ASSIGNMENT COVER SHEET

UTS: FEIT

Student name: VISHAL SINGH BISHT

Student number: 12865693

Course name MASTER OF ENGINEERING

Subject name: NETWORK MANAGEMENT

Assignment topic: TECHNICAL REPORT ASSIGNMENT

Date due: 09-JUNE-2020

Date submitted: 09-JUNE-2020

Tutor name: SARAH FARAHMANDIAN
(SUBJECT COORDINATOR- PROF. DOAN HOANG)

(must be correct to ensure your assignment is delivered to the correct marker)

I hereby certify that this assignment is my own work, based on my personal study and/or research and that I have acknowledged all material and sources used in the preparation of this assignment. I also certify that the assignment has not previously been submitted for assessment and that I have not copied in part or whole or otherwise plagiarised the work of other students or authors.

Signature: _VISHAL SINGH BISHT_____

INTRODUCTION:

Software Defined Networking has been a hot topic of research over the last few years. The transition from legacy networks to programmable networks promises several advantages in terms of network operations, network management and network flexibility. The networks in legacy architecture are configured in a low-level and vendor-specific manner. Moreover, the rapid growth of the network has increased the complexities involved in network configurations and introduced additional configuration errors (Kim and Febster, 2013). Furthermore, the large-scale legacy networks have increased complications due to the tight coupling between management, control and data plane, with control and management tasks implemented in hardware (Fetia et al. 2017). Hence, making network management tasks even more difficult. The involvement of heterogeneous network devices such as routers and switches have their proprietary configuration tools, and work according to the specific protocol, thereby increasing both complexity and inefficiency in the network management (Fetia et al. 2017). SDN is an effective solution for above-mentioned as well as several other limitations which were observed in legacy networks. The programmability of networks gives an immense amount of advantages.

SDN ARCHITECTURE:

The SDN architecture involves three main layers: Infrastructure layer, Control layer and Application layer. The SDN devices in the infrastructure layer perform packet switching and forwarding. SDN device consists of API (Application Program Interface) to communicate with the controller, abstraction layer and packet processing component (Hoang 2015). There are two APIs in SDN architecture, namely, northbound API and southbound API. Southbound API is used by the controller to control and configure the parameters in the network infrastructure devices. Northbound API provides an abstract view of the network to the applications. The network forwarding behaviour is monitored by the controller in the control layer using an open interface. The application layer consists of the applications used by the users consuming SDN services. Furthermore, the OpenFlow Switch communicates with the controller using the OpenFlow protocol. OpenFlow protocol is a standardised protocol which defines the communication between the controller and network devices such as routers and switches in the forwarding plane.

LITERATURE SURVEY:

The evolving SDN architectures demand advanced network management techniques. Since the inception of SDN, the researchers have continuously developed new methodologies related to conventional FCAPS (Fault, Configuration, Accounting, Performance, Security) model of network management. SDN controller can perform management tasks, but recently there have been several management issues identified and considered. According to Abdallah et al. (2018), relying solely on the controller to perform management tasks might overload the controller as it is already responsible for controlling network services. It is also important to designate a significantly right amount of management functions to the controller. There has been constant research in addressing the separation of SDN manager and controller to solve the related issues (Kulinski and Chemoil 2014). Abdallah et al. (2018) suggests that the tasks related to

configuration, accounting and performance can be attributed to the manager because of the longer processing and data analyses. In contrast, the controller can take care of the fault and security aspects. A work proposed by Song et al. (2017), recommended delegating some control tasks to switches to ensure the scalability of the SDN network. Delegating controller tasks to switches can be challenging due to the control and data plane split. However, the idea of assigning some tasks to switches can be used with respect to management tasks. The work presented by Abdallah et al. (2018) builds upon this concept stating that the switches can help in communicating with the management applications directly, thus bypassing the controller. Furthermore, assigning the task of data collection and policy administration to switches would reduce the load on the controller. However, the author didn't provide enough evidence and analysis of the performance management aspect of the system. In reference to the performance management, an interesting model was developed by Lin and Chao (2015) using SNMP4J class library. The network performance system proposed by Lin and Chao (2015) was divided into three different modules, namely: Data acquisition module, Data analysis module and Alarm module. The data acquisition and data analysis module primarily focused on retrieving data and analysing it. However, the subdivision of alarm module into performance alarm and fault alarm was an interesting approach. The performance alarm did a real-time data analysis with constant reporting to the management system and providing the device reports whose performance index exceeded the threshold value. The fault alarm, on the other side generated alerts related to equipment failure or any network misbehaviour. Although the system had advantages of safety and reliability, the hierarchical design and processes in the system increased the complexity in the network management model. Moreover, the fault management system was verified only using generic techniques such as performing close port commands.

