configuration process along with the time required to carry out the configuration selected in step 12. An example is the result of all configurations from the system settings shown in Figure 13. The time required to perform all system setting configurations is 0.62 minutes. If there is only one configuration option, after pressing enter, the automation tool will return to the main menu. However, if there is more than one configuration option, after the process of one configuration section is complete, it will continue with the next configuration. Finally, check whether the inserted tenant configurations matched with the keyed in configuration to the Excel file. Tabel 1 shows all configurations available in the automation tool.

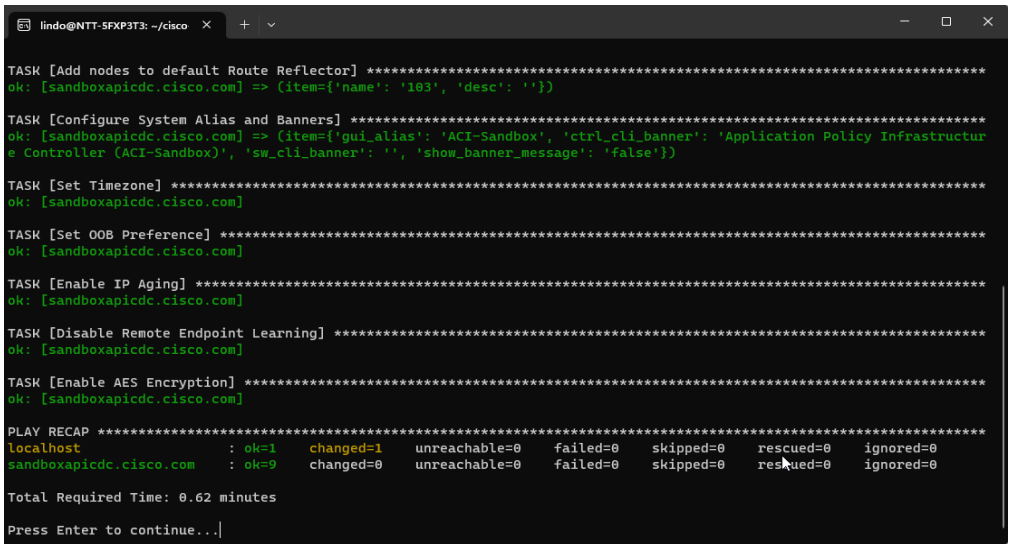

Figure 13. Configuration information

Table 1. Testing results of all configuration available in automation tool

No	Section Name	Configuration Name	Result
1	System Setting	OOB Preference	100 %
		Banner and alias	100 %
		AES Encryption	100 %
		IP Aging	100 %
		Remote EP Learning	100 %
		Time Zone	100 %
		MP BGP	100 %
2	Admin	Security Domain	100 %
		Local User	100 %
		Export Policy	100 %
3	Pod Fabric Setup	Pod and POD TEP Pool	100 %
4	Fabric Membership	Fabric Discovery	100 %
5	Node Management Address	OOB Management IP	100 %
		Site Building	100 %
6	Fabric Policies	NTP Policy	100 %
		SNMP Policy	100 %
		ISIS Policy	100 %
		Power Supply Policy	100 %
		Fabric Node Controls	100 %
7	Switch Policy Group	Switch Policy Group	100 %
8	Switch Profile	Switch Profile	100 %
9	Pod Policy Group	Pod Policy Group	100 %
10	Pod Profile	Pod Profile	100 %
11	Fabric Access Policies	Interface Policies	100 %
		AEP	100 %
		MCP	100 %
12	Pools	VLAN Pools	100 %
13	Domain	Physical Domain	100 %
		L3 Routed Domain	100 %
14	Tenant	Tenant	100 %
		Tenant Policies	100 %
15	VRF	VRF	100 %
16	Bridge Domain	Bridge Domain	100 %
17	Application Profile	Application Profile	100 %

4.2. Discussion

Having done implementing the SDN for auto configuration, then we perform evaluation. This section discusses the performance of the proposed SDN. Firstly, the performance of the automation tool as SDN in term of accuracy is discussed. Secondly, the execution times of the configurations with and without the SDN (manual) are compared. Testing of automation tool is carried out to ensure that the configuration keyed into the Cisco ACI system matches the data on the Excel File. At this stage, testing is carried out on the sample configuration that will be tested on SEC_Tenant_1 by going through the configuration steps as previously described. The test results show that the keyed in tenant configuration is in accordance with the configuration keyed in the Excel file as shown in Figure 14.

