Network Applications Observability on Hosts: A New SDN Privacy Issue

Conference Paper · October 2023

DOI: 10.1109/ICNAS59892.2023.10330440

3 authors:

Serrat Ishak
Harbin Engineering University
3 PUBLICATIONS **24** CITATIONS
SEE PROFILE

Tayeb Kenaza
51 PUBLICATIONS **288** CITATIONS
SEE PROFILE

Islam Debicha
Université Libre de Bruxelles
15 PUBLICATIONS **62** CITATIONS
SEE PROFILE

Network Applications Observability on Hosts: A New SDN Privacy Issue

1st Ishak SERRAT
Computer Security Laboratory
Ecole Militaire Polytechnique
Algiers, Algeria
d_serrat.ishak@emp.mdn.dz

2nd Tayeb KENAZA
Computer Security Laboratory
Ecole Militaire Polytechnique
Algiers, Algeria
ken.tayeb@gmail.com

3rd Islam Debicha
Computer Security Laboratory
Ecole Militaire Polytechnique
Algiers, Algeria
debichasislam@gmail.com

***Abstract*—Software Defined Networking is a relatively new network paradigm that enables unified programmable network control. SDN is still subject to several challenges, such as scalability, availability, security, and reliability. Furthermore, the deployment and expansion of SDN to many specific networks, such as IoT networks, have revealed further challenges, namely privacy issues. Few works in literature have been devoted to certain SDN's privacy problems. In this paper, we highlight a new privacy concern introduced by SDN architecture that has not been considered in the literature. In order to demonstrate the problem and because of the lack of SDN traffic and flow rule datasets, we first implemented *Legacy2SDN*, a solution for reproducing a legacy network traffic into an SDN environment using PCAP files. We have also implemented a third-party network application that passively collects flow rules and their statistics. *Legacy2SDN* along with the developed network application are used on a real privacy sensitive network dataset, to conduct experimental case studies to illustrate the privacy issue. Our experimental results prove the observability of network applications on hosts, which can threaten their owners privacy. Finally, we leave the door open for future research in this area by initiating discussion about possible solutions.**

***Index Terms*—SDN Security, Host-Owners Privacy, Malicious Application plane, SDN privacy.**

I. Introduction

The rapid advance in internet technologies and services has increased the complexity and scale of modern network deployments for both centralized infrastructures such as cloud networks (Production networks), and ubiquitous infrastructures such the IoT and ISP Networks. This complexity has left conventional network control insufficient [1], due to the lack of openness and general control paradigm. The current network deployments require more agile, flexible and cost-effective network architectures, to achieve greater scale, increased security, and more elasticity (VM migration).

Software Defined Networking (SDN) is a relatively new network control paradigm, which has emerged to ease network control, simplify their management [2], and overcome legacy network limitations by enabling unified and on-the-fly programmable network control. SDN architecture segregates the control plane from network devices to a software-based, logically centralized entity known as the SDN controller. The controller communicates with network devices using a standardized protocol such as OpenFlow, so it retains an up-to-date network state and a global network view by collecting flows statistics, and abstracts low-level network resources to the application plane by translating high-level policies into primitive instructions namely flow rules.

Nevertheless the SDN concept started to ease production networks management, it has rapidly evolved toward all types of networks, from enterprise scale to IoT, WAN, and ISP networks [3]. In fact, several studies leverage SDN architecture and its newly introduced potential to address current network challenges, enhance network security, and preserve privacy. On the contrary, other works focus on SDN's new challenges including availability, scalability, reliability, interoperability, and Security. However, SDN deployment and expansion beyond production networks to privacy sensitive networks such as IoT and ISP, have revealed further challenges such as privacy issues. Fewer works have been devoted to these issues, including privacy preservation for cross domain routing, and privacy preservation for inter-domain policies. The goal of our work is to highlight network applications observability on hosts as a new privacy concern brought by SDN itself. To the best of our knowledge, no work has emphasized this privacy issue, and this is the first work to raise this concern that we also call Host-Owners' privacy issue in SDN architecture.

