# OUTSMARTING NETWORK SECURITY WITH SDN TELEPORTATION

## SEMINAR REPORT SUBMITTED BY

JAIKISHAN K C

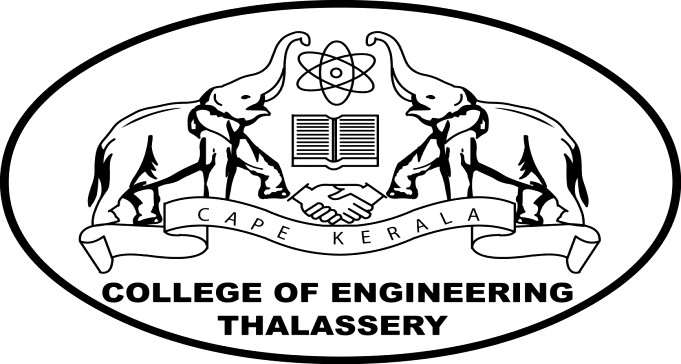
**(12152011)**

*In partial fulfillment of the requirements for the award of Degree of*

*BACHELOR OF TECHNOLOGY*

*In*

*COMPUTER SCIENCE AND ENGINEERING*

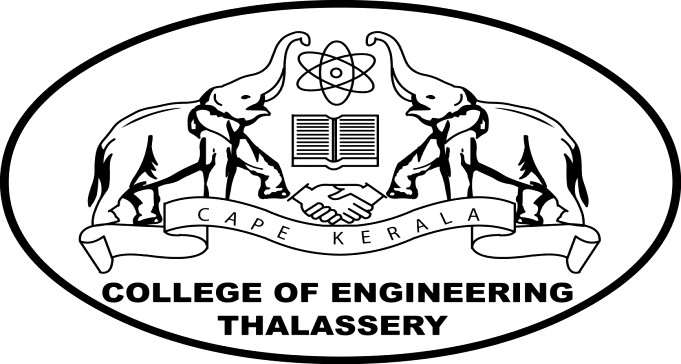


**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

# COLLEGE OF ENGINEERING THALASSERY

**(Engineering College Under CAPE, Estd by Govt. Of Kerala)**

**OCTOBER 2017**



# COLLEGE OF ENGINEERING THALASSERY KANNUR DT:-670107

**CERTIFICATE**

### This is to certify that the seminar report entitled ***“Outsmarting Network Security with SDN teleportation ”*** is a bonafide account of the work carried out by ***Mr. Jaikishan K.C*** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering under CUSAT during the year 2017.

**COORDINATOR GUIDE HEAD OF THE DEPARTMENT**

Ms. Binitha S. Mrs. Reshma T.V Mrs. Ambili M.P.

Asst. Professor Asst. Professor Asst. Professor

Dept of Computer Science Dept of Computer Science Dept of Computer Science & Engineering & Engineering & Engineering

# ACKNOWLEDGEMENT

I owe my heartfelt gratitude to Almighty for all the blessings showered on me during this work. I take this opportunity to express my sincere gratitude to all the people who have been instrumented in bringing out this work to the correct form.

I would also like to express my sincere thanks to Prof.Dr.Joseph O.A. the Principal, College of Engineering Thalassery, for rendering me his kind permission to do this seminar and providing me with all facilities to complete it successfully.

I am very much thankful to Asst.Prof.Mrs.Ambili M.P. Head of the Department, Computer Science and Engineering, for her timely help and encouragement.

I would like to thank Seminar Coordinator Mrs**.**Binitha S. Assistant Professor, Department of Computer Science and Engineering, for her valuable support.

I would like to express my sincere thanks to Seminar supervisor, Mrs Ambili M P Assistant Professor, Department of Computer Science and Engineering, for her valuable suggestions and guidance.

I also thank the staff members of Computer Science and Engineering Department, for their cooperation and support.

Finally i thank my family, friends, classmates for providing me the strength and endurance.

### Jaikishan K.C

# ABSTRACT

Software-deﬁned networking is considered a promis- ing new paradigm, enabling more reliable and formally ver- iﬁable communication networks. However, this paper shows that the separation of the control plane from the data plane, which lies at the heart of Software-Deﬁned Networks (SDNs), introduces a new vulnerability which we call *teleportation*. An attacker (e.g., a malicious switch in the data plane or a host connected to the network) can use teleportation to transmit information via the control plane and bypass critical network functions in the data plane (e.g., a ﬁrewall), and to violate security policies as well as logical and even physical separations. This paper characterizes the design space for teleportation attacks theoretically, and then identiﬁes four different teleportation techniques. We demonstrate and discuss how these techniques can be exploited for different attacks (e.g., exﬁltrating conﬁdential data at high rates), and also initiate the discussion of possible countermeasures. Generally, and given today’s trend toward more intent-based networking, we believe that our ﬁndings are relevant beyond the use cases considered in this paper.

# TABLE OF CONTENTS

CH. NO TITLE PAGE NO ACKNOWLEDGEMENT i

[ABSTRACT ii](#_TOC_250013)

1. [INTRODUCTION 1](#_TOC_250012)
2. PRELIMINARIES……………………………………………….5
3. THREAT MODELS……………………………………………..7
4. MODELLING TELEPORTATION……………………………..8
5. TELEPORTATION TECHNIQUES……………………………9
6. SWITCH-AND-HOST BASED ATTACKS…………………..15
7. COUNTERMEASURES……………………………………….23
8. CONCLUSIONS……………………………………………….25
9. REFERENCES…………………………………………………26

LIST OF FIGURES

### Figure 1:Illustration of firewall……………………………………….2

### Figure 2: Path update teleportation………………………………….11

### Figure 3: Path reset teleportation……………………………………12

### Figure 4: Bypassing critical network functions……………………..16

### Figure 5: Rendezvous and malicious switches discovery…………..17

### Figure 6: Exfiltration………………………………………………..20

### Figure 7: Man- in- the middle..............................................................22

# 1. INTRODUCTION

Computer networks such as datacenter networks or the Internet have become a critical infrastructure . Not only a large fraction of the economic activity critically depends on the availability of such networks, but also governments increasingly rely on existing and shared infrastructures, mainly for their cost beneﬁts.