Modern SDN approaches still addresses the reliability concerns, thus demanding improved fault management techniques and solutions. There have been significant solutions and approaches proposed by researchers for identifying/detecting, locating, solving and avoiding the faults in the network. A failure-free operation in a given time under specified conditions refers to the reliability, and it is maintained using advanced and suitable Fault management techniques. Bu k. et al. (2016), proposed a RuleScope solution, for the detection of missing faults and priority faults in SDN forwarding, by using customised probe packets in complying with the data-plane rules. The algorithm helped in reliable network monitoring and tracked the real-time forwarding status. However, the accuracy of the algorithm was limited to the flow table size of 160 with a further significant decrease in efficiency and accuracy. Moreover, the author didn't take varied inconsistencies related to SDN architecture into consideration. The inconsistent variations between the control plane and data plane may result in packet misbehaviour, thus violating the original policies and rules in the flow tables of the data plane. To alleviate the discrepancies in the packet behaviour, Lei k. et al. (2018) proposed a framework for measuring the control-plane consistency, which meant complying with the defined relationship between the control-plane policies and data plane rules. The framework consisted of three subsystems, namely:

- **Packet Processing Subsystem** for adding tag value in the packet
- **Verification Subsystem** for computing the tags with the information of switches and affirming that the travelled path was consistent

- **Localisation Subsystem** for locating the fault in the switch when the verification process fails

Although the fault localisation in work proposed by Lei k et al. (2018) showed 100% accuracy in a fat-free topology, it didn't confirm the same results with different topologies such as Jellyfish topology. Furthermore, the framework focused primarily on verification and localisation by exploiting the path information in the tag values of the packet, thus increasing the risk of attacks in the whole network architecture. Moreover, the author fails to identify any mitigation techniques to encrypt the path information in the tag values. Another interesting work presented by Perisini et al. (2018), proposed a system- Monocle, which addressed the forwarding issues caused due to hardware or software failures. The suggested system injects its monitoring packet into a switch to examine the switch behaviour. The proposed system by Perisin et al. (2018), was designed for the purpose to monitor, detect and locate the fault, however, it didn't cover all the aspects of Fault management such as solving and avoiding the similar kind of faults in future. Moreover, the system was placed as a proxy layer between controller and router/switches, without addressing any security issues which in future may compromise the whole network infrastructure.

The decoupling of control plane and data plane abstracts the network infrastructure from applications thus enhancing the network security by providing the full view of network state to the logically centralised control plane (Ahmad et al. 2015). However, the separation of control plane and data plane opens up the several possibilities of network attacks such as a man-in-the-middle attack, Denial of Service (DoS) attack, distributed Denial of Service (DDoS) and saturation attacks. The past kinds of literature have raised several security issues in SDN. The availability of SDN rootkits has allowed attackers to gain access to the controllers and thus compromise the whole network. Porras et al. (2015) presented an attack that allowed the attackers to get away from the existing flow rules in the network. Tatang et al. (2017) proposed a novel system (SDN-Guard) for the detection and mitigation of SDN rootkits. The system consisted of two main components, namely: proxy unit and decision unit. The proxy unit acted as a reverse proxy, thus extracting the flow_mod messages for the review of their behaviour. Further, the decision unit was used for comparing the network view with the controller view to detect any malicious changes. However, the model suggested by Tatang et al. (2017), was only limited to the detection of hidden flows as long as the rootkits were unable to detect SDN-Guard calls. In the current situation of security threats where attackers have gained advanced skills, there is a high possibility that the rootkits can be developed for intercepting the proposed system (SDN-Guard) calls and bypass the security architecture. Moreover, the author fails to identify the possibility of rootkits compromising the operating system; thus, as a result taking down the controller process as well as the underlying network.