Furthermore, to handle a noticeable lack of dataset dedicated for SDN, we provide *Legacy2SDN*, a solution for reproducing a legacy network traffic in an SDN environment using PCAP files. The provided solution along with an implementation of a network application are used to conduct experimental tests on a real dataset of IoT Smart Home network and collect relative statistics and flow rules. We highlight host-owners privacy issue to prove network applications observability on hosts. We also initiate discussion about possible solutions.

The remainder of this paper is organized as follows. Section 2 describes briefly the background of SDN and related works. The problem statement and threat model are charactrized in section 3. In Section 4, we present case studies, including our experiments setup, implementation, tests and results. Possible countermeasures are discussed in Section 4. Finally, we conclude the paper in section 5.

II. Background and Related Works

In this section, we first introduce a brief SDN background. We consider open SDN which refers to SDN with plane separation, simple forwarding devices, logically centralized control, network function abstraction, and open interfaces. Then, we discusses the existing literature and research related to privacy issues in SDN, and prior works that have addressed similar and related problems.

A. SDN Architecture

The core idea of SDN is to separate the network control plane from the data plane, making control centralized rather than distributed, thereby providing a new network layering model (Fig. 1), where the data plane encompasses set of devices of the network infrastructure, usually referred to as ”forwarding devices (FDs)” (or ”programmable switches”). FDs do not behave autonomously or run complexes and distributed protocols anymore; instead, they ensure forwarding functionality by adhering to rules set by the control layer.

The control layer is moved to a logically centralized controller, often implemented as software. The controller communicates with network devices (Data plane) using a standardized protocol such as the defacto OpenFlow protocol. Through this communication, the controller collect statistics to get the global view of the entire network and manage network policies. It also abstract network from underlying hardware to network administrators. This abstraction allows to define network behavior and policies using high-level programming languages or specialized network management applications.

The Application plane holds these high-level network applications. Network applications interact with the controller through the Northbound interface (NBI) which allows programming FDs and/or collecting network relevant statistics. SDN's applications tend also to be implemented by third parties, and since their requirements are so divergent, different North Bound APIs where enabled to allow more flexibility for application developers. Nevertheless, some APIs are largely enabled and used in multiple controllers, such as the RESTful API.

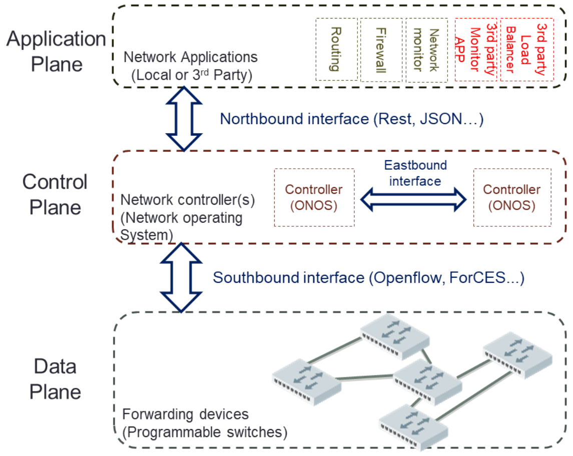
Application Plane
Network Applications (Local or 3rd Party)
Routing
Firewall
Network monitor
3rd party Monitor APP
3rd party Load Balancer
Northbound interface (Rest, JSON…)
Control Plane
Network controller(s) (Network operating System)
Controller (ONOS)
Eastbound interface
Controller (ONOS)
Southbound interface (Openflow, ForCES…)
Data Plane
Forwarding devices (Programmable switches)

Fig. 1. SDN architectural model.

B. Related works

From its appearance, SDN has showed potential to make networking more flexible, easily manageable and efficient. Nevertheless, It is still an active research area with new opportunities to discover, and also subject of open issues and challenges [8]. For decade, SDN challenges mainly included: Availability, Scalability, Reliability, Interoperability, and Security. However, the SDN's privacy issue arises when SDN is expanded beyond production networks to highly privacy-sensitive networks. In fact, production networks are less privacy sensitive since they rely on a purely virtual and dynamic environment. Currently, highly privacy-sensitive networks seem to be IoT networks, internet provider networks, and inter-domains networks.

