SDN 網路安全架構:以防火牆為例

簡旭彤 林盈達
國立交通大學資訊工程系
Email: msgchien.cs03g@nctu.edu.tw; ydlin@cs.nctu.edu.tw
September 30, 2014

摘要

SDN 網路架構將控制面與資料面獨立分開，控制面集中於外部控制器管理，被控管的網路設備只進行資料傳遞，如此可以有效地降低管理複雜度。本篇報告討論 SDN 網路安全架構，以防火牆為例，包含防火牆設置方式、運作原理且尋找將傳統防火牆規則無痛轉移 SDN 網路架構的方法。實驗中，針對 Linux 內建防火牆 iptables 過濾功能的規則做轉移，把規則分成三類，為過濾 IP、protocol 與 service，並分別做了對應 SDN 架構的規則模組。將網路 iptables 規則的範例轉移至預先設計的規則模組，發現 iptables 中，若設有指定網路卡 I/O 或含 iptables 特有參數，則會轉換失敗。原因有二，(1)SDN 架構下，防火牆規則設置於 SDN switch，而 SDN switch 並沒有網路卡的區別；(2)iptables 特別參數(如 -g -f -c)，在 SDN switch 並沒有支援。針對原因(1)，如去掉網卡 I/O 設定不影響規則解釋，則將設定去除，即可轉換；原因(2)，除非 SDN switch 將功能實作，否則無法轉換。所以是無法將傳統防火牆規則完全無痛轉移至 SDN 架構的。

關鍵字：SDN、OpenFlow、防火牆、網路安全、iptables

1. 簡介

對於整個網路發展趨勢，雲端運算成為不可或缺的部分，將分散的伺服器資源集中化管理，便能更方便控制、分配資源。除了將資源有效運用外，同時也提升系統的容錯能力，即是虛擬化。虛擬化技術在硬體面獲得成功後，網路虛擬化也即將來臨。為了將虛擬化的優點落實於網路架構,許多研究團隊提出了 Software-defined networking (SDN)[1]的概念，將控制面(control plane)與資料面(data plane)從網路設備中獨立分開，並將控制面集中到控制器管理，達到路徑運算、環境配置等集中處理的效果，實現網路控制面虛擬化。由於 SDN 架構的集中控管特性，可將設定配置在任何控管範圍內的網路設備，其中 OpenFlow[2]將設置以 flow table[2]的方式記錄於網路設備中，並提供配對過濾機制，因此網路設備具備基礎的封包過濾的能力，達到防火牆功能。如此 SDN 網路安全架構便與傳統網路有所不同，主要的差異在於設定的方式與防火牆的防禦位置。最後探討是否有方法可以將傳統防火牆規則無痛轉移至 SDN 架構下。本篇文章針對 SDN 網路安全架構做討論，並以防火牆為例，討論：(1)SDN 防火牆實現與運作原理、(2)SDN 防火牆與傳統防火牆的差異、(3)無痛轉移傳統防火牆規則至 SDN 架構下的可能性。

2. SDN 介紹

SDN 是一種網路架構，其主要特性是，將控制面(control plane)與資料面(data plane)從網路設備中獨立分開，並集中到控制器上管理，再將商業應用與服務應用從控制面中獨立出來。因此，我們可以將控制面軟體安裝在任何實體機器或虛擬機器上，成為控制器，使其管理所有的網路設備、控制封包轉送需求。不需要在各個網路設備中安裝控制面軟體。如此，可以由控制器定義不同的網路行為模式，讓所控管的網路設備扮演正確的角色。透過這種方式，不需要逐一設定網路設備，使在建置各種不同的網路環境能更有效率且快速。甚至，因為可程式化的管理模式，除了改善以往可觀的環境部屬時間，同時也因集中化的特性，更能即時監控網路設備的狀態及調度資源，使得維護資源成本降低，達到網路虛擬化的目的。

