Protecting OpenFlow Flow Tables with Intel SGX

Nicolae Paladi
Lund University and RISE
nicolae.paladi@ri.se

Jakob Svenningsson
RISE
jakob.svenningsson@ri.se

Jorge Medina
New Jersey Institute of Technology
jmc237@njit.edu

Patrik Arlos
Blekinge Institute of Technology
patrik.arlos@bth.se

1 INTRODUCTION

Flexible and powerful control over network flows is one of the core advantages of Software-Defined Networking (SDN). Flow rules stored in switch network flow tables contain information on packet processing and routing. Flow rules are stored in memory, in a set of data structure rules, and managed by a classifier in flow tables. Network flows are a valuable asset: they contain information about the traffic patterns between the endpoints, while network tenants may be competing for the (limited) entries in flow tables [3, 4].

Commodity software switches do not currently implement confidentiality or integrity protection of flow tables. An attacker can exploit software vulnerabilities to access the switch host memory and observe or modify installed flows. *Observing* installed flows allows an attacker to learn security-sensitive information: topology, flow patterns between endpoints, and flow priority. *Modifying* installed flows allows an attacker to exploit routing loopholes and avoid certain packet steps - e.g. route around a firewall or prevent mirroring packets to an intrusion detection system.

In this demo we present `OFTinSGX`, an approach to protect the confidentiality and integrity of network flows installed on software switches. Our approach is based on decomposing the network switch to reduce the attack surface by isolating the OpenFlow flow tables and the flow rules from the rest of the code base. `OFTinSGX` allows to prevent attacks on the confidentiality and integrity of flow rules in software switches.

Protecting the security assets of network elements is a topic of active on-going research. Jacquin proposed an architecture that used a hardware root of trust to remotely attest the integrity of virtualization hosts in SDN infrastructure [2]. Furthermore, commodity TEEs were used in case studies on securing network applications [8]. TruSDN is a framework for bootstrapping trust in an SDN infrastructure [5]. While it supports secure provisioning of switches to SGX enclaves, it was implemented with a rudimentary software switch in a SGX emulator and incurrs a significant performance overhead. *Trusted Click* [1] explores the feasibility of performing network processing in SGX enclaves. While none of the approaches above adress the integrity and confidentiality of OpenFlow flow tables, they can be complemented with OFTinSGX to achieve this.

Event Handler Eviction mitigates DoS attacks and overflow of OpenFlow flow table [7]. The mechanism uses two independent modules - learning module and flow checking module - that While the event handler mechanism reduces overflow of flow tables and the risk of DoS attacks, it does not provide security guarantees to the OpenFLow flow tables. `OFTinSGX` protects both the integrity and confidentiality of the OpenFlow flow tables, as well as the forwarding logic and eviction process. TLSonSGX protects the cryptographic material used by OvS instances to protect communication with SDN controllers [6]. The approach can be combined with `OFTinSGX` for wider security guarantees for OpenFlow switches.

The prototype is developed based on a modified version of Open vSwitch (OvS) version 1.11.1 and the RYU controller, version 2.8.

2 ARCHITECTURE

We next describe the design and implementation of `OFTinSGX`, a security mechanism allowing OvS to allocate its OpenFlow flow tables and forwarding logic inside enclave memory. `OFTinSGX` is comprised of four components: the *SGX OpenFlow tables*, the *SGX rule mechanism*, the *SGX Eviction Component* and the *SGX Tables dpif*. We illustrate the interaction of these components in Figure 1.

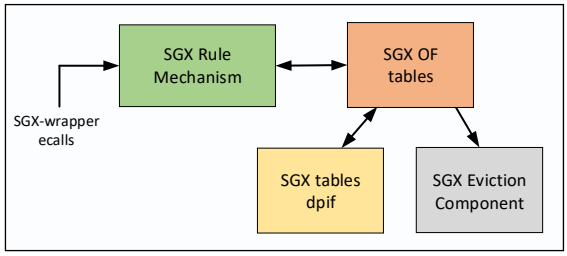

Figure 1: `OFTinSGX` architecture

2.1 `OFTinSGX` Components

SGX OpenFlowtables implement OpenFlow flow tables running in the enclave memory. *SGX OpenFlowtables* host the forwarding logic to match classification rules for unknown incoming flows. Classification rules are translated to instructions installed in the data path.

The *SGX rule mechanism* is an interface that implements the logic to map rules allocated in untrusted memory to classification rules in the enclave for matches in the SGX OpenFlowtables. It ensures that only rules created by an authenticated SDN controller are used to install entries in the data path. The SGX rule mechanism translates the SGX-wrapper calls to direct operations on the OpenFlow flow tables. It is structured as a hash map table (*SGX_cls_table*), enclosing SGX rules (*SGX_cls_rule*). We used the *hmap* OvS library to create and manipulate the SGX hash map table.

The SGX Eviction Component implements the rule eviction process, which based on a given criteria (kept in the OpenFlow flow tables) removes a classification rule from the classifier when a table reaches the maximum number of allowed flows. In addition, the SGX Eviction Component contains the eviction groups (struct `eviction_group`).