Many studies leverage SDN's concepts to deal with known privacy issues, omitting the SDN threat to privacy. From one side, to the best of our knowledge, papers [9], [10] are the only research works that treat a part of SDN-related inter-domain privacy issues, namely the privacy-preserving for inter-controller data plane forwarding policies and cross-domain privacy-aware routing, respectively. Authors in [9] deploy Secure Multi-party Computation (SMC) as the GMW protocol to ensure sharing of private and sensitive inputs and policies between SDN controllers, then optimize the number of controllers involved in the SMC to reduce the SBI's latency. While the work in [10] considers the problem of sensitive information disclosure through routing protocols and has proposed PYCRO. The latter is a cryptographic protocol for privacy-preserving cross-domain routing optimization in SDN, which provides two fundamental privacy-aware routing services: shortest path computing and bandwidth allocation. Furthermore, works such as [11] treat SDN's network isolation attacks and countermeasures and propose a detection and defense scheme, RSDetector and SpoofDefender, respectively. Despite the fact that this scheme does not directly address the privacy issue, it seems that network isolation issues and their related privacy issues are correlated.

On the other side, to the best of our knowledge, no works in the literature have pointed out or addressed the SDN host owners' privacy issue or suggested network application observability on hosts as a privacy issue. Indeed, we are the first work that calls attention to this issue with real experiments and initiate discussion about feasible solutions that maintain SDN's utility while protecting host owners' privacy.

III. Host-Owners Privacy Issue

The rising prevalence of Internet-connected devices and appliances, promises both new opportunities and new privacy concerns. Many IoT devices, unlike traditional network enabled devices, have always-on sensors that continuously monitor the physical environments of users and influence their network communications pattern. Their communication pattern with cloud-based services may expose users to certain privacy risks. Active and passive network observers could potentially deduce sensitive information about users by just analyzing IoT devices traffic.

The SDN architecture might motivate malicious network applications to act as network observer and threaten host owners' privacy, while hosts are any device that may directly or indirectly threaten peoples privacy by either their collected data, their traffic signature or by just their identification, such devices includes: personal computers, laptops, smartphones, and more seriously IoT device (Sensors and actuators) or any smart device which are used by human being and involved in their daily life and habits. Malicious Network applications will probably tries to compromise hosts privacy actively or passively. On one hand, actively means that application installs malicious flow rules to affect the behavior of the network to perform attacks such as man-in-the-middle (MITM); by routing packets through their data collectors, and hence eavesdrop to violate hosts privacy or even more seriously alter packets content, and change data integrity. Active attacks are easily detectable, and several researches have been proposed to mitigate SDN's MITM based attacks [4]. However, less harmful privacy violation attacks consist of injecting into different switchs fine-grained rules which are specific to some network hosts, by for example, defining flow rules with a source Mac address or a destination IP address, then collecting rules relative statistics.

Otherwise, the adversary's network applications may perform a full passive attack where they just collect and analyze flow rules and their statistics to infer sensitive private information. Then, since in the internet architecture, most packets are traceable using network-based traceback techniques [5], adversaries network applications may be able to trace back destination servers (DNS, NTP, cloud service, etc.) and their purpose to identify the host identity, such as a Sense Sleep Monitor that can be identified through DNS queries [7], or other device that is enabled through a manufacturer's specific cloud service on the Internet and has a limited purpose. Furthermore, host location can be inferred from his IP address as well as their real identity (device or person identity) through the analysis of communication patterns and habitual tendencies using machine learning techniques [6]. Despite the fact that this attack is less harmful compared to the MITM attack, it is conceivably more difficult to detect.

On the other hand, the adversary's applications might act as global passive statistics eavesdroppers, which are able to monitor all traffic statistics in the network, thus identify who is communicating with whom, at which traffic rates, and their communication habitual tendency. Such an observability combined with Media Access Control (mac) and Internet Protocol (IP) addresses can even lead to infer host owners identity, thus expose their privacy to risks. From this perspective, it seems evident that malicious behavior of such passive privacy attacks is harder to detect.

Indeed, The latter issue rises more due to the expanding use of SDN paradigm from production networks to privacy-sensitive networks. Production networks are less privacy sensitive, since they rely on machines virtualization, and cloud computing, and that explains why privacy-aware has not been considered in SDN architecture proposals. On contrary, IoT (Smarthome...), ISP networks are privacy-sensitive networks, since they involve smartphones, sensors and actuators which involves human privacy directly either through collected data itself or related traffic or only through the identity of the IoT device itself.