2.1 OpenFlow

實作 SDN 的方法有很多，目前以 OpenFlow 實作的最完整也最為有名，以下將介紹 OpenFlow 如何實作 SDN。OpenFlow 是一個開放的通訊協定(protocol)，使得安裝在控制器(以下稱 controller)上的控制面軟體得以對底層網路設備(以下稱 switch)做有效的控制，例如設置 flow、控管 flow table、監看設備狀態等來實作 SDN。見圖 1，此圖分為三部分：

- Flow table(s)：在 switch 中寫入封包的 flow，封包進入 switch 後依照 flow table(s)所定義的 flow 來傳送封包，使封包能採取正確的動作。
- Controller：當處理未知封包時，也就是在 flow table(s)沒定義的 flow，switch 會將封包送往 controller，由 controller 計算封包接下來的流向且建立 flow entry，以因應未來此類封包的處理。
- OpenFlow protocol：透過 SSL 加密通道，讓 switch 與 controller 溝通

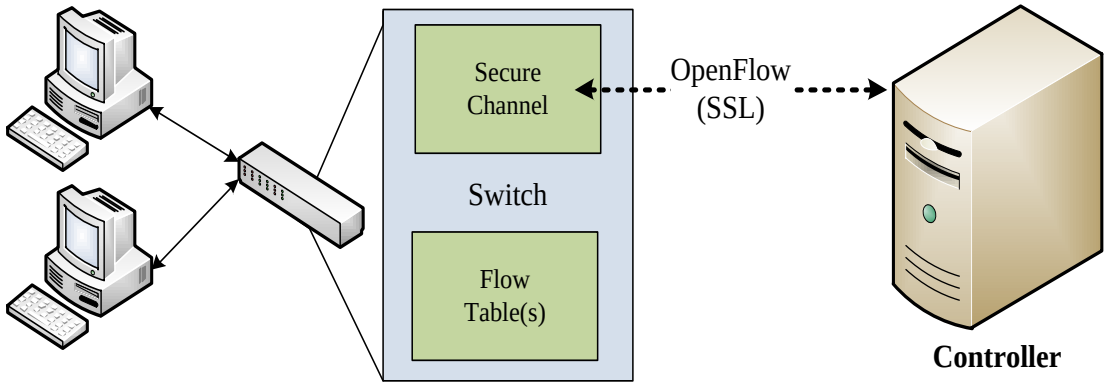

圖 1: OpenFlow 架構示意圖[2]

3. SDN 防火牆

在 SDN 架構下，實現防火牆關鍵就在於 OpenFlow switch 處理封包的流程。

3.1 OpenFlow flow entry matching

OpenFlow switch 存在 flow table(s)，其記錄了預先設置的規則及處理動作。圖 2，當封包進入到 switch 時，會將該封包是由哪個 switch port 進入交換器中記錄在封包內，接著辨別封包種類，若屬於控制封包，就直接送到 controller 處理；若屬於資料封包，則會先比對 flow table，如有符合的規，則執行其預先設定的動作，若沒有配對成功，則經由 SSL 通道送往 controller 計算。

