Practical Labs for Teaching SDN Security

Souvik Das
The University of Texas at Dallas
USA
souvikdas95@utdallas.edu
0000-0002-1744-8309

Kamil Sarac
The University of Texas at Dallas
USA
ksarac@utdallas.edu
0000-0001-7018-5256

***Abstract*—The rapid adoption of Software Defined Networking (SDN) in the industry has exposed certain security risks today some of which are unique to its paradigm. Security issues around the use-cases that expose these risks are fundamentally aligned with the networking and cybersecurity concepts that are taught at the graduate level in academia. In this paper, we present a number of lab activities on SDN security that are inspired from practical use-cases in SDN deployments. The goal of this effort is to help students give a shape to their thought process about the practical security implications of SDN deployments and gain valuable practical domain knowledge in securing an environment with such deployments.**

Keywords—software-defined networking, network security, cybersecurity lab, security education, practical labs, teaching

I. INTRODUCTION

SDN provides certain advantages including effective network management and ability to introduce new and customized network services in a timely manner. However it also exposes a range of network and application security issues such as denial-of-service and unauthorized access. The concern for some of these issues is novel to its paradigm and requires adopting advanced strategies such as moving target defense and closed loop automation. While the domain of solutions to these issues keeps expanding with the increase in the variety of SDN applications, the concern however remains fundamentally aligned with the networking and cybersecurity concepts that are taught at the graduate level in academia. This motivates us to introduce the concept of SDN security as a sequence of lab exercises to students who aspire to revolutionize the industry. Therefore, in this paper, we present a set of eight hands-on lab activities to introduce students to various practical security issues in SDN deployment environments.

SDN is often taught as a graduate course in academia and while most of these courses already include related labs, we believe that addressing certain qualities [1] could help spark a discussion on its practical security aspects. For instance, in Lab 4, we have addressed a use-case that requires outsourcing certain network policies using SDN. By imposing minimal constraint on and generalizing the scope of sources and communication media of these policies, we have tried to introduce the qualities of portability and flexibility in our lab. A practical solution to this use-case would require addressing the possibility of these entities operating asynchronously which raises concern for potential policy conflicts posing as a security risk to the network. Our lab has been designed to address such use-cases and the issues around them to reflect the needs of practical applications in the industry.

Inspired by some of the inductive learning strategies [2], we have structured the lab activities to introduce a kind of competitive demeanor in students that piques their curiosity and encourages them to explore the solution space and potentially come up with their own solution and implement it. Section II describes the environment that we propose for students to stage the lab activities. Section III covers the range of activities in our proposed lab. Section IV describes related work in this domain. Finally, Section V concludes the paper.

II. LAB ENVIRONMENT

Our lab environment consists of several software tools that we have bundled in a Virtual Machine (VM) (Fig. 1) configured to run on VirtualBox. These tools mainly include 1) OpenvSwitch, a virtual-switch software that helps in emulation of Openflow-enabled SDN switches; 2) Open Network Operating System (ONOS), a java-based controller platform for deploying SDN applications; 3) Mininet, a software that orchestrates SDN controllers and switches to emulate an SDN operated network; and 4) Docker, a platform for containerizing applications such as SDN controllers in our lab. Other tools include an IDE for developing SDN applications and networking tools such as hping, iperf, ssh, tc, ip, arp and ethtool available as command-line utilities in Linux.

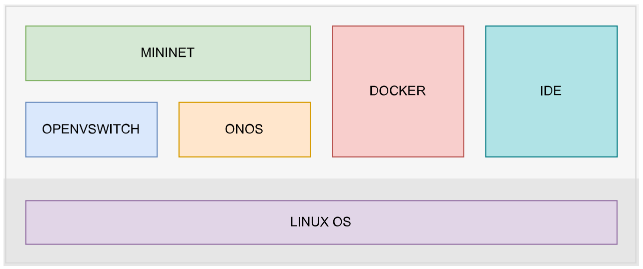


Fig. 1. VM Composition

We recommend students to use our VM (available at our project website personal.utdallas.edu/~ksarac/sdn-security/ as their workspace for end-to-end execution of lab activities.

