Vulnerabilities in SDN Due to Separation of Data and Control Planes

S. Faizullah
Dept. of Computer Science
Islamic University of Madinah

S. AlMutairi
IT Deanship
University of Tabuk

ABSTRACT
Tremendous advancements over the past several decades revolutionized the networking research and technological industry, however, it is still dominated and remains hardware based. Such legacy networks are inflexible, hard and costly to scale and manage. Software defined networking (SDN) is a new approach to networking which enable comprehensive network programmability. SDN architecture bifurcates the data and control plane thereby simplifies network management. In this new architecture, the control plane consists of networking intelligence and the policy making ability is moved to a centralized entity called as controller. Commonly, SDN uses OpenFlow as the communication interface between the data and control planes. This separation while providing great opportunities for scalability, also introduces new vulnerabilities. We identify certain scenarios for vulnerabilities in the OpenFlow semantics that can subject the controller to distributed denial of service (DDoS) attack which is unique to SDN due to the new architecture where the control plane is separated from the data plane. We also explore some reactive mechanisms that can detect and help to devise techniques to prevent impending DDoS attack on an SDN controller.

General Terms
Computer Networks; Software Defined Networking; SDN; SDN Vulnerabilities

Keywords
Software Defined Networking; SDN; SDN Vulnerabilities; DDoS; Cloud Computing; OpenFlow.

1. INTRODUCTION
Our world view of traditional computer network is based on the network composed of routers, switches, and computing devices, interconnected by diverse and often times complex protocols. The management of such network is error prone, hard to scale, difficult to manage routing tables, and is not able to catch the pace with today's growing demand from Internet where in the era of Cloud Computing, the scalability is a defining factor in the face of vast changes that the network goes through in any split millisecond. Another aspect is the cost of scalability, which albeit timid, and even if desired to achieve with premium hardware, the cost will often time be prohibitively high. As a result, programmable networks concept has emerged as the solution to the problems of legacy network architecture. SDN [1, 2, 3, 5] is dynamic, manageable, cost-effective and adaptable, by allowing programmability, making it ideal for dynamic networking world. This is achieved by bifurcating the network so that we decouple functionality of traffic forwarding decisions, control plane, from the functionality of forwarding the traffic to the destinations, data plane, and hence provide the needed flexibility. In these new network architectures, the network intelligence is transferred to logically centralized programmable entity called controller and the rest of the network devices are doing the basic job of packet forwarding. This environment opens new era of possibilities and cost savings but also is prone to security risks and vulnerabilities that can be exploited for attacks. As such new challenges need to be identifies and filled by robust research and technical efforts. In this paper we identify one such area of concern due to the separation of the data and control planes. We show that utilizing some characteristics of the SDN, we can devise attacks that outpaces network provisioning efforts to cope with increase for traffic demand and processing and hence results in denial of service.

The rest of this paper is organized as follows: Section 2 describes SDN architecture. Section 3 describes OpenFlow [6, 7, 8, 10]. Section 4 describes vulnerability in SDN that can lead to DDoS attack on controller [4, 9, 11]. Lastly, some simulation results are given in Section 5.

2. SDN Architecture
We represent the basic Software-Defined Networking (SDN) architecture in Figure 1. From top down, we have applications that uses the two layers of control plane and data plane for communications. Note that the switches in SDN don't construct the forwarding table automatically, but instead relies on controller to construct a flow table [1, 9]. Control plane of the SDN architecture consists of centralized controller which is the most important new feature provided by SDN. This facilitates easier network management and configuration information is stored in simple place instead of distributing it to individual network devices. Additionally, the controller has a centralized view of the network which makes it capable to calculate optimal routes across the network, construct flow tables and insert them into each of the switches without relying on distributed routing algorithms. Lastly, the application layer in SDN allows applications to request specific network behavior from the controller.

As SDN is taking share and several protocols are being studies, OpenFlow is emerging as one of the most used protocols in SDN is OpenFlow. OpenFlow is defined as open standard and as such not tied to any single controller making it suitable for heterogeneous networks. It is preferred protocol that is being utilized between the controller and SDN switches. It transmits messages from controller to the switch to facilitate the flow tables' constructions/updates. In addition, it transmits messages from the SDN switch back to controller so that switches can inform the controller about the various events in the network such as new host or switch additions and other similar functions.

3. OPENFLOW
The OpenFlow protocol that is defined in [9] is an efficient means of communication between the controller and SDN switches. OpenFlow protocol supports three types of messages: controller-to-switch, asynchronous and symmetric

messages. Controller-to-switch messages are initiated by the controller while asynchronous messages are messages sent by the switch without being requested by the controller- these denote packets arrival, switch state changes, error states, etc. Lastly, symmetric messages are the messages sent without solicitation, in either direction. In this new architecture, every SDN switch consists of a flow table which in turn stores a set of flow entries. Whenever a new packet is processed by the switch, it is compared against the flow table to find a matching entry.