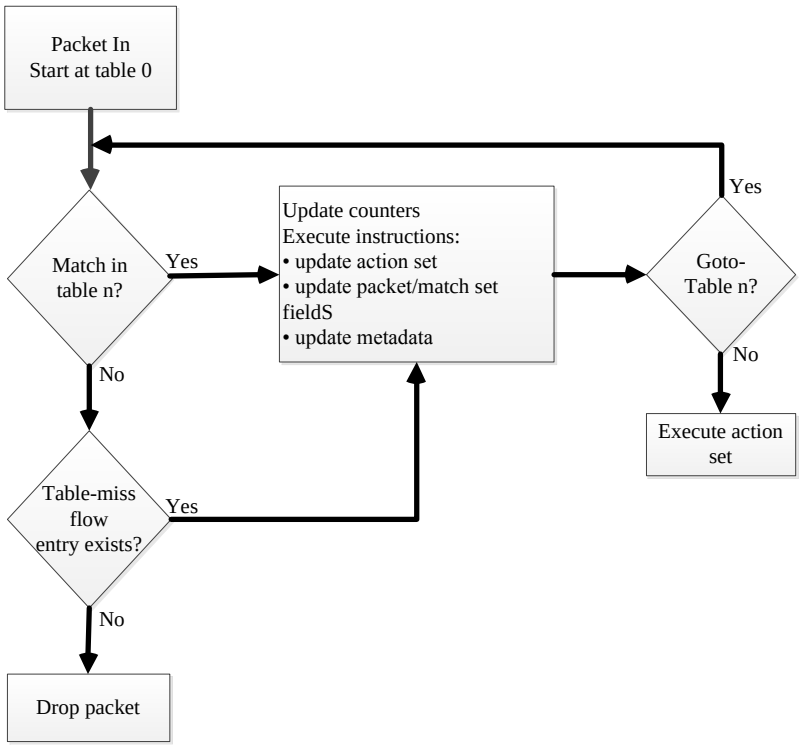

圖 2: OpenFlow switch 處理 packet 流程[2]

Flow table 中 flow entry 的架構由三個主要的元素構成,分別是 Match Fileds, Counters, Instructions。，如圖 3 所示，Match Fields 中，提供多種參數來設置規則，例如：來源 ip 地址、目的 ip 位址、來源 port、目的 port、通訊協定等基本參數，不過依不同 switch 實作的程度不同，會影響到所提供的參數多寡，進而影響到防火牆的功能性。 Counters 是在記錄這個封包狀態，包括收到的流量等。Instructions 則是設置封包動作的欄位，也分為基本動作與選擇性實作的動作，基本的動作有，DROP 及 FORWARD 兩種，其中 FORWARD 可以設置要轉發到 controller、一個或多個 switch port、flow table 等，其預設動作為 DROP。SDN 防火牆實現，是使用以上搭配出不同規則，來達到防火牆的功能。

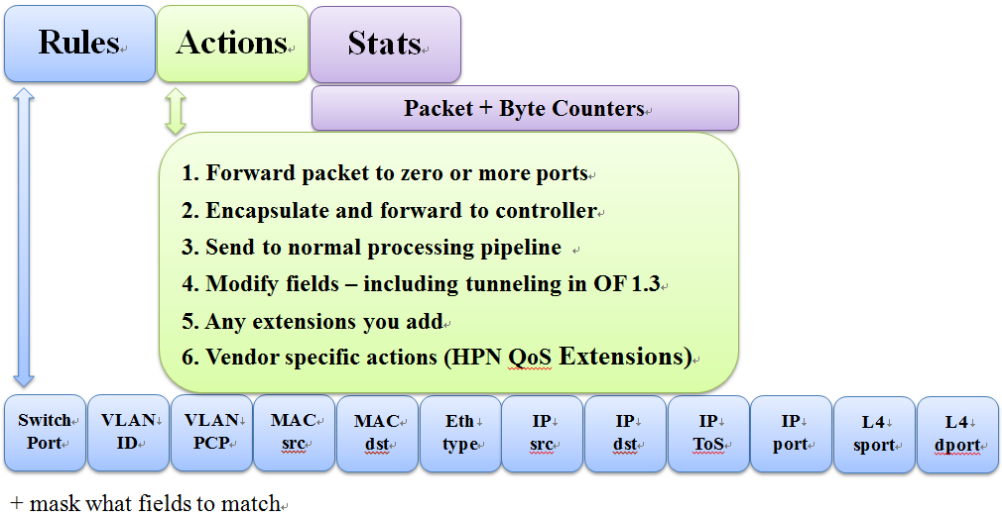

圖 3: OpenFlow flow table entry 架構圖[2]

3.2 Flow entry 設置

SDN 防火牆是透過設置 flow table 來實現，透過圖 4，可以看到設置的方法有三種型態：(1) 透過 applications、(2) 利用 modules、(3) 直接設在 switch 上。透過 applications 與 modules 的差異是在於是否透過 North bound API[3]跟 controller 溝通。例如 REST module[4]是較常被使用的 module，可以提供 API 讓使用者透過較簡單的方式設置 flow；另一常用設置 flow 的 module 是 simple switch[6]，這些 module 同時也是作為 North bound API 的角色。所以假使撰寫一支程式利用 REST module 或者 simple switch 方式去設置 flow，就是屬於利用 application 這種型態的範疇。至於，直接在 switch 上設置，則是透過命令式的工具，就像是傳統網路 switch 上的 framework，提供命令直接對 switch 設定做修改。

