Study Of The Impact Of Side-channel Attacks On Software Defined Networks

Alex De Los Santos Subirats
Supervisor(s): Mauro Conti, Chhagan Lal
EEMCS, Delft University of Technology, The Netherlands
27 June 2022

A Dissertation Submitted to EEMCS faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

Abstract

Software-Defined Networks (SDNs) are a promising new network design paradigm that allows for better control of the network. But as with any new software implementation, there are new security concerns that arise. In the past there have been various papers covering specific side-channel attacks on SDNs; most of them consist of either using time delays in operations to create a covert communication channel between two compromised hosts or using proving packets and their response times to determine the flow rules and configuration of the network.

This paper intents to investigate the impact of different types of side-channel attacks in SDN scenarios. Provide a survey on the state-of-the-art solutions that are proposed to address the side-channel attacks in SDN. Particularly, identifying different ways through which an adversary can launch side-channel attacks, and the different entities and network metrics that are impacted by a specific side-channel attack. Next, identify and survey the solutions available in the state-of-the-art that tackle side-channel attacks. Finally, propose new possible improved solutions to the issue of side-channel attacks in SDNs and future research on the field.

We conclude that current side-channel attacks target the separation control and data planes at the core of the SDN paradigm, by exploiting the response delay created by a centralized logic system in the controller. This as seen can be exploited in two main ways related to the two main attack categories mentioned in this paper: *teleportation attacks* and *recognisance attacks*.

1 Introduction

Software-Defined Networks (SDN) unlike in traditional network architectures makes a point of dividing the control and data layers. This is done by centralizing the gate control logic in *controller* [1]. Such a structure allows for better control of the network allowing the administrator to roll out protocol updates or implement policies with ease. However, this comes with new challenges when it comes to security as a change in architecture leads to new vulnerabilities that can be exploited by malicious actors. In particular, this new design paradigm opens new side channels to be exploited caused by the separation and logical centralization of the control plane in the entity of the controller [2], how this happens will become clear as the attacks are explained.

There have been some papers that have dealt with the security issues of Software Defined Networks (these papers will be addressed in the related papers section 2), but there has not yet been a comprehensive survey of their vulnerabilities regarding side-channel attacks. That is what will be the focus of this paper, part of a group of five other papers focusing on SND vulnerabilities to different types of attacks. To be precise this paper aims to provide the reader with a survey of the current state-of-the-art vulnerabilities and solutions to side-channel attacks in Software Defined Networks, and finally propose new solutions to be used to make SDNs less vulnerable to side-channel attacks. To do that the paper will follow the following structure: Firstly we will go over some necessary background information and the related work that has already been done in this field, the section after that will cover the attacks and their solutions, following that we will discuss the limitations and compare the performance of the different solutions, the second to last section will be reserved for discussing the problems that remain unsolved and the possible future research, and finally the last section will be a conclusion where we will explore some final remarks.

2 Background and Related Work

Software-Defined Networks (SDN) is a new network architecture paradigm in which the data and control layers are separated. This allows for the data layer to consist of switches that simply follow the instructions given by a logically centralized controller[1]. This has the advantage that new protocols are policies can be easily implemented with access to the controller.However, this has the side effect that if an attacker is to compromise the controller the entire system would be in danger[3]. In addition to this more specifically in the case of side-channel attacks, the added logic and centralization of the controller adds new side channels that can be exploited[2][4]. By using the architecture you gain control over the network but also open the door to new attacks.

For this paper, it is also important to define and understand what a side-channel attack is and its characteristics. A side-channel attack is an attack that targets the physical implementation of the cryptosystem other than the system itself [5]. This is done by using so-called *side channels* that are ways where data and information is being “released" without that being the intention. A typical example of this would be the response time of the network when receiving different packages, this would be an example of a side-channel attack where the response time is the side-channel [5]. Other notable side-channels are[5]: the power consumption of the network, the traffic of the network, or even the sound and thermal energy being dissipated by the network.

In the case of side-channel attacks on SDNs, there is usually an interaction with the network for which you monitor the response to gain the information you are interested in, this is usually the time it takes for you to receive a response.

In the past, there have been many papers discussing the several side-channel attacks on SDN. But those papers focused on specific attacks and did not provide an overview of the security issues with side-channel attacks on SDN like this paper intends to. Some notable examples of these papers are:

1. *“A Novel Stealthy Attack to Gather SDN Configuration-Information"* [4]: A paper covering the details of a novel *know you enemy* attack that would allow the attacker to gather information about the network configuration.
2. *“Preventing Timing Side-Channel Attacks in Software Defined Networks"* [6]: Is a paper that looks into preventing timing attacks in particular.

3. *"Flow Reconnaissance via Timing Attacks on SDN Switches"* [2]: Is another paper covering side-channel attacks but this time focuses on the delay that packages can encounter when the switch has not matching forwarding rule. using that to determine if covering rules for the given package were previously installed in the switch.