III. PROPOSED LAB ACTIVITIES

Each activity in the proposed lab targets one or more of the cybersecurity problems including but not limited to denial-of-service, unauthorized access, configuration issues, man-in-the-middle, etc. The manual for each activity is provided with technical hints and reference materials to help students practically apply the concepts of SDN to develop solutions to these problems. These include comprehensive tutorials to help students explore features and capabilities of the SDN controller (i.e., ONOS) that are relevant to these activities. The manuals also include guidelines for students to debug and test their approach as they make progress towards perfecting their solutions. Instructors are separately provided with a reference manual that contains a demonstration of an approach to implement the solutions in each activity.

Each activity also includes a “hackathon” section in which students are asked to draw suitable inference(s) from a given variation to the problem scenario and potentially design a solution of their own using suitable heuristics or in consultation with the published solutions suggested in relevant academic papers. The goal of this section is to help build the level of competency in graduate students that is required to practically address the many variations to these problems which they would inevitably face as future industry professionals. We intend to introduce this section in the curriculum by motivating students to attempt it for bonus points.

In the rest of this section, we first present a couple of warm-up activities (Section III-A) that we propose to help students get familiar with the various aspects of SDN technologies and then present the activities in our proposed lab (Section III-B).

A. Getting Started

As a precursor to the proposed lab activities, we provide two warm-up activities to illustrate how the concepts of SDN can be applied in the practical sense before moving forward to address its complex security issues.

In the first activity, students build a MAC tracker which captures the MAC address of network hosts and then publishes them using the controller’s southbound and northbound interfaces respectively. This activity gives an example of end-to-end flow of information in an SDN operated network.

In the second one, students implement reactive forwarding where they adaptively set up flow-paths across switches to establish communication between network hosts when required. Unlike proactive forwarding, reactive forwarding assumes a dynamic state of the network and helps reduce unnecessary waste of switch memory. This activity gives an example of practical deployment of flow-paths in SDN.

B. Lab Activities

In this section, we present the set of developed lab activities pertaining to SDN security in some detail as given below:

Lab 1: Unauthorized Access by Compromised Controllers

Problem Definition: Network control is often distributed across multiple SDN controllers for scalability and resiliency purposes. This requires coordination between controllers that control different parts of a network for synchronous execution of network policies. Lack of proper coordination could be exploited by an adversary in the form of a man-in-the-middle (MITM) attack contemplated by a set of compromised controllers and network hosts to hijack parts of a network.

Lab Objectives: Students are instructed to 1) deploy a multi-controller SDN setup on a given topology, 2) develop a controller module to segregate the control of switches across different controllers (Fig. 2), 3) structure a set of network policies given in the form of flow-paths across them using an appropriate configuration markup, and 4) import them at runtime in each controller for installation.

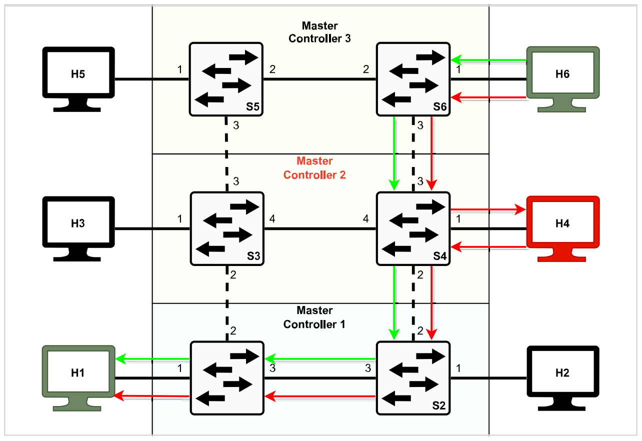


Fig. 2. Multi-controller SDN setup with (red) and without (green) MITM attack

Lab Challenges: Students need to 1) learn to apply ONOS-Atomix integration (*atomix.io*) (Fig. 3) to set up the given multi-controller network, 2) learn to use the mastership services of the controller in their modules to implement the proposed segregation of network control and 3) ensure that each controller complies with it when installing the network policies.