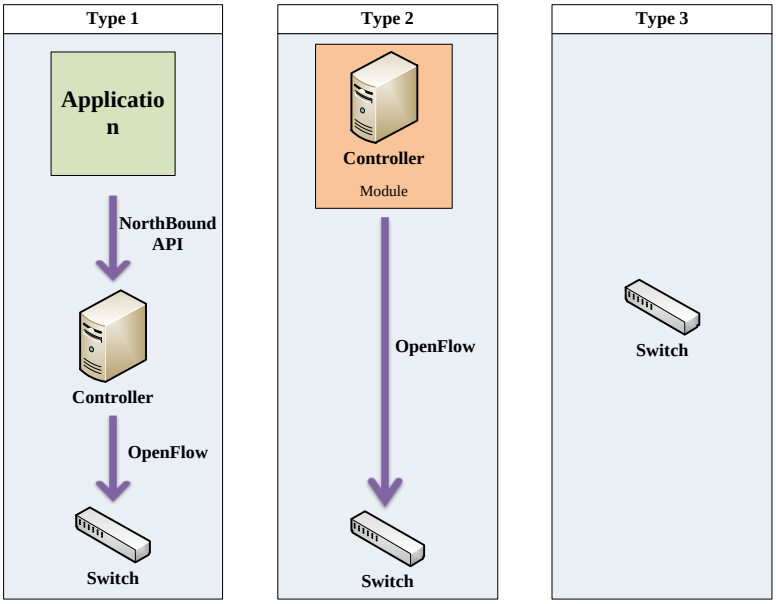

圖 4: 設置 flow entry 方法示意圖

上述提到的方法不論是透過 applications，還是利用 modules，去設置 flow，都會需要透過 OpenFlow message [2]的協助，OpenFlow message 是 controller 與 switch 溝通的語言，例如 controller 詢問 switch 狀態、switch 向 controller 請求設定等。不過，在每個不同的 OpenFlow 版本中，OpenFlow message 都有些許調整。其中 FlowMod message 則是從 OpenFlow1.0 至最新版本都還在使用的 message，是 controller 用來設定 flow 的 message。透過表 1 可以看到，command 是可以選擇要新增、修改或刪除 flow 的欄位；match 欄位是設置封包要 match 的規則；instructions 則是設置封包的動作的欄位，可參考 3.1 節中提及的資訊。

表 1: modify flow entry message 重要欄位[6]

Attribute	**Description**
table_id	ID of the table to put the flow in
command	ADD, MODIFY, MODIFY_STRICT, DELETE ,DELETE_STRICT
match	Instance of OFPMatch
instructions	list of OFPInstruction* instance

3.3 傳統防火牆與 SDN 防火牆差異

見表 2，從設定的面相來說，在傳統防火牆的設定是依靠設定檔配置來管理規則，且設定檔位於防火牆本身。但是在 SDN 架構中，透過 OpenFlow 協定去控管網路設備，因為利用 flow table entry 來實現防火牆功能，所以能夠由多種方法(applications, modules, switch)設置 flow entry，且 flow entry 可以配置於任一指定網路設備中，不在侷限於防火牆本身。

表 2 : 傳統防火牆 與 SDN 防火牆設定比較表

	Traditional Network	**SDN**
How to setup	configure file	flow entry
Where to setup	Firewall	Application Controller Switch

就防禦位置來說，如圖 5 所呈現，傳統防火牆只能防禦在一個點上，容易被集中火力攻擊或者繞過，且一旦防火牆失效，內部網路等同於暴露在外。反觀 SDN 架構下的防火牆，因為 SDN 特點，任何一項設備都能成為防火牆，只需設置 flow entry。