4. *"Control Plane Reflection Attacks in SDNs: New Attacks and Countermeasures"*[7]: This paper once again focuses on timing-based side-channel attacks, this time the emphasis is put on exploiting the limited possessing power of the hardware switches and forcing the controller to expensive control messages towards SDN switches.

As mentioned all these papers cover specific side-channel attacks that can be performed on an SDN but no attempt to make a comprehensive survey and summary of the current state-of-the-art attacks and deterrence methods.

3 Attacks and their countermeasures

In this section we will go over several attacks that have been covered in papers; a brief explanation of the attack will be given, followed by a section discussing the components affected by the attack, after that we will discuss the metrics that are affected and finally the proposed solutions for that attack.

When looking at the papers reviewed for these attacks you can classify them into two broad categories. The first one would be attacks that create a covert communication channel through which the attacker can create a covert communication channel between two points of the network. The second category of attacks or those that give the attacker information about the configuration of the SDN and the limitations and protocols it has, this information can then be used to compromise the network by attacks like DDoS. In the following paragraph, a more in depths look at both of these categories of attacks will be explained.

The first category of attacks, that from now on we will refer to as *teleportation attacks* [8], are exploits that create a covert communication channel between two compromised hosts [9] or switches [8] inside the network. This is worrisome because it compromises network security by permitting these two points to communicate without even the most rudimentary elements of network security [9], allowing them to share information such as secret keys, network details or material that is protected under copyright. *It is important to note that for this sort of attacks the attacker has to have compromised or have access to two switches or hosts in the network*[9][8]. This type of attacks utilize a timing side-channel in which the *sender* can considerably affect the timing of an operation performed by the *receiver*, then with a previous agreement between the *sender* and *receiver* you can modulate a "0" as a short version of the operation and a "1" as an extended version of it. This channel however is slow to transmit data as each transmission of a bit happens in a pre-accorded *frame* of time that is not negligible, this style of communication can achieve a rate of 20 bits per second and an accuracy of 0.9 [8], so the transmission of a 2048 bit RSA key would take less than 2 minutes to transmit.

The second type of attack covered is a lot more varied and we shall refer to them as *two stage recognisance attacks*, the basis of this attack is to gather information about the network that can then be used to compromise the network. These attacks are divided into two phases. The first is a proving phase in which the attacker sends packets through the network with the objective of gathering information about network policies and how to trigger events in the network. To do this the attacker uses the timing side-channel in order to measure the response times to the different packets being sent in order to determine what actions have taken place in the network and their cost (see figure 3). Following that there is usually an attacking phase where you use the information you gained from the proving phase to trigger the network behaviour the attacker desires; this is frequently used to craft Denial of Service attacks, as now know you to trigger expensive operations in the network. Due to the focus of this paper being side-channel attacks the focus will be on the first stage of the attack.

In table 1 the classification used for the attacks covered in this paper is displayed together with the section where they are covered.

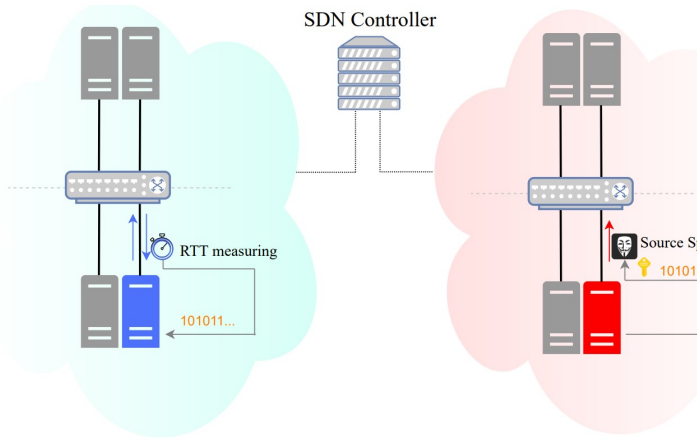
Teleportation attacks:	Recognisance attacks:
Cache Collision attacks: Section 3.1	Flow Reconnaissance: Section 3.3
Switch Identification: Section 3.2	Control Plane Reflection: Section 3.4
Proposed Attack: Section 5	Stealthy Configuration Information Gathering: Section 3.5

Table 1: Table Showing the classification of the attacks covered.

3.1 Cache Collision Attacks [9]

This attack is the first example of the *teleportation* attack category mentioned earlier. As such, this attack uses cache address collisions in order to allow two hosts that might be separated by a firewall or physically to communicate and leak secrets to each other, as long as they are connected to the same controller [9].