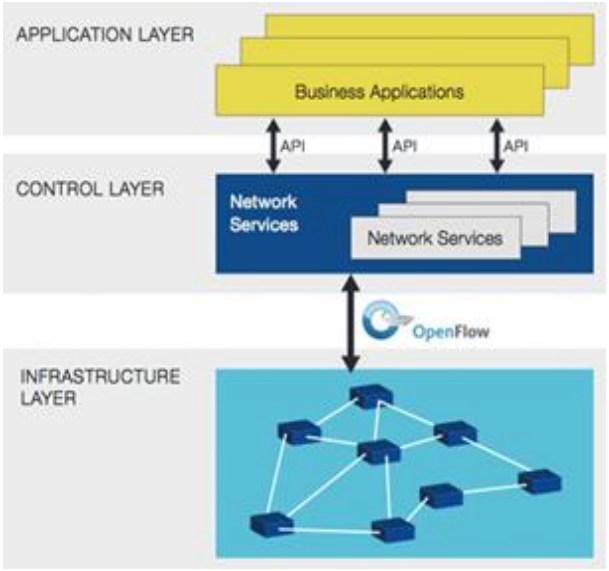

Fig. 1: SDN architecture

Once a matching entry is located, actions specified for that entry are performed on the packet e.g. forward a packet out a specified port. If no matching entry is found, the packet is forwarded to controller which can decide the action to be taken on the packet. We depict different components of a flow table entry in Figure 2. Each flow table entry consists of rule or header field to match against the incoming packet. Please note that every flow entry is associated with zero or more actions that dictate how the switch handles matching packets. If no actions are specified for an entry, then such a packet must be dropped. Flow entry also contains counters that are maintained per -table, per-flow, per-port and per-queue.

4. DDoS ATTACK ON SDN CONTROLLER AND RESULTS

OpenFlow semantics dictates that an incoming packet is handled according to a matching flow entry looked up in the flow table, if such a matching exists. On the other hand, if there is no matching flow entry found, then that packet must be forwarded to the controller for further processing. This is a major vulnerability even if the controller is physically distributed among many computing devices and hence this vulnerability can be exploited by attackers who can out do the computing power of the controller by increasing the attack. The attackers can send out a very large number of packets at once, or in distributed bursts, where no matching entries exist in the flow table which will force the controller to intervene. Once forced to intervene by forcing massive number of packets, it can be overwhelmed- keeping it busy to serve the attackers packets while resulting in denial of the service to other users. The architecture as it is can encourage distributed denial-of-service attacks on the SDN controller.

5. SIMULATION RESULTS

Below are several scenarios in which the switches will be forced under which the switch is forced to forward the packet to the controller. In a real DDoS attack, the attacker sends such packets in enormously large numbers at once hence a potential DDoS attack on the controller. Some scenarios are as follows:

1) Bursts of packages: attackers send continuous bursts of packages faking real packets but with spoofed or

2) to unknown destinations which will trigger table-misses resulting in traffic to controller.

3) New host addition: as a host is added to network, no flow entry is added to flow table until first packet is sent to the host which per this architecture makes the initial packets forwarded to the controller.

4) Per the architecture as a host is removed from the network flow entry then this host is also removed from the flow table. As no new flow entry will be added in flow table for this host unless a packet is sent to the host after it has been added again which as noted above will direct the first packet sent to the host after it is inserted again to the network will be forwarded to controller.

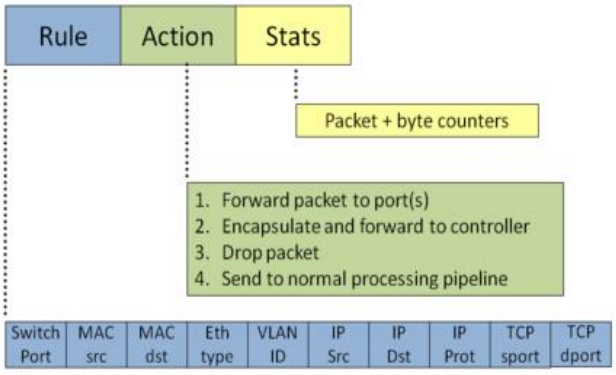

Fig. 2: Flow table entry

5) As packets are sent to a non-existing hosts, these will be forwarded to the controller since obviously no flow entries exists for such hosts. This can be easily exploited.

6) In SDN architecture each switch flow expiry mechanism is run by that switch independently of the controller. There is an idle_timeout and an associated hard_timeout for each flow entry. In case an hard_timeout field is non-zero, the switch must note the flow entry's arrival time, as it may need to remove the entry later and such entries cause the flow entry to be removed after the given number of seconds, regardless of how many packets it matches. Additionally, in case the idle_timeout field is non-zero, the switch must note the arrival time of the last packet associated with the flow, as it may remove the entry later as a non-zero idle_timeout field causes the flow entry to be removed when it has matched no packets in the given number of seconds. Hence, the switches must implement flow expiry mechanisms and remove flow entries from the flow table when one of their timeouts is exceeded. This creates vulnerability as

these values are few seconds which enable attackers to utilize t devise their DDoS attacks.

