Synaptic: A formal checker for SDN-based security policies

Nicolas Schnepf, Rémi Badonnel, Abdelkader Lahmadi, Stephan Merz

Synaptic: a Formal Checker for SDN-based Security Policies

Nicolas Schnepf, Rémi Badonnel, Abdelkader Lahmadi, and Stephan Merz
Université de Lorraine, CNRS, Inria, LORIA, Nancy, France

ABSTRACT

Software-defined networking offers new opportunities for protecting end users by designing dynamic security policies. In particular, security chains can be built by combining security functions, such as firewalls, intrusion detection systems and services for preventing data leakage. The configuration of these security functions and their associated policies is based on behavioural models of end-user applications when accessing the network. In this demo, we present our tool Synaptic, a SDN-based framework intended for the formal verification of security policies as well as for automatically generating such policies based on automata learning methods applied on NetFlow records of end-user applications collected at the device level.

I. BACKGROUND

The programmability that characterizes SDN [1] simplifies the specification of network policies by decoupling the control and the data planes. Based on this paradigm, it is possible to enforce chains of security functions, such as described in [2] for protecting end users. Such chains are composed of security functions, such as intrusion detection systems, firewalls or data leakage prevention mechanisms, combined in sequence or in parallel. However, due to their complexity and dynamics, these chains of security functions may give rise to misconfigurations in the network. Formal methods provide techniques that can help the validation of security chains before they are deployed. Formal verification of SDN policies is an important topic in the literature [3], [4]. Current approaches focus mostly on the verification of the data plane, and miss aspects related to the control plane. Nevertheless, the Pyretic language [5], part of the Frenetic framework [6], provides an intuitive way for specifying SDN security policies, and verification facilities are provided for the control plane, through its Kinetic extension [7].

II. SYNAPTIC: GENERATING AND CHECKING POLICIES

Synaptic combines two functionalities. First, it provides techniques for verifying both the control and data planes related to security chains, as described in [8]. These chains correspond to security policies combining security functions using software-defined networking. Second, it includes a component for profiling applications based on logs of their behavior, and for configuring SDN security policies from the inferred profiles. We developed the prototype using Python 2.7, as an extension of Pyretic and Kinetic. The interactions among the different components of Synaptic for the verification of a policy is depicted in Fig. 1. The input received by our checker is a security policy specified in Pyretic together with several logical properties. This input is then translated into either a SMTlib model that can be verified by SMT solving, or into a nuXmv model that can be verified by model checking. If the behavior of this policy is controlled by a Kinetic automaton, Synaptic will use the verification procedure implemented by this framework, then verify the correctness of each data plane policy used by this control automaton. Otherwise, Synaptic will directly verify the data plane policy that it receives. Concerning the possibilities in terms of formal verification, we integrated the following SMT solvers: CVC4 [9] v1.4 and veriT [10] v201506. We also included the nuXmv [11] v1.0.1 model checker, as a backend of our tool.

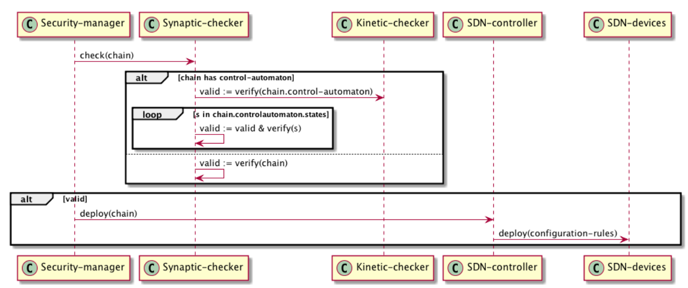

Figure 1. The verification steps of a SDN-based security policy.

While the checker accepts arbitrary (such as hand-written) security policies expressed in Pyretic, we also support their automatic generation based on automata learning [12]. Synaptic includes a component for learning a Markovian model that captures the networking behavior of an application and for deriving a corresponding security policy in Pyretic. This generation process is divided into four phases depicted in Fig. 2: NetFlow acquisition, NetFlow aggregation, automata learning and rule generation. NetFlow records are collected directly on the end-user device by a dedicated probe, such as Flowoid [13] developed in our research team or from available datasets. Collected NetFlows are transmitted to the aggregation module deployed in the cloud: it will aggregate NetFlows based on the responses of `whois` requests for the IP addresses and on the ranges of ports contained in the traces. From the aggregated NetFlows, the automaton learning module will infer a Markov chain summarizing the behavior of the application: to do so, we create a state for each pair of netname/port range and we compute the probability of each

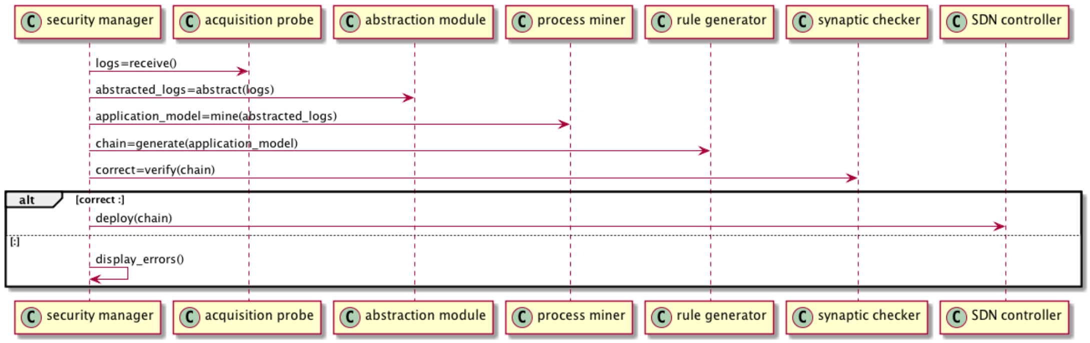

Figure 2. Generation and verification of SDN-based security policies based on application profiling.

outgoing transition. Finally, this automaton is provided to the rule generation module that will generate a Kinetic automaton matching the behavior of the application.