This method works by having a host (from here on referred to as *sender*) send a message to an arbitrary destination while faking the address MAC of a second host (from here on referred to as *receiver*). The switch in contact with the *sender* will then contact the controller in order to communicate a new MAC address discovery. This will cause the controller, running a mobility application, to interpret this as the *receiver* (blue in figure 1) migrating to the location of the sender (red in figure 1) [9]. Accordingly, the controller will perform a flow reconfiguration by acting like *receiver* is in the new location. This reconfiguration consists of deleting the flow rules used in the "old" switch where the *receiver* is still connected and installing them in the "new" switch where the *sender* is connected. Next time the *receiver* attempts to send a message the switch will not have any matching flow rules installed forcing it to request them from the controller, this creates a delay in *receiver*'s packets that can be measured as longer



[9, p. 1]

Figure 1: Macchiato attack diagram. The sender (red) modulates data by causing cache misses for the receiver (blue).

than normal [9]. We can then proceed to repeat the process to modulate data to transmit it from the *sender* to the *receiver*. As a modulation example: a longer message time is a "1" and a shorter one is a "0". This communication channel bypasses the fundamental network security policies and allows for the leaking of network keys, secret parameters or other secured information contained in the network.[9]

Components and network metrics effected

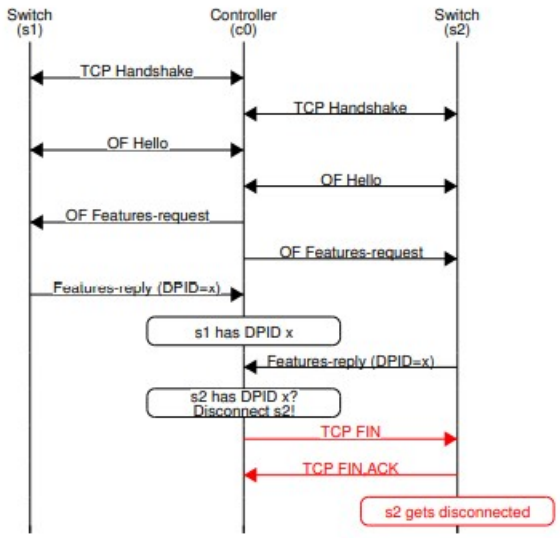
For this attack, it assumed that the malicious agent has been able to compromise and gain control over at least two hosts in the network connected to different switches. In addition to that this attack exploits the time delay introduced by the mobility application of the controller, so it is separate from the SDN protocol used [9].

The main metric this attack affects is the privacy and security of the network, as it allows two hosts to communicate directly and ignore all network security protocols [9]. This makes it possible for the attacker to easily leak sensitive information like security keys (RSA private keys for example) or other protected content, as this channel is under no supervision or security protocol.

Countermeasures

There are various proposed ways of tackling security from this attack some of them are better tailored than others:

- First of all it can be mitigated by adding security to the control plane with tools like *Topoguard*[10] and *Sphinx*[11]. This however has the disadvantage that it critically reduces the network's ability to adapt to moving hosts.
- Finally, the paper [9] suggests a solution that as said in the paper, "breaks the strong coherency imposed by the mobility application: the MAC forwarding at all switches are consistent with the last known location of that MAC address"[9, p. 13]. It is this coherency that is exploited by the attack by using it to delete flow rules from the *receiver's* switch. This solution proposes breaking this by substituting the immediate revoking of the rules in the "old" switch by predefined flow rule idle-timeout for all rules in a switch.



[8, p. 2]

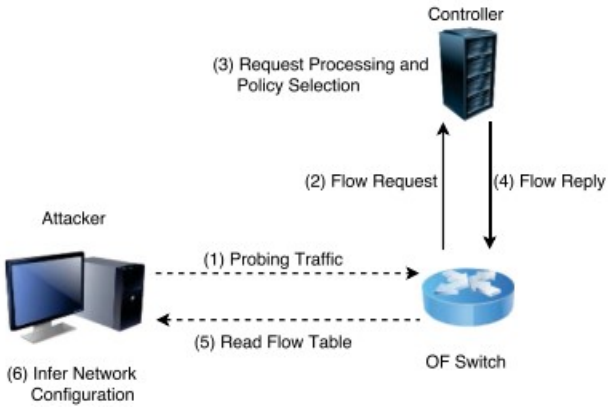
Figure 2: OpenFlow connection example for two switches in case 1.

3.2 Switch Identification Covert Timing Channel [8]

This attack, as the name implies creates a covert side-channel between two switches, for this reason it has been classified as a *Teleportation attack*. In this case, the side-channel being exploited is the connection state of the switch to the network at a point in time to modulate the bits of the data [8]. Before we go into further detail it is important to review the OpenFlow connection protocol used to initialize connections with a new switch: First of all, in order to identify switches OpenFlow uses a Datapath ID (DPID) in order to uniquely identify each switch to the controller; secondly, if a new switch shows up with the DPID of an existing switch the controller running OpenFlow has 4 possible ways to handle this[8]:

1. It can choose to refuse the incoming connection from the new switch (figure 2).
2. The controller can accept the incoming connection and terminates the existing connection with the previous switch.
3. The controller accepts the connection from both switches.
4. The controller accepts both connections but assigns a different ID to the switches.