This dependency on public and shared infrastructures raises concerns. While the Internet has certainly been a huge success, and over the last years, there has been much innovation on the higher network layers (e.g., application layer) and the lower network layers (e.g., data-link and physical layer), the Internet core suffers from ossiﬁcation [3]. In particular, it is questionable whether today’s network technology is sufﬁcient to ensure essential security, resilience and dependability properties. For instance, today’s Internet does not provide any means of path control, and we are still struggling to render routing protocols more secure.

However, Software-Deﬁned Networks (SDNs) also introduce new security challenges. In particular, we in this paper study threats introduced by an *unreliable south-bound interface*, i.e., we consider a threat model in which switches or routers do not behave as expected, but rather are malicious and e.g., contain hardware backdoors. While many existing network security and monitoring tools rely on the trustworthiness of switches and routers, this assumption has become questionable: Attackers have repeatedly demon- strated their ability to compromise switches and routers, thousands of compromised access and core routers are being traded underground, networking vendors have left backdoors open, national security agencies can bug network equipment hacker tools to scan and eventually exploit routers with weak passwords, default settings are openly available on the Web, etc.In particular, this paper shows how an outsourced and consolidated control plane—as it lies at the heart of the SDN paradigm, introduces an opportunity for *teleportation.* Malicious SDN switches may transmit information via the logically centralized software control plane, *completely* bypassing data plane elements (such as other switches, middleboxes, etc.). By violating logical and even physical separations, teleportation can constitute a serious security threat. For example, teleportation could be used by one malicious switch to discover (and communicate information to) other malicious switches, bypassing security checks in the data plane. As we will show in this paper, teleportation can also be exploited by malicious hosts, triggering (benign) switches to teleport information for them.

We argue that teleportation can be seen as a ﬂexible communication channel which constitutes a threat in various situations, for example (see Figure 1):

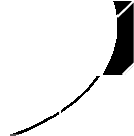
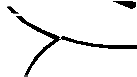
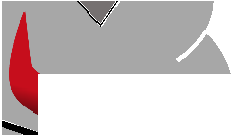
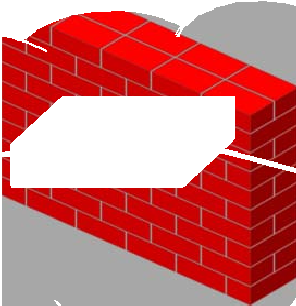
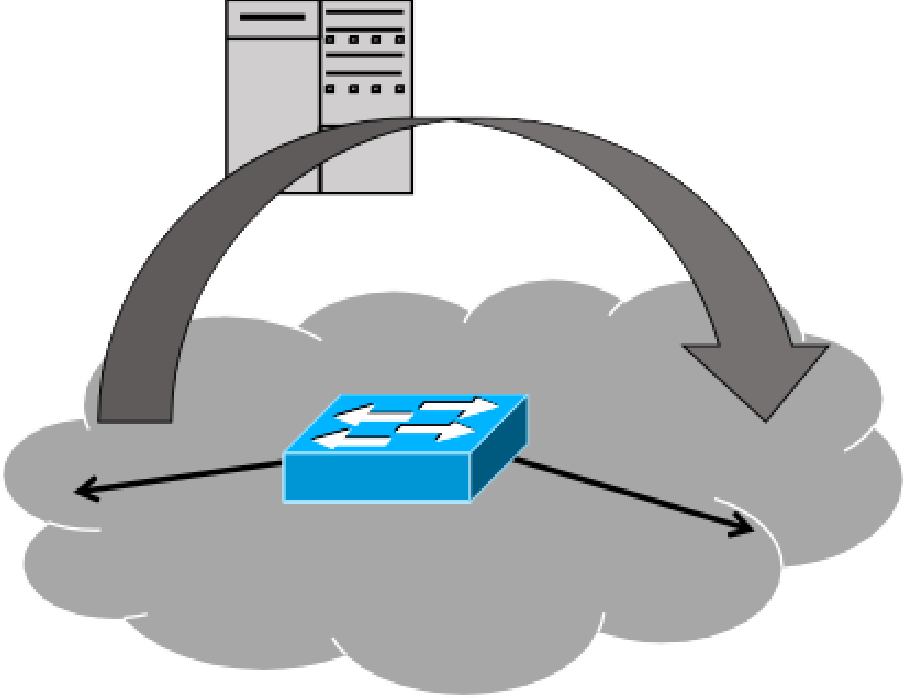


Figure 1: Illustration of teleportation: Malicious switches (with *red horns*) exploit the control platform for hidden communication, possibly bypassing data plane security mechanism such as firewall.

## **Bypassing critical network components:**

By implicitly communicating information via the control plane, it is possible to circumvent critical network components, such as switches, middleboxes or pol- icy enforcers located in the data plane. For example, teleportation can in principle be used to bypass middleboxes performing security checks (e.g., network intrusion detection systems), middleboxes in charge of billing (e.g., a Radius server), or QoS enforcers (e.g., a leaky bucket policer).

## **Rendezvous and attack coordination:**

While al- ready a single malicious switch, for example located *inside* an enterprise network, may cause signiﬁcant harm and violate basic security policies, the situa- tion becomes worse if multiple malicious switches cooperate [21]. Malicious or Trojan switches (e.g., switches containing a hardware/software backdoor) may use teleportation as a rendezvous protocol, to discover each other, and subsequently coordinate an attack.

## **Exﬁltration:**

Teleportation can also be used to exﬁltrate sensitive information between networks that have no data plane connectivity.

## **Eavesdropping and data tampering:**

Particularly serious threats are introduced if malicious hosts and switches collude. For instance, in a scenario with collusion, teleportation can be used for eavesdrop- ping. We show that a malicious switch and host can carry out a man-in-the-middle attack that serves benign hosts with malicious web pages.