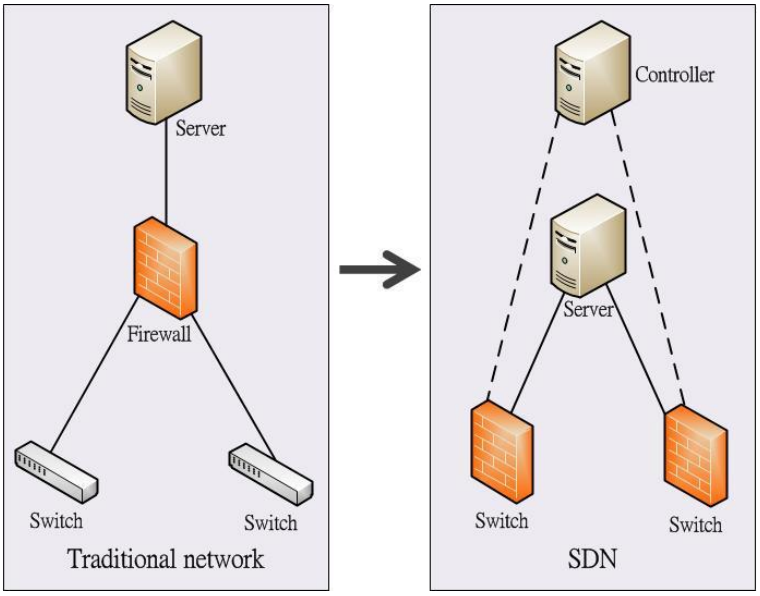

圖 5 : 傳統防火牆與 SDN 防火牆防禦位置示意圖

4. 防火牆規則轉移

表 3 中，分別列舉了設定傳統防火牆針對過濾 IP、port 號、通訊協定的 command script，可以發現傳統防火牆規則其實只是去設定 IP、port 號、通訊協定這些基本資訊。在 SDN 中，防火牆是建立在 flow table match 的機制下，且 flow 中的 match entry 也可以去設定配對這些資訊。因此，覺得有將傳統防火牆規則無痛轉移至 SDN 架構下的可能性。

表 3: 傳統防火牆設定 command script

Filter type	**iptables command**
IP	iptables -A INPUT -s 10.0.0.1 -j DROP
protocol	iptables -A INPUT –p icmp -j DROP
prot number(service)	iptables -A INPUT -p tcp -s 10.0.0.0/24 --dport 22

另外，在網路搜尋後發現，其實是有廠商提供防火牆轉移這項服務，見表 4，像是 Palo Alto Networks[7]、Juniper Networks [8]、DTS-Solution [9]，但是這些廠商所提供的服務對象往往是針對擁有大型網路架構業者，或者是自家產品的轉移，且並不是每家廠商都有聲稱提供轉移至 SDN 架構的服務。因此，更加希望能找出一個能無痛移防火牆規則的方法，並討論方法的優缺點及可用性。

轉移方法構想

撰寫一支程式，將傳統防火牆規則轉換成 REST API 形式。選擇 REST API 原因是，目前知名的幾套 controller 軟體皆支援 REST API，不同套 controller 只需要微調參數即可。

程式構想如圖 9，讀取設定傳統防火牆的 Scripting commands，去將傳統防火牆指令中的參數去對應 OpenFlow 中的 match fields，最後產出是以 JSON 的形式。

表 4: 各家 migration service vendor 比較表

Service vendor	**Palo Alto Networks**	**Juniper Networks**	**DTS - Solution**
To SDN firewall	O	N/A	N/A
Migrate to their products	O	O	X
Auto migrated APP	X	X	O