Form these options number 3 is the only one that does not allow us to create a covert communication channel, however it does create other security issues as the controller is now unable to differentiate the two switches [8]. For the remaining 3 options, the method of creating this side-channel is slightly different but the same idea remains in each case: interfere at



[4, p. 3]

Figure 3: Basic diagram of traffic during the proving phase.

a regular schedule with the connection status of the *receiver* switch in order to modulate a “1" or do not interfere during said time to transmit a “0" (or vice-versa). As an example, we will cover the transmission of a bit in case 2 [8]. As can be seen in figure 2 when the second (the *receiver*) switch attempts a connection but the DPID is already taken over by the *sender* the connection to it is denied. However, if the connection was to be attempted during a period of time where the *sender* is not connected the *receiver* would be able to connect. The result of this connection attempt would then be interpreted as the value of a bit in the message that is being transmitted. [8]

Components and network metrics effected

To be able to perform this attack it is required for the attacker to have compromised at least two switches with connections to the controller in the network [8]. It exploits the ability of a switch to disconnect or effect the connection of another switch in the network by faking its identity.[8]

The components affected by this attack and the metrics of the network the attack has an effect on are similar to the other covered *teleportation* attack (see respective section of section 3.1). Security is compromised as this covert channel allows for the transmission of data that would otherwise be blocked by the network.

Countermeasures

The paper [8] does not provide any solutions for this kind of attacks. However, there is a solution that would completely shut down the covert channel, keeping the controller configuration on option 3 (where it accepts connection from both switches). As long as none of the switches can detect a connection from the other the communication channel does not exist. However, this opens the door to other exploits caused by the controller not being able to tell the two malicious switches apart. Another option when it comes to shutting down the side channel could be adding a “cool-down timer" for each switch, not allowing another connection with the same DPID to the controller until a certain amount of time has passed. This would heavily slow down the transmission rate of the side-channel making it almost unusable, as it would force the *frame* used to transmit one bit to be at least as big as the cool-down [8].

3.3 Flow Reconnaissance [2]

This is a simple version of a *recognisance attacks* that uses the networks response time to certain packages (for example packets with a destination IP address for which the switch does not have an applicable flow rule) in order to gain information on what flow rules are currently installed in the switches. By measuring the response time of the packets the attacker is able to extrapolate if a flow rule covering a flow stream has been installed on the switch [2].

Before we delve into more specific details of the attack let us cover what causes this extra delay, this information will be useful when talking about more advanced *recognisance attacks* later in this section. When a packet reaches the switch in an SDN the switch checks its flow table to see if any flow rule matches the headers of the given packet, if it finds said matching rule the package is forwarded accordingly. However, if the switch fails to find a matching rule then the switch sends a message to the controller requesting a flow rule that matches the IP address in the package, the controller will then provide it with a new flow rule to be added to the switch cache. As is obvious the first operation takes notably less time than the second one, it is this time delay that the attacker can measure to know if the flow rule was installed in the switch.[2]

This sort of attacks can be used to obtain different information, notably, it can be used to prepare a DDoS attack (by repeatedly triggering the expensive operations) or to check what websites a member of the network has been visiting [2]. As an example, if an attacker wanted to see if a host X has visited a web server Y they could send two packages to the web server, one from their own IP address and one with a spoofed IP address from host X and measure the response times. Assuming that the attacker has not visited the website in question they should get a response with added time (created by the switch communicating with the controller), this response time can be used as a reference. On the other hand, the package sent from the spoofed IP address should either take a similar amount of time (if the user has not visited that website recently) or in a shorter amount of time (if he in fact has visited said website).

Components and network metrics effected

In order to be able to execute this attack, the attacker needs to have compromised a host in the network with a connection to a switch communicating with the controller.

This being an information-gathering attack on the network it has little effect on the efficiency of the network when it is being executed, however, it greatly affects the privacy of the network (by allowing you to know websites visited by other hosts). In addition to that, the security of the network is also affected as information regarding the configuration of the network is now available to an attacker that could potentially use it in a future attack or sell it to another agent that will.

Countermeasures

In the paper [2] there a three proposed countermeasures:

1. *Artificially adding delays*: By artificially delaying the first few packages of a flow regardless of if they have

a matching rule or not the attacker would be unable to know if the rule was installed in the switch or not. This solution has the upside of making sure no information can be gathered from the side-channel by eliminating it, however, it has a severe downside as it effectively removes the "cache" that the switches have as they can no longer benefit from the speed it having pre-installed rules provides.

2. *Proactively setting up rules*: The controller could proactively install the rules in the switches during the set-up phase. By doing this the attacker would no longer gain any valuable information. This solution is however limited by the capacity of the switches used, as the more precise the flow rule is and the better it can be tailored to a specific flow the more individual rules you require to cover the total flow
3. *Using a transforming rule structure*: Finally by transforming the rule structure by merging or splitting rules the uncertainty faced by the attacker after proving would be increased. This solution can be implemented easier than the other given however it does not make the attack impossible, simply harder.