Teleportation can be difﬁcult to detect: The teleported information follows the normal trafﬁc pattern of control communication, not between switches directly but *indirectly* between any switch and the controller. Moreover, the telepor- tation channel is *inside* the (typically encrypted) OpenFlow channel. Accordingly, it cannot easily be detected with modern Network Intrusion Detection Systems (NIDSs), even if they operate in the control plane.

## **Our Contributions:**

* We identify a new vulnerability, namely teleporta- tion, which targets the very core of the software- deﬁned networking paradigm, namely the separation of the control and the data plane. In particular, we consider the threats introduced by a malicious data plane. Indeed, recent incidents related to the trustworthiness of routers and switches, indicate that our threat model is a relevant one.
* We model and characterize possible teleportation channels theoretically.
* We recognize and demonstrate four teleportation techniques in software-deﬁned networks utilizing state-of-the-art OpenFlow controllers, in particular ONOS among others.
* We present and demonstrate multiple simple and sophisticated attacks. In particular, we show that teleportation can also be exploited by malicious hosts in scenarios where all switches are benign.
* We evaluate the performance and quality of the teleportation channel in terms of throughput, jitter and packet loss respectively, and also evaluate the resource footprint in terms of CPU and memory consumption at the controller.
* We initiate the discussion of possible countermea- sures. In particular, we propose to combine intru- sion detection with waypoint-enforcement, forcing *Packet-out* messages (from controller to switches) to pass through the waypoint if mandated by a security policy.

# 2. PRELIMINARIES

This paper considers Software-Deﬁned Networks (SDNs) which outsource and consolidate the control over the network switches to a logically centralized software controller. The separation of the control and data plane has the potential to simplify the network management, as many networking tasks are inherently non-local. Moreover, SDN and especially its de facto standard protocol, OpenFlow, introduce interesting new ﬂexibilities, e.g., in terms of trafﬁc steering: Routes may not necessarily be destination based, and can depend on layer-2, layer-3 and even layer-4 properties of the packets. At the heart of an SDN lies a control software, running on a set of servers. These controllers receive information and statistics from switches, and depending on this information as well as the policies they seek to implement, issue instructions to the switches.

Open Flow follows a match-action paradigm: The con- trollers install rules on the switches which consist of a match and an action part; the packets (i.e., ﬂows) matching a rule are subject to the corresponding action. That is, each switch stores a set of (ﬂow) tables which are managed by the controllers, and each table consists of a set of ﬂow entries which specify expressions that need to be matched against the packet headers, as well as actions that are applied to the packet when a given expression is satisﬁed. Possible actionsinclude dropping the packet, sending it to a given egress port, or modifying its header ﬁelds, e.g., adding a tag. The match-action paradigm is attractive as it simpliﬁes formal reasoning and enables policy veriﬁcation.By default, if a packet arrives at a switch and does not match an existing rule, the packet (usually without payload if the switch supports packet buffering) is forwarded to the controller.

This event is called a *Packet-in*. Upon a *Packet-in* event, the controller can decide how to react to packets of the corresponding type, and add/delete/modify ﬂow rules accordingly issuing *Flow-mod* messages to the switch (and maybe to other switches proactively on this occasion as well). A controller can also decide to send out a packet explicitly from a switch, issuing a so called packet out command to control switch.

An attractive alternative to the hop-by-hop installation of new ﬂows, reacting to a new packet repeatedly along the path (multiple *Packet-in*s), is the so-called “pave-path technique”: Once the controller receives a ﬁrst *Packet-in* event from some switch, it proactively updates the

Other switches along the path. Such an “intent-based” controller behavior can render the reaction to network events and set up of host-to-host/network connectivity (according to current policies) more efficient.

While SDNs are logically centralized, the control plane can be physically distributed, e.g., for fault-tolerance or performance reasons. Accordingly, Open Flow supports multiple controllers for a single switch. The controllers and switch exchange *Role-request* and *Role-reply* messages respectively to assert the various roles (*Master*, *Equal* and *Slave*). There may be only one *Master* controller for a given switch while multiple *Equal* and *Slave* controllers are permitted.

The Open Flow standard [24] speciﬁes basic security mechanisms. For example, the communication between the controller and switch can be authenticated and encrypted, using TLS over TCP/UDP.

Finally, we note that although some of our techniques are generally applicable in networks separating the control plane and the data plane, while others exploit OpenFlow speciﬁc features, when clear from the context, in this paper we will treat SDN and OpenFlow as synonyms.

# 3. THREAT MODEL

We in this paper consider a threat model where OpenFlow switches, hosts, or both, may not behave correctly but are malicious.

We do not place any restrictions on what a malicious switch can and cannot do. For example, a malicious switch can fabricate and transmit any type of OpenFlow message, it can arbitrarily deviate from the OpenFlow speciﬁcation, and it can even use multiple identities, all at the risk of being detected. However, the malicious switch cannot choose where it will be placed in the network. In order to collude, the malicious switches have been programmed to recognize some data and/or timing pattern. Similarly, we do not place any restrictions on what a malicious host can and cannot do. For example, a malicious host may masquerade its Media Access Control (MAC) and/or Internet Protocol (IP) addresses, use an incorrect gateway, falsify Address Resolution Protocol (ARP) requests/responses, and so on. The attacker could also be an insider, i.e., an authorized user who intends to subvert his/her current organization. We also consider the case where malicious hosts and switches collude. We assume that an attacker has sufﬁcient resources and know- how to compromise hosts/switches and therefore do not concern ourselves with how the host/switch is compromised. For example, the attacker can exploit a buffer overﬂow vulnerability in the switch software to compromise the switch.

The OpenFlow controller and its applications on the other hand are trusted entities and are available to the switches: For example, they are based on static and dynamic program analyses. The OpenFlow channel is reliable and may be encrypted.

