The Vision of Self-aware Reordering of Security Network Function Chains

Vision Paper

Lukas Iffländer
University of Würzburg
Germany
lukas.ifflaender@uni-wuerzburg.de

Jürgen Walter
University of Würzburg
Germany
juergen.walter@uni-wuerzburg.de

Simon Eismann
University of Würzburg
Germany
simon.eismann@uni-wuerzburg.de

Samuel Kounev
University of Würzburg
Germany
samuel.kounev@uni-wuerzburg.de

ABSTRACT

Services provided online are subject to various types of attacks. Security appliances can be chained to protect a system against multiple types of network attacks. The sequence of appliances has a significant impact on the efficiency of the whole chain. While the operation of security appliance chains is currently based on a static order, traffic-aware reordering of security appliances may significantly improve efficiency and accuracy. In this paper, we present the vision of a self-aware system to automatically reorder security appliances according to incoming traffic. To achieve this, we propose to apply a model-based learning, reasoning, and acting (LRA-M) loop. To this end, we describe a corresponding system architecture and explain its building blocks.

CCS CONCEPTS

• **Networks** → **Security protocols**; *Middle boxes / network appliances*; *Deep packet inspection*; *Control path algorithms*; *Network performance modeling*; *Denial-of-service attacks*; *Firewalls*; • **Hardware** → *Modeling and parameter extraction*; • **Software and its engineering** → **Software system models**; • **Security and privacy** → *Intrusion/anomaly detection and malware mitigation*; Virtualization and security;

ACM Reference Format:
Lukas Iffländer, Jürgen Walter, Simon Eismann, and Samuel Kounev. 2018. The Vision of Self-aware Reordering of Security Network Function Chains: Vision Paper. In *Proceedings of ACM/SPEC International Conference on Performance Engineering (ICPE'18 Companion).* ACM, New York, NY, USA, 4 pages. https://doi.org/https://doi.org/10.1145/3185768.3186309

ICPE'18 Companion, April 9–13, 2018, Berlin, Germany

ACM ISBN ISBN 978-1-4503-5629-9/18/04…$15.00
https://doi.org/https://doi.org/10.1145/3185768.3186309

1 INTRODUCTION

Today's network attacks are based on huge bot networks. Their attacking power raises as the number of online devices rapidly grows in times of Internet of Things (IoT). The opportunity to throw additional resources to fight attacks is limited since Moore's Law (promising doubled resources every two years) ended [7]. Moreover, booking additional resources on demand is very costly also considering that the owners of bot networks do not have to pay for their attack resources.

IT systems providing services via a network can be attacked in various ways. Common attack types include, for example, "HTTP Flood" or "ping of Death". For each type of network attack, there are dedicated Security Appliances (SAs) to defend the system. For example, a firewall fights HTTP flooding attacks and excessive pinging can be antagonized by a DDoS Protection System (DPS). To protect a system against a set of attack types, multiple SAs have to be chained in a Security Service Function Chain (SSFC).

For most systems, there is a direct correlation between consumed resources and the number of processed packages. In contrast, SAs (and therefore SSFCs) stand out, as they (i) behave differently under various traffic conditions (package characteristics and overload [8]) and (ii) drop packets deemed as malicious causing lower load on subsequent SAs.

Security can be taken out of service by attacks during suboptimal configuration long before available resources are purposefully utilized. We illustrate this by a chain of two SAs, a firewall, and a DPS. We assume that each SA can handle a throughput of 100 Mbps while accurately filtering malicious packages. The described system is now attacked by a Distributed Denial of Service Attack (DDoS) attack on a port not filtered by the firewall. The traffic rate is 1 Gbps of which 90% of the traffic is malicious. The throughput and required resources of the SSFC for this attack highly depend on the order of SAs. If we put the firewall first, there would be no filtering. Consequently, ten instances of each SA type would be required. Otherwise, if we place the DPS in front, the first layer would still require ten instances but 90% of the traffic would be dropped before reaching the second layer. Instead of ten instances, the remaining 10% percent of traffic can be handled by a single firewall instance. The attack could be survived using eleven instances in total, which means a reduction of required resources by 45% compared to the

firewall in front. A second attack could be on a port blocked by the firewall. Then the placement of the firewall in front of the DPS inverts the discrepancy in efficiency. While dropping rates might differ for different SAs and traffic compositions, our illustrative example demonstrates potential efficiency gains when tailoring the order to the incoming traffic.

