Early detection of network threats using Software Defined Network (SDN) and virtualization

Project submitted in partial fulfillment of the requirements for the Degree of Master of Engineering in Technology Innovation Management

By: Michael Rolbin

For: Professors Michael Weiss and Marc St-Hilaire

Carleton University, Ottawa, Ontario, Canada

December 2013

Abstract.

Our potential customers require a cost efficient, flexible and easy to maintain solution for network vulnerability identification and prevention.

We propose to consider a new approach to network security, which is rapidly developing in modern computer industry – Software Defined Networking (SDK) with OpenFlow protocol technology. The new approach, which we have adapted to the problem of network access control and Zero-Day attack prevention, to achieve maximum efficiency of network devices such as routers and firewalls, and eliminate usage of expensive physical devices.

The proposed solution is the foundation for further research and development of final products.

Table of Contents

1. Introduction

Internet architecture was developed in the 60s and hopelessly outdated. At the time, no one could predict the current speed and amount of transferred data. The level of access network differentiation was not foreseen in the development of basic network protocols. It was assumed, that different applications could communicate with each other without any restrictions in a global network. This assumption required an adjustment quite quickly and there were developed specialized devices such as firewalls, Intrusion Prevention Systems (IPS), network anti-viruses and Web Application Firewalls (WAF).

We argue that the innovative concept of Software Defined Networking (SDN) with OpenFlow protocol can provide a natural opportunity to solve the problem of Zero-Day attack not just from host resolution, but from network level as well. Our project is to show idea of how to identify potential vulnerability of a host in virtual network environment. Our project focused on identification of abnormal behavior in virtual network of virtual network devices and hosts that could be easy observed and investigated without damage of the real production environment. Our method allows us to dramatically reduce the cost of equipment, intellectual property and data recovering.

We suggest building easy to recover virtual network environment based on Mininet Open Source solution and SDN based malicious activity emulator. We developed a basic network activity identifier that could find if the host trying to connect to another network objects in the local network or external devices and how. This solution works in principle of thread prevention “sandbox” but for network level instead of single host level.

Sandbox is a security mechanism for separating running programs. It is often used to execute untested code, or untrusted programs from unverified third parties, suppliers, untrusted users and untrusted websites (Goldberg et al. 1996). We suggest bringing the sandbox idea to the network level and executing untrusted hosts with untrusted programs in this environment. In the network level we can identify not just malicious activities of untrusted programs, but also Bot Nets or untrusted attempts to connect to external networks. Therefore, we can identify not just known viruses or programs but Zero-day threats as well.

2. Background

OpenFlow originated in Stanford Clean Slate project group. CleanSlate ultimate goal of the project is to re-invent the Internet, aimed at changing the design has been slightly outdated and difficult to evolutionary development of the existing network infrastructure. In 2006, Stanford student Martin Casado led a project on network security and management of the project Ethane, the project attempts a centralized controller, allowing network administrators to easily define network-based traffic safety control strategy, and these security policies applied to a variety of network devices, in order to achieve the security of the entire network communication control. Affected by the project inspired, Martin and his mentor, Professor Nick McKeown (when he was Clean Slate Project Faculty Director) found that if Ethane design is more general, the traditional network equipment data forwarding (data plane) and routing control (control plane) two functional modules separated by a centralized controller with standardized interfaces to various network device management and configuration, then this will be the network resources the design, management and use of the possibility to provide more, making it easier to promote innovation and the development of the network. So, they put forward the concept of OpenFlow, and Nick McKeown, who in 2008 published in ACM SIGCOMM entitled OpenFlow: Enabling Innovation in Campus Networks of the paper, the first detailed description of the OpenFlow concept. In addition to describing the papers OpenFlow works, but also cited the OpenFlow several scenarios, including: 1) the campus network on experimental protocol support (its title indicates); 2) network management and access control; 3) network isolation and VLAN; 4) WiFi-based mobile networks; 5) Non-IP networks; 6) Web-based packet processing. Of course, the current research on OpenFlow far beyond these areas.

OpenFlow is based on the characteristics of programmable network brings, Nick and his team (including the University of California at Berkeley professor Scott Shenker) further proposed the SDN (Software Defined Network, currently more literally translated as "software-defined networking") concept - In fact, SDN said the concept was first used by KateGreene in 2009 Technology Review site selection made the top ten frontier technology. If all network devices in the network to be managed as a resource, then refer to the operating system’s principle, can abstract a network operating system (Network OS) concepts - the network abstraction of the underlying operating system on the one hand the details of network equipment, and also for the upper application provides a unified management view and the programming interface. Thus, the

network operating system based on this platform, users can develop a variety of applications by software to define logical network topology to meet the different needs of network resources, without concern for the underlying physical network topology.

From the above description, it can be seen OpenFlow / SDN principle is not complicated, in the strict sense can hardly be regarded as a revolutionary innovation. However OpenFlow / SDN has attracted more and more attention in the industry in recent years become a veritable hot technology. Currently, including HP, IBM, Cisco, NEC and Huawei and ZTE and other domestic traditional network equipment manufacturers have been added to the OpenFlow camp, while some support OpenFlow network hardware equipment has been published. In 2011, the Open Network Foundation at Nick, who, driven by established standards and norms responsible for OpenFlow maintenance and development; same year, the first session of the Open Network Summit (Open Networking Summit) held for OpenFlow and SDN in academia and industry have done a good introduction and promotion. Held in the beginning of 2012 the second summit, from Google's Urs Holzle in order OpenFlow @ Google Keynote speech titled Google has announced its global backbone network data centers around the large-scale use in OpenFlow / SDN, thus proving the OpenFlow is no longer just stay in academia, a research model, but has been fully equipped can be applied in a production environment technology maturity. Recently, Facebook also announced that its data center using OpenFlow / SDN technology.

2.1. OpenFlow standards and norms

Since early 2010 released the first version (v1.0), OpenFlow specification has undergone recently released 1.1, 1.2 and 1.3 versions. Meanwhile, earlier 2012 to manage and configure OpenFlow protocol also released the first version (OF-CONFIG 1.0 & 1.1). The following diagram lists the OF and OF-CONFIG specification development process and the various versions of the changes can be seen from the figure, the current use and support is still up 1.0 and 1.1 versions.