# 4. MODELING TELEPORTATION

With these concepts in mind, we now model and charac- terize a novel threat called *teleportation* which targets the heart of SDNs: The outsourcing and consolidation of control over multiple data plane elements. In particular, we argue that we can see an OpenFlow controller as a “reactor”: It reacts (in a best-effort and timely manner) to events generated by the network operator, the OpenFlow switches, and timeouts; as a response, the controller sends OpenFlow commands to switches. Accordingly, we argue that the following 3-stage functionality is fundamental in the SDN paradigm.

**Switch to controller:**

A source switch intentionally or unintentionally sends modulated information to the controller (e.g., by adding speciﬁc events, delaying existing events, etc.).

**Controller to switches:**

The controller reacts to the received events, by sending commands to one or multiple other switches.

**Destination processing:**

A destination switch pro- cesses incoming commands from the controller. In case of a malicious switch, the switch may search for some message properties, temporal patterns, etc., and hence infer the information modulated by the source, or by simply forwarding the information (to a potentially malicious host).

Based on this controller model, we can identify two kinds of teleportation channels.

**Explicit teleportation:**

The teleported information actually appears in the messages exchanged. The message may for example contain steganographic contents.

**Implicit teleportation:**

The teleportation relies on modulating information implicitly. For example, it is based on timing (e.g., message transmissions are delayed according to some pattern) or it is based on shared resources, whose availability is changed over time (e.g., leveraging mutual exclusion).

# 5. TELEPORTATION TECHNIQUES

Having established a conceptual model of teleportation, we next present techniques that can realize teleportation in today’s SDNs. In particular, we have identiﬁed the following three fundamental SDN functionalities which can be exploited for teleportation:

**Flow (re-)conﬁgurations:** In an SDN, a controller needs to react to various data plane events (such as so-called *Packet-in*s in OpenFlow or link failures), and conﬁgure and reconﬁgure ﬂows and paths accordingly. Triggering and exploiting such events can be used for teleportation.

**Switch identiﬁcation:** In an SDN, switches are responsible for introducing and uniquely identifying themselves to the controller. This is required as policies are often speciﬁc to the switch. Unique switch identiﬁers are also necessary to correctly construct and enforce policies on the switches and in the controller. We will show that such switch identiﬁcation mechanisms can be exploited for teleportation.

**Out-of-band forwarding:** An SDN controller must not only be able to receive events and control packets from switches, but also to instruct switches to forward speciﬁc messages. This basic functionality in SDNs can be exploited by a malicious switch or host to forward entire packets via the controller

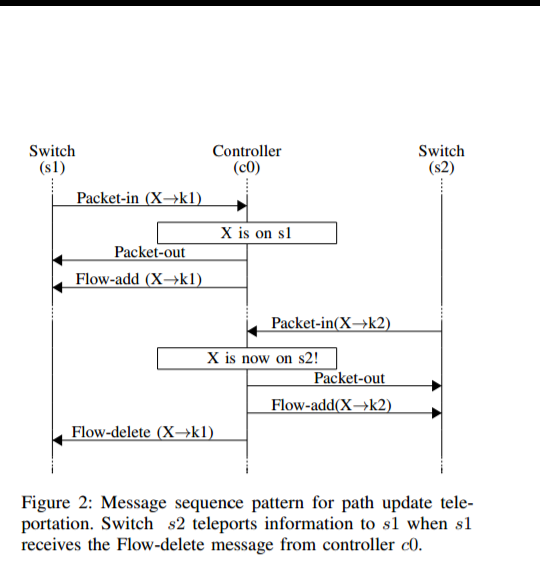
## Flow (Re-)Conﬁgurations

We distinguish between two types of ﬂow re- conﬁguration events: *path update* and *path reset*.

**Path Update:** Our ﬁrst teleportation technique is based on *path updates*. Path updates are a fundamental controller functionality, and come in the form of different controller features such as *Mobility*, *VM Migration* or simply *MAC Learning*. The basic scheme is as follows: A controller typically maintains some mapping of which hosts (MAC addresses) are connected to which ports (on the switch). If a host suddenly appears on another switch, the controller installs new ﬂows for the host on the new switch, and also deletes the corresponding ﬂow rules on the old switch. We deﬁne this type of installation and deletion of ﬂows by the controller on switches as *path update*. Speciﬁcally, a path update involves the use of *Packet-in*, *Flow-mod* and *Packet- out* messages. Malicious switches can use path update for implicit teleportation.

For the teleportation with path update, a switch triggers the deletion of rules at other switches. Malicious switches can teleport information between themselves by prompting path updates for the same host using *Packet-in*s. Note that during a path update, the *Packet-out* is be sent to the destination reported in the *Packet-in* which may generate data plane trafﬁc. To prevent data plane trafﬁc, the malicious switch can use a destination host that is connected to itself (so that the *Packet-out* is sent back to it). The message sequence pattern for path update teleportation is shown in Figure 2.

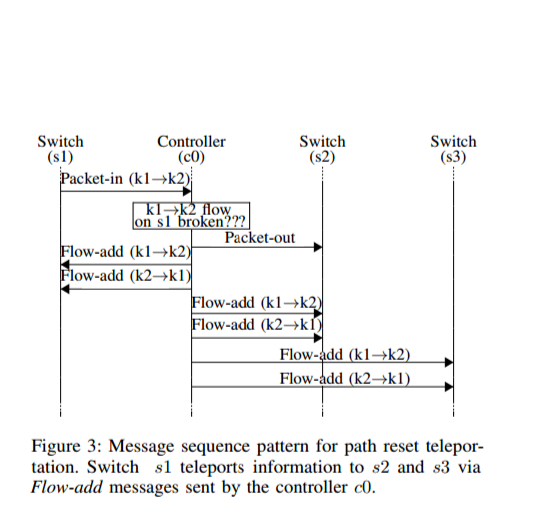
We can summarize the scheme presented so far with the following abstract steps: A switch *s*1 announces *X*, a switch *s*2 announces *X* thereby stealing *X* from *s*1, where stealing is detected by the “victim” ( *s*1). Also note that announcing is possible once some host which is connected to the malicious switch (e.g., *k*1 at *s*1) is learned by the controller.