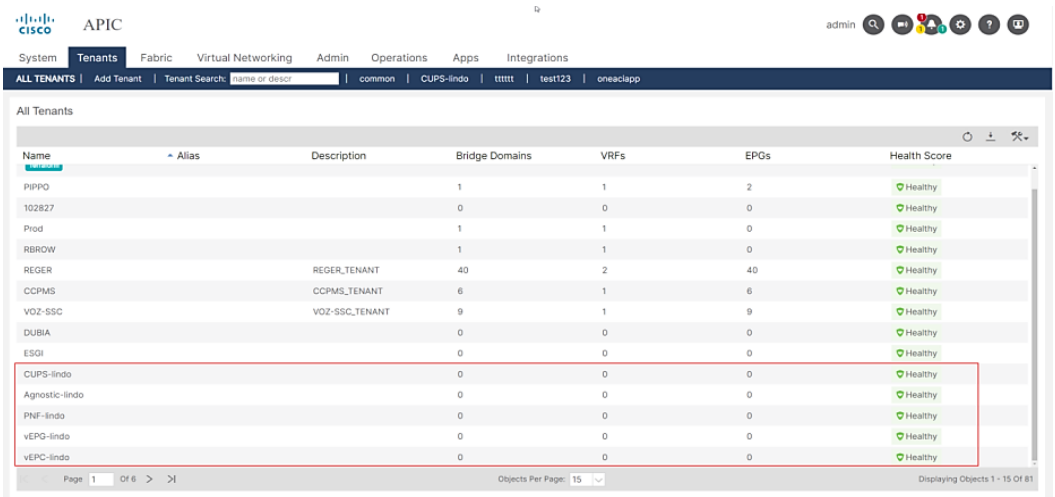

Figure 14. Test results

Next, we measure the configuration time. In this experiment, a comparative test of configuration time was carried out between manual configuration testing and configuration testing using the automation tool. Configuration time testing is measured for a total of 100 configuration rules with different configurations including Tenants, TN_Policies, virtual routing and forwarding (VRF), and bridge domain. In this test, one Excel sheet was used, namely "SEC_Tenant_1", which is the configuration with the most oftenly used by engineers when configuring Cisco ACI and even though using another configuration, the time required will not be much different from the Tenant configuration because it uses the same method. The result of the time comparison between manual configuration and using automation tool is shown in the graph in Figure 15. It shows that the time required for manual configuration is 50 minutes, while the automatic configuration time takes 5.44 minutes or approximately 6 minutes. Thus, the proposed SDN improves the configuration time by 833.33%.

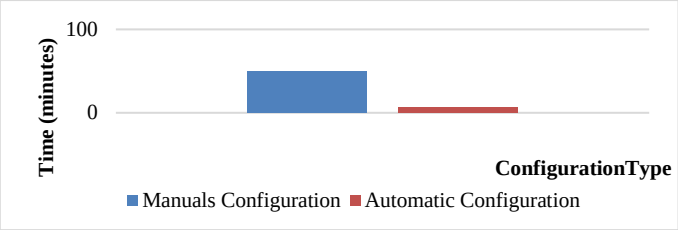

Figure 15. Configuration time comparison

5. CONCLUSION

This research has implemented an automated configuration tool for Cisco ACI as an SDN by combining Ansible and Python scripts in Ansible Playbook. The tool is able to speed up significantly the configuration time and show a high configuration accuracy level. The input to the tool is configuration parameters in the form of Excel template file to produce optimal router configuration simultaneously. Thus, the proposed tool as an SDN assists the network engineer in managing the enterprise networks. Experimental results showed that the proposed SDN achieved a significance improvement in configuring a complex configuration, i.e.: 6 minutes configuration time, compared to50 minutes for manual configuration, which means 833.33% improvement. In addition, the correctness of the configuration achieved 100% for all scenarios. For the future research, it is proposed to expand the automation tool by incorporating more configurations to increase the range of automatable configurations. The automation of input module is also considered as future work. Additionally, there is a plan to develop a user-friendly interface in the form of a website, aiming to facilitate users in utilizing the automation tool more effectively.

ACKNOWLEDGEMENTS

The author would like to express heartfelt gratitude to PT. NTT Indonesia Technology for assistance and support throughout this research.

REFERENCES

[1] S. Badotra and S. N. Panda, "Evaluation and comparison of OpenDayLight and open networking operating system in software-defined networking," *Cluster Computing*, vol. 23, no. 2, pp. 1281–1291, 2020, doi: 10.1007/s10586-019-02996-0.