One solution to detect the hidden flows can be to use monitoring sensors across the network and to verify the behaviour of the flows. The dynamic nature of SDN gives the flexibility to use cheap monitoring sensors across the network to listen to the traffic and detect any suspicious behaviour (Adam et al. 2018). However, these sensors are not reliable enough to mitigate any network failures or behaviours. Adam et al. (2018), proposed an adaptive model combining network monitoring and security visualisation to effectively provide information to the system administrator in case of any intrusion. The idea of using two aspects into one is undoubtedly useful, however, Adam et al. (2018), only provided a very brief demonstration

strategy against MitM attacks using MitM detector. The author did not provide strong solution or architecture against several vulnerabilities found in SDN architecture. Monitoring and security visualisations are no doubt a good solution to view the network behaviour, but there is a need to implement a technique for secure communication within the network practically. TLS is the recommended implementation for secure communication in SDN. However, TLS also has several vulnerabilities, and it is imperative to scale up the security level of communication in SDN. Midha S. and Tripathi (2019), proposed an extended secure algorithm for TLS where the client message was sent along with the 16-bit sequence, which was a result of the XOR operation between a 16-bit public key and 16-bit random sequence. The suggested TLS extended algorithm made the channel communication stronger with strict authentication. Learning the behaviour of the attacker and gaining access to malicious data is one of the techniques that can be used to strategise network security. One of the most common techniques suggested in the past literature is the deployment of the honeypot system. Honeypots systems are used as intentionally deployed systems as a decoy for attackers to let them exploit the system by exposing some set of vulnerabilities and in return gain the attacker's information (Han et al. 2016). Further, the utilisation of multiple honeypots creates a network architecture which is known as Honeynet. However, several issues, such as inefficient data capturing capability and vulnerabilities to fingerprinting attacks, were addressed in Honeynet. Kyung et al. (2017), proposed the next generation Honeynet architecture with improved design and mitigation of limitations that existed in the honeypot systems. The main idea was to operate Honeynet as one large honeypot and run several vulnerable services which would be the union of every individual honeypot. Hence, allowing attackers to have more interaction with honeypots and in return helping network administrators to gain more information about the attackers and their malicious behaviour. Furthermore, the adaptive technique to prevent fingerprinting attacks by using proxy module eliminated the potential drawback of deploying honeypot in a network architecture.

PROPOSED MODEL:

Physically distributed SDN architectures have gained an immense amount of attention in the past few years and are known to mitigate problems concerned with the conventional centralised architectures. Fetia et al. (2017), presented a detailed survey and a comparative study on centralised and distributed architectures in SDN. Hierarchical SDN control is a promising approach to improve network scalability and performance by vertically partitioning the control plane into multiple levels depending on the requirements. Yeganeh and Ganjali (2012), proposed a framework: Kandoo, suggesting two-layer control structure partitioning control applications into local and global. The framework reduced the overall load on control plane by establishing a two-level hierarchy where local controllers handled the events occurring near the data path and non-local controllers monitoring local controllers and global network.

As a network manager, it is imperative to strategise and propose the best-suited methodologies to maintain overall network and adhere to the FCAPS model of network management. After a careful review of the above-mentioned pieces of literature, this work proposes a three-level hierarchical SDN control framework, with secure communication and deployment of honeypot controller.

The framework includes four physically distributed controllers: Local Controller, Honeypot (Honeyproxy) controller, Main controller and Clone controller.

The first controller, which is a local controller consists of two alarm modules: performance alarm and fault alarm, similar to the system proposed by Lin and Chao (2015). The local controller will process data acquisition and real-time data analysis and generate an alarm if the performance index exceeds a threshold value. On the other side, the fault alarm would be responsible for providing information related to equipment failure or network misbehaviour. Moreover, for fault localisation, the packet will be injected with a tag value and, tag computation will be processed to verify the information of switches and further confirm the travel path. The fault location will be detected if the tag values don't verify. The tag value computation is dedicated to the local controller.

The second controller which is a honeypot controller called HoneyProxy Kyung et al. (2017), will be a decoy for any attacks, and this controller will be providing the important information about any malicious activity. The three modes: Transparent mode(T-mode), Multicast mode(M-Mode) and Relay mode(R-Mode) as explained in Kyung et al. (2017) can be taken into consideration for HoneyProxy. To monitor this controller and gain important information about the attacker behaviour and details, the first controller, will have a security visualisation model similar to the one explained in Adam et al. (2018).

Furthermore, the third controller, which is the main controller, will be responsible for monitoring the local controller and configuring all the network devices in the network architecture. The reason for dedicating the task of configuration to the third controller is simply to avoid single point failure and any compromise in the configuration. So, even if the local controller or honeypot controller is compromised, the main controller can detect any intrusion or fault and take an adaptive approach towards changing the network configurations. Further, the main controller will be responsible for overall network security functions, and the communication between the local controller and the main controller would be using strong AES256 encryption. The main controller will be designed with advanced machine learning algorithms to independently change the overall of network configurations in case of any fault or intrusion events. Furthermore, for the high availability, the main controller will be connected to the clone controller, which will have exact same configurations as main controller and will only come to picture in the event of failures in main controller. The advanced TLS algorithms can be used for secure communication similar to the algorithm presented by Midha and Tripathi (2019), to create a strong authentication channel within the network. The idea is to make a highly adaptive SDN system with the network manager involved in the end-to-end loop of entire network architecture with a secure and sophisticated technique.