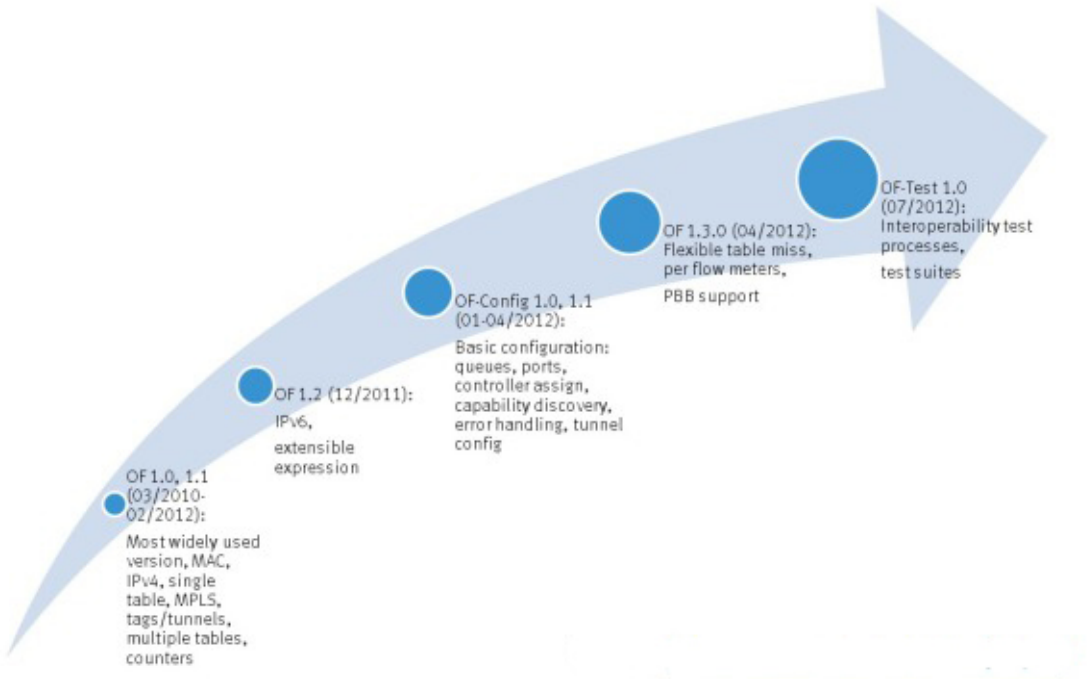


Figure 2.1.a OpenFlow specification development process

Here, we will explain in detail the latest OpenFlow Switch Specification (i.e. OF-1.3). Nick, who is selected from the following figure, papers OpenFlow: Enabling Innovation in Campus Networks. This chart is often used to illustrate the principle and basic structure OpenFlow. In fact, this figure is also a good indication of the OpenFlow Switch Specification as defined range - can be seen from the figure, OpenFlow Switch Specification defines the Switch main function modules as well as its communication channel between the Controller and other aspects.

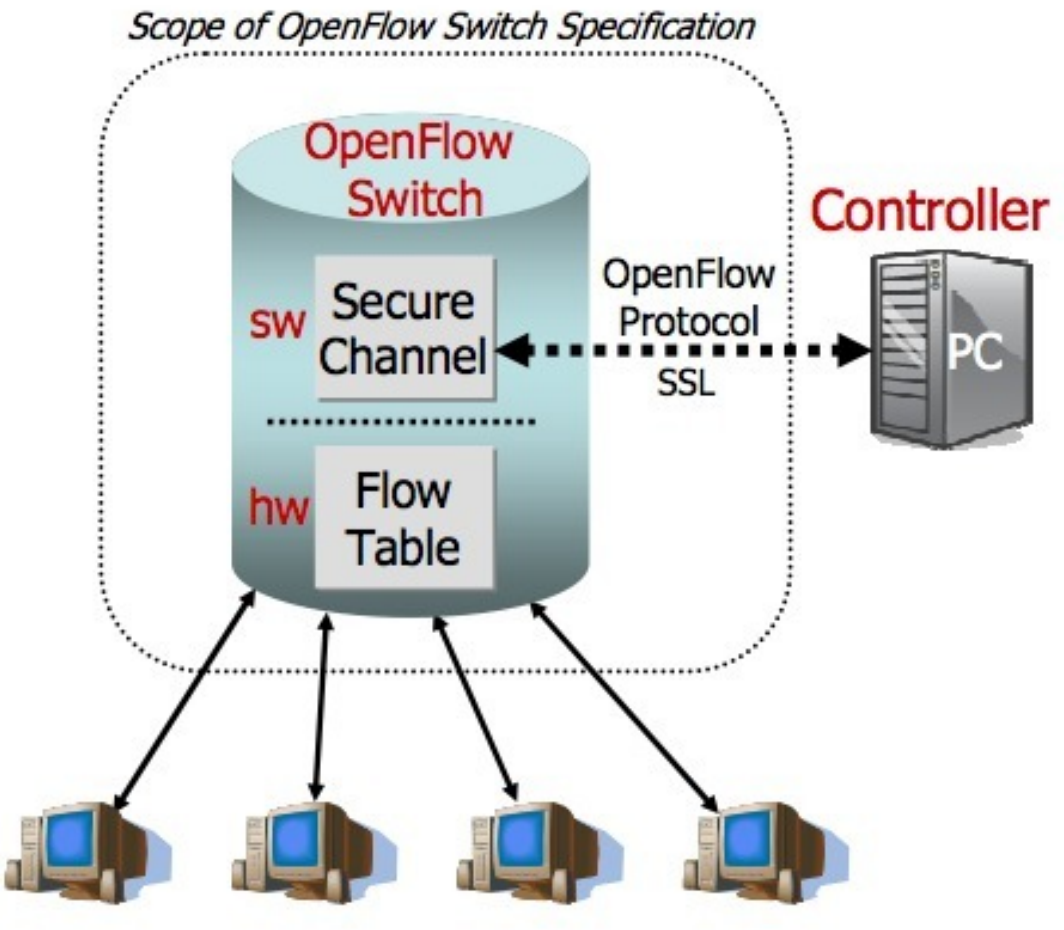


Figure 2.1.b OpenFlow protocol layers

OF specification is divided into the following four parts:

2.1.1. OpenFlow port (Port)

OpenFlow specification ports on the Switch into three categories:

a) The physical port, the port on the device physically visible;

b) Logical port, on the basis of the physical port Switch device abstracted from the logical port, such as a tunnel or aggregation functions to achieve logical port;

c) OpenFlow defined port. OpenFlow currently defines a total ALL, CONTROLLER, TABLE, IN_PORT, ANY, LOCAL, NORMAL and FLOOD 8 kinds of ports, of which the latter three kinds of non-essential ports, only in the mixed type OpenFlow Switch (OpenFlow-hybrid Switch, which also support traditional network protocol stack and OpenFlow Switch device, as opposed to OpenFlow-only Switch concerned) exists.

2.1.2. OpenFlow the FlowTable

OpenFlow through user-defined or default rules to match and handle network packets. An OpenFlow rule by the matching domain (Match Fields), priority, processing instructions and statistics (e.g. counters) and other fields, as shown below.