Modifying any rule in the classifier requires flow validation or an update of the instantiated flow in the corresponding facet. This is necessary to monitor entries in the data path and collect statistics according to the modified flow. The dpif SGX tables implement struct `table_dpif` to maintain updated information about the existing rules in the OpenFlow flow tables. OvS uses dpif SGX tables when performing update routines to the facets and to maintain an updated datapath cache.

2.2 Workflow Summary

Prior to sending any OpenFlow request to OvS, a remote SDN controller attests the integrity of the enclave where the OpenFlow flow tables are hosted (1). Upon a successful attestation, the controller sends OpenFlow requests to operate the OpenFlow flow tables (2). If the requested operation is a rule addition, a rule is first allocated in untrusted memory and an SGX rule containing a classification rule is allocated in the enclave memory. The SGX Rule Mechanism receives the rule operation requests and handles the requests inside the enclave by calling the services of the respective processing component (3): lookup, eviction or flow revalidation.

The SGX Rule Mechanism contacts the OpenFlow flow tables when the data path does not find instructions to handle an incoming packet. In this scenario, an upcall request for a rule lookup is passed to user space, containing the incoming packet the data path cannot handle(4). If a matching classification rule is available, the SGX Rule mechanism returns the address of the corresponding untrusted rule and the fundamental information, about the matching classification rule, to install an entry in the data path cache for the incoming packet (i.e., a flow and a mask) (5). The entry in the data path is installed once the rule, allocated in untrusted memory, has been identified and the actions to be executed on that packet provided (6). The OpenFlow flow tables deployed in SGX enclaves are protected from malicious tampering (7).

The SDN controller maintains the hash measurement of the enclave and the private key used to signed the enclave, and later uses them to establish an authenticated channel with the enclave. As a result, only an authenticated SDN controller can operate on the OpenFlow flow tables.

3 USE CASES AND DEMONSTRATION

The approach is based on three scenarios introduced below: rule lookup, flow addition, and flow deletion[1]. *Rule lookup* is invoked when the user space contacts OpenFlow flow tables for a matching classification rule. *Flow Addition* is invoked when adding new flows, contains a large number of ported functions interacting with OpenFlow flow tables. *Flow Deletion* flows installed in OpenFlow flow tables are removed using the *del-flows* command in *ovs-ofctl*. This operation is implemented in two steps: *collecting* flows and *deleting* flows from the OpenFlow flow tables.

[1]Demo video: https://vimeo.com/217788815

4 FUTURE WORK

Isolating only the contents of OpenFlow flow tables does not address all security risks, since a classifier keeps only references of the classification rules. Classification rules are allocated in the struct `rule`, which is pointed by a struct `rule_dpif`, in untrusted memory. We limited the implementation scope to porting the contents of the classification rule to enclave memory and leave porting classification rules for future work. Ongoing work is focused on improving the performance of rule addition and eviction and will be reported in an upcoming publication. Future work will extend the security features of the prototype to support network slicing.

5 ACKNOWLEDGEMENTS

This work was financially supported in part by the Swedish Foundation for Strategic Research, grant RIT17-0035 and EU H2020 project ASCLEPIOS, grant 826093.

REFERENCES

[1] Michael Coughlin, Eric Keller, and Eric Wustrow. 2017. Trusted Click: Overcoming Security Issues of NFV in the Cloud. In *SDN-NFVSec '17*. ACM, 31–36.
[2] L. Jacquin, A. L. Shaw, and C. Dalton. 2015. Towards trusted software-defined networks using a hardware-based Integrity Measurement Architecture. In *Proc. 1st IEEE Conf. Network Softwarization (NetSoft'15)*.
[3] Teemu Koponen, Keith Amidon, Peter Balland, Martín Casado, Anupam Chanda, Bryan Fulton, Igor Ganichev, Jesse Gross, Paul Ingram, Ethan J Jackson, et al. 2014. Network Virtualization in Multi-tenant Datacenters.. In *NSDI*, Vol. 14. 203–216.
[4] J. Matias, J. Garay, N. Toledo, J. Unzilla, and E. Jacob. 2015. Toward an SDN-enabled NFV architecture. *IEEE Communications Magazine* 53, 4 (April 2015), 187–193. https://doi.org/10.1109/MCOM.2015.7081093
[5] Nicolae Paladi and Christian Gehrmann. 2016. TruSDN: Bootstrapping Trust in Cloud Network Infrastructure. In *Proc. 12th International Conf. on Security and Privacy in Communication Networks (SecureComm'16)*. Springer.
[6] Nicolae Paladi, Linus Karlsson, and Khalid Elbashir. 2018. Trust Anchors in Software Defined Networks. In *Computer Security*, Javier Lopez, Jianying Zhou, and Miguel Soriano (Eds.). Springer International Publishing, Cham, 485–504.
[7] Ying Qian, Wanqing You, and Kai Qian. 2016. OpenFlow flow table overflow attacks and countermeasures. In *2016 European Conf. on Networks and Communications (EuCNC)*. 205–209. https://doi.org/10.1109/EuCNC.2016.7561033
[8] Ming-Wei Shih, Mohan Kumar, Taesoo Kim, and Ada Gavrilovska. 2016. S-NFV: Securing NFV States by Using SGX. In *Proc. 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization (SDN-NFV Security '16)*. ACM, 45–48.