[2] S. K. Keshari, V. Kansal, and S. Kumar, "A systematic review of quality of services (QoS) in software defined networking (SDN)," *Wireless Personal Communications*, vol. 116, no. 3, pp. 2593–2614, 2021, doi: 10.1007/s11277-020-07812-2.

[3] D. S. Rana, S. A. Dhondiyal, and S. K. Chamoli, "Software defined networking (SDN) challenges, issues and solution," *International Journal of Computer Sciences and Engineering*, vol. 7, no. 1, pp. 884–889, 2019, doi: 10.26438/ijcse/v7i1.884889.

[4] W. Li, W. Meng, Z. Liu, and M. H. Au, "Towards blockchain-based software-defined networking: Security challenges and solutions," *IEICE Transactions on Information and Systems*, vol. E103D, no. 2, pp. 196–203, 2020, doi: 10.1587/transinf.2019INI0002.

[5] K. B. Sowmya and A. Thejaswini, "Systematising troubleshooting of disputes in network," *International Journal of Reconfigurable and Embedded Systems*, vol. 10, no. 1, pp. 32–36, 2021, doi: 10.11591/ijres.v10.i1.pp32-36.

[6] A. Abdulghaffar, A. Mahmoud, M. Abu-Amara, and T. Sheltami, "Modeling and evaluation of software defined networking based 5G core network architecture," *IEEE Access*, vol. 9, pp. 10179–10198, 2021, doi: 10.1109/ACCESS.2021.3049945.

[7] B. Sokappadu, A. Hardin, A. Mungur, and S. Armoogum, "Software defined networks: issues and challenges," *2nd International Conference on Next Generation Computing Applications 2019, NextComp 2019 - Proceedings*, 2019, doi: 10.1109/NEXTCOMP.2019.8883558.

[8] M. Mujib and R. F. Sari, "Performance evaluation of data center network with network micro-segmentation," *ICITEE 2020 - Proceedings of the 12th International Conference on Information Technology and Electrical Engineering*, pp. 27–32, 2020, doi: 10.1109/ICITEE49829.2020.9271749.