CONCLUSION:

The proposed network management framework promises advanced scalability, reliability, performance and security. The designation of performance, fault and accounting management to local controller and configuration and security management with overall monitoring capabilities to the main controller, make it highly adaptive and promises high-performance results. Moreover, the deployment of honeypot controller is a smart strategy to decoy the attackers in case of any intrusion and thus retrieving important information and passing it to

the local and main controller to take necessary steps for the network management. Finally, taking high availability into consideration a clone controller is used in the network in case of main controller failures.

REFERENCES:

Abdallah, S., Elhajj.H.I, Chehab, A. & Kayssi, A. 2018, "A Network Management Framework for SDN," *9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Paris, pp. 1-4

Adam, I., Ahola, T., Sailio, M., Vallivaara, V. & Eye, V. F. 2016, "Adaptive monitoring and management of security events with SDN", *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, Istanbul, pp. 817-820

Ahmad, I., Namal, S., Ylianttila, M. & Gurtov, A. 2015, "Security in Software Defined Networks: A Survey," in *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2317-2346, Fourthquarter

Bu, K., Wen, X., Yang, B., Chen, Y., Li, E. L. & Chen. X. 2016, "Is every flow on the right track?: Inspect SDN forwarding with RuleScope," *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, San Francisco, CA, pp. 1-9

Fetia, B., Sami, S. & Abdelhamid, M. 2017, "Distributed SDN Control: Survey, Taxonomy and Challenges", IEEE Communications Surveys & Tutorials. PP. 1-1

Han, W., Zhao, Z., Doupé, A. & Ahn, J-G. 2016, "Honeymix: Toward sdn-based intelligent honeynet", *Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, pp. 1-6

Hoang, D. (2015). Software defined networking - shaping up for the next disruptive step? *Australian Journal of Telecommunications and the Digital Economy*, *3*(4), 48–62.

Kim, H. & Feamster, N. 2013, "Improving network management with software defined networking.", IEEE Communications Magazine, vol. 51, no. 2, pp. 114–119

Kyung, S., Han. W., Tiwari, N., Dixit.H.V., Srinivas, L., Zhao, Z., Doupé, A. & Ahn, J-G. 2017, "HoneyProxy: Design and implementation of next-generation honeynet via SDN," *2017 IEEE Conference on Communications and Network Security (CNS)*, Las Vegas, NV, pp. 1-9

Lei, K., Li, K., Huang, J., Li, W., Xing, J. & Wang, Y. 2018, "Measuring the Control-Data Plane Consistency in Software Defined Networking," *2018 IEEE International Conference on Communications (ICC)*, Kansas City, MO, pp. 1-7

Slawomir, K. & Prosper, C. 2014, "Network Management Challenges in Software-Defined Networks". IEICE Transactions on Communications

Midha, S. & Tripathi, K. 2019, "Extended TLS security and Defensive Algorithm in OpenFlow SDN," *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, India, pp. 141-146

Tatang, D., Quinkert, F., Frank, J., Röpke, C. & Holz, T. 2017, "SDN-Guard: Protecting SDN controllers against SDN rootkits," *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Berlin, pp. 297-302

Lin, T. & Chao, W. 2015, "The research on network performance management system based on SDN technology," *2015 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, Ningbo, pp. 1-5

Perešíni, P., Kuźniar, M. & Kostić, D. 2018, "Dynamic, Fine-Grained Data Plane Monitoring with Monocle," in *IEEE/ACM Transactions on Networking*, vol. 26, no. 1, pp. 534-547

Porras, P., Cheung, S., Fong, M., Skinner, K. &Yegneswaran, V. 2015, "Securing the Software-Defined Network Control Layer", *Symposium on Network and Distributed System Security*

Ping, S., Yi, L., Tianxiao, L. & Depei, Q. 2017, "Controller-proxy: Scaling network management for large-scale SDN networks. *Computer Communications"*, *108*, 52–63

Yeganeh, H.S. & Ganjali, Y. 2012, "Kandoo: A frame- work for efficient and scalable offloading of control applications," in Proceedings of the First Workshop on Hot Topics in Software Defined Networks, ser. HotSDN '12. New York, NY, USA: ACM, pp. 19–24