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie

In a rule, according to the network packet at L2, L3 or L4 and other network packet header to match any field, such as Ethernet frame's source MAC address, IP packet protocol type and IP addresses, or TCP / UDP port numbers. The specification of the current OpenFlow Switch also provides equipment manufacturers can optionally support wildcard matching. It is said, OpenFlow in the future also plans to support the entire packet to match any field.

All OpenFlow rules are organized in a different FlowTable, in the press with a FlowTable priority rule has matched. One of OpenFlow Switch can contain one or more FlowTable, numbered from 0 arrangement. OpenFlow specification defines streamlined processes, as shown below. When a packet enters Switch, you must match sequentially from FlowTable 0; FlowTable can leapfrog jump sequence from small to large, but not from a FlowTable Skip forward to number of smaller FlowTable. When a packet is successfully matched a rule, the rule will first update the corresponding statistics (such as the total number of packets successfully matched and the total number of bytes, etc.), and then according to the rules of the Directive and operate accordingly - for example jump to the subsequent a FlowTable continue processing, or to immediately execute the modified packet corresponds action set the like. When the packet is already in the last FlowTable, its corresponding action set all the action will be executed, including forwarding to a port, a

field to modify the data packets, dropped packets, etc. OpenFlow specification for the currently supported instructions and actions for a complete detailed description and definition.

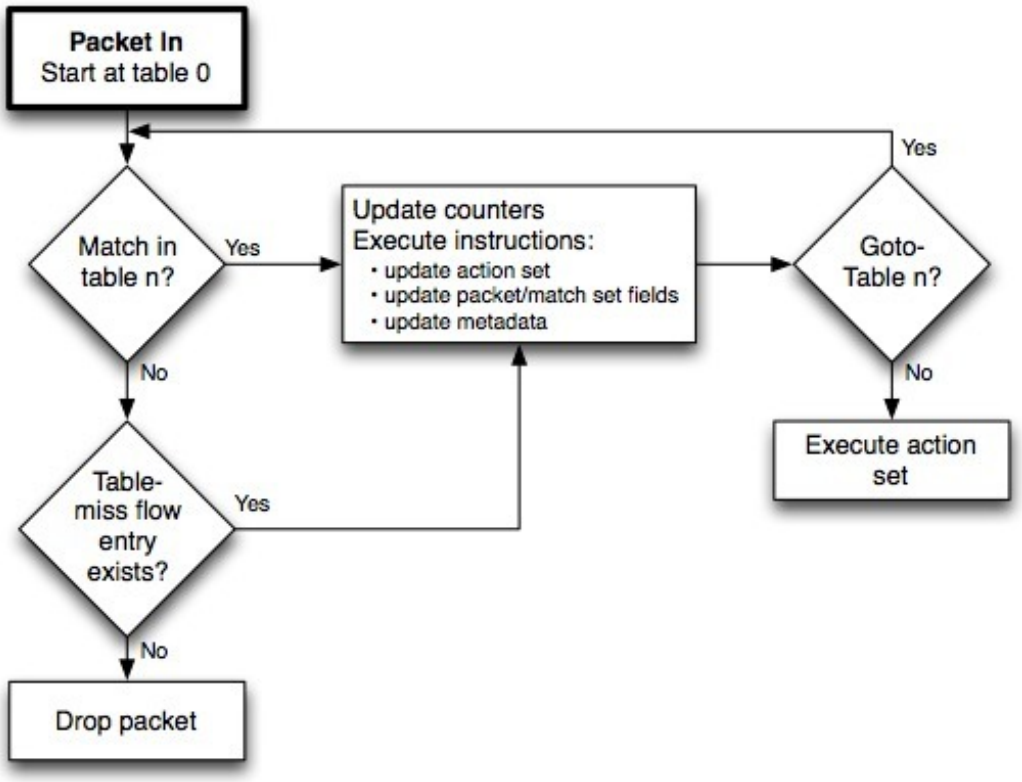


Figure 2.1.2 OpenFlow algorithm

Further, OpenFlow specification also defines a number of other features and behaviors such as OpenFlow support for QoS (i.e. MeterTable and the definition Meter Bands), for GroupTable definitions and rules out processing, etc.

2.2. OpenFlow communication channel

This section, OpenFlow specification defines how an OpenFlow Switch and Controller to establish a connection, communication and relevant message types and so on.

OpenFlow specification defines three types of messages:

a) Controller / Switch message is initiated by the Controller, Switch receives and processes the message, including Features, Configuration, Modify-State, Read-State, Packet-out, Barrier and Role-Request other news. These are primarily used by the Controller for Switch for state operations such as query and modify the configuration.

b) Asynchronous message is sent by the Switch Controller, used to notify Switch asynchronous events that occur on some of the messages, including Packet-in, Flow-Removed, Port-status, and Error and so on. For example, when a rule is deleted due to timeout, Switch will automatically send a Flow-Removed message notification Controller, to facilitate the Controller to make the appropriate action, such as re-set the relevant rules.

c) Symmetric message, the name suggests, these are two-way symmetrical news, mainly used to establish a connection, detecting whether the other online, including Hello, Echo and Experimenter three kinds of messages.

The following figure shows a typical OpenFlow and Switch exchange of messages between processes, for security and high availability considerations, OpenFlow specification also defines how to Controller and Switch between channel encryption, how to create a multi-connections (primary and secondary connections).

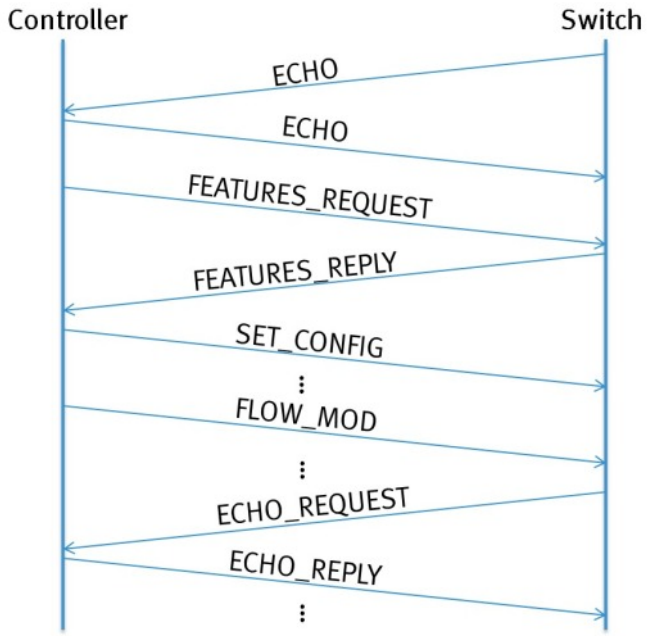


Figure 2.2 OpenFlow and Switch protocol exchange process

2.3. OpenFlow protocol and related data structures

In the last part of OpenFlow specification primarily defines various OpenFlow detailed data structure of the message, the message includes a message class OpenFlow. Here is not to enumerate here, learn more about OpenFlow source code can refer to in openflow.h header files for various data structure definitions.