3.4 Control plane reflection [7]

This attack has two variations, one more complex than the other. The first one we will call "*Table-miss striking attack*" and the second one which we will cover after we have gone over the simpler version is called "*Counter Manipulation attack*". [7]

Table-miss striking attack is an attack consisting of two phases. The first one of these is the proving phase, which consists of sending packages whose headers are deliberately faked values at a low rate. The attacker can then use the round trip response time of these packets to gain confidential information about the network [7]. If there are multiple packets with the same header, in particular, if the first one had a considerably longer response time than the other we can deduce that the first packet got sent to the controller while the other packets were redirected by the switches without issues [7]. This indicates that that packet does not match any flow rules in the switch invoking *Flow-Mod* and *Packet-in* instructions. From that, the attacker can manipulate the header fields and with no more than 42 trials (that being the max number of header fields OpenFlow supports) they are able to know which of the fields is triggering the controller. From that, a stream of packets can be crafted in order to cripple the network by deliberately causing expensive downlink messages. [7]

Counter Manipulation attack is a more sophisticated attack when compared to the table miss attack. In order to perform this attack, the attacker creates three different packet streams that they will send to the network. [7]

- The first stream, which we will call "*timing proving packets*", consist of packages that should be sent at regular intervals and cause as little load as possible to the controller and switches. The change in response time of these packets will be used to measure under how much stress the network is under at that moment.
- The second packet stream we shall call the "*test packets*" and these packets exist to complement the function of the "*timing proving packets*". Their objective is to create as many table-misses as possible in order to burden the switch software agent
- Finally the last stream of packets we will call the "*data plane stream*". This stream will consist of packets that travel directly through the switches (the data plane) without triggering the controller (control plane), and their function is to collect more advanced data.

With this setup the attacker can learn information about the setup, the basis for this lies in the idea that different kinds of downlink messages have to take a different amount of time and have a different cost for the downlink channel [7]. For example, the three main types of downlink messages can be differentiated by their time cost. In descending order they are: Flow-Mod, Statistics Query and Packet-Out. These cost differences will cause the timing probing packets to vary their RTTs (round trip response time), from this we can learn what messages were triggered when. [7]

Components and network metrics effected

This attack assumes that the attacker has compromised one or more hosts or virtual machines of the SDN and can use said host to send probe packets, monitor the response times and generate traffic. With this the attacker can affect the controller and learn the configuration details it is operating under. [7]

Regarding the network metrics affected, both attacks discussed in this section have the same basic objective, to gather information about the SDN configuration details. However, the first attack gives a lot more limited information than the second one. Since the information gained from the proving phase of the **Table-miss striking attack** can be used to craft an effective denial of service attack we can consider that the main affected metric is the performance of the network [7]. However, we should note that the effect on the performance of the network is NOT caused by the side-channel component (the probing phase) but by the follow-up attack that uses said information. Moving on to the **Counter Manipulation attack**, the information provided by this attack is quite more exhaustive than its more simple counterpart. The information from it could be used in multiple ways to compromise the network, therefore this attack affects the security and performance of the network greatly [7].

Countermeasures

In the paper [7], several solution proposals are mentioned such as *limiting the use of dynamic features for network applications*. This would make it harder (or depending on the limit impossible) for the attacker to overwhelm the switches. On the other hand, this solution comes at the cost of less control and visibility of traffic management.

Another proposed solution is *limiting the downlink message transmission rate directly in the controller* in order to prevent the switches from being overwhelmed, however, this is hard to implement as the exact downlink message capabilities vary from switch to switch, and even if that was accounted for the controller can not guarantee an underload or uperload for a remote switch [7].

The last simple solution proposed is to add a random increase of latency to downlink messages making data plane events harder to monitor and read. This would make it so readings based on the response time are unreliable, as long as the random latency added is similar to the latency added when triggering patterns/policies of direct/indirect data plane events. However this countermeasure has the downside of increasing the latency in downlink messages and "would inevitably violate the latency requirements of some latency-sensitive downlink messages, making it high cost and infeasible"[7, p. 15]. These are simple solutions but all have some significant downsides.

However, the authors of the paper [7] propose their own system called *SWGuard*, that according to them *discriminates between "good" and "evil"* and then uses this discrimination result to prioritize the downlink messages. This discrimination is made using a multi-queuing scheduling strategy, that archives different latency to different messages. This is done on the basis of a statistical analysis on the messages during the past period that "takes both fairness and efficiency into consideration"[7, p. 15]. When there is starting to be congestion in the downlink channel then the malicious messages are given low priority, heavily hindering the attack.[7]

3.5 Stealthy Configuration Information Gathering [4]

This attack is another case of a side-channel proving attack that aims to gather information about the target network such as [4] the configuration of security tools, general network policies and network virtualization. For this attack, however, unlike other proposed attacks where the attacker was only able to compromise a host of the network, the attacker has gained access to a switch and can see the flow table and rules in the switch (how the attacker has gained such access is outside the scope of this paper).