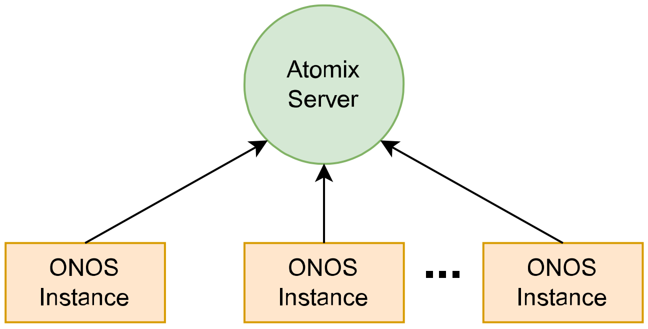


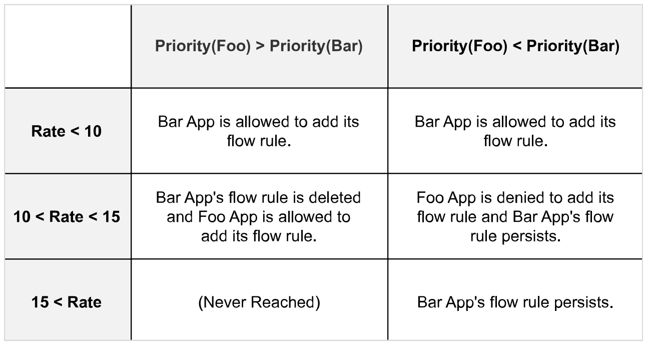
Fig. 3. ONOS-Atomix Integration

Hackathon: Students are instructed to 1) emulate a malicious network behaviour with a single compromised controller and network host (Fig. 2) and 2) inspired by Byzantine Generals Problem, develop a module to configure each of the controllers to publish an integrity heuristic on their northbound (REST) interface that may be inferred upon to identify the compromised controller.

Learning Outcome: Students learn about the functioning of a multi-controller SDN setup with segregated network control. They also learn by practice that such setups require protection against hijack attacks.

Lab 2: Unauthorized Access by Unauthorized Applications

Problem Definition: The SDN paradigm enables multiple tenants to operate on the same network. A tenant here uses controller resources to install network flows. It is often assumed to be trusted as it runs as an application module inside the controller. However, this enables it to potentially cause unauthorized operations in the network.



	Priority(Foo) > Priority(Bar)	Priority(Foo) < Priority(Bar)
Rate < 10	Bar App is allowed to add its flow rule.	Bar App is allowed to add its flow rule.
10 < Rate < 15	Bar App's flow rule is deleted and Foo App is allowed to add its flow rule.	Foo App is denied to add its flow rule and Bar App's flow rule persists.
15 < Rate	(Never Reached)	Bar App's flow rule persists.

Fig. 4. Expected workflow events for tenants, Foo and Bar

Lab Objectives: Students are instructed to develop a controller module to broker the task of installing tenant flows. A tenant here is required to register with a certain priority heuristic with the broker and request permission for installing flows from it. The broker must detect conflicts for requested flows and resolve them based on this heuristic on their behalf. Lastly, students execute a workflow (Fig. 4) involving two conflicting tenant modules: 1) Foo, which drops traffic with rate > 10 packet-per-second (PPS) and 2) Bar, which forwards traffic with rate < 15 PPS.

Lab Challenges: Students need to 1) identify appropriate criteria for determining flow conflicts and 2) learn to provision tenants with service to internally communicate with the broker.

Hackathon: Students are required to configure their broker to address a set of operations that a tenant is permitted to perform on network flows. A tenant here is required to register with a set of flags (get, add, remove) denoting these operations with the broker and request permission from it for performing them.

Learning Outcome: Students learn about handling flows from multiple tenants in an SDN environment. They also learn by practice that mismanagement of tenants can compromise network integrity.

Lab 3: DoS Attack Detection and Mitigation

Problem Definition: Networks in general are designed to be reachable to others. denial-of-service (DoS) describes a problem scenario in which the nature and volume of traffic overwhelms a network, potentially compromising it by disrupting communication and making destinations unavailable for service. The introduction of centralized control and separation of planes in SDN introduces new dimensions to the solution space for this problem.