While a traffic-aware reordering would provide benefits, today's operation of SSFC is based on a static order of SAs. To remedy this deficiency, we propose a self-aware approach to automatically reorder SAs based on incoming traffic. At this, SAs report detected attacks to a central instance, the Function Chaining Controller (FCC). We model the behavior of different SA types based on measurements as well as different traffic types for attacks, benign workloads, and combinations of them. Based on these models, we then infer a configuration tailored to incoming traffic which can be instantiated by dynamically reordering the SSFC.

The benefit of our approach is to improve the efficiency of the traffic processing inside the SSFC. This results in an increased throughput to resources ratio. Since overload situations are avoided or at least mitigated, we additionally expect a reduction of the false-positive and false-negative rate. Finally, the chances of the security system being permanently disabled due to an attack are reduced.

The remainder of this paper is structured as follows: Based on the problem statement postulated in Section 2, we present our vision of self-aware reordering of Service Function Chains (SFCs) including major building blocks in Section 3. Section 4 highlights deficiencies in the state-of-the-art. Finally, we conclude the paper and lay out our next research steps in Section 5.

2 PROBLEM STATEMENT

The order of an SSFC has a significant impact on its efficiency and accuracy. The optimal order depends on the incoming traffic and cannot, therefore, be static. However, the order of existing SSFCs is determined once prior to initialization. While virtualized SA would allow for quick traffic and attack analysis *and* instant reordering at insignificant cost using Software Defined Networks (SDNs), SFCs mechanisms do not take advantage of the combination of these two abilities yet.

3 RESEARCH VISION

The high-level objective is to automatically reorder SAs inside the SSFC to improve its efficiency and accuracy. Basically, we tailor the order of SAs to the characteristics of arriving network packages. A tailored configuration provides (i) a more efficient processing of traffic inside the security system, (ii) a reduction of false positives (accidental dropping) and false negatives (accidental forwarding of malicious packages), and (iii) increased survivability during attacks. In general, we expect insignificant overhead since we efficiently gather information using push notifications, reasoning happens on a small set of components, and acting can be considered instantly based on existing SDN solutions. If required, preprocessing may further reduce overhead at runtime. The main benefit of our approach is that it can optimally cope with arbitrary attack mixes. Assuming an SDN-based SFC is already in place, self-aware reordering adds no extra vulnerabilities. Attacks aiming at triggering oscillating

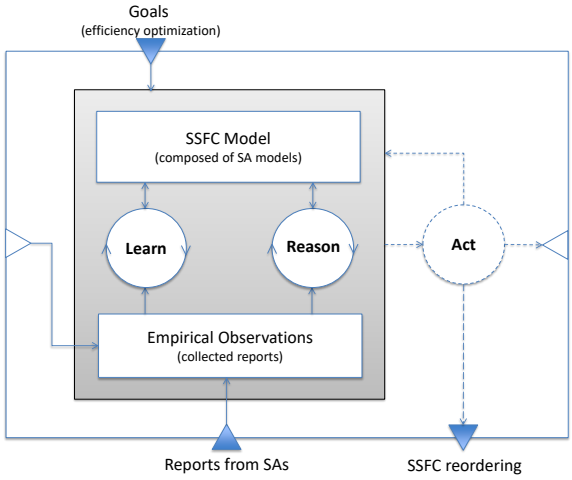

Figure 1: Structure of Self-aware Security Function Chain Reordering System

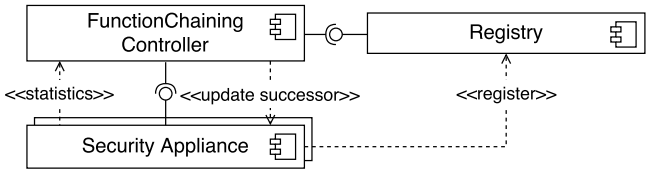

Figure 2: Architecture for Self-aware reordering of SSFCs

behavior can be neglected due to insignificant SDN adaptation costs.

3.1 Architecture

To tailor the sequence of SAs to incoming traffic, it is necessary to determine the most efficient order for a certain traffic mix. Therefore, we require a performance model of each SA inside the SSFC as well as a mechanism to automatically combine them to predict the performance of the entire SSFC. Self-awareness in computing systems is achieved by implementing a model-based learning, reasoning, and acting loop (LRA-M loop) [2] as shown in Figure 1.