6. SIMULATION RESULTS

We have done some preliminary implementations of the controller in the lab where, utilizing the cloud computing environment. We have implemented the combinations of the above scenarios in the environments where we also dynamically increase the computing power of the control underlying hardware, taking advantage of the elasticity of the cloud computing paradigm, mimicking a provider's management functions (scalability) to add more power as it see loads and taking it for increase in utilization . To conduct the simulation studies, we will be using randomly generated networks of varying complexities and sizes. This ensures that the simulation results are independent of the characteristics of any particular network topology. We noticed that we can overwhelm the controller with these scenarios.

The results presented in his paper are based on averages for 10-20 iterations of same network setup (topology, traffic, groups, etc.) with varying degree of the four different scenarios representing DDoS, see Table 1. The results were logged in intervals where the end points kept increase sending packets in each interval. As the attack increases, more CPUs are provisioned and we could increase the traffic with the above scenarios that overwhelmed and surpassed the increase the computing power being continually added- attack can exceed the throughput in any given time with increase in the velocity of the flooding with fake packets. This is shown in the last row of the Table 1. This can also happen in the real world, where the additional CPU power that the controller can acquire can be overwhelmed by the increase in attack utilizing similar vulnerabilities that are depicted by several scenarios in Section IV, hence the attack can exceed the throughput and result in denial of service to legitimate users. In near future, we will be conducting more experimentations and also utilize a CPU/GPU based lab as well as a GPU based lab to increase the controller's computing power even more.

Table 1: Simulation Parameters/Settings and Results

Simulation Parameters	Settings
# Network Nodes	100
Environment - CPUs	8-120
Attack Scenarios	#1/#2, #1/#2/#3, #2/#3/#4/#5, #1-#5
Packet Servicing Time	0.09s-500s (normal-denial)

We would like to conduct the attack scenarios and add other scenarios to see the impact on network operations. The results are, however, very clear that we do have potential for vulnerabilities with this centralized approach to controller even if physically it is distributed and beefed up.

7. ACKNOWLEDGMENTS

This work was done while the author was a V. Research Professor at Rutgers University Thanks to my Cloud Computing Graduate Class students for their contributions in the initial phase of the work.

8. REFERENCES

[1] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka and T. Turletti, "A survey of software-defined networking: Past, present, future of programmable networks", *IEEE Commun. Surv. Tut.*, vol. 16, no. 3, pp. 1617-1634, 2014

[2] N. Feamster, J. Rexford and E. Zegura, "The road to SDN", *Queue*, vol. 11, no. 12, pp. 20:20-20:40, 2013

[3] K. Ahokas, "Software-defined networking", Aalto University School of Science.

[4] S. Shin and G. Gu, "Attacking software-defined networks: A first feasibility study (short paper)" , In HotSDN'13.

[5] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with OpenSketch". In Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI'13). April (2013).

[6] http://archive.openflow.org/documents/openflow-spec-v1.0.0.pdf

[7] A. Doria, J. Hadi Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern, "*Forwarding and control element separation" (ForCES) protocol specification*, RFC 5810 (Proposed Standard), March 2010,

[8] Devolved Control of ATM Networks. http://www.cl.cam.ac.uk/research/srg/netos/old-projects/dcan/#pub.

[9] H. Wang, L. Xu, and G. Guofei, "Of-Guard: A DoS Attack Prevention Extension in Software-Defined Networks', In USENIX Open Network Summit, 2014.

[10] T. Limoncelli, "Openflow: a radical new idea in networking",. Commun. ACM, 55(8):42–47, August 2012.

[11] K. Benton, L. J. Camp, and C. Small. OpenFlow Vulnerability Assessment. HotSDN '13, pages 151--152, 2013.

9. AUTHOR'S PROFILE

Safi Faizullah received his Ph.D. in Computer Science from Rutgers University, New Brunswick, New Jersey, USA in 2002. He also received MS and M. Phil. Degrees in Computer Science from Rutgers University, New Brunswick, New Jersey, USA in 2000 and 2001, respectively. Dr. Faizullah also earned his BS and MS degrees in Information and Computer Science from KFUPM, Dhahran, KSA in 1991 and 1994, respectively. His research interests are in computer networks, mobile computing, wireless networks, distributed and enterprise systems. He has authored over twenty refereed journals and conference papers. Dr. Faizullah works for Hewlett-Packard and he is a Visiting Research Professor of Computer Science at Rutgers University. He is currently Professor with Dept. of Computer Science, IUM. He is a Senior member of IEEE, SCIEI, PMI and ACM.

Dr. Saad Al-mutairi (BSc, MSc, Ph.D) has completed his Ph.D & master in De Montfort University, UK. Currently, he is working as a Dean of Information Technology Deanship and also an Associate Professor in Computer and Information

Technology Faculty at Tabuk University, Saudi Arabia. His research interests are software engineering & Developments, Cloud computing, security requirement engineering for context-aware systems using the Uniform Modeling Language (UML). In connection with his research, he has published a number of papers in various international journals and conferences. He is a good team leader and he has established many software's to support regular activities of university.