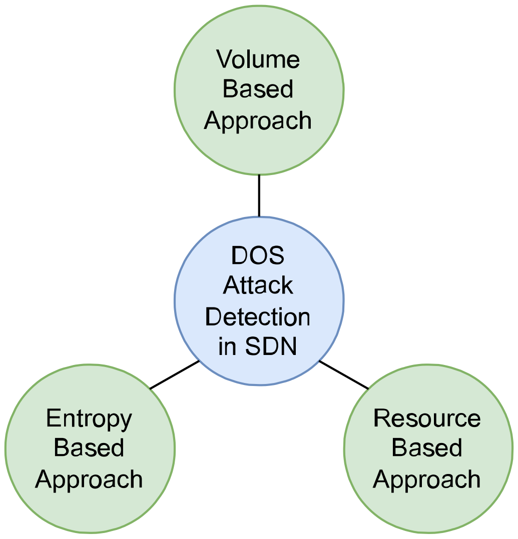


Fig. 5. DoS attack detection mechanisms

Lab Objectives: Students are instructed to develop a controller module to detect DoS attacks for a given traffic-type, i.e., ICMP using two different approaches: 1) volume-based, which uses a threshold over traffic-volume, 2) entropy-based [3] which uses a heuristic over the mean entropy over traffic-volume across different network destinations. Lastly, they execute a workflow and mitigate the attack using suitable heuristics alongside these approaches.

Lab Challenges: Students need to 1) determine appropriate window and threshold for sampling traffic for given workflow and 2) learn to retrieve the traffic statistics from the switches instead of inspecting traffic at the controller.

Hackathon: Inspired by elements of Adaptive Bubble Burst technique [4], students are given a resource-based approach for which they choose a file-storage application, deploy replicas of it with identical content across different network hosts, and develop a controller module to 1) detect the attack by designing a heuristic over application load and 2) mitigate the attack by scaling the replicas based on it.

Learning Outcome: Students learn about different approaches (Fig 5) to detect DoS attack. They also learn by practice that different scenarios call for different measures to mitigate it.

Lab 4: Network Configuration Issues

Problem Definition: SDN enables a network to offload its policy configuration to a variety of client applications. Some of them function externally over controller's northbound interfaces (NBI). These interfaces are often designed to be open-ended so as to welcome a variety of such applications. Without proper management, they could potentially compromise the integrity and consistency of a network.

Lab Objectives: Students are instructed to develop an application to manage policies for clients from outside the controller. They need to configure it to comprehensively address the given problem using two approaches: 1) proactive approach that resolves conflicts between incoming and existing client policies and 2) reactive approach that monitors the network and resolves conflicts between client (managed) and other (unmanaged) policies. Lastly, they develop a general application for clients to test these approaches using comprehensive workflows.

Lab Challenges: Students need to 1) implement the interface between manager and clients for communicating policies, 2) determine an appropriate data-structure to handle policies (Fig. 6) at the manager and 3) learn to invoke controller's REST API from the manager to communicate with the controller.

Field	Description
id	ID of the client (e.g. "App 1", "App 2")
type	Type of policy - "forwarding", "forbidden" or "unmanaged"
src-ip	IP address of the source host in the policy
dst-ip	IP address of the destination host in the policy
priority	Priority of flows in the policy
flows	Comma separated list of flows in the policy. Format: "<switch-dpid, in-port, [out-port(s)]>, ..." Example: "<of:0000000000000002,1,[3]>, <of:0000000000000001,3,[2]>"

Fig. 6. Example of a policy schema

Hackathon: A network-cycle is an example of a consequence of faulty network policies that has a negative effect on network state and infrastructure. Given an example of an SDN-operated cyber-physical system (CPS), students are asked to design a mechanism to efficiently detect the cycle (Fig. 7) without indiscriminately spanning the entire topology.

Learning Outcome: Students learn about several ways to manage external network configuration. They also learn by practice that faulty policies can compromise the integrity and consistency of a network.