**Path Reset:** We next discuss a second ﬂow reconﬁguration based teleportation which we refer to as *path reset*. Recall that at the heart of any SDN controller lies the functionality to set up host-to-host/network connectivity, ac- cording to the network policy (e.g., deﬁning constraints such as bandwidth, link type and waypoints), which is translated into device level conﬁgurations (e.g., ﬂow rules). The “pave- path technique” is an attractive alternative to the hop-by-hop installation of new ﬂows: Once the controller receives a ﬁrst *Packet-in* event from some switch, it proactively updates the other switches along the path.

In order to provide high availability, a controller also monitors the network state and makes necessary changes, such as rerouting or resetting ﬂows on switches, when needed (e.g., due to a link failure). For example, triggered by a *Packet-in* event, a controller may learn that (parts of) the path may no longer be available, and hence initiates the reconﬁguration/repair of the path. We deﬁne the reinstallation of ﬂows by the controller on switches along a path as *path reset*. Accordingly, the path reset technique involves *Packet- in*, *Flow-mod* and *Packet-out* messages.

Malicious switches may use path reset for implicit teleportation: If the controller resets the complete path between hosts when it receives a *Packet-in* from a switch that ignores the ﬂow rule, then information can be communicated. By doing this at multiple and speciﬁc times, a malicious switch can teleport information to other malicious switches along the path. Figure 3 illustrates the message sequence pattern for teleportation using path reset.



## Switch Identiﬁcation

This teleportation type exploits the fact that a switch typically must uniquely identify itself whenever it connects to the controller. For example, in OpenFlow this is usually done using the Datapath-ID (DPID) ﬁeld in the *Features- reply* message. We deﬁne two switches attempting to use the same DPID to connect to the same logical controller as *switch identiﬁcation*. The outcome can be used for implicit teleportation.

Three basic ways an OpenFlow controller can react to using the same DPID are as follows:

* The controller denies the second switch a connection.
* The controller terminates the ﬁrst switch and connnects to the second.
* The controller accepts both switches but sends them different *Role-request* messages.

With any of the above outcomes, a switch can infer the (mis)use of the same DPID by another switch. By using a-priori conﬁgured single or multiple DPID values, a pair of malicious switches can establish teleportation. For example, consider the message sequence pattern in Figure 4, and assume that ﬁrst switch *s*1 tells controller *c*0 that its DPID is 1. At a later time, switch *s*2 tells *c*0 that its DPID is 1. At this point, *c*0 does not allow *s*2 to connect with DPID 1. Since *c*0 denied *s*2 to connect with DPID 1, *s*1 teleported information to *s*2 via *c*0. With a similar message sequence pattern, the second outcome can be used for teleportation as well.

Interestingly, switch identiﬁcation is not limited to scenar- ios with a single controller: We have found additional threats in the presence of *distributed control planes*. Moreover, we can generalize the ﬁrst *switch identiﬁcation* outcome to scenarios with *m* malicious switches, see the event-handler algorithm, Algorithm 2. The other two outcomes discussed can also be seen as event-handler algorithms.

## Out-of-band Forwarding

The third and potentially most powerful teleportation technique is called *out-of-band forwarding*. It is an example of explicit teleportation. Out-of-band forwarding exploits the fact that an SDN controller is typically connected to multiple switches: Accordingly, a packet from one switch can potentially reach multiple other switches in the network via the control plane. Out-of-band forwarding involves a *Packet-in* from one switch and a *Packet-out* message at another switch, with the possible side effect of *Flow-mod* messages on the switch that sent the *Packet-in* message. Out- of-band forwarding could for example include the complete Ethernet frame (typically 1500 bytes), and can even serve as a “multicast service”. Out-of-band forwarding can be a serious threat to network security, not only because malicious trafﬁc can bypass critical security functions in the data plane, but also because it can be exploited by switches and hosts. Figure 6, illustrates the message sequence pattern for teleportation using out-of band forwarding.

# 6.SWITCH-AND HOST BASED ATTACKS

We now demonstrate how the identiﬁed teleportation techniques can be exploited to carry out speciﬁc attacks. In particular, we show how teleportation may be exploited.

1) To bypass critical network functions such as Firewall and NIDS.

2) As a rendezvous protocol for malicious switches;

3) To exﬁltrate sensitive data from remote locations;

4) To conduct a man-in-the-middle (mitm) attack.

## **Bypassing Critical Network Functions**

We believe that the possibility to bypass network ele- ments is a serious threat in modern computer networks. For example, many network policies today are deﬁned in terms of adjacency matrices or big switch abstractions, specifying which trafﬁc is allowed between an ingress port *s* and an egress network port *t* [26]. In order to enforce such a policy, trafﬁc from *s* to *t* needs to traverse a middlebox instance (waypoint) inspecting and classifying the ﬂows. The location of every middlebox may be optimized, but is subject to the constraint that the route from *s* to *t* should always go via the waypoint.

**Firewall and NIDS.** In order to demonstrate how a ﬁrewall may be circumvented by hosts (or switches), we set up Mininet and ONOS as shown in Figure 7. The switches do not have ﬂow rules for *k*1 and *k*2 to communicate. The ﬁrewall *fw*1 prevents hosts on the left to communicate with hosts on the right and vice-versa. ONOS has the *Intent Reactive Forwarding* (*ifwd*) application enabled. *ifwd* uses the reactive “pave-path technique” (discussed above) to install ﬂows in the switches. By default, the *ifwd* application establishes host-to-host connectivity when it receives a *Packet-in* for which no ﬂows exist.

We send a ping packet from *k*1 to *k*2. Despite the presence of the ﬁrewall, *k*1 receives the reply from *k*2 using out-of-band forwarding teleportation. In the absence of out- of-band forwarding teleportation, the packet would have been dropped by *fw*1.

