SICS: Secure and Dynamic Middlebox Outsourcing

Huazhe Wang, Xin Li and Chen Qian
Department of Computer Engineering, University of California Santa Cruz

***Abstract*—Outsourcing middleboxes brings threats to the enterprise's private information including the traffic and rules of middleboxes. We present a secure and dynamic middlebox outsourcing framework SICS, short for Secure In-Cloud Service. SICS encrypts each packet header and uses a label for in-cloud rule matching, which enables the cloud to perform its functionalities correctly with minimum header information leakage.**

I. Introduction

Network functions, also known as middleboxes, are vital parts of modern networks. Middleboxes are configured with processing rules, each of which may include the matching field that specifies a set of packets and the action field that specifies the corresponding processing action applied to these packets. Middlebox policies typically require packets to go through a sequence of middleboxes , which is called a *service function chain*. There is an increasing trend that enterprises outsource their middleboxes to a cloud for lower cost and ease of management. Existing middlebox outsourcing solutions [2] require enterprises to provide the cloud with the processing rules for outsourced middleboxes and to redirect their traffic to the cloud in plaintext. The resulting privacy issues impede the broad adoption of middlebox outsourcing.

In this poster, we design and implement a middlebox outsourcing scheme SICS, short for Secure In-Cloud Service chaining. SICS protects the private information of packet headers and rules by only sending packets with encrypted headers into the cloud. Since encrypted headers cannot be used for forwarding and rule matching, SICS utilizes the concept and techniques of Atomic Predicate (AP) [4][3] to assign a label to each encrypted packet. Each label uniquely identifies the forwarding and rule-matching behavior of the packet, revealing minimal information of the in-cloud middlebox processing chain. We first logically group packets with the same middlebox processing chain and actions into equivalence classes and thus we eliminate the need to assign a unique label to each single flow. Second, building on domain expertise and predefined new headers, we propose a label-to-label replacement scheme. The new labels correspond to the new modified headers and are used for subsequent processing.

II. Design

A. The SICS Outsourcing Architecture

SICS contains two parties: an enterprise and a third party cloud that provides in-cloud middlebox processing. The enterprise has a *gateway* that connects each router or switch to the external network. All incoming packets to the enterprise will be forwarded to the gateway. The gateway encrypts the packet headers and payloads and sends the packets to the cloud for middlebox processing. The in-cloud middleboxes process

978-1-5090-6501-1/17/$31.00 ©2017 IEEE

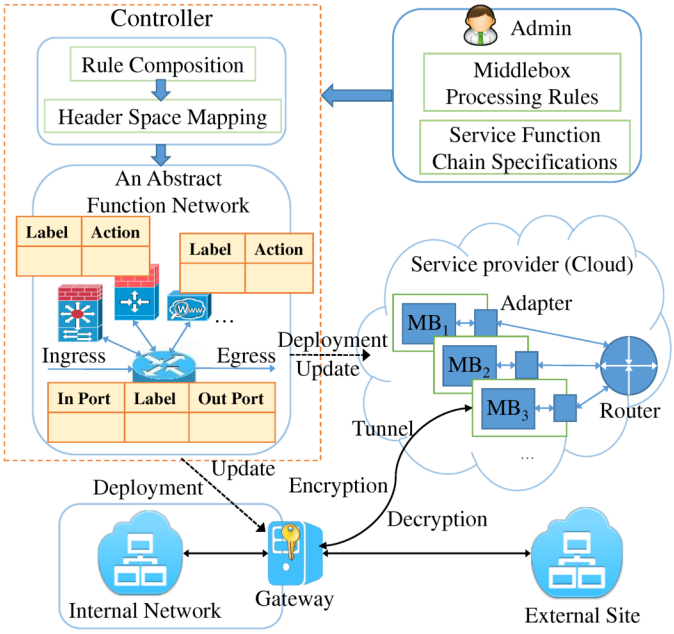

Fig. 1: The system model of SICS

packets following the service function chains and then the cloud transmits the packets back to the enterprise. The gateway decrypts the packets and sends them to the internal network.

The key challenge in this architecture is how the in-cloud middleboxes correctly match packets to rules given that the packet headers are encrypted. To enable correct rule-matching with encrypted headers, SICS assigns each packet a label. The label represents all behavior of the packet in the cloud, including to which middleboxes the packet should be forwarded and in which order, as well as which rules the packet should match at a middlebox.

B. Design Framework

Fig. 1 shows the design framework of SICS. Modules in the dashed box are rule preprocessing procedures running on a controller in the enterprise network. The controller also installs configurations and updates onto the gateway and in-cloud middleboxes. The rule preprocessing procedures take the service function chain specifications and the processing rules at each middlebox as their input. Service function chains specify what traffic should be handled in which way. The output is an *abstract function network* which consists of all middleboxes required in the service function chains and a *virtual switch* that connects all middleboxes. At each middlebox, there is a rule table identifying the action applied to each packet based on the label. The virtual switch is equipped with a forwarding table that determines the next hop of a packet based on its label and input port. For example, if the label indicates a service chain

IDS $\rightarrow$ *Proxy* and the input port is from *IDS*, then the next hop should be *Proxy*. The abstract function network should ensure that packets are processed by required middleboxes in a correct sequence. The enterprise gateway hosts a tunnel between itself and the cloud, performs the header manipulation and encryption/decryption.