A. Threat Model

We consider a full SD-IoT network, where all IoT devices communicate to LAN and to internet through SDN, and all the paths hops between packets sources and destination are Openflow programmable switches. We do not place any restriction on communication encryption.

In term of privacy, we do not consider controller threat to hosts' privacy. Eventually, the main controller's purpose is to implement the core of network functionalities and to provide an abstraction of network capabilities to applications. Considering the privacy threat of controller itself is similar to privacy and trust issues related to services providers.

We consider malicious or honest but curious network applications that tries to learn sensitive information about hosts of the network, consequently of users and peoples owners of these hosts. Sensitive information includes traffic rates, communication habitual tendency and even their identity. Malicious application may try to alter the traffic flow to collect information whereas honest but curious network application may performs their tasks honestly but in the same time tries to learn about the network owners. In such a scenario, application may firstly register to the controller as a network tool that collect only flow rule and statistics to extract useful information in varied forms, or as a load balancer tool that do really performs its tasks, but threat host owners' privacy while performing their tasks. We assume that adversaries applications has sufficient resources to collect, store and analyse large scale networks flow rules. We also assume that adversaries have a prior knowledge about IoT devices communication patterns.

IV. Case studies

In order to prove the observability of network applications on hosts, and since the lack of SDN traffic datasets for privacy sensitive networks, our key idea is to reproduce a highly privacy-sensitive network traffic in an emulated SDN and collect relative flow rules and their statistics. In this section, we detail our conducted experiments setup, implementations, test scenarios and results.

A. Experiments setup overview

For our experiments, we used a laptop with an I7-4700HQ (8 cores) 2.4GHz processor, 8 GB of memory, and Windows 10. Fig. 2 illustrates our experiments setup scheme. On one side, we put together two virtual machines within Oracle VirtualBox, both running the Linux Debian OS. The first virtual machine is dedicated to a network controller running Open Network Operating System (ONOS) [12] version 1.5 on the Docker platform. ONOS is one of the most widely used, open-source, and community-wide supported state-of-the-art SDN controllers. We run only one node (instance) of

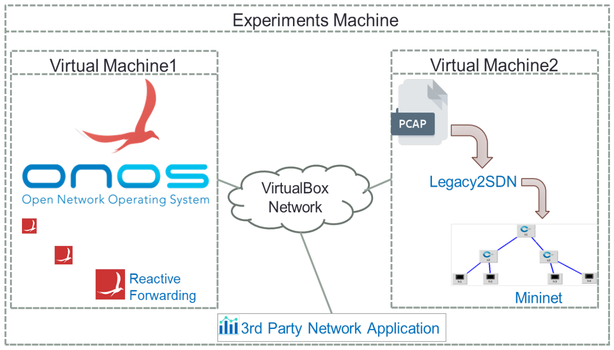

Fig. 2. Experiments setup

ONOS due to resource limitations and since our objective is to present a proof of concept. Nevertheless, we believe that cluster of nodes will give the same results, considering that cluster mode ensures data consistency between nodes. While the second virtual machine is dedicated for hosts and network emulation, it runs Mininet [13] version 2.3.0d1, which is a network emulator that builds virtual networks of links, switches, controllers, and hosts. Mininet is used as an SDN testbed to perform different network traffic scenarios. We set up Mininet using OVSwitch with OpenFlow version 1.3. We also configure our Mininet-emulated networks to use our ONOS remote controller through an Out-Of-Band channel.

On the other side, Flow rules and their relative statistics are collected by a third-party network application. The application runs on the laptop (Windows 10). Our network application is unable to reach directly emulated network switches or Hosts. However, it is able to authenticate to our controller and pull statistics and flow rules through the controller's northbound interface. The application performs a passive data collection. Hence, we configured and ran ONOS reactive forwarding application (org.onosproject.fwd): an application that intercepts packets for which there is no flow matching (Packet-In) and computes the shortest path to destination using the ONOS path service, then provisions short-lived flow rules for hop-to-hop packet forwarding. Furthermore, reactive forwarding application works at layer 2 and by default uses only rule matching by MAC addresses, hence we activated IP address matching and configured static routing at each host to ensure that packet reach their first connected switch.

