Towards an Open Middlebox Platform for Modular Function Composition

Shinae Woo*student*, Keon Jang*, Dongsu Han, and KyoungSoo Park
*KAIST, *Microsoft Research*
shinae@ndsl.kaist.edu, keonjang@gmail.com, {dongsuh, kyoungsoo}@ee.kaist.ac.kr

With the growing diversity of middlebox features, software-based middleboxes have become increasingly popular over traditional hardware-based middleboxes [1, 2]. The movement towards software-based implementation enabled middleboxes to consolidate multiple functionalities into a single box [7]. Such consolidation, however, introduces fundamental challenges in developing high-performance middleboxes.

First, existing networking stacks for end hosts, lack support for extracting flow-level information, which is often required for middlebox applications. Also, different applications may be interested in different pieces of information. For example, a WAN accelerator that performs redundancy elimination is interested in the payload, whereas a firewall that detects a port scan may be interested in the first packet of each flow (e.g., a SYN packet). However, no existing framework effectively handles issues in multiplexing applications, making consolidation more challenging.

Second, middlebox applications often take actions in response to a diverse set of transport- and application-flow level "events". For example, an application protocol parser may only be interested in established TCP flows while application-specific events can be of interest to others. A Web firewall may particularly be interested in HTTP requests, whereas an anti-virus scanner may be interested in various forms of file transfers (e.g., e-mail, Skype, and Dropbox). Nevertheless, no existing framework allows setting up such "flexible" events.

Finally, existing platforms [3, 4, 7] do not provide a programmable processing pipeline. A programmable pipeline must 1) be able to dynamically direct the processing of a packet or flow from one module to another and 2) efficiently multiplex the inputs and outputs of these modules. While xOMB takes a first step, its usage model is limited to protocol acceleration [3]. To support multi-protocol acceleration, xOMB requires setting up a chain of machines. SDN-based approaches [4] provide a programmable control plane for composing a "pipeline" over a set of middleboxes, but do not easily extend to consolidated architectures. Finally, CoMB demonstrates the benefits of a consolidated middlebox, but it lacks programmability to support building modular applications and their composition.

This work aims to provide a consolidated middlebox development platform that addresses these issues. The framework supports modular design and composition of middlebox applications as shown in Figure 1 with three key building blocks.

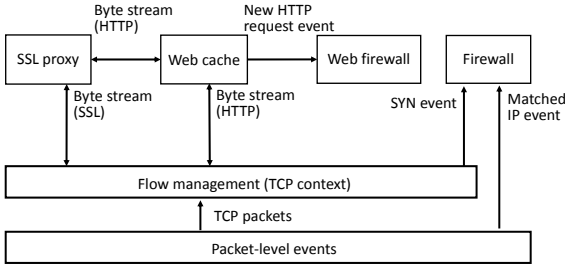

Figure 1: Modular composition of applications

Flow management: Our framework provides a unified flow management that exposes state information for TCP and application-level contexts. Our TCP module, based on user-level TCP [6], exposes its state information through a well-defined set of APIs. Our API enables observation of TCP contexts and byte streams, including passive monitoring of TCP sessions. Based on the API use, our TCP stack selectively enables features at runtime for performance. Similarly, we allow modules to create user-defined flows (e.g., HTTP sessions) and share their state information with others.

User-defined events: Our platform allows users to define events related to TCP (or user) context changes (*e.g.*, retransmission) or TCP (or user-defined) byte streams (*e.g.*, byte streams starting with "HTTP 1.1") and associate callbacks to these events. User-defined events and callbacks allow efficient extension of the basic TCP and application functionality. For example, malicious retransmission can be easily detected by adding a callback for retransmission without building a complex TCP module from scratch [5]. Also, application protocols can generate file transfer events that may be consumed by an anti-virus module.

Module composition: Modules in our system can either be composed statically or dynamically based on module's input/output types. For example, when a module exports an "HTTP session" and another one imports it, the system automatically links them together.

In sum, we propose an extensible platform for modular middlebox development. While the implementation and evaluation is in progress, we believe it can effectively address key issues in middlebox development.

References

[1] Brocade Vyatta. `http://www.brocade.com/index.page`.

[2] SilverPeak VX Software. `http://www.silver-peak.com/products-solutions/wan-optimization/vx-software`.

[3] J. W. Anderson, R. Braud, R. Kapoor, G. Porter, and A. Vahdat. xOMB: Extensible Open Middleboxes with Commodity Servers. In *Proceedings of the eighth ACM/IEEE symposium on Architectures for networking and communications systems (ANCS)*, 2012.

[4] A. Gember, A. Akella, A. Anand, T. Benson, and R. Grandl. Stratos: Virtual Middleboxes as First-Class Entities. Technical Report Technical Report TR1771, University of Wisconsin-Madison, 2012.

[5] Y. Go, J. Won, D. F. Kune, E. Jeong, Y. Kim, and K. Park. Gaining Control of Cellular Traffic Accounting by Spurious TCP Retransmission. In *Proceeding of the Network and Distributed System Security Symposium (NDSS)*, 2014.

[6] E. Jeong, S. Woo, M. Jamshed, H. Jeong, S. Ihm, D. Han, and K. Park. mTCP: A Highly scalable user-level TCP stack for multicore systems. In *Proceedings of the USENIX conference on Networked systems design and implementation (NSDI)*, 2014.

[7] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi. Design and Implementation of a Consolidated Middlebox Architecture. In *Proceedings of the USENIX conference on Networked systems design and implementation (NSDI)*, 2012.