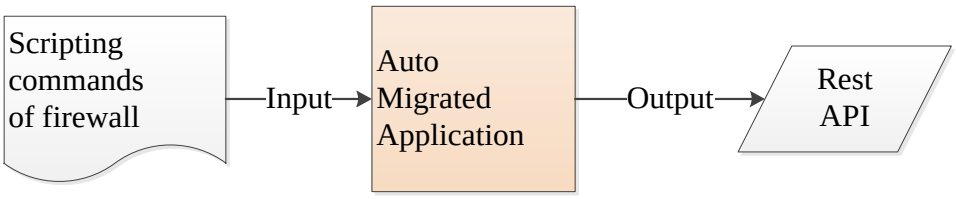

圖 9: Auto Migrated Application I/O 示意圖

5. 防火牆規則轉移實驗

本次實驗將探討上述的 Auto Migrated Application 方法是否可行。並找一套常用的傳統防火牆來做實驗。

實驗架構與方法

圖 10 中可以看出，本次實驗環境是在 VMware Workstation 9 [10]中，架設兩台虛擬機皆安裝 Ubuntu Server14.04。其中一台虛擬機上面執行 OpenFlow controller Ryu 3.13[11]，另一台則執行 SDN Simulator: mininet 2.1.0 [12]。在 mininet 中，則是創建了兩個 host 分別為 h1、h2，與一台 open vSwitch[12]，s1。

以 Linux 內建防火牆 iptables [13]作為傳統防火牆，iptables 是由 Linux 核心 2.4 之後所提供的簡單封包過濾軟體，其主要有三個部分 filter、nat、mangle，都使透過 iptables 這個指令去完成防火牆相關設定。實驗所採用的是 filter 這部分的功能，與 iptables-extensions[14]中 filter port number 的部分。實驗主要將 iptables 的規則分成三類，(1)filter ip address 、(2)filter protocol、 (3)filter service，分別做好了對應的 REST API 規則模組，只需要將 iptables 指令的參數對應到 OpenFlow 的 match fields 即可。設定完 flow 之後，再由 h1、h2 互 ping 的方式驗證防火牆規則是否有成功轉移。

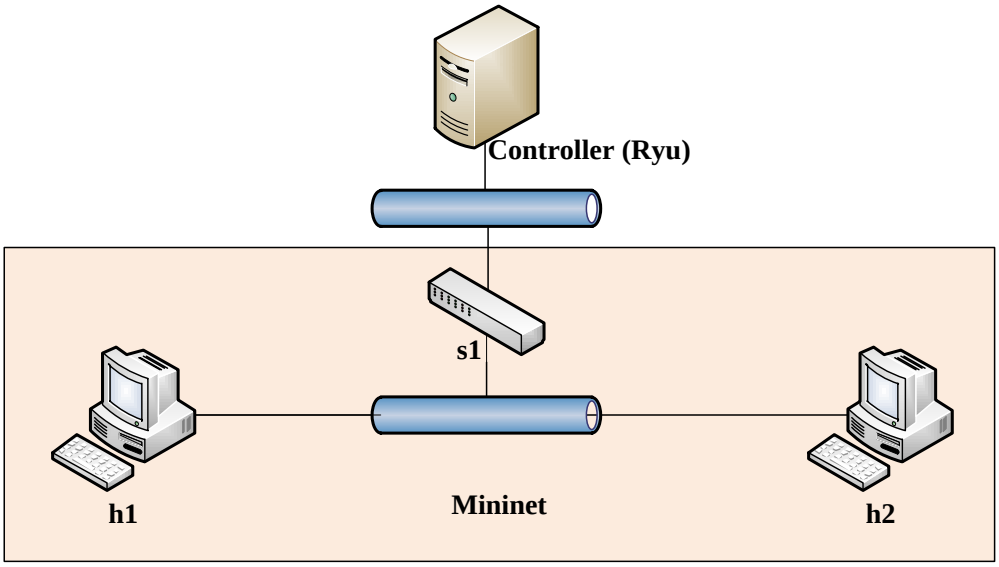

圖 10: 實驗環境架構圖

實驗結果

經由實驗發現，轉換失敗原因，主要有兩類，(1)在 iptables command 中，有指定網路卡 I/O(iptables -i -o)，這類規則就會轉換失敗。原因是，SDN 架構下的防火牆是利用 flow table 實現，所以是設置在 switch 上，然而這些 switch 並沒有網路卡的區別，所以無法指定 I/O 介面。(2)帶有 iptables 獨有的參數(iptables -g -f -c)，這類型的規則也會失敗。因為這些是 iptables 中獨有的參數，Openflow switch 並沒有支援這些參數所提供的功能，所以包含以上兩類參數的規則命令是無法成功轉換的。表 5 是將 iptables 的參數與可對應的 match fields 整理出一張對照表格供參考。

