# Controlling Your Neighbour's Bandwidth for Fun and for Profit

Jonathan Weekes(✉) and Shishir Nagaraja

Lancaster University, Lancaster, UK
{j.weekes,s.nagaraja}@lancaster.ac.uk

**Abstract.** We carry out a systematic study of the attack-resilience of flow-rule replacement strategies for switch caches in Software-Defined Networks. Flow Rules are inserted into the switch at the request of the network hosts to direct traffic- replacing older rules when the switch flow table is full. Malicious hosts can leverage the flow rule replacement strategy to launch cache flushing attacks. This results in substantially reducing the network throughput of their neighbours forcing their traffic to slow down dramatically. We describe and evaluate the attack on the First-In-First-Out strategy- the defacto SDN flow-rule replacement strategy.

## 1 Introduction

Programmable switches are a foundational component of Software-Defined Networks (SDN). One of the key strengths of SDN switches is its fast line rate. These switches contain rules in TCAM which provide instructions as to where to forward the traffic they receive. TCAM memory provides fast lookup of rules (tens of nanoseconds) [1]. However, to reduce manufacturing costs, the size of TCAM memory on a switch is limited. Thus a switch can only hold a small number of rules (between a few thousand to a few hundred thousand) within its flow table. Being a scarce resource, a well configured flow table is essential for switch performance as it will otherwise result in 'cache-miss' for arriving packets, increasing their transmission time. An attacker who can exert influence on the state of the flow table has the potential to lower network throughput and increase latency.

SDN uses cache-misses to trigger controller activity to configure the flow-table. A Cache-miss however forces the packet to register a significant delay in arriving to its target due to the fact that it takes additional time for the controller to reconfigure the flow table to accommodate it. Under normal circumstances, it is only the first packet of the flow that registers this delay (of the order of ms) [5] with the remaining packets of the same flow being processed at line rate.

SDN's intrinsic nature of allowing users to influence the state of the switch's flow table allows malicious users in the network to launch various attacks aimed at reducing or controlling the performance/throughput of the network. A malicious user of the network may launch a cache flushing attack by requesting


F. Stajano et al. (Eds.): Security Protocols 2017, LNCS 10476, pp. 214–223, 2017.
https://doi.org/10.1007/978-3-319-71075-4_23

enough rules to fill the entire flow table with his/her own flows causing all other flow rules to be evicted from the switch. The rules inserted by the attacker are purely for the purpose of ensuring the legitimate rules are removed from the switch. By doing this, the attacker ensures that the next packet of a legitimate flow experiences the delay of a "first packet" while the switch awaits instructions from the controller. By repeatedly causing legitimate rules to be evicted, the attacker can ensure that a high percentage (if not all) of the packets of the legitimate flows experience this delay which results in the aggregate speed or throughput of the flows in the network being reduced.

The default replacement policy in SDN is the First-In-First-Out policy. However it does not perform optimally even under benign circumstances due to the fact that network traffic follows a power-law distribution [2]. Since FIFO does not consider the usefulness or popularity of a flow-rule before evicting it, it does not service network traffic particularly well. In the context of an attacker, FIFO faces an extreme vulnerability as the attacker can maliciously force a switch to dump heavy-hitter rules from the cache resulting in service-denial attacks.

So what rule-replacement strategies can designers of SDN switches and controllers adopt, when building a network, to provide performance and attack-resilience? There is significant literature on cache-design in distributed systems which suggest a number of strategies. Is it possible to achieve attack-resilience by retaining heavy-hitter rules in the flow table, as current traffic distributions suggest (LFU)? Or, is the key to attack-resilience to minimise the attacker's control of which rule is evicted by evicting randomly selected flow-rules (Random Replacement). Or, is evicting the least-recently-used-rule the best way to rid the switch of spurious rules induced by attacker traffic?

## 2 Motivation and Threat Model

### 2.1 Motivation

The ability to control a target host's throughput in a network can to be a powerful weapon. An attacker may be motivated to throttle a target host's throughput in order to launch a number of insider attacks. The attacks proposed in this work provide an attacker with the ability to introduce arbitrary delays to the target host's incoming and outgoing traffic. Here we look at two scenarios demonstrating how this ability can be useful to a malicious entity while detrimental to the rest of the network as well as the abilities required of an attacker attempting to perform these attacks.