Figure 2 illustrates the coarse-grained architecture for self-aware function chaining. It consists of a `Function Chaining Controller`, a central `Registry`, and a set of `Security Appliances`. New SAs are subscribed at the registry and report their statistics to the FCC. The registry holds the set of connected SAs, and provides this information to the FCC. The FCC receives the statistics from the SAs and can reorder them by updating successors of selected SAs. To ensure scalability, we suggest detecting attacks directly on the SAs. They then forward gathered statistics (e.g., frequencies) about the attacks to the central FCC where they are evaluated. The controller decides on the optimal configuration and if it is not the current configuration issues the reconfiguration of the SSFC. For the communication, we propose a push-based approach according to the observer design pattern.

To ensure a broad applicability, we formulate requirements for self-aware function chaining of SA: (i) the system must be able to handle an arbitrary number of SAs, (ii) as well as arbitrary types of

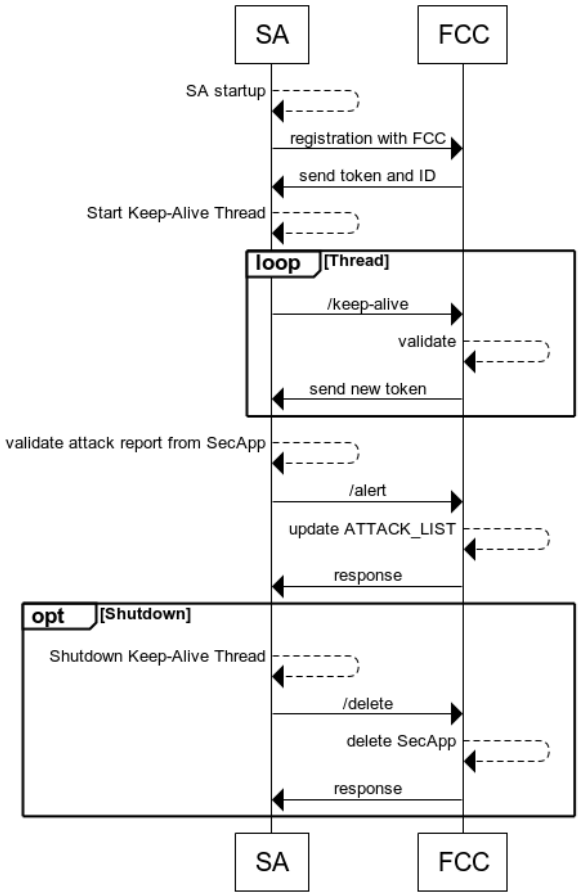

Figure 3: Communication life-cycle between the SA and the FCC

SAs, (iii) it must capable of working on all ISO/OSI layers to be able to model attacks and defense for different levels (frames, packets, and segments).

3.2 Attack Detection

To manage the communication with the central `Function Chaining Controller` instance, we propose to deploy a wrapper software aside from the SAs. All active SAs are registered at the central controller at the beginning of their life-cycle and de-registered at their shutdown. The SAs detect attacks while handling them. In regular intervals, the SAs send these statistics as push messages to the central controller. These messages contain all relevant information on the detected attacks like source and destination address and port, the attack rate and the attack type. The central controller aggregates the incoming messages deriving the composition of the incoming traffic. The LEARN process over the life-cycle of an SA wrapper is depicted in Figure 3.

3.3 Deriving Tailored Configurations for Incoming Traffic

To reason about which ordering of the SSFC will be suggested it is necessary to model the workload classes, considering benign traffic as well as attacks. To this end, the model of the arrival rate has to consider the content (e.g., relevant for Deep Packet Inspection (DPI) based Intrusion Detection Systems (IDSs)) and the composition of the different workload classes.

To model the behavior of SAs, we require a common model for every SA inherited by specialized models for the SA types. It is necessary to model parametric dependencies between the incoming traffic and the SA's behavior. Relevant behavior to be modeled includes the drop rate, overload behavior (especially relevant for IDS). as well as the detection accuracy (rate of false-positive and false-negative results).

The models can be parameterized based on measurements under a variety of different workloads. For the parametrization, the workload models can be re-used.

