SymNet: Static Checking for Stateful Networks

Matei Popovici and Costin Raiciu
{matei.popovici, costin.raiciu}@cs.pub.ro
University Politehnica of Bucharest

1. INTRODUCTION AND MOTIVATION

Static checking of network configurations is a promising approach to help understand whether a network is configured properly. Techniques such as Anteater [2] and Header Space Analysis [1] are great as they allow us to check basic network properties e.g. reachability and loop-freeness. Unfortunately these tools assume stateless middleboxes and fail short in today's Internet where stateful middlebox processing abounds. Furthermore, the set of questions they answer are limited to packets; with stateful processing everywhere, we must also be able to answer questions about ***packet flows***.

Consider the simple example shown in Figure 1: the client's packets are NATed by box A and then reach box B that is running a traffic scrubber deployed by the destination (to clean flows of malicious payload). After scrubbing, the traffic is proxied to its real destination. En-route an ISP is required by law to log traffic, so it deploys its logger box C. We would like to know:

1. Can SRC open a TCP flow to DST?
2. DST wants to accept only scrubbed traffic. Is it safe to accept all the packets received from B?
3. Can C log an end-to-end view of the traffic?

2. SYMBOLIC NETWORK ANALYSIS

We propose to use symbolic execution—a technique prevalent in compilers—to check network properties more general than basic reachability. The key idea is to track the possible values for specified fields in the packet as it travels through a network. Each middlebox or router will impose constraints on certain fields of the packet via forwarding actions, packet modifications and filtering. The symbolic information makes it possible to answer many questions. We now briefly describe our technique.

Packets are modelled as sets of variables that represent header fields and even payload. Each variable can be free or bound to a symbolic expression. A symbolic expression is either another variable, set of variables, or set of possible values. For instance, a variable binding for the SYN Flag could specify that it is set or cleared.

The topology is given as graph where vertices are networking boxes (middleboxes or routers) and edges are links. As in HSA [1], we model boxes as transformations on (symbolic) packets arriving on ports. Transformations are specified by rules; a rule describes where it applies (packets and ports) and what the output is (a list of (packet, output port)).

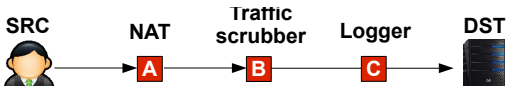

Figure 1: Example middlebox configuration

Modelling flow state is achieved using work variables added to packets. For instance, stateful firewalls add a new, firewall-dependent, variable to the packet to represent the firewall state associated to the packet's flow. At the receiver, the variable is copied into the return packets, allowing the firewall to recognize them as being part of an allowed outgoing connection. NATs and tunnels are modelled similarly.

Symbolic analysis verifies network behaviour between a source s and destination port d as follows: (i) take a generic symbolic packet at the source port, (ii) apply all matching rules and (iii) forward the resulting packets on all outgoing ports. The steps (i), (ii) and (iii) are repeated for each resulted pair of packets and outgoing ports, until the destination is reached, on all possible paths. The result first tells us which packets are accessible at destination port d from source port s. The ability to read a certain variable v from a packet amounts to checking whether the variable remains free or is bound by the same expression at both source and destination ports. To check whether a packet can be modified at a certain point, we ask if these modifications are visible downstream (i.e. they can be read).

3. EVALUATION AND CONCLUSIONS

We have implemented **SymNet**—our symbolic analysis tool—in Haskell. SymNet includes a set of common functionalities including filter, NAT, proxy and tunnel. Running the toy example above we found that 1) TCP flows are allowed as long as the scrubber allows packets through. 2) The server can safely accept all packets from B because C does not modify the packets in any way. 3) C can read the original packet payload, but it will see B as the source of the traffic instead of A.

We are now adding support for common vendor configuration files to be able to analyze real networks. We are confident that symbolic analysis is a cheap and effective way of analyzing complex middlebox behaviours.

Acks. This work has been supported by Change, an FP7 project funded by the European Commission.

4. REFERENCES

[1] P. Kazemian, G. Varghese, and N. McKeown. Header space analysis: static checking for networks. In *NSDI*, 2012.
[2] H. Mai and et al. Debugging the data plane with anteater. In *Sigcomm*, 2011.