**Trading Markets:** Environments such as currency trading, stock markets and energy trading, require traders to be able to buy and sell commodity units in real-time. A trader with the ability to delay another trader's transactions, can gain a significant upperhand. For example, a trader company - Malequities Ltd, deploys malware within another - WeakTraders Ltd's recently upgraded network infrastructure composed of SDN. Launching cache-flushing attacks on the SDN switches at crucial times from within WeakTrader's network, gives Malequities

Ltd an edge in purchasing cheaper stock before its victim does so, and selling it just after WeakTraders Ltd makes a purchase. Thus WeakTraders Ltd ends up paying for Malequities Ltd's profit margin by inducing latency at the right times. The attack completely undermines the fairness property of the stock exchange and allows the attacker to obtain a first-mover advantage.

**Video and Audio:** Video and audio streaming have quality-of-service requirements in terms of the maximum latency the application codecs will tolerate. For instance, both video and audio codecs will drop packets that don't arrive at the destination within a certain amount of time, or reduce bit-rate in response to deteriorating network (bandwidth) conditions. Cache-flushing attacks proposed here can reduce the network bandwidth in such a way as to render video and audio streaming inoperable while under attack. Organisations are increasingly dependent on the availability of VoIP to conduct business and any disruptions have the potential to cause significant financial and reputational damage.

**Denial-of-Service Attacks on Critical Infrastructure:** SDN lowers capital cost and offers centralised management of network infrastructure. For these reasons, SDN is a good candidate for network infrastructure in cyber-physical systems (CPS). Latency plays an important role in ensuring systems integrity. A delay in flows or packets from sensors to actuators can lead to a rapid build-up of risks from simply unsafe to outright dangerous. For instance, delays in reporting the moisture levels of a tank of sulphuric acid can result in a blast with potentially fatal consequences. Thus an attacker who is able to induce flow latency can cause damage in a wide range of situations: A hospital under such an attack can result in alerts notifying staff about critical changes in their patient's conditions being delayed causing lives to be lost; An airport which has its network throughput reduced may be unable to co-ordinate the departures and landings of planes properly resulting in collisions and traumatic near misses; A sewage system unable to receive timely instructions can cause tanks to overflow causing urban pollution.

### 2.2 Threat Model

From the above scenarios, we can see that there is sufficient motivation to warrant this attack. In evaluation of these attacks, we make the following assumptions and place the following restrictions on the attacker.

- We assume the attacker(s) is a regular user of the network who does not have admin access to the network equipment or any hosts on the network except the ones he/she has compromised.
- The attacker may have compromised or otherwise has root access to a small subset of the hosts in the SDN network excluding the hosts whose traffic he/she is attempting to control.
- The attacker is not able to directly access or control the physical switches or controllers but is able to influence the state of the network using traffic from the hosts he/she controls.

These assumptions show that our attacker is a relatively standard user of the network with no special abilities and a secure SDN network should have the robustness and intelligence to handle a user (or small subset) who suddenly turns malicious. To reduce the potency of such a situation leading to these attacks, the network should be able to distinguish between legitimate and malicious flow requests- classing malicious flow requests as ones which are simply issued for the purpose of disrupting regular use of the network. Ideally it should be able to distinguish between malicious and legitimate requests from a single host such that in the event of the host becoming infected with malware, the network is still able to provide appropriate service to the host while defending itself against the effects of the malware. In addition to this, the issue of fairness as a security requirement arises. As a QoS assurance the network should ensure that use of the network's resources are appropriately apportioned among its users and no single user has the ability to abuse more than their fair share of the resources.

## 3 Background

The key to inducing a delay in network traffic is flow rule eviction. SDN switches hold flow rules in the TCAM which direct traffic through the switch. TCAMs typically hold 4000–8000 rules (some hold more). Due to the size of the network and the large number of flows per switch, SDN networks often employ active flow eviction strategies in which the controller explicitly removes flow rules from the switch to make room for new flows.

The ability of a user to request a rule which leads to the eviction of another rule in the switch provides a vector for malicious users to launch attacks. In the absence of a rule to direct a packet arriving at the switch, the switch is forced to send the packet to the controller for direction. This is called a cache miss and adds additional latency to the packet. Subsequent packets of the flow usually do not register this delay because the controller places a flow rule in response to the cache miss, however by causing the regular eviction of this flow rule, a malicious user causes a large majority of the packets in the flow to register cache misses giving the aggregate flow a higher latency, inducing a delay.