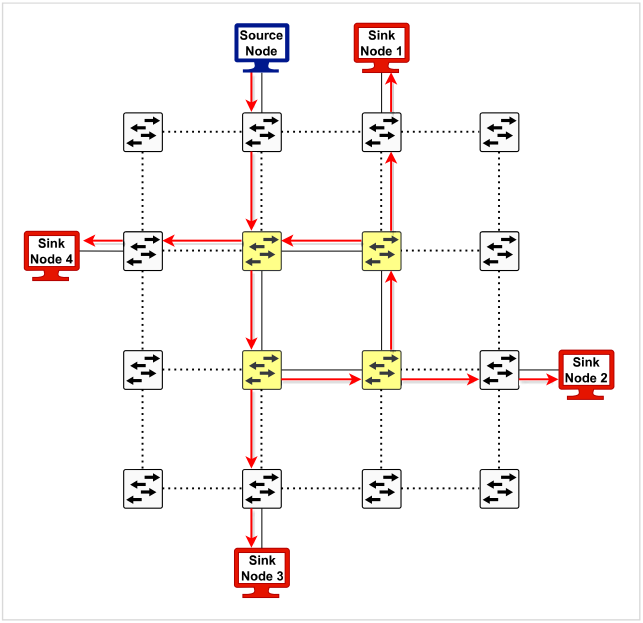


Fig. 7. Example of network cycle due to poor network policies

Lab 5: ARP Spoofing Attack Mitigation

Problem Definition: Address Resolution Protocol (ARP) is widely used for producing a mapping between IP and MAC addresses in network switching domain. However it is known to be vulnerable to a range of spoofing attacks [5]. SDN introduces a new dimension to the solution space to mitigate them.

Lab Objectives: Inspired by some of the strategies described in [5], students are instructed to develop a controller module to address the given problem in three stages: 1) MITM-based attack prevention, that detects mismatch between ethernet and ARP headers to prevent ARP cache poisoning attacks, 2) Host tracking and filtering, that tracks host-identities to prevent impersonation attacks and 3) Stateful ARP Inspection, that correlates ARP request and response packets to thwart unsolicited ARP messages.

Lab Challenges: Students need to 1) build host-identities using a combination of IP, MAC and switch-port information retrieved from openflow and data packet headers and 2) determine a way to use them to efficiently track ARP request and response messages.

Hackathon: Given that host-location given by switch-port can arbitrarily change, students are instructed to configure their module to apply an entropy heuristic over it to mitigate the attack.

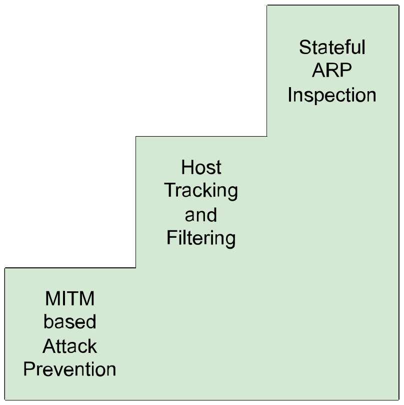


Fig. 8. ARP spoofing attack mitigation strategies

Learning Outcome: Students learn about several ways (Fig. 8) to prevent ARP attacks in an SDN setting. They also learn by practice the significance of location and identity of hosts in addressing the given problem.

Lab 6: Moving Target Defense

Problem Definition: Communication between network hosts depends on the ability of 1) the hosts to identify each other and 2) the network to identify the paths between them. Typically services such as DNS are deployed and a routing process such as Dijkstra's shortest path algorithm is employed to aid with (1) and (2) respectively. However, given enough time, an adversary can discover target hosts and predict the path between them to stage an attack. Moving Target Defense (MTD) describes a range of mechanisms to introduce unpredictability in the network to address this problem.