OpenFlow applications

With OpenFlow / SDN concept development and promotion of its research and applications have also been expanding. Currently, about OpenFlow / SDN research areas include network virtualization, security and access control, load balancing, converged networks, and green energy and so on. In addition, there are about OpenFlow and traditional network device interaction and integration studies.

The following will give some typical case studies to demonstrate the application of OpenFlow.

2.3.1. Network virtualization - FlowVisor

Network virtualization is the nature of the underlying network to be able to abstract the physical topology of the network can be logically fragmented or integration resources to meet a variety of applications for the different needs of the network. In order to achieve the purpose of network fragmentation, FlowVisor implements a special OpenFlow Controller, can be regarded as different users or applications other Controllers layer between network devices and agents. Therefore, different users or applications can use your own Controllers to define different network topologies, while FlowVisor can guarantee these Controllers can be isolated from each other and between each other. The following figure shows the use FlowVisor can be defined on the same physical network different logical topology. FlowVisor OpenFlow is not just a typical application cases, is also a good research platform, there are already a lot of research and applications are based on FlowVisor do.

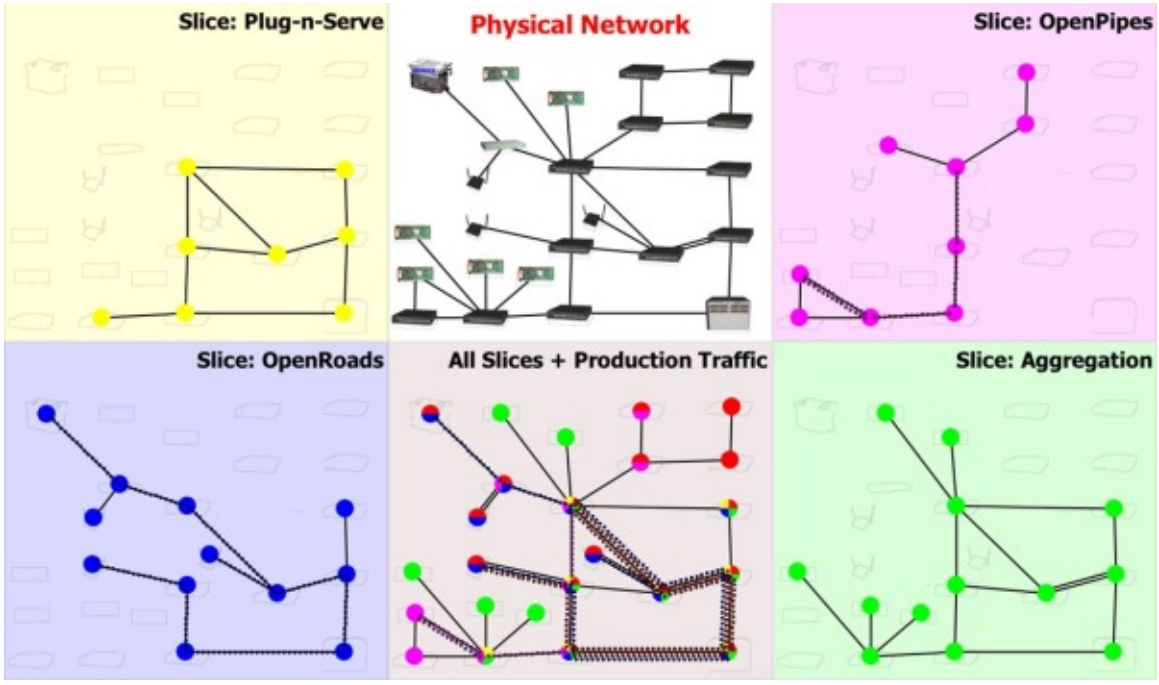


Figure 2.3.1 Flow Visor examples

2.3.2 Load balancing - Aster * x

The traditional load balancing solutions in server clusters generally require entrance through a gateway or router to monitor, statistics server workloads and dynamically allocated accordingly relatively light user requests to the load on the server. Since all network devices in the network are available through OpenFlow centralized control and management, and application server load can be timely feedback to the OpenFlow Controller there, then OpenFlow is very suitable for load balancing work. Aster * x through the Host Manager and Net Manager to monitor server and network respectively workload information is then fed back to Flow Manager, so Flow Manager can be based on these real-time load information to redefine the OpenFlow network equipment rules, which will user requests (that is, network packets) to be adjusted in accordance with the capabilities of the server and distributed.