Various strategies can be used in deciding which flow rule to remove from the switch TCAM. The most common one is First in First Out- meaning the rule which has been in the TCAM the longest is removed. Least Frequently used- the rule which has the lowest number of packet hits per second, Least Recently Used- the rule which has not been used for the largest amount of time and Random Replacement- selecting a rule at random to be removed are also potential rule replacement strategies. Without active flow removal, once the switch flow table is filled, no new flows can be routed resulting in denial of service to some hosts.

## 4 Attacks

### 4.1 Determining the Switch Table Size

The aim of the attacks is to fill the switch's flow table with flows such that the legitimate flows in the network are continuously removed. As such, the attacker

must, in each instance, determine the size of the flow table in question. Here we describe a "reconnaissance" attack an attacker can use to determine this value. Depending on the flow replacement strategy, this will involve probing the network in various ways. Because this part of the attack is not the focus of this work, we only demonstrate a probe under the FIFO strategy which gives the attacker an idea of the flow table size.

Abiding by the attacker model defined above, the attacker controls two physical hosts on the network. Determining the size of the flow table is simply a matter of requesting rules until one of them expires. Within one of these hosts (Host2), the attacker creates a large number of virtual interfaces, assigning each a valid network IP address. From Host1 the attacker can then regularly ping one IP address (IP1) on Host2. Once this is in action, the attacker sends a single ping to each of other the IP addresses on the interfaces created on Host 2. The number of other IPs which can be contacted in this manner before Host1-IP1 pings register a delay (which indicates a cache-miss) will give an idea of the number of rules in the flow table.

The main factor affecting the accuracy of this probe is the amount of surrounding network traffic as the attacker conducts his/her experiments. It is crucial that the attacker picks a time when there is expected to be as little traffic as possible on the network (e.g. late at night). Repeating the experiment several times will also help reduce the noise-induced errors (due to possible other traffic on the network). Even with this, there is not a 100% guarantee of accuracy, however it will give the attacker a general idea of how large the flow table size is.

### 4.2 First In First Out

The First In First Out (FIFO) flow rule replacement strategy is the default for OpenFlow Switches. When the switch's flow rule capacity is filled, it evicts the oldest flow rule in favour of new ones requesting space.

To induce a delay in a network user's traffic, the attacker must ensure that as many of the victim's packets as possible register a cache-miss forcing the switch to go to the controller for direction. To remove the user's rule (to achieve the cache-miss), the attacker must send enough traffic to fill the switch's flow table. The FIFO policy means that the number of packets the attacker sends in a particular flow after the first is irrelevant, as long as they are sent after the victim's rule has been put in place, the attack rules will be favoured over the victim's rule. With this in mind, the attacker sends a single packet to a range of destinations large enough to cause rules which fill the flow table and have the legitimate user's rule removed. Enough destinations cause the table to be filled with the attacker's rules which is the goal of the attack. Because the initial packet of a flow experiences some delay while the controller maps out the path and puts the rules in place, the constant removal of a legitimate flow's rule will have the effect of slowing down the entire flow effectively reducing the bandwidth of network.

Deterministic rule replacement strategies give the attacker a 100% chance of success if he/she is able to meet the criteria. In this instance, the criterion is

simply to cause enough rules to get the victim's rule evicted as often as possible. The use of virtual interfaces and PINGs mean that the attack does not require a large amount of computing power and is only limited by the attacking host's bandwidth. The bandwidth will dictate the rate at which he/she can request new flows. The rate at which the victim sends packets through this flow also affects the attack. If the victim sends 1000 packets at line rate, the attack will be much less effective than sending 1000 packets at a rate of 10 per minute, in which case the attacker will likely have enough time to remove the flow rule between each packet. The number of compromised hosts the attacker controls also factors into the effectiveness of the attack. The more hosts the attacker controls, the more attack rules that can be requested per second. Very little physical effort is required of the attacker as well since the entire process can be automated with a short script and a list of the ip addresses.

## 5 Evaluation

### 5.1 Network Setup

To evaluate the various attacks described above, we utilise the network shown in Fig. 1 below. Up to this point we have only evaluated the success of the attack on the FIFO strategy. The results from this indicate a promising line of research. The key evaluation criteria is how much we can slow the bandwidth of the benign users when an attack is launched.