In order to gather the network security configuration information, the attacker uses analysis of the flow table together with repeated proving using a spoofed IP address with the objective of determining if a security measure has been triggered and therefore the triggering threshold. The type of flow rules that c, van be installed as a response to said attacks vary from [4]: traffic redirection [12], traffic filtering [13], rate-limiting [14], honeypot redirection [15] or whitehole network approaches [16]. For all of these mechanism-specific flow rule installations are required that can be read by the attacker to be made aware of what defence they are dealing with.

When it comes to gathering the SDN configuration information the attacker uses a similar procedure to the one used when trying to achieve the same goal using a timing side-channel. A series of proving packets are sent in order to get the desired information [4]. These packets will take advantage of the fact that flow tables can only have a limited amount of flow rules, when this number is surpassed the controller can implement so-called *wildcard* rules. These rules usually will mean the fusion of two or more flow rules from the switch in order to make sure that the switch is not saturated. A saturated switch will ignore new inbound network flows so that usually wants to be avoided. When a *wildcard* rule is implemented it targets the smallest flows of the switch (packets per second or packet size), making sure the "large" flows are given their own individual flow rules. With this attack, the opponent aims to discover what rate and/or size of flow is required in order to install targeted control flow rules.

Components and network metrics effected

In order to be able to execute this attack, the attacker needs to have compromised a switch of the network to such a degree that they can read the rules in the flow table [4]. In addition to that, this attack affects the security of the network as it exposes the configuration and logic of the controller.

This attack on its own, like most information-gathering attacks, has a minor effect on the performance of the network while the proving is taking place. However the metric that is most affected by this attack is the security of the network, as now the information can be used to perform a variety of attacks [4].

Countermeasures

In order to solve the proposed vulnerabilities, it is suggested by the paper [4] to take advantage of the programmability and flexibility of SDNs and use a solution they call *flow obfuscation*. The basis of this solution is to stop the attacker from discerning the flow rules that are installed in response to their traffic. To do this it uses OpenFlow's ability to modify packets while in transit in order to stop the attacker from being able to make the distinction between their transit and other users' transit.

When a network flow enters a switch the controller installs a single flow rule by doing the following actions: the first action is a modification of the packets by changing some identifying header fields (such as IP addresses), and the second action then informs the switch of the port to use in packet forwarding.[4] This can stop attackers that control more than one switch, even stronger than the one proposed in this section. This is then repeated for all but the last switch in a path, in which the proper flow rule for the flow is installed. This is what is referred to as the *obscured path* [4].

4 Limitations and comparison of solutions

In this section we will be comparing solutions and their effectiveness in mitigating SDN side-channel vulnerabilities.

When dealing with side-channel attacks for SDNs most of them exploit the timing side-channel that emerges from the logic centralization of the control and its subsequent separation from the data plane. The problem when addressing those issues is that this separation is intrinsic to the nature of SNDs [1]. Consequentially, many of the simpler solutions proposed end up limiting the benefits that an SDN design offers in order to try and diminish this side channel.

The first group of solutions that will be addressed are the various solutions to tackle *Teleportation attacks* and how to avoid the creation of covert communication channels. The main issue that needs to be solved is allowing the compromised host/switch to influence the amount of time it takes from another compromised host/switch to preform a certain operation. These attacks mainly rely on the sender host/switch being able to impersonate the receiver

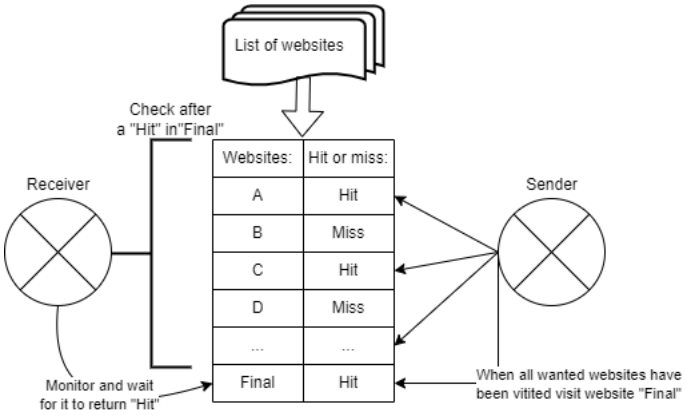


Figure 4: Diagram of the purposed showing the communication between 2 hosts using a pre-approved list of websites. The sender is visiting websites designated with a HIT and the sender will check what websites have been visited after it detects a visit to website "final".

host/switch in the eyes of the controller to be able to influence the flow rules of the switch[8] [9] that the receiver host/switch is connected to or is. Because of that solutions that prevent host/switch identity faking from influencing operations by other hosts/switch, like the ones proposed by [9] 3.1 are extremely effective. However some of these solutions compromise the topographic flexibility of SDNs. On the other hand there are solutions like the ones targeted at attacks similar to the one covered in [8] and covered in section 3.2 have the down side of making the network less usable for a host that is experiencing connection issues. When it comes to stopping teleportation attacks a valid and worth while solution is the use of algorithms like the one proposed in [9] to detect and shut down suspicious behaviour.