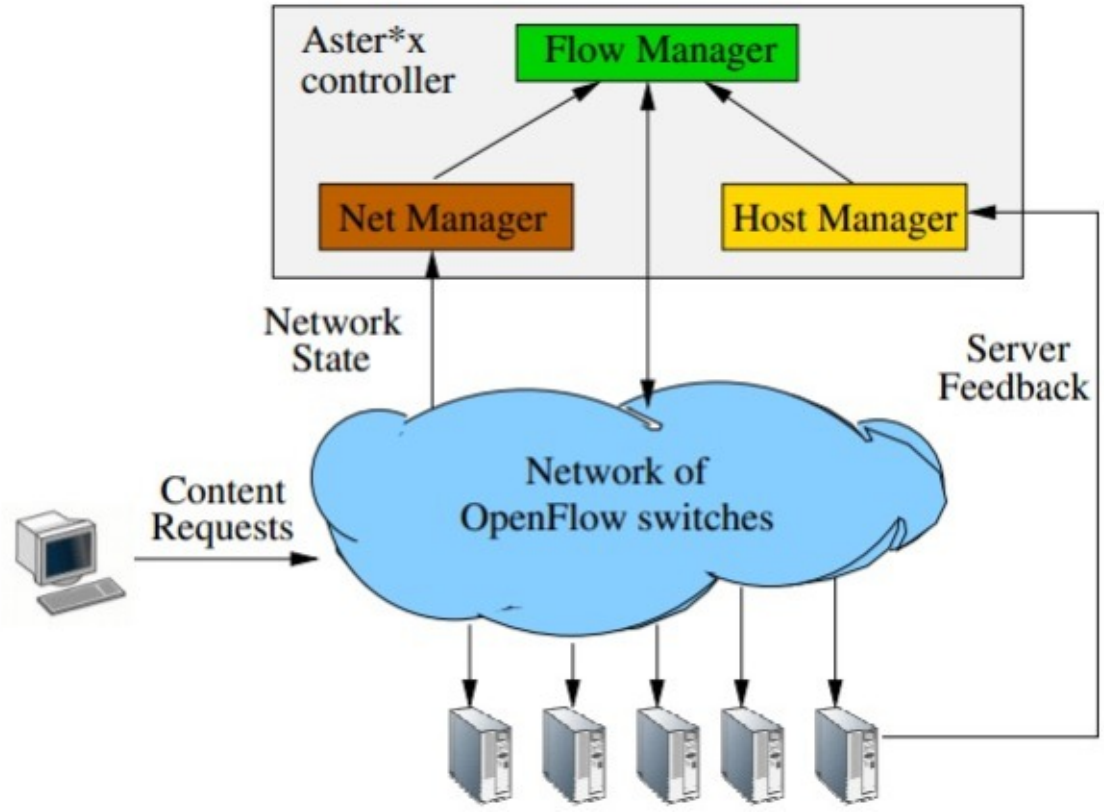


Figure 2.3.2 Load balancing

2.3.3. Green energy network services - ElasticTree

In the data center and cloud computing environments, how to reduce operating costs is an important research topic. According to the workload on-demand, dynamic programming resources, not only can improve resource utilization, but also can achieve energy saving purposes. ElasticTree innovative use of OpenFlow, without compromising performance, according to the network load dynamic programming routing, allowing network load is not high in the case of selectively close or suspend part of the network device, it enters a power saving mode to achieve Energy saving, lower operating costs.

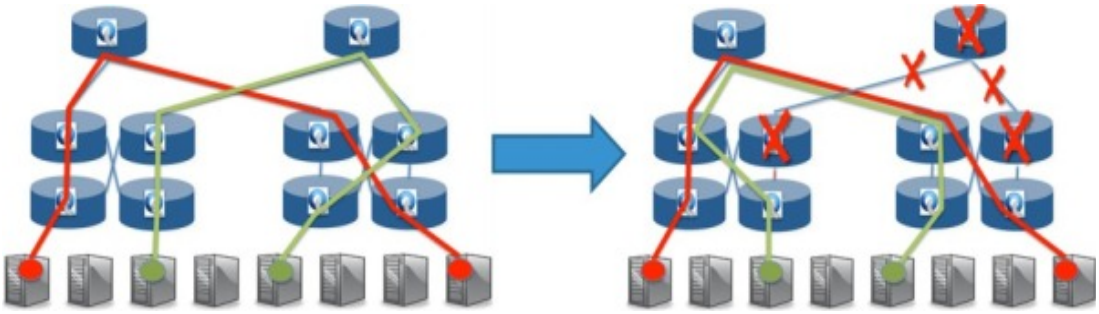


Figure 2.3.3 Example of Elastic Tree

3. Project Definition and Scope

3.1 Objective

Over many years we observe widespread malicious activities in LAN (one broadcast domain) networks have resulted in growth in Enterprise and SMB (Small and Medium Businesses). Computers in these networks are often could be compromised by viruses, “Trojan horses” and other kinds of malwares. We argue that SDN OpenFlow technology can provide more flexible and effective solution for the problem. Our objective is to provide a new cheaper and effective method for malware detection instead of existing methods and solutions. We argue that the concept of Software Defined Networking (SDN) could bring the known methods to the new level of network defense. Implementation of the proposal method can be easier for deployment instead of the existing by using well known switches supporting OpenFlow protocol, which managed by a network controller module (NOX in our project) to control the forwarding behavior of the OpenFlow supporting switch device.

3.2 Deliverables

Our project is the SDN OpenFlow protocol based solution to control and detect abnormal network behavior in LAN by using OpenFlow supporting switches as network security appliances. We will provide implementation of network anomaly detection algorithm on SDN OpenFlow protocol and compare our solution with existing non-SDN Open Flow solutions and methods.

3.3 Relevance

Existing network security solutions of malware detection based on an external devices (for instance firewall) that collecting a whole traffic from LAN and analyzing the traffic following predefined policy, deep packet inspection, behavioral algorithms or another methods (Figure 2.3.1).

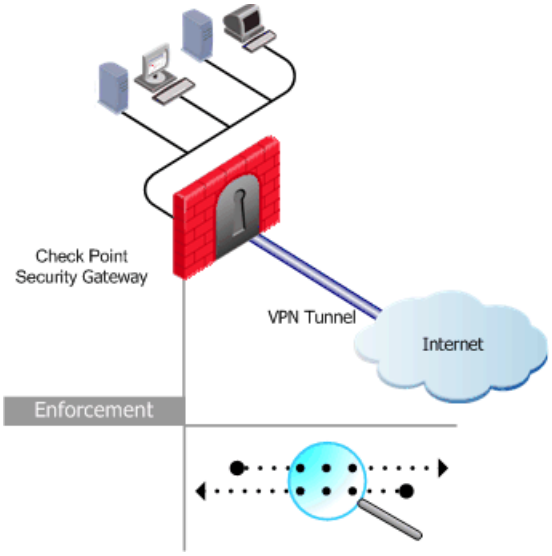


Figure 3.3.a Traditional Fire Wall with Statement Inspection solution

The weak point of the method is the central traffic appliance that cannot monitor and manage communication between hosts connected to the same switch or communicating with hosts behind another switch. It means, that we cannot control internal traffic between “protected”, by firewall and hosts. Any malware could be fast distributed from one host to all LAN in few seconds without to be detected by

Firewall. Another known solution OS Antivirus, but the solution requires periodic update and control that all hosts are recently updated. In addition, Antivirus solution cannot prevent connection and control of mobile devices that could be connected by employees to the internal network.

Our target is to provide central control of switches that provide connect to the internal hosts and can deploy flow based policy on the forwarding traffic.

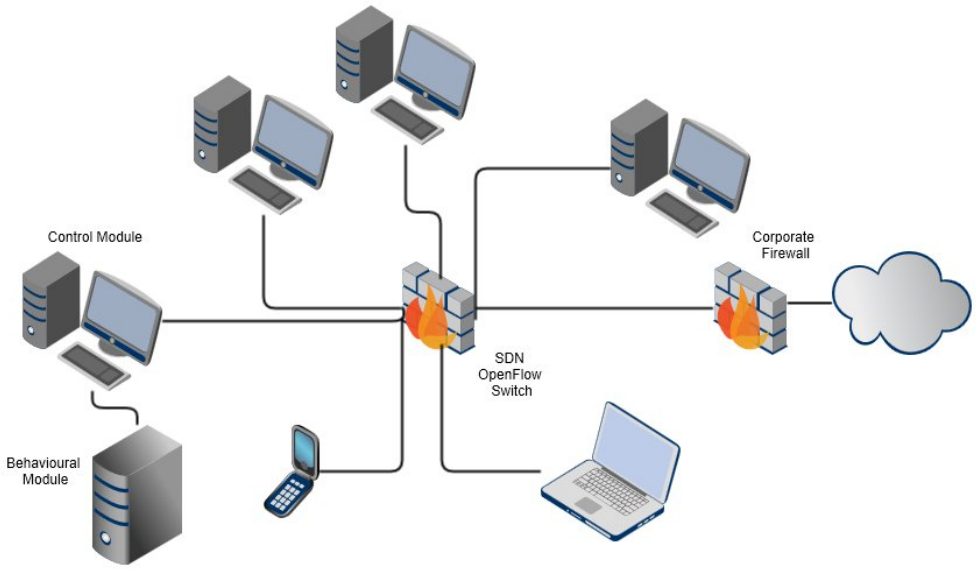


Figure 3.3.b OpenFlow Switch controls traffic of broadcast domain

3.4 Contribution

Our project can bring the existing network security solutions to a new security level and make network administration easier and more effective. Instead of using expensive solutions we can deploy Open Source applications for the existing network devices with user friendly interface and flexible configuration.