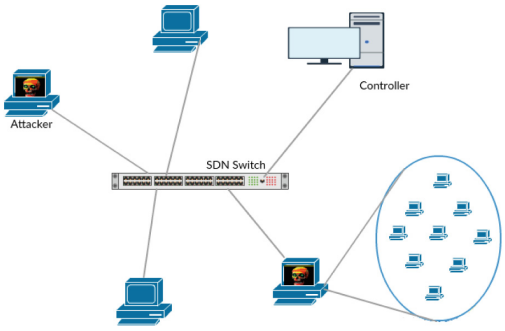



**Fig. 1.** Network setup

The network contains a Pica8 3297 SDN Switch, a ryu controller which installs the flow rules for communication in the network and four (4) Linux hosts. Two (2) of the hosts are used as "compromised hosts" and the other 2 as benign hosts on the network. Because of the small size of the experimental network, we constrict the size of the switch table to hold 90 flows. In a regular network where the switch flow table size is unrestricted, a switch can hold several thousand flows. The attacker would simply have to scale up the attack to be effective i.e. compromise enough hosts to support enough flows to facilitate the attack. This is not an unlikely scenario.

### 5.2 Determining the Switch Table Size

We evaluate this attack by testing the ability to accurately determine the number of flow spaces available in the switch. We launch 100 virtual interfaces in one compromised host and carry out 5 rounds of the attack described above. The results are given below.

**Table 1.** Determination of flow table size

| Round | Num of Pings |
|---|---|
| 1 | 45 |
| 2 | 44 |
| 3 | 46 |
| 4 | 45 |
| 5 | 44 |

From the results we determine we are able to ping between 44 and 46 interfaces before the first rule was replaced. Thus, we conclude that the switch has enough flow spaces for communication with no more than 46 machines. This is relatively accurate since the switch was set to hold 90 rules (each ping requires two (2) rules- source to destination and destination to source for the reply). With this in mind, each attack shall be carried out with a minimum of 50 destinations.

### 5.3 FIFO Attack

To evaluate the effectiveness of the attack, we conduct 10 file transfers from one benign machine to the other under both normal and attack conditions and compare the differences. Each file is 1 GB in size. The length of time the file transfer takes and the average bandwidth available to the benign hosts under both conditions will determine whether the attack has been successful or not.

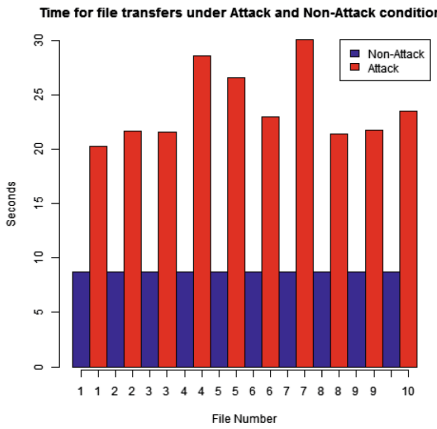
We carried out the attack by launching 250 virtual interfaces on one "compromised" host, assigned them valid network IP addresses and had the other compromised host rapidly ping each of these destinations in quick succession. From the recon attack, 250 targets would be more than enough destinations to fill the switch flow table. The attack was repeated every 0.5 s thereby constantly renewing the rules in the table. It should be noted that this attack can be scaled up or down by increasing or decreasing the time between attack repetitions and/or using more compromised hosts, allowing the attacker to increase or decrease the victim's bandwidth at will. The results are documented below in Table 2 and Fig. 2.

## 6 Discussion

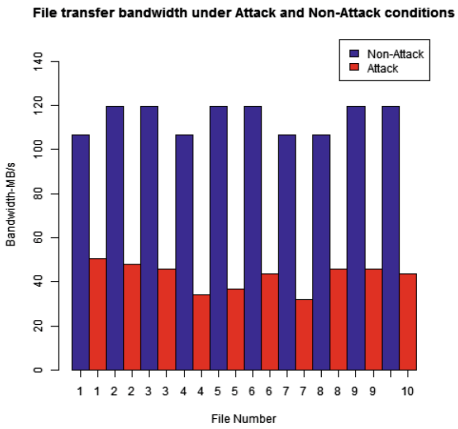
This work examines the impacts of cache-flushing attacks within an SDN network. The results in Table 1 show that the cache-flushing attacks slow down file