Now let's focus on solutions proposed in order to stop *two stage reconnaissance* timing side-channel attacks. One of the simplest and yet effective solutions to these attacks is artificially adding delay to the first packet of a flow to simulate the flow rules being installed even if they are already present[2]. However the simple solutions proposed on these attacks all have the downside of either limiting network flexibility or slowing down the network by adding latency. The papers consulted [4] [7] seems to concur that a more fine grain option is required that detects and filters (or gives lower priority) to flows that are acting in abnormal ways.

5 New attack Proposal

In this section, a new possible attack of the *teleportation* category will be explained. This attack proposal takes inspiration from the ones proposed in [2] and [9], as it uses the same principle used in that paper to determine if a user has visited a certain website in order to create a communication between the two hosts. As with all the attacks of this kind we are assuming the attacker has control of both hosts but wants to send information from one to the other while circumventing all network security.

In order to do this, the receiver would check if the sender has recently visited a certain per accorded website (this would be done using the same method that was used on the *Flow reconnaissance* [2] attack). If the website in question has been visited then this will be interpreted by the sender as a 1, and if not a 0.

This attack at first seems quite inefficient as there are other attacks that can achieve similar goals (for example [9]), however, the strength of this attack lies in the fact that if there is an agreement on a list of websites/web-servers that are being used to transmit the message could be sent without having to use time coordination between the two compromised hosts (no frames[8]). As an example the sender and the receiver could have agreed on websites A, B, C, D and E (in this case E is the "*final*" website seen in 4) to represent a 4 bit message and the sender could visit websites A, C and E to modulate the message 1010, the sender could then check if what websites have been visited in order to receive the message. The receiver would know if the message has been sent by using the binary value represented by website E, acting as a sort of signal for the receiver indicating to it that the message is ready to be read. This action by the receiver does not need to temporally coincide with when the sender sent the message, so this check could be done periodically until the website returns a "hit". In order to determine how often the website needs to be checked, how fast all the websites need to be visited by the sender or how many bits the message can contain *recognisance attacks* like [4] can be performed. The objective of these *recognisance attacks* would be to check how far back can a website visit be detected, be it in time or in other website visits.

This attack would excel in the sending of relatively short bit strings (the length depending on the results of the *proving attacks*) containing valuable pieces of information (such as private RSA keys) that the attacker would like to transmit covertly. Another situation where this attack could see some use is in situations where it is not known when the sender will be able to send the information making time coordinating between the *sender* and the *receiver* exceedingly difficult since it allows for the receiver to periodically check for the message.

The solutions for this attack would be the same as the ones mentioned previously 3.3 [2] as they would stop the receiver from being able to tell the websites the sender has visited and therefore shutting down the communication channel.

6 Responsible Research

When dealing with cybersecurity in academia it is important to keep in mind that the information that you are sharing can be sued to cause harm, because of that this paper makes a point of sharing both the vulnerabilities and solutions to said vulnerabilities. In addition to that the fact that this paper is mainly an academic survey of the state-of-the-art solutions that have been published in academia, and therefore a great majority of the given information could also be accessed by reading other already published papers (that also provide their solutions) makes it of more value for academic research than for would-be malicious actors.

7 Discussion and Future Research

Over the course of this paper, it has become evident that there is a vulnerability being exploited by attackers when it comes to side-channels in SDNs. A lot of the proposed solutions end up limiting what made an SDN network so appealing, that being its flexibility and logically centralized control plane [6]. In the future, there are two avenues of research that if given more time are worth expanding upon in order to attempt to solve this issue. Firstly looking into machine learning defence measures and secondly investigating further anti header spoofing mechanisms.

When referring to machine learning security measures there is promising research being done. In the paper, [17] big data machine learning (ML) is used in order to detect threats to a traditional network. As opposed to traditional means, which they claim to be inefficient when dealing with large amounts of data (such as the data flowing through a decently sized network), they use the emerging field of machine learning cybersecurity to better protect the network. In the paper [17] they apply the techniques to a traditional network, however, due to the promising results of published [17] and those of other papers starting to look into ML security in SDNs [18] the field of ML security in SDN networks is a promising direction to go. This is particularly promising when referring to side-channel attacks, as most if not all attacks need to use abnormal packet streams and patterns that are when ML algorithms excel in detecting. Researching how well machine learning can detect these patterns and how attackers can avoid this detection would be an intriguing research focus.

Secondly, as mentioned, another interesting research opportunity would be looking into a way to make sure no ID (Mac address, DIDP or IP addresses) or another header spoofing can occur, therefore shutting down a considerable number of attacks [8][4][2][9] or making them harder to execute. There has been research on this topic for years now [19] and there has not been a specific all-encompassing solution, however, SDN networks with their centralized control logic offer an exciting new possibility by monitoring the topology of the network to monitor the position in the network and detect location jumps that could be caused by an address spoof. Finding a solution to this problem that maintains network flexibility is something that would bring great value.