B. Implementations

We have implemented two main codes to conduct our experiments. The first tool is Legacy2SDN which replays a legacy network traffic on a Mininet SDN network, by analysing network PCAP files per host, then run packets between hosts according to related PCAP files. Legacy2SDN takes as input a Json encoded file and PCAPs folder. The file contains device names, their IP and MAC address, and their PCAP filenames. Legacy2SDN is still scenario specific but we plane make it more automated and not scenario specific. The second tool represent a third party network application that collects flow rules statistics using ONOS pull based approach through REST API. Our both implementations are written in Python language. We used SCAPY package to handel PCAP files, Mininet API to create virtual SDN with real kernel OpenFlow switchs(OVSwitch), virtual hosts and also emulate the traffic between them.

TABLE I
IoT DEVICES AND THEIR DETAILS USED IN OUR TESTS

Device name	IP Packets(IN/OUT)	Used traffic duration
Blink Indoor Home Security Camera 1	14621/9091	7 days(full)
Blink Indoor Home Security Camera 2	13378/9317	7 days(full)
Blink Indoor Home Security Camera 3	15226/10033	7 days(full)
Geeni Aware Surveillance Camera 1	8811	two hours and half (part)
Geeni Aware Surveillance Camera 2	9652	two hours and half (part)
Schlage Encode Smart WiFi Deadbolt	194/166	two hours and half (part)
Philips Hue Bridge (connected to 5x Philips Hue Lights)	3371	one day (part)

C. Test and results discussion

Our main idea is to reproduce highly privacy-sensitive network traffic in SDN and collect relative statistics to show through evidence the privacy issue. For this purpose, we first targeted IoT and smart home networks since they are highly privacy-sensitive networks. We looked for a traffic dataset for IoT or smarthome networking; therefore, we have analyzed several online available datasets. The IoT Sentinel [14] and IoT traffic [15] datasets were the only more realistic ones since they deploy commercial IoT devices and both have published their Pcaps traffic files. However, IoT Sentinel was capturing only the packets sent by each device during the setup process, which is not suitable for our purpose. We adopted the IoT traffic dataset for our experimental tests since it met our needs and also because authors published labeled events over time along with Pcap files for different IoT devices. The dataset includes seven days of traffic for 57 IoT devices. Owing to the fact that we rely on ONOS, we were unable to accelerate traffic scenarios; hence, we have used for some devices the full traffic and for others just a time portion of the traffic. Table I represents the set of IoT device traffics and time portions used in our experiments.

In our first scenario, we set up a network of three Blink indoor home security cameras using our Legacy2SDN and their traffic Pcap files. Then we run our network application to collect flow rules from the first switch connected to the Blink cameras. We did manually treat and analyze the collected flow rules and their related statistics. We recall that our network application pulls flow rules every one second, which is a frequency sufficient to not miss any flow rule update since ONOS by default updates flow rules and their statistics every 5 seconds.

By analyzing and viewing packets and bytes counts of flow rules collected by our network application, Fig 3 shows the number of packets sent and received per day for traffic of three

Blink indoor home security cameras in the UTC timezone. We can notice that on March,12th there were a few packets compared to other days, while in March, 14 only one camera recorded traffic. Further, we have focused on flow rules packet count of about three hours on March 10th as shown in Fig 4. We have noticed that the captured IP packet traffic aligns perfectly with labeled motion events of all cameras. In fact, this kind of camera generates traffic only when there is motion. Hence, our third-party application can infer the presence or not of camera owners and also other private information depending on camera placement, such as activity time. In a similar scenario, we tested more than two hours of two Geeni-aware security cameras, and the packets counts per flow rules are shown in Fig 5. Unlike the Blink camera, the Geeni camera keeps continuous traffic over time, however we are still able to link movement events by traffic picks, which are as accurate as the flow rule pull period. Further analysis of the collected flow rules reveals that same devices have the same traffic patterns, so they communicate with the same deported servers at the same rates. Using this information and the Organizational Unique Identifier (OUI), which identifies the manufacturer of a device using its MAC address, will significantly simplify the task of 3rd-party application to identify the kind of device and further use machine learning techniques to extract sensitive information.