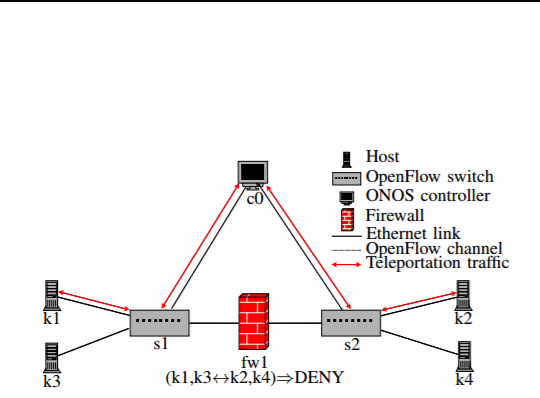


Figure 4: An SDN topology with OpenFlow switches *s*1 and *s*2 and an OpenFlow controller *c*0 (ONOS). *k*1 and *k*3 are connected to *s*1 while *k*2 and *k*4 are connected to *s*2. *s*1 and *s*2 are separated by a ﬁrewall *fw*1 that denies hosts on *s*1 to communicate with hosts on *s*2 and vice-versa. *k*1 can use *out-of-band forwarding teleportation* to transfer data to *k*2, bypassing *fw*1.

Similar to the ﬁrewall scenario, we can also use out-of- band forwarding teleportation in the presence of *Snort*, an NIDS. In particular, we can generate attack trafﬁc using nmap to conduct TCP ﬂag attacks or even port scans. Indeed, by masquerading the source MAC address, one can effectively carry out a wide enough port scan without having the scan pass through the ﬁrewall and being detected by the latter.

By replacing the ﬁrewall we previously described with *Snort*, we use nmap from *k*1 to carry out a TCP port scan on *k*2 using out-of-band forwarding teleportation. By inspecting the alerts in *Snort* we veriﬁed that no alerts were generated for the port scan.

Note that the host-to-host connectivity setup involves a *Packet-in* and *Flow-mod* messages whereas the out-of-band forwarding teleportation only involves *Packet-in* and *Packet- out* messages with the side effect of *Flow-mod* messages. Therefore, security policy enforcers that do not inspect and correlate *Packet-in* with *Packet-out*s, will miss out-of-band forwarding teleportation based attacks. Of course, violating *Flow-mod*s may eventually be detected, but only after the data has been teleported.

## **Rendezvous and Malicious Switch Discovery**

We next consider a rendezvous protocol in which mali- cious switches wish to discover one another. A rendezvous or discovery protocol can be also seen as a precursor to a much more damaging attack such as a denial-of-service, exfiltration.

Teleportation can be an attractive solution: A rendezvous protocol can rely on steganography, i.e., embedding patterns in teleported benign information or modulating patterns in legitimate messages. Without teleportation, by going through the data plane directly, the malicious switches risk detection.

We show how three of our techniques, namely path update, path reset and switch identiﬁcation teleportation may be used as a rendezvous protocol for malicious switches.

**Path Update.** To demonstrate a rendezvous with path update teleportation, we set up Mininet and ONOS as shown in Figure 8. Instead of instrumenting code for the malicious switches, we keep them as simple Open vSwitches and we deﬁned dedicated Mininet hosts (*k*3 and *k*4) for each of them. We use the dedicated hosts (*k*3 and *k*4) to generate the packet that the malicious switch sends as a *Packet-in* to the controller. The *host mobility* and *ifwd* applications are enabled on ONOS. The controller has already installed ﬂows for *k*1 to *k*2 and vice-versa. Accordingly, we use *k*4 connected to *s*2, to send *k*2 a packet using *k*1 as the source MAC address. This triggers the controller to issue *Flow- mod* commands to *s*1, *s*2 and *s*3. *s*2 thereby teleported its presence to *s*1.

**Host**



**Open flow switch**

**Ethernet link**

**Host to host traffic**

**Teleportation traffic**

Figure 8: An SDN topology of OpenFlow switches *s*1, *s*2, *s*3 and *s*4, OpenFlow controller *c*0 (ONOS). Hosts *k*1 and *k*3 are connected to *s*1 and *k*2 and *k*4 are connected to *s*2. *c*0 has installed ﬂows on *s*1, *s*3 and *s*2 so that *k*1 and *k*2 can communicate bi-directionally. Teleportation trafﬁc is via *c*0.

By inspecting the ﬂows on the switches, we veriﬁed the successful path update teleportation: *s*2 was able to cause a ﬂow deletion in *s*1 without exchanging any packets with *s*1 directly (except for a normal ﬂow in the past).

Note that path update may trigger alerts in systems that keep track of moving MAC addresses by inspecting *Packet- in* and *Flow-mod* messages. In such cases, many moving MAC addresses may introduce suspicious activity within the network. Also worth noting is that port-based security (that associates MAC addresses with speciﬁc ports) may not be applicable in the presence of malicious switches.

**Path Reset.** To demonstrate that path reset teleporta- tion can be used as a rendezvous protocol, We modulate trafﬁc between *k*1 and *k*2 using ping packets with 100 microsecond intervals. Instead of manipulating the Open vSwitch code for sending a *Packet-in* for an existing ﬂow from *s*1, we simply remove the ﬂow for *k*1 to *k*2 on *s*1, using the ovs-ofctl del-flow command. This causes *s*1 to send *c*0 a *Flow- removed* message which triggers the controller to add the ﬂow back onto *s*1. But due to the high rate of ping trafﬁc, at least one packet triggers a table-miss before *s*1 adds the ﬂow and a *Packet-in* is sent to *c*0.

When *c*0 receives the *Packet-in* it sends the packet to *s*2 directly as a *Packet-out*, bypassing *s*3, and then sends *Flow mod*s to *s*1, *s*2 and *s*3 resetting the bi-directional path between *k*1 and *k*2. By checking the lifetime of the ﬂow ruleson S1,S2,S3 we verified that path reset teleportation succeeded. In this manner, *s*1 teleported its presence to *s*2 by having the controller send *Flow-mod* commands for existing flow rules.