C. Enterprise Modules of SICS

To enable secure middlebox outsourcing, SICS dynamically maps the header spaces specified by the middlebox processing policies to labels at the enterprise gateway. We present the design of three key modules at the SICS enterprise side.

Rule composition: This module takes the service function chain requirements and the middlebox processing rules as its input. It combines the different requirements and determines the service function chains for each set of packets. Based on the service function chains, it generates the forwarding rules at the virtual switch. Then it uses a rule aggregation algorithm to combine rules at the virtual switch and the middleboxes to a list of *predicates*, each of which specifies a set of packet headers that has identical behaviors on a middlebox or the virtual switch.

Header space mapping: This module takes the list of predicates from the rule composition module as its input. It calculates the equivalence classes using the predicates. An equivalence class specifies a set of packet headers that exhibits identical cloud-wide behaviors. SICS uses a label to represent an equivalence class.

Packet classification: This module classifies packets into equivalence classes based on their headers. When an equivalence class is found for a given packet, the gateway assigns the label corresponding to the equivalence class to the packet.

D. In-Cloud Modules of SICS

Static middleboxes. If a packet is only processed by static middleboxes, we can use a single label to identify its cloud-wide behavior. At a middlebox, a label is saved as a key-value pair in a hash table. The key represents a label and its value is the action of packets with that label.

Header transformers. When a middlebox modifies packet headers, the behavior of a packet in downstream boxes is determined by its new header. In label-matching, the subsequent packet behavior should be determined by the new label applied to the new header. To address the above problem, we design a label-to-label replacement strategy. We define a property of per-field equivalence classes in which an equivalence class can be identified by a set of per-field equivalence classes. Based on the property of the per-field equivalence class, a label replacement table is built for each equivalence class when one header field is modified. All entries in the label replacement table are generated by the enterprise controller and sent to the cloud at either setup or update time.

E. Update operations

A rule insertion or deletion can be converted to predicate changes [4]. If there exist predicate changes after the rule updates, SICS performs the following methods to update both the enterprise side and the in-cloud boxes.

Update at the enterprise side. SICS starts by updating the packet classifier at the gateway. When a new predicate is added, SICS adds the new predicate to the bottom of the packet classifier. Updates to the classifier can be executed very fast. In our experiments, the average cost of adding/deleting a predicate is less than 0.5 ms.

Update in the Cloud. In SICS, a rule update in the cloud is the updating of the rule tables (hash tables) at each middlebox and the virtual switch. The forwarding table of the virtual switch is partitioned into several sub-tables which are updated independently. When a new equivalence class is added into the representation list of a predicate, its label-action pair is inserted into the rule table of the in-cloud box that produced the predicate. In contrast, a label-action pair is removed from the rule table when the corresponding equivalence class is deleted from the representation list of the predicate.

III. Implementation and Evaluation

We have built a SICS prototype using middleboxes running in the Amazon Virtual Private Cloud (VPC) and a gateway running on a general purpose desktop computer with quadore@3.2G and 16GB memory. The gateway redirects traffic from another machine using the same model. On the cloud side, the abstract function network can be easily converted into a practical deployment within the Amazon VPC. We implemented middleboxes using Click and rule tables using the Cuckoo hash table [1]. To enable in-cloud middlebox chaining, we added an adapter layer which holds a sub-forwarding table from the virtual switch at each middlebox instance. Based on their labels, the adapter decapsulates incoming packets for current processing and encapsulates outgoing packets with the addresses of next required middleboxes.

Evaluation results show that SICS achieves 20% higher throughput, 5 times faster construction and fast rule update, and lower resource overhead at the enterprise and in the cloud when compared to existing solutions.

IV. Conclusion

SICS is a middlebox outsourcing framework that protects the private information of packet headers and middlebox rules. Compared to existing methods, SICS has several unique advantages including a stronger security guarantee, high-throughput processing, and support of quick update. SICS assigns each packet a label identifying its matching behavior in a service chain and all middlebox processing in the cloud is based on labels. We use a prototype implementation and evaluation on VPC and local computers to show the feasibility, high performance, and efficiency of SICS.

References

[1] Pagh, R., and Rodler, F. F. Cuckoo hashing. In *Proc. of ESA* (2001).

[2] Sherry, J., Hasan, S., Scott, C., Krishnamurthy, A., Ratnasamy, S., and Sekar, V. Making middleboxes someone else's problem: network processing as a cloud service. In *Proc. of ACM SIGCOMM* (2012).

[3] Wang, H., Qian, C., Yu, Y., Yang, H., and Lam, S. S. Practical network-wide packet behavior identification by ap classifier. In *Proc. of ACM CoNEXT* (2015).

[4] Yang, H., and Lam, S. S. Real-time verification of network properties using atomic predicates. In *Proc. of IEEE ICNP* (2013).