表 5: iptables 參數與 OpenFlow match entry 對照表

parameters of iptables	**match fields of OpenFlow**
-p, --protocol	nw_prot
-s, --source	nw_src
-d, --destination	nw_dst
--mac-source	dl_src
--source-port, --sport	tp_src
--destination-port, --dport	tp_dst
-j, --jump (ACCEPT, DROP)	Action: {"output":"all“} , {}

6. 結論

從本篇文章中，可以知道 SDN 防火牆運作即是透過 OpenFlow 中 flow table 的 match 機制。且 SDN 下的網路安全架構與傳統網路有所差異，以防火牆為例，從設定面來說，SDN 防火牆除了控制集中化使控管更加方便之外，也有多種方式可以設定，像是透過 applications、modules 或直接設置在網路設備上，而傳統網路卻只能設定在本身防火牆上；防禦位置佈署方面，因為 SDN 架構的關係，只要是屬於控制器控管的網路設備皆可成為防禦位置，如此佈署變得更彈性也更多元，但傳統防火牆卻只能堅守自己所在的位置。另外，透過實驗得知，欲撰寫傳統防火牆規則自動轉換程式，就必須要去針對每一套傳統防火牆個別撰寫，且大部分無法做到無痛的轉移。除了遇到 SDN 網路設備本身不支援傳統防火牆的功能外，也有硬體架構不同所導致的問題。傳統防火牆架設於伺服器上，SDN 架構下卻是設置於網路設備中，兩者本身的硬體介面有所不同，因此，如遇到有指定網路卡介面的設定時，則必須評估變更網路卡介面設定的影響並手動處理。綜合以上，如要將傳統防火牆規則轉換至 SDN 架構，還是建議需要重新規劃適合 SDN 架構的新規則。

參考文獻

[1] K. Greene, Software defined networking, Technology review-the 10 emerging technologies of 2009, March 2009.

[2] Open Networking Foundation, OpenFlow Switch Specification, openflow.org, October 14, 2013

[3] Firewall REST API, http://www.openflowhub.org/display/floodlightcontroller/Firewall+REST+API

[4] simple_switch http://osrg.github.io/ryu-book/en/html/switching_hub.html

[5] Stanford University, OpenFlow wiki, http://yuba.stanford.edu/cs244wiki/index.php/Overview

[6] Ryu, OpenFlow v1.4 Messages and Structures, http://ryu.readthedocs.org/en/latest/ofproto_v1_4_ref.html

[7] Palo Alto Nerworks, Firewall Migration Service, https://www.paloaltonetworks.com/content/dam/paloaltonetworks-com/en_US/assets/pdf/services/professional-services-firewall-migration.pdf

[8] Juniper Networks, Firewall Migration Service, http://www.juniper.net/us/en/local/pdf/datasheets/1000231-en.pdf

[9] DTS Solution, Firewall Migration Services http://www.dts-solution.com/services/professional-services/firewall-migration-services/

[10] VMware Workstation http://www.vmware.com/tw/products/workstation

[11] Ryu http://osrg.github.io/ryu/

[12] mininet http://mininet.org

[13] Open vSwitch http://openvswitch.org/

[14] iptables
http://ipset.netfilter.org/iptables.man.html#lbAK

[15] iptables-extensions
http://ipset.netfilter.org/iptables-extensions.man.html

[16] Michelle Suh, Sae Hyong Park, Byungjoon Lee, Sunhee Yang, Building Firewall over the Software-Defined Network Controller, ETRI, February 16~19, 2014

[17] Hongxin Huy, Gail-Joon Ahnz, Wonkyu Hanz and Ziming Zhao, Towards a Reliable SDN Firewall, USENIX, February 24, 2014

[18] 戴錦友，余少華，汪學舜，朱國勝, Openflow flow table storage and optimization method based on resource reuse, BiBTeX, EndNote, RefMan, March 6, 2013

[19] SDN Use Case: Firewall Migration in the Enterprise,
http://etherealmind.com/sdn-use-case-firewall-migration-in-the-enterprise/