**Switch Identiﬁcation.** We now demonstrate how two malicious switches may teleport their presence using switch identiﬁcation. We set up Mininet and ONOS as shown in Figure 8 with only *s*1, *s*3 and *s*4 having connected to *c*0 with DPID 1, 3 and 4 respectively. Also, there are no ﬂows installed on the switches for hosts to communicate. We modiﬁed the Mininet script to conﬁgure *s*2 with the same DPID as *s*1.

When *s*2 tries to connect to *c*0 with DPID 1 after *s*1 has connected to *c*0, it is denied a connection. This way, *s*1 teleports its presence to *s*2.

In Floodlight and OpenDaylight, when *s*2 attempts to connect to *c*0 with DPID 1 after *s*1 has connected, Floodlight terminates the connection with *s*1 and accepts *s*2’s connec tion. *s*2 thereby teleports its presence to *s*1. Interestingly RYU allowed switches with the same DPID to co-exist which potentially introduce additional issues.

## **Exﬁltration**

Our next attack is related to data exﬁltration. This is a key concern for many organizations that own intellectual property, personal data or any kind of sensitive information. Once an attacker gets into a network, one possible goal of the attacker is to stealthily exfiltratr sensitive data.

We demonstrate exﬁltration by considering a scenario where a small number of hosts are networked together in a remote location. The data plane isolation is meant to improve security. However the data plane elements are managed by a controller that handles other similar remote locations. We show that in such a network, not only malicious switches can exﬁltrate data using out-of-band forwarding teleportation but even malicious hosts.

We set up Mininet and ONOS as shown in Figure 10. ONOS has the *ifwd* application activated. By showing how *k*2 can exﬁltrate data to *k*1, we also demonstrate how *s*2 can exﬁltrate data to *k*1 or *s*1.

Given that *s*1 and *s*2 do not have ﬂow rules for trafﬁc from *k*2 to *k*1 (as they are located in disconnected data planes), *k*2 can exﬁltrate data to *k*1 by simply sending a packet (e.g., UDP packet) to *k*1 thereby exploiting out-of- band forwarding teleportation. The controller will receive the packet from *s*2 and send it to *s*1 which will then forward the packet to *k*1.



Figure 10: An SDN topology with OpenFlow switches *s*1, *s*2, *s*3 and *s*4 and an OpenFlow controller *c*0 (ONOS). *k*1 and *k*3 are connected to *s*1 while *k*2 and *k*4 are connected to *s*2. Note that *s*2 is not connected to the other switches, and thereby is isolated in the data plane. *k*2 can still exﬁltrate data to *k*1 using *out-of-band forwarding teleportation* attacking the data plane isolation.

## **Evading Policy Conﬂicts**

For an attacker, remaining stealthy is key to persistent existence. One of the side effects of using the out-of-band forwarding teleportation is the *Flow-mod* messages issued by the controller. The *Flow-mod* messages may generate policy conﬂicts (unauthorized/conﬂicting ﬂow rules), alerting the administrator. A stealthier version of using the out-of-band forwarding teleportation would be to prevent the *Flow-mod* side effect. This would not only prevent policy conﬂicts, but also leave minimal traces on the source and sink switches.

*k*2 can exﬁltrate data to *k*1 using out-of-band forwarding teleportation without triggering *Flow-mod*’s on *s*2 and *s*1 by masquerading its source MAC address *and* ETHER TYPE.

If the packet processor and intent framework cannot correctly identify a packet, their behavior may violate security policies. Note that it is enough if the ETHER TYPE is set to a value that ONOS does not recognize, and we are not restricted to Jumbo frames only.

**Remark on a Denial-of-service Attack.** Interestingly, we observed that a side effect of our out-of-band forwarding teleportation is a novel denial-of-service attack. If in our evading policy conﬂicts example, the host sends the same packet (Jumbo frame) again, then *ifwd* encounters a null- pointer exception and disconnects the switch that sent it the packet. This shows how a malicious host can cause the switch it is connected to, to be disconnected from the controller even when a packet it sends is not corrupted.

We emphasize that this is a side effect of out-of-band forwarding teleportation only, and not a teleportation issue in itself. Fortunately, the issue has been resolved by the ONOS community after we contacted them.

## **Man-In-The-Middle**

While we have so far focused on attacks where either only switches or only hosts are malicious, we now detail an attack that involves a malicious switch and a malicious host. The damage of such a collaboration can be severe,for example, the attackers could serve benign hosts with malicious web pages. In order to exemplify the attack we use HTTP rather than HTTPS.

For this attack, we set up Mininet and ONOS with *ifwd* activated as shown in Figure 12. *s*1 and *k*2 are both malicious while the others are not. *k*3 is a benign web server. *s*1 teleports speciﬁc HTTP trafﬁc towards *k*2. *k*2 modiﬁes the HTTP trafﬁc and teleports it back to *s*1 who then forwards it to *k*1. In order to emulate the malicious switch, we introduced a ﬂow rule (shown in Listing 1) that rewrites the destination MAC address for TCP trafﬁc with PSH and ACK ﬂags sent from *k*3 to *k*1, to *k*2. This modiﬁed packet is then passed through the ﬂow table lookup again by using the resubmit action in Open vSwitch. *k*2 runs *ettercap* to modify the TCP/HTTP payload and forwards the packet to the correct destination. Speciﬁcally, we created an *ettercap* ﬁlter that looks inside HTTP responses from *k*3 for the word “good”, replaces it with “evil”, and sends it to *k*1. The ﬁrewall *fw*1 is meant to block trafﬁc between hosts on the right and the left.

When *k*1 requests the index.html page from *k*3, based on the ﬂow rule installed on *s*1, only HTTP responses from *k*3 are teleported to *s*2 and forwarded to *k*2, through the out-of-band forwarding teleportation. Subsequently, *k*2 modiﬁes only the index.html web page and has *s*2 teleport it back to *s*1 via out-of-band forwarding teleportation. Indeed, the side effect is *Flow-mod* messages to *s*1 and *s*2.