Open Source entrepreneurs could find great opportunity in SDN and OpenFlow technology to build Open Source communities, provide custom made features, services and develop new modules such as provided in this project.

3.5 Scope and Platform Definition

To implement and test out theory, we will build a simple SDN OpenFlow based configuration with one OpenFlow supporting switch and NOX controller. The idea is that there is no need for the controller to inspect every packet. The SDN OpenFlow controller applies distributed communication with switch to detect performance and security problems in the LAN networks.

To implement Behavioral Network Security in SDN OpenFlow, we implement the following configuration (Figure 2.5): NOX control module, OpenFlow Ethernet switch (on virtual machine or CISCO switch) and two or more hosts (on virtual machines) with traffic generator such as tcpreplay.

The traffic generator will generate traffic with different rates, than SDN OpenFlow controller will send specific commands to switch based on the behavioral algorithm, and switch will deploy flow policy based on the received commands.

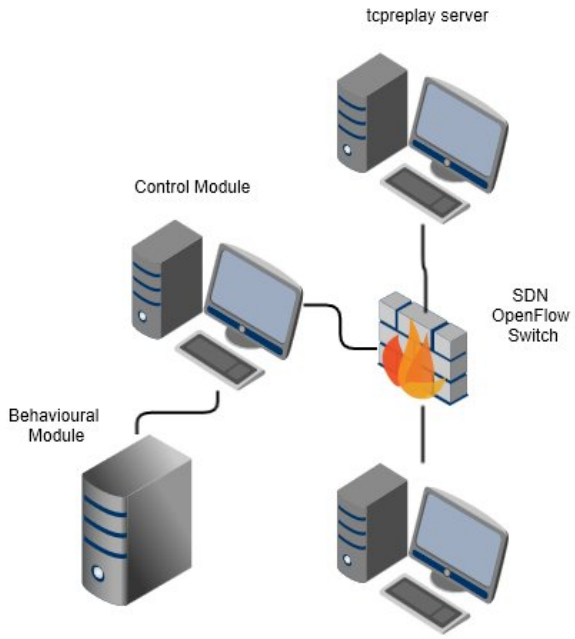


Figure 3.5 Test lab

4. Literature Review

We review academic and technical literature related to computer network, nework security and SDN.

4.1. Computer Network

Computer network, refers to a different location, having individual functions multiple computers and peripheral equipment, connected via a communication line, the network operating system, network management software and network communications protocol management and coordination, sharing of resources and information transfer computer system. Computer Network consists of following features:

1. Can achieve the data transmission and centralized information processing speed
2. Can be shared computer system resources
3. To improve the reliability and availability of computer
4. Can collaborate with each load balancing

Communications inside LAN are based on packets exchange between network peers through the network devices such as switches and routers. Switch is working in Layer 2 (Data Link Layer) and reads each frame at is passing through the network. The Layer 2 switch takes and store every source interface hardware address (MAC address) in a local filter table and keeps track of which physical port the frame was received on. Router is working in Layer 3 (Networking) and connects networks together and route packets of data from one network to another. They break a broadcast domain – the set of all devices on a network segment that hear all the broadcasts sent to the segment (Lammle, T., 2011).

4.2 Security risks

Verizon RISK (Response, Intelligence, Solutions and Knowledge) team published a highly regarded annual Data Breach Investigations Report (www.verizonenterprise.com). From the report we can find that 98 percent of the cyber incidents were done from external agents; 85 percent of attacks took at least weeks to discover and investigate; 81 percent used hacking and 69 percent used malwares. Cybercriminals might break into organization networks and get access to sensitive data that could be sold to the competitors.

Every host in the network is vulnerable to a cyber-attack. If somebody can access the network it means that any host could be compromised.

The cost of the risks is very high. In the worst cases, it can kill a company. Very important to know that cost of attacks including investigation, customer and partner costs, public relation costs, lost of revenue, fines, civil claims and legal fees.

4.3. Introduction to Network Security

Ganguly (2012) defined the security triangle as: confidentiality, integrity and availability. Confidentiality means to secure privacy and sensitive data from illegal usage. Data Integrity refers to the verification and approving of data and source of the data. It means to check data consistence and source of content. Data availability means to make data available to user anytime even under system failover.

Firewalls can provide all parts of the security triangle. Following Skoudis and Liston (2009), the first attempt to implement a firewall was in early 1990 when transferred between computers packets were inspected. The inspection was performed on every packet based on the following five-tuple construction: source and destination IP addresses, source and destination ports and protocol

The next generation of firewalls provided further inspection of the OSI model. In fact, they are now able to inspect layer 4 and performed stateful inspection. It means that firewalls are building connection table of TCP/IP flows dynamically and verify if new packets are part of a registered flow in the table and if the packets are eligible based on the TCP/IP protocol. Stateful inspection firewalls are still widely used today.

Based on Reichenberg (2013), another kind of firewall is known as Unified Threat Protection (UTM) where devices provide Antivirus, IPS, etc. protections.

Next Generation Firewalls (NGFW) provides protection of all OSI layers. The firewalls are designed to filter traffic based on application and user traffic as well. IPS and Antivirus could be implemented on the devices to provide additional protection against malicious traffic.

Another kind of firewalls, referred to as Hypervisor level firewalls, inspect traffic and communication between Virtual Machines (VM).

Cloud-based firewalls are in the beginning of the technology that offers security in the cloud.

We would like to focus on the principles of stateful inspection to explain the idea behind OpenFlow network security. Stateful inspection devices store flow information in a single or multiple flow tables and different data structures. Stateful inspection refers to an extension of packet-by-packet filtering process that tracks protocol flows, enabling predefined packet verifications that extend across flows of packets. Stateful inspection requires at least one flow table that includes five-tuple entries. The table looks up for a match for every arriving packet. The table entry could be shown as the format of <src-addr, src-port, dst-addr, dst-port, ip protocol, state, time> and will be created when the first packet of new flow appears. Subsequent packets of an established session are checked against the session table rather than against the policy configuration and RFC compliances. If packets belong to an existed session, session state and session time will be updated respectively. If a flow session entry has overtime (for example 60 seconds for regular TCP flow), it will be deleted for security and performance reasons (Li et al. 2005).

