# OF-GUARD: A DoS Attack Prevention Extension in Software-Defined Networks

Haopei Wang, Lei Xu and Guofei Gu (Texas A&M University)

## 1 Motivation: Stopping the threats of DoS attack

Software Defined Networking (SDN) is becoming widely discussed in recent years and many researchers are using OpenFlow which is a reference implementation. It is a physically distributed but logically centralized framework. Decoupling the control plane from data plane, SDN is designed to support fine-grained policy. In OpenFlow protocol, network devices handle network flows based on the flow rules sent by the controller. When new packets that data plane does not know how to handle (table-miss) are coming, the data plane has to ask the control plane for actions and flow rules. So a table-miss consumes much resource in SDNs.

**Challenge and Threat Model:** OpenFlow "southbound" protocol leads to a scalability challenge. New data plane events may flood the control plane and far exceed the throughput of control plane. An attacker could exploit it by launching dedicated *denial of service attack* (or *data-to-control plane saturation attack*) that floods SDN networks [2]. The attacker may generate a large number of fake packets, which means all or part of fields of each packet are spoofed as random value. These coming packets will trigger table-miss and send a lot of packet-in messages to controller. As a result, this attack could overload the buffer memory of network devices, generate amplified traffic to occupy the data-to-control plane bandwidth and consume the computation resource of controller in a short time. This paper aims to solve above three research challenges.

**Problem Statement:** Our work aims to design a framework in SDNs to prevent data-to-control plane saturation attacks. In our problem domain, the flooding traffic includes a small amount of normal packets and large number of fake packets. Our design has two functional objectives. The first one is to keep both control plane and data plane working when suffering from data-to-control plane saturation attacks. For this, we introduce an extension called *packet migration*. The second objective is to distinguish fake packets from normal packets and discard them in order to reduce table-miss. To achieve this, we introduce another extension called *data plane cache*. Our design has the following advantages. First, our framework is attack-driven, i.e., under normal circumstances, only monitoring component works but others keep dormant. Second, our design does not change controller applications and end hosts. Third, we merely add ignorable overhead and latency.

**Related Work:** Avant-Guard [3] tries to halt the same threat but could only defeat TCP based flooding attack. Our approach aims to defeat all kinds of malicious flooding packets. DIFANE [4] focuses on the general scalability problem. However, it is not easy to directly apply its approach to our problem. Generated fake packets may still cause large communication and computation burden. DIFANE needs to work all the time, while our design is attack-driven. DIFANE keeps flow rule update in data plane, which may lose information about new incoming flows from controller's point of view. In our design, packet migration guarantees transparency to controller applications and end users. Because this DoS attack does not target on specific server, traditional DoS prevention solutions seem not suitable. Our work focuses on data-to-control plane saturation attack and provides a new approach to address it.

## 2 Approach Overview

Our basic idea is, when attack happens, we migrate table-miss packets and use proactive flow rules to distinguish fake packets. We introduce OF-GUARD, a scalable, efficient and lightweight framework for SDN networks to prevent *data-to-control plane saturation attack* by using *packet migration* and *data plane*

*cache*. Figure 1 shows the architecture and defense process of our approach. Packet migration detects flooding attacks and aims to protect switch and controller when attack occurs. Data plane cache is a machine that stores proactive flow rules, caches table-miss packets and distinguishes fake packets from normal packets.

**Packet migration** uses a controller application to keep monitoring the rate of packet_in messages from data plane. From the rate, we can calculate current usage percentage of the capacity of our SDN network. We identify there may be potential flooding attack based on certain anomaly threshold. Then this application will write a wildcard flow rule which has the lowest priority to ingress switch. Since wildcard rule matches all matching space, so it will redirect all table-miss packets in data plane to a remote machine we call it *data plane cache*. Our design is attack-driven. Only when suffering from flooding attack, our tool will shift all table-miss packets and protect OpenFlow switch and controller. After being checked in data plane cache, filtered packets will be set VLAN tag and forwarded back to ingress switch to recreate packet-in messages to controller. This process guarantees the transparency to controller applications and end hosts.

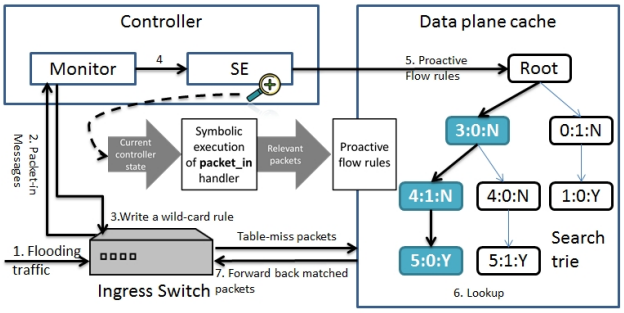


Figure 1: OF-GUARD architecture

**Data plane cache** stores proactive flow rules, caches table-miss packets and distinguishes fake packets. The primitive of resource consumption attack lies in that OpenFlow network needs to generate actions and flow rules for every table-miss packet. Hence, when network is under attack, our solution is to leverage the logic of applications in controller to generate proactive flow rules, which, to great extent, covers the possible upcoming packets. The challenge here is how to dynamically generate proactive flow rules. Symbolic Execution is a means of program analysis approach, which bears symbol variables as input other than concrete ones. It is capable to efficiently traverse possible branches in a program. In this sense, it is helpful for us to derive the logic of a specific controller application. So first we collect current controller state to initialize the variable of each application. Then we symbolize the packet header of packet-in event and feed it to packet_in handler which normally acts as trigger of flow rule update. By symbolically exploring the packet_in handler, we can get correlation between a chain of constraints ( a.k.a., path conditions) and the final handling decision. The handling decision is limited to a small set which is generating Modify State Message and Packet Out Message (defined in OpenFlow Spec. 1.4.0). Finally we get proactive flow rules based on current controller state.

As mentioned above, using the packet migration extension, all table-miss packets will be forwarded to data plane cache. Data plane cache also stores proactive flow rules. One packet is considered as a normal packet if its header matches any flow rule. We design an efficient data structure to store matching rules and fast check every incoming packet. Inspired by VeriFlow [1] we utilize an improved trie. The trie in VeriFlow has some limitations. Each level in the trie corresponds to a specific bit in a forwarding rule. So the number of levels in the trie will be very large. A path from the tries root to a leaf represents the set of packets that a rule matches. But in fact most levels in one path may be * (wildcard). That inspires us to optimize the data structure. In our trie design, each node stores one bit value of a matching rule. Each node has three fields: the order of bit in the matching rule, the value of this bit (0 or 1) and if it is a leaf of the trie (Y or N). We do not store wildcard value. Here is an example. Suppose we have three matching rules: ***010***, ***001*** and 10*******, the trie is shown in Figure 1. Our figure highlights the path for the matching rule ***010***. Actually the total number of nodes in our trie is equal to the number of non-wildcard nodes in the trie designed in VeriFlow. We reduce the depth and the total number of nodes in the trie.

# References


[1] A. Khurshid, X. Zou, W. Zhou, M. Caesar, and P. B. Godfrey. Veriflow: Verifying network-wide invariants in real time. In *NSDI'13*.

[2] S. Shin and G. Gu. Attacking software-defined networks: A first feasibility study (short paper). In *HotSDN'13*.

[3] S. Shin, V. Yegneswaran, P. Porras, and G. Gu. Avant-guard: Scalable and vigilant switch flow management in software-defined networks. In *CCS'13*.

[4] M. Yu, J. Rexford, M. J. Freedman, and J. Wang. Scalable flow-based networking with DIFANE. In *SIGCOMM'10*.