In our third scenario, we deploy the Schlage wifi deadbolt traffic through our Legacy2SDN, and we view packets and bytes counts of relative collected flow rules (Fig 6). Through the figure, we notice that this IoT device has an easily identifiable communication pattern, so it communicates slightly every exactly 5 minutes after the last communication, and any other traffic that exceed 10 packets per five seconds matches exactly with the locking and unlocking action events of the labeled dataset.

In our last scenario, we replayed a capture of oneday(March 09th) of Philips Hue Hub in our SDN environment. Unlike other devices, monitoring the full flow rules of traffic did not disclose valuable information. Hence, we have inspected flow rules per network source and destination, which shows continuous traffic with several servers that we have tracedback mainly to Google Cloud, Amazon data servers and Google time servers. We notice that this device communicates with four time servers at a notable frequency, every hour and teen minutes to every server, such a pattern may be related to number of connected smart bar lights. In addition, we found that the most active flow rule connect our Hue Hub with a Google cloud server in Netherlands (Philips brand homeland), has a network pattern that align perfectly with lights triggering events, with peaks of bytes received from the server revealing the user's light interactions.

In summary, we have shown through our investigation that several IoT and smarthome devices can reveal users' potential sensitive data and interactions by simply plotting flow rules traffic (packets or bytes).

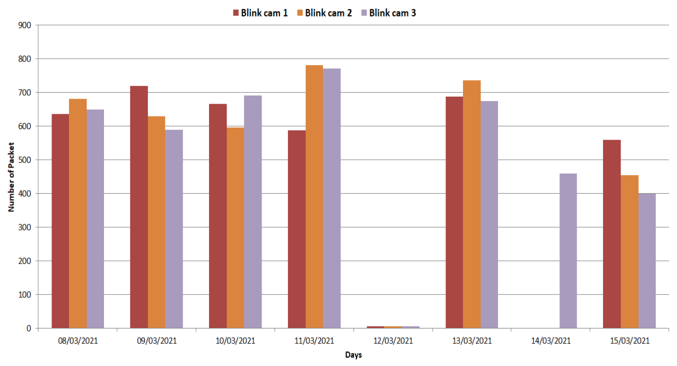

Fig. 3. Flow rules IP packets days (utc time) distribution for three Blink Indoor Home security Cameras.

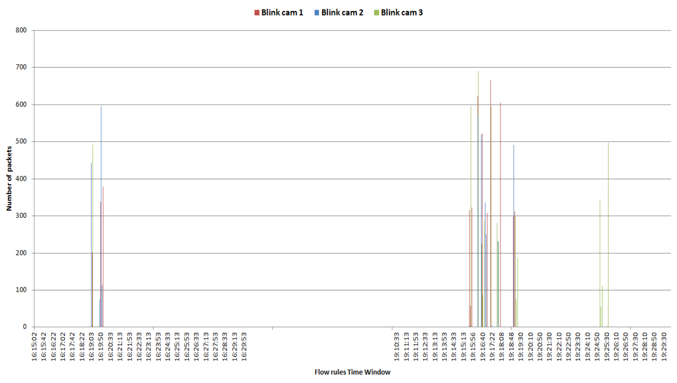

Fig. 4. Flow rules IP packets for three Blink Indoor Home security Cameras for March 10th, from 16:15 to 19:30 utc time.

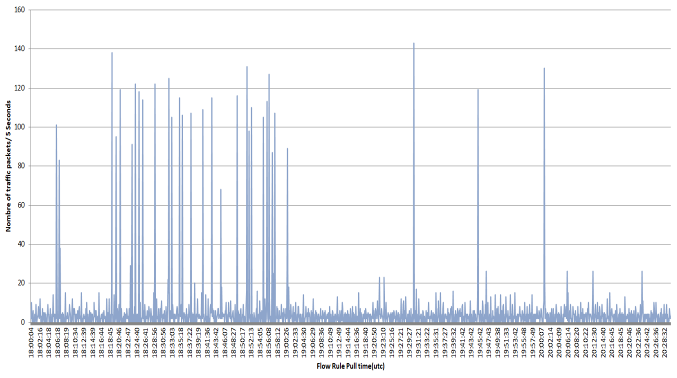

Fig. 5. Flow rules IP packets traffic of Geeni Aware Surveillance Camera for March 8, from 18:00 to 20:30 utc time