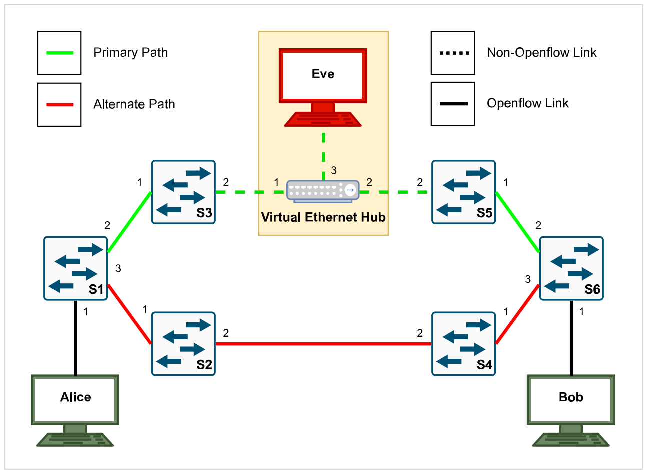


Fig. 9. Example of RRM with an adversary (Eve)

Lab Objectives: Inspired by an MTD mechanism called Random Host Mutation [6], students are instructed to develop a controller module that randomly and frequently changes network identity of hosts across IP and ARP packets to give an adversary very little scope to discover them and stage an attack.

Lab Challenges: Students need to 1) consistently generate virtual identities (addresses) for network hosts and 2) ensure that ongoing communication isn't disrupted by identity changes. Hackathon: Inspired by an MTD mechanism called Random Route Mutation [7] (Fig. 9), students are asked to develop a controller module that randomly and frequently alters between network paths across actively communicating hosts to give an adversary very little scope to stage an attack to interpret their conversation. Students need to determine a way to seamlessly apply the required mutation.

Learning Outcome: Students learn about two MTD mechanisms popular in an SDN setting. They also learn by practice about concerns around the dynamics of these mechanisms and how to address them to ensure consistency of network communication.

Lab 7: ML-based Network Intrusion Detection

System Problem Definition: Networks are constantly under the threat of malicious attacks. However the heuristics and mechanisms used today to detect them are not suitable for all network scenarios. This has motivated network designers to employ learn-ability in the design of Network Intrusion Detection System (NIDS) to train machine learning (ML) models to predict malicious behavior in a desired network environment. Lab Objectives: Students are instructed to implement an ML-based NIDS setup (Fig. 10) to detect and mitigate a TCP-SYN-based flooding attack in an SDN-operated network. In this setup, Argus (openargus.org) captures network flows and Elasticstack (elastic.co), a big-data toolset, harvests and stores them. Students need to develop a service that 1) periodically retrieves flows from Elasticsearch, 2) trains an ML-classifier (i.e., logistic regression) with a suitable dataset [8] to classify them and 3) instructs the controller (i.e., ONOS) to drop malicious flows.

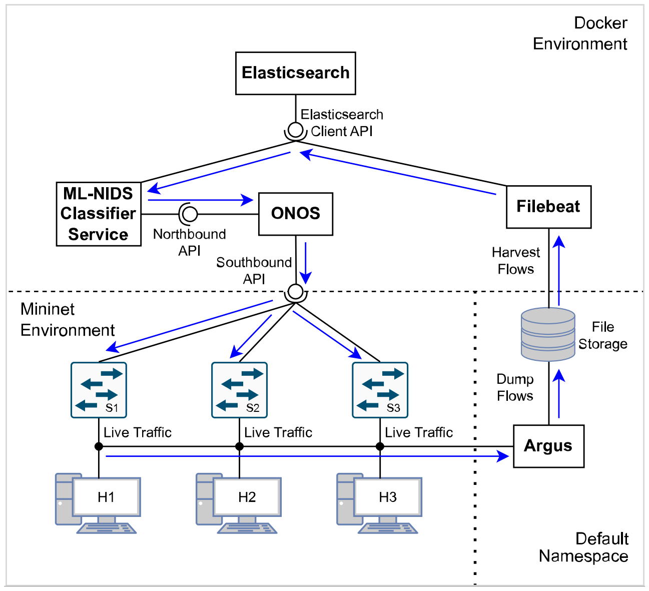


Fig. 10. ML-based NIDS setup overview

Lab Challenges: Students need to 1) learn to configure an ML-classifier using scikit-learn (scikit-learn.org), 2) learn to interface with Elasticstack and ONOS using provided (REST) APIs and 3) ensure consistency of flow predictions across them.