**Table 2.** Time taken and average bandwidth for file transfers under bening and attack conditions

| Round | Benign conditions | | Attack conditions | |
|---|---|---|---|---|
| | Time - secs | Avg BW - Mb/s | Time - secs | Avg BW - MB/s |
| 1 | 8.741 | 106.5 | 20.234 | 50.4 |
| 2 | 8.733 | 119.8 | 21.667 | 47.9 |
| 3 | 8.732 | 119.8 | 21.580 | 45.6 |
| 4 | 8.730 | 106.5 | 28.557 | 34.2 |
| 5 | 8.732 | 119.8 | 26.605 | 36.9 |
| 6 | 8.734 | 119.8 | 22.985 | 43.6 |
| 7 | 8.732 | 106.5 | 30.115 | 32.0 |
| 8 | 8.733 | 106.5 | 21.428 | 45.6 |
| 9 | 8.735 | 119.8 | 21.803 | 45.6 |
| 10 | 8.733 | 119.8 | 23.534 | 43.6 |




(a) Figure 2a




(b) Figure 2b

**Fig. 2.** Experimentation results

transfer significantly. Figure 2a and b show the file-transfer duration and the average bandwidth recorded. Under normal conditions, the recorded bandwidth was 106.5–119.8 MB/s while under attack conditions the recorded bandwidth was 32.0–50.4 MB/s. Under attack conditions, the transfer duration was increased by 400–600% compared to normal conditions, making this a successful attack.

In the evaluating the attack, we flush the flow table twice every second to achieve the increase in file-transfer duration. Several factors must be considered in the attack including the rate of flushing, size of the table and rate of flow rule installation. Increasing or decreasing the rate of flushing is likely to drive

the available bandwidth down or up respectively. For example, flushing the table twenty times per second will cause more first-packet delays than twice per second providing the attacker with a tool to adjust the victim's throughput as desired. The size of the flow table dictates the number of destinations the attacker must send to in order to flush the table. As the flow table size grows, it is likely to become more difficult to flush the flow table unless he scales up the number of attack sources. This is achievable through the use of botnets and should not present a difficult challenge. Finally, the rate at which the controller can install rules will determine how quickly the table can be flushed and by extension the maximum number of flushes which can be executed per second.

Our analysis of cache-flushing attack is in its preliminary stages and we make several assumptions. First, we assume that the controller is sufficiently provisioned and does not suffer overloading as discussed by Kloti et al. [3] and Shin and Gu [4] during the attack. Second, we assume that controller-switch link [6] also does not suffer congestion as a result of the attack.

Both controller-capacity and the availability of the link between the controller and the switch are important considerations for attack performance. Due to the attack, a large number of flow requests are generated. When the number of requests exceeds the controller's capacity, it develops a backlog of flow requests or crashes. In the event of a crash, it no longer provides routes for flows and neither the attacker nor victim can make further progress. Similarly, with a backlog, the response times for flow requests increases, slowing down both attack and benign flows. Another side effect of the attack is congestion in the control-channel (switch controller link) which occurs when the switch-controller channel is filled to capacity causing delays in the controller response times when a flow request is made. This delay brings about a larger than usual delay in the first packet of the flow while the switch awaits the flow rule. We have not studied the impact of these variables on attack efficiency. Future work will move towards isolating the effect of rule replacement from the other issues in order to bring about a better understanding.

## 7 Conclusion

In this work we propose attacks against rule replacement strategies in Software Defined Networks and evaluate the attack resiliency of the FIFO flow-rule replacement strategy with the intention of carrying out similar evaluations of other prominent strategies. We demonstrate that the FIFO flow-rule replacement strategy is easily susceptible to attack which can then be used to throttle the bandwidth of neighbouring hosts in the network.

## References

1. Kannan, K., Banerjee, S.: Compact TCAM: flow entry compaction in TCAM for power aware SDN. In: Frey, D., Raynal, M., Sarkar, S., Shyamasundar, R.K., Sinha, P. (eds.) ICDCN 2013, vol. 7730, pp. 439–444. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35668-1_32

2. Katta, N., Alipourfard, O., Rexford, J., Walker, D.: Rule-caching algorithms for software-defined networks. Technical report. Citeseer (2014)
3. Kloti, R., Kotronis, V., Smith, P.: Openflow: a security analysis. In: 2013 21st IEEE International Conference on Network Protocols (ICNP), pp. 1–6, October 2013
4. Shin, S., Gu, G.: Attacking software-defined networks: a first feasibility study. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN 2013, pp. 165–166. ACM, New York (2013)
5. Tavakoli, A., Casado, M., Koponen, T., Shenker, S.: Applying nox to the datacenter. In: HotNets (2009)
6. Wang, A., Guo, Y., Hao, F., Lakshman, T., Chen, S.: Scotch: elastically scaling up SDN control-plane using vSwitch based overlay. In: Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies, CoNEXT 2014, pp. 403–414. ACM, New York (2014)