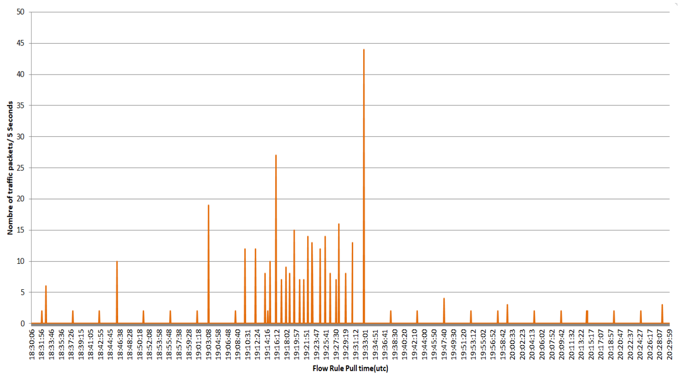

Fig. 6. Flow rules IP packets traffic of Schlage lock for March 13, from 18:00 to 20:30 utc time

V. Possible countermeasures

At the first sight, tunnels and end-to-end communication encryption appear to be a solution to the problem. Unfortunately, these techniques do not prevent observability. Inspired by our previous work on privacy preservation in participatory sensing [16], we believe that from the network perspective, observability can be prevented through communication anonymization techniques. The latter aims to hide the identities of the sender and receiver in the network by removing identifying patterns from the network flow. Most of the proposed anonymization techniques are based on either relying on trusted entities (centralized solutions) or laundering traffic through collaborative intermediate nodes (distributed solutions). Trusted entities are often used to help network hosts hide their traffic destination from their side and the source from the servers' side, such as through VPN obfuscation and anonymity enhancement. Distributed collaborative solutions include Mix and Tor networks, which enhance privacy by rerouting traffic through multiple nodes, and K-anonymity by setting k devices to communicate through an anonymous communication protocol to reach their specific internet servers, such as the bulk transfer protocol.

From the SDN perspective, we believe that the controller can be trusted and hence does not threaten privacy of hosts but, on the contrary can protect their privacy by providing a privacy-aware abstraction to the application plane. The first solution seems to be rule aggregation, which will help to break the link between traffic statistics and specific hosts. The network application then collects aggregated statistics and will be unable to extract individual host statistics. Another basic solution is perhaps the use of pseudonyms rather than Mac and IP addresses in flow rule presentation. Those pseudonyms should be short-lived and periodically changed by the controller host management, and only this service can link back hosts' aliases to their addresses.

Another possible solution to this issue is perhaps the use of legacy routers as gateways. Such a deployment may look like a controller-based hybrid SDN [17]. In fact, legacy routers have an obfuscation effect on controller network view and, hence, on network application statistics monitoring. When using legacy routers to route packets, the MAC and IP addresses of the router interfaces take their place at each packet cross, forward, and backward, resulting in the concealment of the MAC and IP addresses in the flow rules of the surrounding FDs. Legacy routers effects on SDN flow rules are similar to VPN anonymity enhancement, by hiding the IP addresses of their clients.

VI. Conclusion and future works

In this paper, we have presented two main contributions. We highlighted a new privacy concern introduced by SDN architecture that has not been considered in the literature. In addition, to handle the lack of SDN datasets, we provided a solution for replaying legacy network traffic into an SDN environment using PCAP files. The provided solution, along with an implementation of a network application, were used to conduct case studies and experimental tests on a privacy-sensitive network. Experimental results have proven the observability of network applications on hosts, which can potentially threaten their owners privacy. We also sow the seeds of possible solutions that may preserve privacy while maintaining SDN utility. We believe that more attention should be paid to SDN privacy issues. Our next step is to enhance Legacy2SDN to enable more scenario automation. Our future efforts will be focused on enabling privacy preservation for SDN architecture.