8 Conclusions

After analyzing the current state-of-the-art attacks and solutions for side-channel attacks on SDNs it can be determined that one of the biggest upsides to SDNs is also one of its main vulnerabilities, the logically centralized controller. The attacks abuse the switches' necessity to sporadically communicate with the controller in order to gain information.

As established earlier there are two main broad categories of side-channel attacks are directed at SDNs that are currently being discussed. The first one being a covert channel communication (or *teleportation*) exploit where a host can affect the controller in such a way that it affects the time it takes for another switch to complete an operation that can be used to modulate a covert communication channel.

The second one is information gathering attacks, whereby by sending packets with different headers and measuring the response time you can extrapolate information about the flow rules and configuration of the switch or the controller that can be used for future attacks.

The first type of attack can be avoided by stopping the actions of a host from influencing the time it takes for another host to perform an action. The second one is a more complex issue to fix as it uses the own functions of the controller to create the situations the attacker desires, a possible solution for this type of attack (as proposed in the section relating to *Discussion and Future Research*) would be a machine learning approach in order to detect hosts or switches that are taking part in suspicious and possibly malicious behaviour.

References

[1] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[2] S. Liu, M. K. Reiter, and V. Sekar, "Flow reconnaissance via timing attacks on sdn switches," in *2017 IEEE 37th international conference on distributed computing systems (ICDCS)*, pp. 196–206, IEEE, 2017.

[3] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 623–654, 2016.

[4] M. Conti, F. De Gaspari, and L. V. Mancini, "A novel stealthy attack to gather sdn configuration-information," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 2, pp. 328–340, 2018.

[5] G. Joy Persial, M. Prabhu, and R. Shanmugalakshmi, "Side channel attack-survey," *Int J Adva Sci Res Rev*, vol. 1, no. 4, pp. 54–57, 2011.

[6] F. Shoaib, Y.-W. Chow, and E. Vlahu-Gjorgievska, "Preventing timing side-channel attacks in software-defined networks," in *2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, pp. 1–6, IEEE, 2021.

[7] M. Zhang, G. Li, L. Xu, J. Bi, G. Gu, and J. Bai, "Control plane reflection attacks in sdns: New attacks and countermeasures," in *International Symposium on Research in Attacks, Intrusions, and Defenses*, pp. 161–183, Springer, 2018.

[8] R. Krösche, K. Thimmaraju, L. Schiff, and S. Schmid, "I dpid it my way! a covert timing channel in software-defined networks," in *2018 IFIP Networking Conference (IFIP Networking) and Workshops*, pp. 217–225, 2018.

[9] A. Sabzi, L. Schiff, K. Thimmaraju, A. Blenk, and S. Schmid, "Macchiato: Importing cache side channels to sdns," in *Proceedings of the Symposium on Architectures for Networking and Communications Systems*, pp. 8–14, 2021.

[10] S. Hong, L. Xu, H. Wang, and G. Gu, "Poisoning network visibility in software-defined networks: New attacks and countermeasures.," in *Ndss*, vol. 15, pp. 8–11, 2015.

[11] M. Dhawan, R. Poddar, K. Mahajan, and V. Mann, "Sphinx: detecting security attacks in software-defined networks.," in *Ndss*, vol. 15, pp. 8–11, 2015.

[12] A. Mahimkar, J. Dange, V. Shmatikov, H. M. Vin, and Y. Zhang, "dfence: Transparent network-based denial of service mitigation.," in *NSDI*, vol. 7, pp. 327–340, 2007.

[13] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments," *Computer Networks*, vol. 62, pp. 122–136, 2014.

[14] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," in *International workshop on recent advances in intrusion detection*, pp. 161–180, Springer, 2011.

[15] S. W. Shin, P. Porras, V. Yegneswara, M. Fong, G. Gu, and M. Tyson, "Fresco: Modular composable security services for software-defined networks," in *20th annual network & distributed system security symposium*, Ndss, 2013.

[16] M. Ambrosin, M. Conti, F. De Gaspari, and R. Poovendran, "Lineswitch: Efficiently managing switch flow in software-defined networking while effectively tackling dos attacks," in *Proceedings of the 10th ACM symposium on information, computer and communications security*, pp. 639–644, 2015.

[17] M. Amrollahi, S. Hadayeghparast, H. Karimipour, F. Derakhshan, and G. Srivastava, "Enhancing network security via machine learning: opportunities and challenges," *Handbook of big data privacy*, pp. 165–189, 2020.

[18] T. N. Nguyen, "The challenges in sdn/ml based network security: A survey," *arXiv preprint arXiv:1804.03539*, 2018.

[19] T. Ehrenkranz and J. Li, "On the state of ip spoofing defense," *ACM Transactions on Internet Technology (TOIT)*, vol. 9, no. 2, pp. 1–29, 2009.