Hackathon: Students are instructed to configure their setup with different ML-classifiers and report their end-to-end performance with appropriate reasoning.

Learning Outcome: Students learn about the application of ML-based NIDS in an SDN setting. They also learn by practice the dynamics of such a setup and its significance in mitigating malicious traffic.

Lab 8: Closed Loop Automation

Problem Definition: Closed loop automation (CLA) is a high-level term for introducing a range of feedback mechanisms to seamlessly automate control across parts of a system and its resources. The controllers in SDN have mainly two sources of feedback for network operations: 1) underlying network and 2) its various applications. Automating control without addressing feedback in SDN can potentially introduce a range of security issues such as buffer overflow and side-channel vulnerability.

Lab Objectives: Students are instructed to develop a controller module to address a quality-of-service (QoS) challenge using CLA. They need to configure it to support a given streaming workflow on Apache Flink (flink.apache.org), an open-source distributed data-stream processing application. They need to retrieve suitable metrics for the workflow as feedback from flink and apply a suitable heuristic over it to configure a QoS mechanism using Openflow Meters in SDN to aid streaming endpoints in processing occasional large bursts of application-layer streaming traffic.

Lab Challenges: Students need to 1) learn to implement the interface between flink and the controller since they belong to different network planes, and 2) ensure seamless deployment of Openflow meters on top of existing network flows.

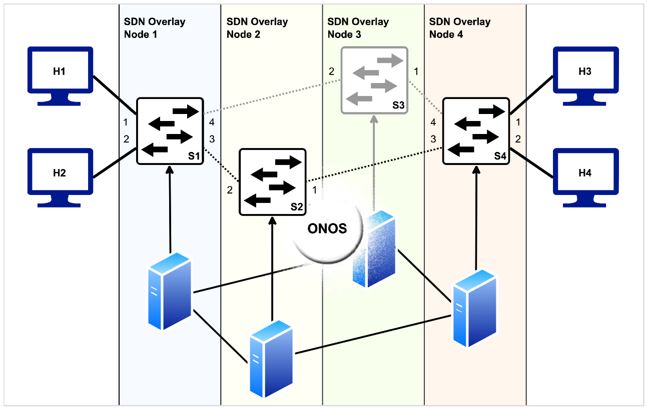


Fig. 11. SDN Overlay Setup

Hackathon: Given an SDN overlay network (Fig. 11) built on top of nodes constituting commodity servers, students are asked to configure it to support the streaming workflow. They need to allocate additional nodes (and links) in the overlay and migrate its flows across them when required and deallocate and revert them afterwards to avoid potential side-channel vulnerability. They need to configure their module to assess feedback from the network (statistics) to determine this requirement and automate the task using secure shell.

Learning Outcome: Students learn about the importance of CLA in securing an SDN operated network. They also learn by practice the significance of feedback in CLA.

IV. RELATED WORK

The related work in this domain can be grouped into two main categories. In the first group, we have many efforts mainly focusing on building a scalable and accessible medium of delivery of lab content. These include a variety of lab platforms [9] [10] [11] [12] [13] [14]. Our work focuses on developing labs to contain in a standard Linux VM that may be easily ported to some of these platforms.

The second group of efforts have been on developing lab content that focus on a diverse set of cybersecurity issues. These include cybersecurity labs [15] [16]. Our work mainly focuses on practical security issues in SDN networks.

The work [17] that is closely related to ours focuses on developing labs on SDN security using CloudLab platform. Most of the labs in this work are at the introductory level exposing students to some of the well-known network security issues within the context of SDN. Our work can be seen as an extension of this study where we focus on security challenges faced in practical SDN deployment scenarios.

V. CONCLUSION

In this paper, we presented our work on developing a number of SDN security lab activities. The selected activities are inspired from realistic and practical SDN network deployment scenarios and provide students with a structured framework to try out new solutions that they may come up with. We plan to include these lab activities in our graduate level network security class to get feedback and improve on them.

ACKNOWLEDGEMENT

This work is partially supported by NSF award DGE-1820640.