Based on the requirements, we propose to apply architectural performance models to model the SAs. Architectural models capture the semantics, allowing for a comprehensible view on the SAs, in contrast to low-level stochastic formalisms. Figure 4 illustrates our proposed approach to model security appliances. Each security appliance is modeled as a software component. Based on the distribution of the input traffic of a security appliance, the corresponding output traffic can be derived. We define the distribution of the input/output traffic as $P_{in/out}(t_i)$ with $i \in [1, n]$ for n different types of traffic. Exemplary, a security appliance which drops all packets of the traffic type k the output traffic can be described by:

$$P_{out}(t_i) = \begin{cases} P_{in}(t_i)/(1 - P_{in}(t_k)) & \text{for } i \neq k \\ 0 & \text{for } i = k \end{cases}. \quad (1)$$

Based on the capability to investigate a single configuration, we have to decide which configurations to analyze. For small numbers of SA types, each permutation can be tested. For example, a chain of four SA types results in the analysis of 24 permutations. In later stages, we plan to reduce the number of analyzed configurations using greedy approaches and/or machine learning. Finally, the controller proposes the configuration with the highest throughput of all analyzed configurations.

3.4 Enacting the Reordering of Security Appliances

The reordering happens by updating the successor of the respective SAs, as illustrated in Figure 2. To send the update, we apply SDN features. The FCC sends ordering in form of coarse-grained flow specifications to the SDN controller. Then the SDN controller automatically executes them on the network devices, as shown in Figure 5. Existing SDN controllers provide the capability to deduce the detailed flow specifications.

4 RELATED WORK

Existing work on SDNs focuses on internals and the connections between the used devices as well as their placement inside the network, e.g., [1, 3, 4]. None of these works targets the performance of security appliances with their characteristics.

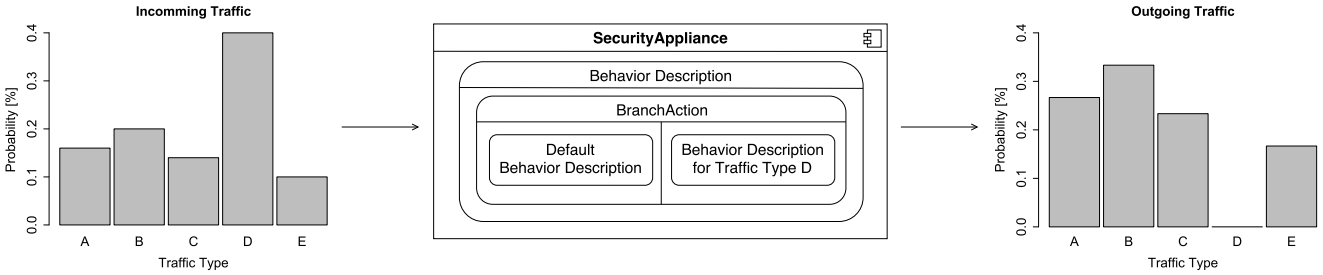

Figure 4: Modeling of Filtering Behavior for Security Appliances

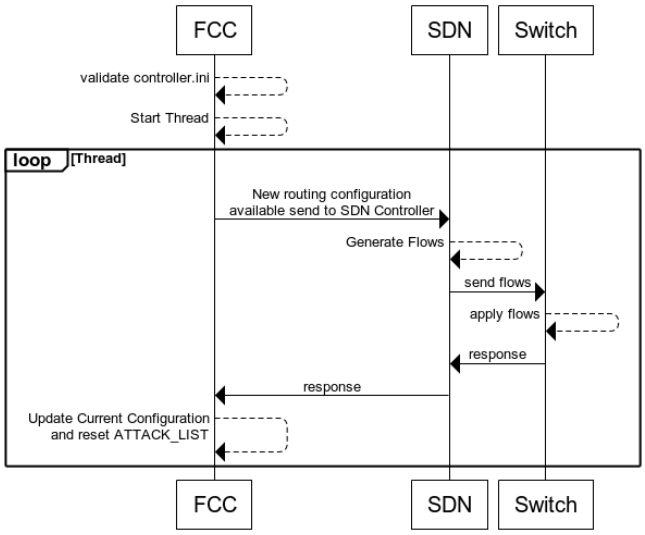

Figure 5: Execution of a new configuration by communication between the FCC and the SDN-enabled network

.