III. The Demonstration

In this demo, we will present both the verification and the process learning features of Synaptic. The verification feature will be illustrated with several examples, including the verification of a Kinetic based control automaton or the verification of a stateless firewall. For each of these examples, we will show (1) the policy that input to Synaptic, (2) the properties to be verified, and (3) the verification options offered by Synaptic. We will exhibit correct and erroneous policies, show the corresponding responses of our checker, and provide timing information. For the automata learning feature, we collected traces of different applications. For each of them, we will show how Synaptic can be used for learning a probabilistic model of their networking behavior, and how this can be used to synthesize and then verify a security chain. Through these different examples, we will show how the complexity of NetFlow records influences the automata produced by Synaptic and the response time of the subsequent verification procedure. We will also show how to configure the aggregation module in order to reduce this complexity as much as possible. Finally, we will illustrate how automata can be stored in external files in order to be processed by other services: an example of such an automaton appears in Fig. 3.

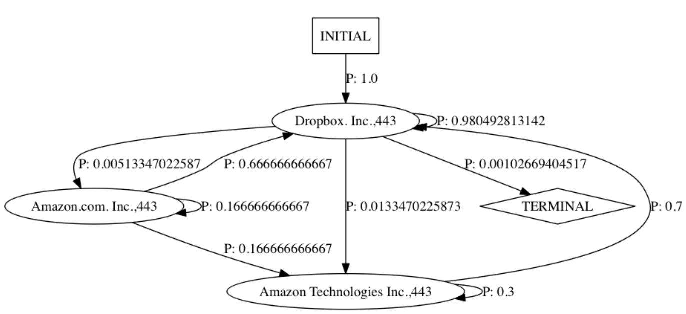

Figure 3. Automaton describing the behavior of the Dropbox application.

References

[1] N. Feamster and H. Kim, “Software-Defined Networks: Improving Network Management with SDN,” in *IEEE Communications Magazine*, February 2013.

[2] G. Hurel, R. Badonnel, A. Lahmadi, and O. Festor, “Towards Cloud Based Compositions of Security Functions for Mobile Devices,” in *IFIP/IEEE International Symposium on Integrated Network Management (IM’15)*, 2015.

[3] E. Al-Shaer and S. Al-Haj, “FlowChecker, Configuration Analysis and Verification of Federated OpenFlow Infrastructures,” in *Proceedings of the 3rd ACM Workshop on Assurable and Usable Security Configuration (CCS’10)*, 2010.

[4] T. Ball, N. Bjørner, A. Gember, S. Itzhaky, A. Karbyshev, M. Sagiv, M. Schapira, and A. Valadarsky, “Vericon: Towards Verifying Controller Programs in Software-Defined Networks,” in *Proc. 35th ACM SIGPLAN Intl. Conf. Programming Language Design (PLDI’14)*, Edinburgh, UK, 2014, pp. 282–293.

[5] N. Foster, M. J. Freedman, A. Guha, R. Harrison, N. P. Kata, C. Monsanto, J. Reich, M. Reitblatt, R. Jennifer, C. Schlesinger, A. Story, and D. Walker, “Languages for Software-Defined Networks,” in *Software Technology Group*, 2016.

[6] N. Foster, M. J. Freedman, R. Harrison, C. Monsanto, and D. Walker, “Frenetic, a Network Programming Language,” in *Proceedings of the 16th ACM SIGPLAN International Conference on Functional Programming (ICFP’11)*, 2011.

[7] H. Kim, J. Reich, A. Gupta, M. Shahbaz, N. Feamster, and R. Clark, “Kinetic: Verifiable Dynamic Network Control,” in *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation (NSDI’15)*, 2015.

[8] N. Schnepf, S. Merz, R. Badonnel, and A. Lahmadi, “Automated verification of security chains in software-defined networks with Synaptic,” in *Proceedings of the 3rd IEEE Conference on Network Softwarization (NetSoft’17)*, 2017.

[9] C. Barrett, C. L. Conway, M. Deters, L. Hadarean, D. Jovanovic, T. King, A. Reynolds, and C. Tinelli, “CVC4,” in *Proc. 23rd Intl. Conf. Computer Aided Verification (CAV 2011)*, Snowbird, UT, USA, 2011, pp. 171–177.

[10] T. Bouton, D. C. B. D. Oliveira, D. Déharbe, and P. Fontaine, “veriT: An Open, Trustable and Efficient SMT-Solver,” in *Proc. 22nd International Conference on Automated Deduction (CADE-22)*, Montreal, Canada, 2009, pp. 151–156.

[11] R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, A. Mariotti, A. Micheli, S. Mover, M. Roveri, and S. Tonetta, “The nuXmv symbolic model checker,” in *Proc. 26th Intl. Conf. Computer Aided Verification (CAV 2014)*, Vienna, Austria, 2014, pp. 334–342.

[12] N. Schnepf, S. Merz, R. Badonnel, and A. Lahmadi, “Towards generation of sdn policies for protecting android environments based on automata learning,” in *Proceedings of the 16th Network Operations and Management Symposium (IEEE/IFIP NOMS’18)*, 2018.

[13] A. Lahmadi, F. Beck, E. Finickel, and O. Festor, “A platform for the analysis and visualization of network flow data of android environments,” IFIP/IEEE International Symposium on Integrated Network Management (IM), May 2015, poster. [Online]. Available: https://hal.inria.fr/hal-01242911