REFERENCES

[1] D. Joyce *et al.*, “Developing laboratories for the sigcse computing laboratory repository: Guidelines, recommendations, and sample labs,” *SIGCUE Outlook*, 1997. [Online]. Available: https://doi.org/10.1145/274382.274779

[2] E. Teiniker, G. Seuchter, and W. Farrelly, “Engaging part-time students in software security by inductive learning,” in *IEEE Global Engineering Education Conference*, 2019. [Online]. Available: https://doi.org/10.1109/EDUCON.2019.8725029

[3] S. M. Mousavi and M. St-Hilaire, "Early detection of ddos attacks against sdn controllers," in *International Conference on Computing, Networking and Communications*, 2015. [Online]. Available: https://doi.org/10.1109/ICCNC.2015.7069319

[4] D. Sattar, A. Matrawy, and O. Adeojo, "Adaptive bubble burst (abb): Mitigating ddos attacks in software-defined networks," in *17th International Telecommunications Network Strategy and Planning Symposium*, 2016. [Online]. Available: https://doi.org/10.1109/NETWKS.2016.7751152

[5] Z. Shah and S. Cosgrove, "Mitigating arp cache poisoning attack in software-defined networking (sdn): A survey," *Electronics*, 2019. [Online]. Available: https://doi.org/10.3390/electronics8101095

[6] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow random host mutation: Transparent moving target defense using software defined networking," in *Proc. 1st Workshop Hot Topics Software Defined Networking. Association for Computing Machinery*, 2012. [Online]. Available: https://doi.org/10.1145/2342441.2342467

[7] Q. Duan, E. Al-Shaer, and H. Jafarian, "Efficient random route mutation considering flow and network constraints," in *IEEE Conference on Communications and Network Security*, 2013. [Online]. Available: https://doi.org/10.1109/CNS.2013.6682715

[8] "Darpa 2009 ddos attack-20091105," University of Southern California - Information Sciences Institute, 2009. [Online]. Available: https://doi.org/10.23721/109/1358116

[9] R. Abler, D. Contis, J. Grizzard, and H. Owen, "Georgia tech information security center hands-on network security laboratory," *IEEE Transactions on Education*, 2006. [Online]. Available: https://doi.org/10.1109/TE.2005.858403

[10] J. Mirkovic and T. Benzel, "Teaching cybersecurity with deterlab," *IEEE Security & Privacy*, 2012. [Online]. Available: https://doi.org/10.1109/MSP.2012.23

[11] C. Pérez, J. M. Ordun˜a, and F. R. Soriano, "A nested virtualization tool for information technology practical education," *SpringerPlus*, 2016. [Online]. Available: https://doi.org/10.1186/s40064-016-2041-8

[12] S. Mohanarajah, G. Ross, and S. Suthaharan, "Towards encapsulated cyber security labs: A container based approach," in *Proc. 50th Technical Symposium Computer Science Education. Association for Computing Machinery*, 2019. [Online]. Available: https://doi.org/10.1145/3287324.3293832

[13] K. Salah, M. Hammoud, and S. Zeadally, "Teaching cybersecurity using the cloud," *IEEE Transactions on Learning Technologies*, 2015. [Online]. Available: https://doi.org/10.1109/TLT.2015.2424692

[14] C. Kuo, K. Chain, and C. Yang, "Cyber attack and defense training: Using emulab as a platform," *International Journal of Innovative Computing, Information and Control*, 2018. [Online]. Available: https://doi.org/10.24507/ijicic.14.06.2245

[15] W. Du, "Seed: Hands-on lab exercises for computer security education," *IEEE Security & Privacy*, 2011. [Online]. Available: https://doi.org/10.1109/MSP.2011.139

[16] A. Pattanayak, S. Steiner, and D. C. de Leon, "Hands-on educational labs for cyber defense competition training," in *Journal of The Colloquium for Information Systems Security Education*, 2022. [Online]. Available: https://doi.org/10.53735/cisse.v9i1.144

[17] Y. Park, H. Hu, X. Yuan, and H. Li, "Enhancing security education through designing sdn security labs in cloudlab," in *Proc. 49th Technical Symposium Computer Science Education. Association for Computing Machinery*, 2018. [Online]. Available: https://doi.org/10.1145/3159450.3159514