Fischer [1] describes an approach to automated model building for function chains using stochastic Petri nets. It consists of two building blocks, a single Virtualized Network Function (VNF) and a physical host. His description of the VNF model lacks essential features required for our intended self-aware reordering. Especially, VNF with non-forwarding behavior (e.g., firewalls dropping packets) cannot yet be modeled. Further, he does not differentiate different types of traffic. Ocampo et al. [4] propose an optimized composition of an SFC. However, their approach does not support characteristics (required computing power, drop rate, throughput) of the VNFs depending on the incoming traffic type. Mechtri et al. [3] present a scalable algorithm for the placement of service function chains. Concerning our vision, their approach is limited to the optimal placement assuming a preconfigured ordering of the network functions.

To implement our vision, we can refer to initial previous work on SDN performance, automated model building, and self-aware system adaptation. We applied Queueing Petri Nets (QPNs) to model and predict SDN performance [5] and proposed an approach to creating a network performance model based on network dumps [6]. Further, we proposed a generic framework for the automated extraction of architectural performance models from application performance monitoring data [9].

5 CONCLUDING REMARKS AND NEXT STEPS

The operation of today's SSFCs is based on a static order of SAs. This can be highly inefficient as the processing of packages depends on the characteristics of the incoming traffic. An unfitting configuration may cause overload situations that may even infer the detection of malicious packages. In this vision paper, we describe how to automatically reorder the system on changing workload and attacks in order to improve its efficiency and accuracy. We expect this vision to affect security mechanisms for all services provided online. Towards accomplishing, we propose a self-aware approach, describe its architecture, and discuss the implementation of major building blocks. As next steps, we will implement the software to LEARN, REASON, and ACT. After evaluating the efficiency improvements of an initial prototype, we plan to investigate how to further improve reasoning.

6 ACKNOWLEDGMENTS

This work was co-funded by the German Research Foundation (DFG) under grant No. (KO 3445/15-1) and grant No. (KO 3445/16-1).

REFERENCES

[1] A. Fischer. Performance evaluation for service function chains through automated model building. In *11th EAI International Conference on Performance Evaluation Methodologies and Tools (ValueTools2017)*, 2017.

[2] S. Kounev, P. Lewis, K. L. Bellman, N. Bencomo, J. Camara, A. Diaconescu, L. Esterle, K. Geihs, H. Giese, S. Götz, et al. The notion of self-aware computing. In *Self-Aware Computing Systems*, pages 3–16. Springer, 2017.

[3] M. Mechtri, C. Ghribi, and D. Zeghlache. A scalable algorithm for the placement of service function chains. *IEEE Transactions on Network and Service Management*, 13(3):533–546, Sept 2016.

[4] A. F. Ocampo, J. Gil-Herrera, P. H. Isolani, M. C. Neves, J. F. Botero, S. Latré, L. Zambenedetti, M. P. Barcellos, and L. P. Gaspary. *Optimal Service Function Chain Composition in Network Functions Virtualization*, pages 62–76. Springer International Publishing, Cham, 2017.

[5] P. Rygielski, M. Seliuchenko, and S. Kounev. Modeling and Prediction of Software-Defined Networks Performance using Queueing Petri Nets. In *Proceedings of the Ninth International Conference on Simulation Tools and Techniques (SIMUTools 2016)*, pages 66–75, August 2016.

[6] P. Rygielski, V. Simko, F. Sittner, D. Aschenbrenner, S. Kounev, and K. Schilling. Automated Extraction of Network Traffic Models Suitable for Performance Simulation. In *Proceedings of the 7th ACM/SPEC International Conference on Performance Engineering (ICPE 2016)*, pages 27–35. ACM, March 2016.

[7] T. N. Theis and H.-S. P. Wong. The end of moore's law: A new beginning for information technology. *Computing in Science & Engineering*, 19(2):41–50, 2017.

[8] S. Thian-ngam and M. Lertwatechakul. False positive decrement for snort intrusion detection. *Engineering and Applied Science Research*, 36(3):251–259, 2012.

[9] J. Walter, C. Stier, H. Koziolek, and S. Kounev. An Expandable Extraction Framework for Architectural Performance Models. In *Proceedings of the 3rd International Workshop on Quality-Aware DevOps (QUDOS'17)*. ACM, April 2017.