[9] S. S. S and S. Praveen, "Automation of CISCO SDWAN Controllers Upgrade Process," *Gradiva Review Journal*, vol. 8, no. 8, pp. 187–191, 2022.
[10] M. F. Mohd Fuzi, K. Abdullah, I. H. Abd Halim, and R. Ruslan, "Network automation using ansible for EIGRP network," *Journal of Computing Research and Innovation*, vol. 6, no. 4, pp. 59–69, 2021, doi: 10.24191/jcrinn.v6i4.237.
[11] S. E. e V. G. J. S. Saini, J. John, J. Fincher, L. J. Cockrell, N. D. Thorve, "Implementing VersaStack with Cisco ACI multi-pod and IBM HyperSwap for high availability," *US: International Business Machines Corporation*, 2018.
[12] A. Baginyan *et al.*, "JINR network infrastructure for megascience projects," *3rd International Science and Technology Conference "Modern Network Technologies 2020", MoNeTeC 2020 - Proceedings*, 2020, doi: 10.1109/MoNeTeC49726.2020.9258004.
[13] B. Santoso and M. W. Sari, "Improvement of setup time on server infrastructure automation using ansible framework," *Journal of Engineering Science and Technology*, vol. 17, no. 5, pp. 3660–3671, 2022.
[14] S. Dalla Palma, D. Di Nucci, and D. A. Tamburri, "AnsibleMetrics: A Python library for measuring Infrastructure-as-code blueprints in Ansible," *SoftwareX*, vol. 12, 2020, doi: 10.1016/j.softx.2020.100633.
[15] V. Shvetcova, O. Borisenko, and M. Polischuk, "Using ansible as part of TOSCA orchestrator," *Proceedings - 2020 Ivannikov Ispras Open Conference, ISPRAS 2020*, pp. 109–114, 2020, doi: 10.1109/ISPRAS51486.2020.00023.
[16] V. M. Ionescu, M. Patel, and D. Hindocha, "Alternatives for running Linux applications in windows," *Proceedings of the 11th International Conference on Electronics, Computers and Artificial Intelligence, ECAI 2019*, 2019, doi: 10.1109/ECAI46879.2019.9042127.
[17] R. Badhwar, "The CISO's next frontier: AI, post-quantum cryptography and advanced security paradigms," *The CISO's Next Frontier: AI, Post-Quantum Cryptography and Advanced Security Paradigms*, pp. 1–387, 2021, doi: 10.1007/978-3-030-75354-2.
[18] P. Kochberger, A. Tauber, and S. Schrittwieser, "Assessment of the transparency of the windows subsystem for Linux (WSL)," *Proceedings - 2019 International Conference on Software Security and Assurance, ICSSA 2019*, pp. 60–69, 2019, doi: 10.1109/ICSSA48308.2019.00015.
[19] M. Kowsher, F. S. Tithi, M. Ashraful Alam, M. N. Huda, M. Md Moheuddin, and M. G. Rosul, "Doly: Bengali chatbot for Bengali education," *1st International Conference on Advances in Science, Engineering and Robotics Technology 2019, ICASERT 2019*, 2019, doi: 10.1109/ICASERT.2019.8934592.
[20] B. Wang, "Programming for qualitative data analysis: towards a YAML Workflow," *ACIS 2022 - Australasian Conference on Information Systems, Proceedings*, 2022.
[21] S. Rasheed, J. Dietrich, and A. Tahir, "Laughter in the wild: A study into DoS vulnerabilities in YAML," *Proceedings - 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering, TrustCom/BigDataSE 2019*, pp. 342–349, 2019, doi: 10.1109/TrustCom/BigDataSE.2019.00053.
[22] A. Ehsan, M. A. M. E. Abuhaliqa, C. Catal, and D. Mishra, "RESTful API testing methodologies: rationale, challenges, and solution directions," *Applied Sciences (Switzerland)*, vol. 12, no. 9, 2022, doi: 10.3390/app12094369.
[23] A. Belkhir, M. Abdellatif, R. Tighilt, N. Moha, Y. G. Gueheneuc, and E. Beaudry, "An observational study on the state of REST API uses in android mobile applications," *Proceedings - 2019 IEEE/ACM 6th International Conference on Mobile Software Engineering and Systems, MOBILESoft 2019*, pp. 66–75, 2019, doi: 10.1109/MOBILESoft.2019.00020.
[24] I. O. Suzanti, N. Fitriani, A. Jauhari, and A. Khozaimi, "REST API implementation on android based monitoring application," *Journal of Physics: Conference Series*, vol. 1569, no. 2, 2020, doi: 10.1088/1742-6596/1569/2/022088.
[25] B. M. Adam, A. Rachmat Anom Besari, and M. M. Bachtiar, "Backend server system design based on REST API for cashless payment system on retail community," *IES 2019 - International Electronics Symposium: The Role of Techno-Intelligence in Creating an Open Energy System Towards Energy Democracy, Proceedings*, pp. 208–213, 2019, doi: 10.1109/ELECSYM.2019.8901668.

BIOGRAPHIES OF AUTHORS

Lindo Prasetyo works at PT NTT Indonesia Technology as a Network Engineer and Automation Engineer. Currently, he is a final year undergraduate student at Department of Informatics, Faculty of Computer Science, Universitas Mercu Buana, Jakarta, Indonesia. His research interests include network automation, SDN, and QoS. To contact him, you can reach him via email: 41519110015@student.mercubuana.ac.id.

Ifan Prihandi is currently a lecturer at Department of Information System, Faculty of Computer Science, Universitas Mercu Buana, Jakarta, Indonesia. He received a Master degree in Computer Science from Universitas Budi Luhur, Jakarta in 2014. His research interests include software engineering, data solution and business intelligence. To contact him, you can reach him via email: ifan.prihandi@mercubuana.ac.id.

Muhammad Rifqi received a bachelor degree in Informatics Engineering in 1999 and a master degree in Informatics Engineering in 2011. He works as a lecturer in Informatics department, Universitas Mercu Buana, Jakarta, Indonesia. 14 years of experience in industry (Panasonic SC Indonesia, KIIC Karawang). His research focuses on computational intelligence, cybersecurity, blockchain and network function virtualization. To contact him, you can reach him via email: m.rifqi@mercubuana.ac.id.

Rahmat Budiarto received B.Sc. degree in Mathematics from Bandung Institute of Technology, Indonesia in 1986, M.Eng. and Dr.Eng. in Computer Science from Nagoya Institute of Technology, Japan in 1995 and 1998, respectively. Currently, he is a full professor at Dept. of Informatics, Universitas Mercu Buana Indonesia. His research interests include intelligent systems, brain modeling, IPv6, network security, Wireless sensor networks, and MANETs. He can be contacted at email: rahmat.budiarto@mercubuana.ac.id.