Our target is Behavioral Profiling solution to analyze and prevent network attacks from inside the network. Network behavioral analysis (NBA) is a sophisticated network security system that creates network traffic baselines to detect network abnormal anomalies (Piper, S., 2013). Existing solutions are recording all traffic that is coming to and form the network and analyze with sophisticated algorithms to uncover potential risks. These devices actually provide “Big Data Security” solution that is not flexible and cost effective for most LAN networks. Our solution based on the OpenFlow technology that provides us with a unique opportunity to get information from the network devices themselves instead of installing an additional network device and allows few additional effective tools to control and block network attacks. We will explain our solution in this work.

4.4. SDN and OpenFlow

McKeown et al (2008) argue, that OpenFlow was created for researchers to investigate network protocols and provide researchers with useful and effective tools. OpenFlow is based on an Ethernet Switches with flow-tables and standard network interfaces that could add or remove flow entries. OpenFlow technology allows to run experimental or heterogeneous switches with they own flow-tables and rules and at the same time, vendors do not need to expose the internal workings.

Many people do not understand the difference between OpenFlow and SDN. This is not surprising, since both the technical terms of the very close relationship. However, the two are not interchangeable. OpenFlow API is similar to the process of configuring the network using switches of the agreement. The SDN is the term used to describe the network infrastructure to provide a programmable interface to automatically provide network services. SDN there is a suspicion of abuse marketers.

In fact, SDN is can be precisely defined. SDN network architecture consists of three levels: the physical network, SDN and SDN controller applications as well.

4.4.1. Physical network

At the lowest level of the physical network, composed of the entire IT infrastructure including networks of all physical devices. We use the " switc.h "to change the term because OpenFlow Ethernet switch works. In this article, you might also consider the physical infrastructure of the virtual switch parts.

4.4.2. SDN Applications

The most visible part of the SDN is designed to provide services for applications such as switch / network virtualization, firewalls and data flow equalizer. (Note that the OpenFlow-based load balancing is known as flow equalizers. they are not in the traditional sense of the load balancing devices, because they cannot read the contents of the packet.) Similar or equivalent to those applications software now runs on dedicated hardware when use application. Most network technology innovations come from SDN applications.

4.4.3. SDN controller

SDN controller is like a whole structure of middleware. The network controller must integrate all physical and virtual devices. Controller to work with the equipment from the SDN software extracts the physical network device, controller and high degree of integration between network devices. In OpenFlow

environment, the controller will use the OpenFlow protocol and NETCONF protocol with the switch contact. (OpenFlow stream data is sent to the switch API, while NETCONF is the network configuration API.)

4.5. OpenFlow impact on the industry chain

As stated previously, OpenFlow vague with PC of the network equipment market may be, but there is still lack of a similar to Microsoft's OpenFlow-based network equipment operating system provider. Theoretically, the operators will prefer network control interface, technology flow operators are happy to DIY their own network, such as data center owners Google, Facebook's servers have been customized with a large number of cheap computing services to build their own server equipment, for network they also have begun ONF started DIY trip.

DIY, the core network equipment value chain differentiation, network switch chip, especially FlowTable processor will be a core value, the controller (which is the aforementioned network operating system) software system is the core value, with these two components a large number of inexpensive network device hardware will emerge on top in the market, which makes the hardware market profits diluted. But this openness will enable network innovation faster, especially when this area have a new birth of Moore's Law Intel stubborn and open source Linux operating system.

Communications industry cycle, innovative features taking an operator raised demand -> equipment supplier analysis needs -> Standard Organization Standardization -> equipment suppliers to achieve -> operator to test and adopt the long path, cycle to calculated for several years, and the Internet service providers often set operators and suppliers in one, from the demand the adoption of innovative features found -> Design -> Development -> on-line commercial entity in a complete and functional development process can continue to receive user feedback and improve the design, which is called Web Business Development "will always be Beta versions" concept, applied to network design, operators can design their own network topology algorithm, and network performance statistics can be iteratively adjusted. Thus the prospect of such OpenFlow network innovation may result from the giant monopoly supplier into operator-led innovation.

4.6. OpenFlow faced technical difficulties

OpenFlow promotion is facing technical difficulties. One of them is the router / switch is widely used to quickly find TCAM memory costs. In conventional devices, the need to adopt TCAM table including FIB, MAC, MPLS Label and ACL tables, each matching field length varies, designed separately and is designed for maximum capacity, in order to achieve minimal overhead. In OpenFlow devices, no longer distinguish matching short length FIB, MAC, MPLS Label ACL tables and long tables, all using the maximum record length FlowTable substitute for OpenFlow1.0 terms, Flow table matches the field length long up to 252 bits, and the general IPV4 RIB TCAM matching field length of only 60-80 bits, spending increased more than 3 times, and for the purposes of the router line cards, TCAM costs accounted for about 20-30%, power consumption also accounted for a large part. Therefore, how to reduce FlowTable size will be OpenFlow system faces an enormous problem; in addition, TCAM system under dynamic insertion algorithm FlowTable record will be more complicated.

OpenFlow1.1 designed a multi-level flow table to reduce the overhead FlowTable will flow table for feature extraction, broken down into several steps, forming pipeline processing form, thus reducing the total number of records flow table, for example, a total of 10,000 original flow table , broken down into the first match VLAN + MAC, and then match the IP and transport layer header table two separate streams, each stream is less than the number of tables may 1000, making the total number of flow table is less than 2000. But pipelined architecture makes matching delay increase, and traffic generation, maintenance algorithm complexity increase, so far have not seen for a real network effect assessment report.

OpenFlow is the key to the network through the OpenFlow forwarding abstract atomic operations, and to drive the process flow table, we discussed all the benefits is based on OpenFlow FlowTable can well be Primitive and Workflow abstraction to support the design of new protocols indeed in most cases can only modify the Controller logic to achieve on this assumption. OpenFlow Switch initial application in the field, OpenFlow has a suspicion of overkill. But the route network, especially in the control logic contains a large number of users border router, such as BRAS, wireless network GGSN / PDSN / xGW other equipment, OpenFlow need to expand the user control logic is abstracted as atomic operations and processes that may have been unsuitable called FlowTable, called AccessControlTable more appropriate matching rules forwarding operation itself, forwarding operations also need to increase access to existing protocols and

characterization of a variety of access session protocol field of support, such as PPPoE, GTP-U, PDP activation of matching and forwarding packages support.