# 7. COUNTERMEASURES

Having showcased a variety of attacks using teleportation, we now start exploring possible countermeasures. Although we have demonstrated all the attacks using ONOS we believe that these issues are likely to become more general in nature. They are becoming important with the shift towards automated and intent aware controller frameworks allowing for simpler and agnostic controller applications. Based on our experiments we have also seen that the resources required and utilized for teleportation, even at high rates are moderate. Therefore, it may be difﬁcult to distinguish the attack trafﬁc from the benign trafﬁc. Accordingly we believe that, with the separation of the control and data plane, it is now important to monitor and police the communication channel between the separated planes due to the increased attack surface.

## **Packet-in-Packet-out Watcher**

In order to prevent the out-of-band forwarding teleporta- tion, we strongly advise the use of a *Packet-in* and *Packet- out* watcher. It can either exist as a controller application or as an application that resides between the controller and switches akin to hypervisors. It would involve tracking and enforcing security policies for *Packet-in*s and their corresponding *Packet-out*s. Existing security enforcement kernels, hypervisors and security applications must account for *Packet-in*s and *Packet-out*s in addition to *Flow-mod*s to detect and prevent out-of-band forwarding teleportation.

Note that the out-of-band forwarding teleportation could also be used by malicious controller applications. In a non- adversarial scenario, the order in which a packet’s fate is decided upon by various applications can inadvertently teleport the packet. Therefore, verifying that the *Packet-out* does not reach an undesired switch/host can prevent out-of-band forwarding teleportation.

## **Audit-Trails and Accountability**

We propose controllers to introduce secure audit-trail capabilities, and accounting, that enable network adminis- trators to thoroughly investigate events in their networks. For example, controllers must log and alert sensitive events such as a moving MAC addresses, or, receiving a *Packet- in* when a ﬂow has not yet timed out. Such capabilities can aid detection and prevention mechanisms. It is also useful for investigating security incidents. We recommend administrators to frequently view controller logs, investigate failed events and suspicious identities in the network.

## **Enhanced IDS with Waypoint Enforcement**

Network intrusion detection systems are an important means to detect and limit cyber attacks today, and accordingly intrusion detection systems constitute an integral part of most networks. We strongly suggest the use of an IDS application on top of or before the controller, that can inspect *Packet- in*s and *Packet-out*s and alert on suspicious trafﬁc. Indeed, some controllers today already offer basic functionality for waypoint enforcement. In particular, we suggest waypoint enforcement and coordinating intrusion detection systems from the control plane with the data plane. This is non-trivial, but vital for network security.

# 8.CONCLUSION

As OpenFlow networks transition from research to production, new levels of reliability and performance are necessary [44]. This paper has identiﬁed and demonstrated a novel security threat introduced by software-deﬁned networks separating the control plane from the data plane. In the presence of an unreliable south-bound interface (containing malicious switches): We have shown that state-of-the-art controller(s) are vulnerable to teleportation. Teleportation has numerous applications (cf. the summary in Table 2): It can be exploited to bypass security-critical network elements (e.g., to exﬁltrate conﬁdential information), as a discovery protocol for malicious switches, to evade policy conﬂicts as well as for man-in-the-middle attacks. Based on our preliminary evaluation, we can say that even a teleportation channel of over 10 Mbps can easily be used inside a loaded control channel.

Our work can also be seen as a ﬁrst security analysis of the increasingly popular intent-based network mechanisms: While intent-based mechanisms are attractive for allowing (cloud) network operators resp. SDN applications to focus on “what to connect” rather than “how’, we have shown that controller managed intents need to be used with care. Indeed, our experiments with controllers that are only starting to introduce an intent based mechanism are not yet vulnerable to all the speciﬁc attacks presented in this paper. Moreover, while intent mechanism implementations can vary across controllers, we believe that the underlying issues are fundamental.

We understand our work as a ﬁrst step, and believe that our paper opens several relevant directions for future research. In particular, we plan to extend our vulnerability analysis to other SDN protocols and conduct a more in-depth performance analysis. Another relevant avenue for future research regards the development of countermeasures.

# 9.REFERENCES

[1] M. Armbrust et al., “A view of cloud computing,” *Communications of the ACM*, pp. 50–58, April 2010.

[2] P. Kazemian, G. Varghese, and N. McKeown, “Header space anal- ysis: Static checking for networks,” in *Proc. Usenix Symposium on Networked Systems Design and Implementation (NSDI)*, 2012, pp. 113–126.

[3] A. Khurshid, X. Zou, W. Zhou, M. Caesar, and P. B. Godfrey, “Veriﬂow: Verifying network-wide invariants in real time,” in *Proc. Usenix Symposium on Networked Systems Design and Implementation.*

[4] M. Reitblatt, N. Foster, J. Rexford, and D. Walker, “Consistent updates for software-deﬁned networks: Change you can believe in!” in *Proc. ACM Workshop on Hot Topics in Networks (HotNETs)*, 2011, pp. 7:6

[7J] . Mudigonda, P. Yalagandula, J. Mogul, B. Stiekes, and Y. Pouffary, “Netlord: A scalable multi-tenant network architecture for virtualized datacenters,” in *Proc. ACM SIGCOMM*, 2011, pp. 62–73.

[8]. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, “A security enforcement kernel for OpenFlow networks,” in *Proc. ACM Workshop on Hot Topics in Software Deﬁned Networking (HotSDN)*, 2012, pp. 121–126.

[9]. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, and M. Tyson, “Fresco: Modular composable security services for software-deﬁned networks,” in *Proc. Internet Society Symposium on Network and Distributed System Security (NDSS)*, 2013.

[10] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, “AVANT-GUARD: Scalable and vigilant switch ﬂow management in software-deﬁned net- works,” in *Proc. ACM Conference on Computer and Communications.*

[11] S. Lee, T. Wong, and H. S. Kim, “Secure split assignment trajectory sampling: A malicious router detection system,” in *Proc. IEEE/IFIP Transactions on Dependable and Secure Computing (DSN)*, 2006, pp. 333–342.