References

[1] Bari, M. F., Boutaba, R., Esteves, R., Granville, L., Podlesny, M., Rabbani, M., Zhang, Q., and Zhani, M. F. (2013). Data center network virtualization: A survey. IEEE Communications Surveys Tutorials, 15:909–928.

[2] Kim, H., Feamster, N., 2013. Improving network management with software defined networking. IEEE Commun. Mag. 51, 114–119.

[3] Rahouti, M., Xiong, K., Xin, Y., Jagatheesaperumal, S.K., Ayyash, M., Shaheed, M., 2022. SDN Security Review: Threat Taxonomy, Implications, and Open Challenges. IEEE Access 10, 45820–45854.

[4] Sebbar, A., Zkik, K., Baddi, Y., Boulmalf, M., and Kettani, D. (2020). Mitm detection and defense mechanism cbna-rf based on machine learning for large-scale sdn context. Journal of Ambient Intelligence and Humanized Computing, 11, DOI: 10.1007/s12652-020-02099-4.

[5] Wang, Xinyuan, Reeves, et al. Traceback and Anonymity[M]. Springer Publishing Company, Incorporated, 2015.

[6] K. Kostas, M. Just and M. A. Lones, "IoTDevID: A Behavior-Based Device Identification Method for the IoT," in IEEE Internet of Things Journal, vol. 9, no. 23, pp. 23741-23749, 1 Dec.1, 2022, doi: 10.1109/JIOT.2022.3191951.

[7] N. Apthorpe, D. Reisman, and N. Feamster, "A Smart Home is No Castle: Privacy Vulnerabilities of Encrypted IoT Traffic", Tech. Report, ArXiv, [Online]. Available: https://arxiv.org/abs/1705.06805. [Accessed: 15-Juily- 2023]

[8] Sezer, S., Scott-Hayward, S., Chouhan, P. K., Fraser, B., Lake, D., Finnegan, J., Viljoen, N., Miller, M., and Rao, N. (2013). Are we ready for sdn? implementation challenges for software-defined networks. IEEE Communications Magazine, 51(7):36–43, DOI: 10.1109/MCOM.2013.6553676.

[9] Zarezadeh, M., Mala, H. Khajeh, H. Preserving Privacy of Software-Defined Networking Policies by Secure Multi-Party Computation. J. Comput. Sci. Technol. 35, 863–874 (2020).

[10] Q. Chen, S. Shi, X. Li, C. Qian and S. Zhong, SDN-Based Privacy Preserving Cross Domain Routing. in IEEE Transactions on Dependable and Secure Computing, vol. 16, no. 6, pp. 930-943, 1 Nov.-Dec. 2019.

[11] Yu, Z., Zhu, H., Xiao, R., Song, C., Dong, J., Li, H., 2021. Detection and defense against network isolation attacks in software-defined networks. Trans Emerging Tel Tech 32.

[12] "ONOS wiki web page", [Online] Available: https://wiki.onosproject.org/, accessed: 15-07-2023.

[13] "Mininet official web page", [Online] Available: http://mininet.org/, accessed: 15-07-2023.

[14] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A. -R. Sadeghi and S. Tarkoma, "IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT," 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, USA, 2017, pp. 2177-2184, doi: 10.1109/ICDCS.2017.283.

[15] Daniel Campos and TJ OConnor. 2021. Towards Labeling On-Demand IoT Traffic. In Cyber Security Experimentation and Test Workshop (CSET '21). Association for Computing Machinery, New York, NY, USA, 49–57. https://doi.org/10.1145/3474718.3474727

[16] Li, Y., Zhao, Y., Ishak, S. et al. An anonymous data reporting strategy with ensuring incentives for mobile crowd-sensing. J Ambient Intell Human Comput 9, 2093–2107 (2018). https://doi.org/10.1007/s12652-017-0529-x

[17] Sajad Khorsandroo, Adrián Gallego Sánchez, Ali Saman Tosun, JM Arco, Roberto Doriguzzi-Corin, Hybrid SDN evolution: A comprehensive survey of the state-of-the-art, Computer Networks, Volume 192, 2021, 107981, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2021.107981.