5. Project Design

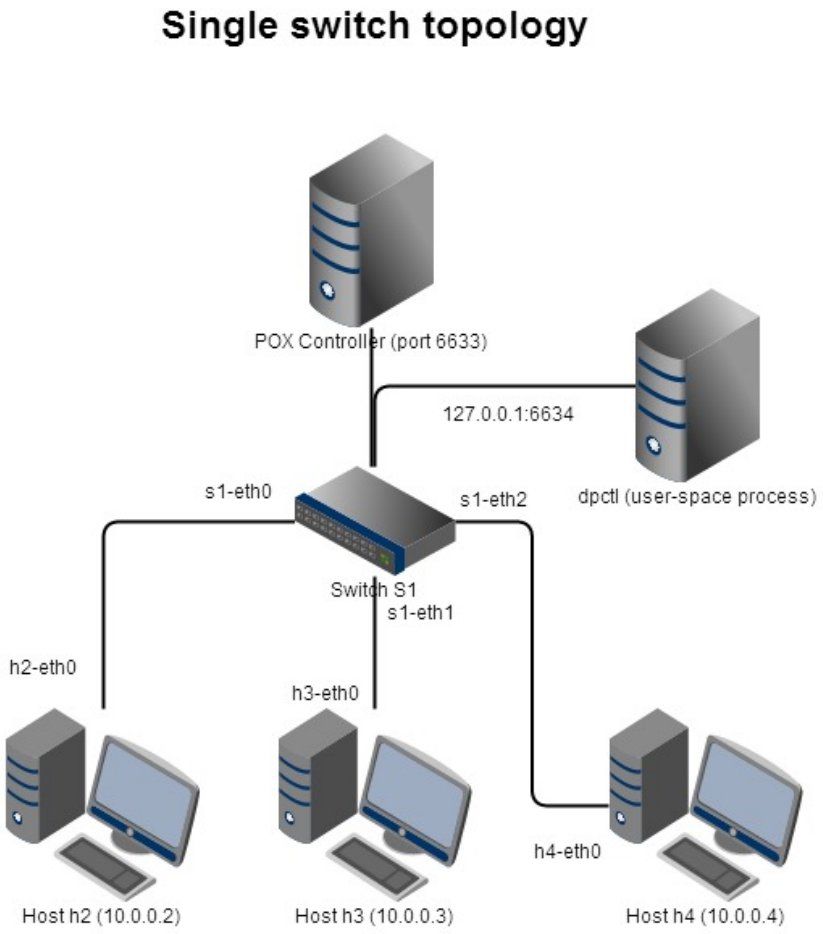
The project based on two main parts: the basic configuration and firewall implementation.

5.1. The basic configuration

We built test environment of virtual network of three hosts connected by switch to test basic OpenFlow functionality. To verify the lab functionality we tested connectivity between hosts through the virtual switch.

In the second stage we managed switch with POX controller, reconfigured switch to hub and vice versa. This test provided us with information how to manage OpenFlow switch, control traffic routing and deploy flows to the switch.

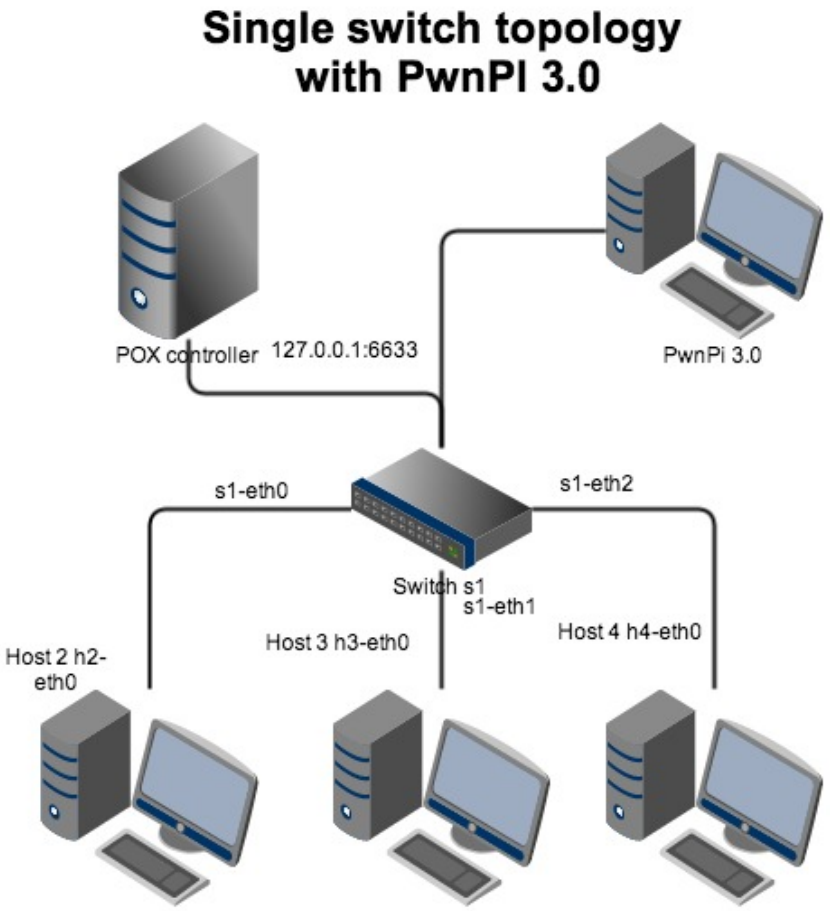
In the final stage we will test POXFW and switch behavior. We will send HTTP (port 80) traffic to IP address 10.0.0.2 (h2) to verify if POXFW will send right commands to the OpenFlow switch to drop traffic.



5.2. Firewall implementation

To implement firewall functionality we added traffic filtering and decision making mechanisms to POX. POX is a Python-based SDN controller platform geared towards research and education and allows flexibility in development.

To test our solution we used PwnPi 3.0 penetration test tool kit that was installed on the Raspberry PI hardware platform. Raspberry PI was connected to the virtual switch through the Virtual Box interface and generated a list of attacks that the POX controller had to identify and make the appropriate actions as a drop or allow traffic.



6. Project Implementation

In this section we will explain and provide with guide how to create and test our solution on virtual environment Mininet.

6.1. OpenFlow virtual model

We built a Single Switch OpenFlow topology based on Ubuntu virtual environment Mininet:

https://github.com/downloads/mininet/mininet/mininet-2.0.0-113012-amd64-ovf.zip

First of all we created Virtual Machine with Mininet network based on one POX controller, one OpenFlow switch and three hosts connected trough the controller. The machine was installed on VritualBox Server version 4.2.16.

6.2. Penetration tool PwnPi 3.0 on Raspberry PI 512.

To generate malicious traffic we used PwnPI 3.0 on Raspberry PI tool.

PwnPi is a Linux-based penetration testing dropbox distribution for the Raspberry Pi. It currently has 200+ network security tools pre-installed to aid the penetration tester. It is built a stripped down version of the Debian Wheezy image from the Raspberry Pi foundation's website and uses Openbox as the window manager. PwnPi can be easily setup to send reverse connections from inside a target network by editing a simple configuration file (http://pwnpi.sourceforge.net/). Appendix 9.2.

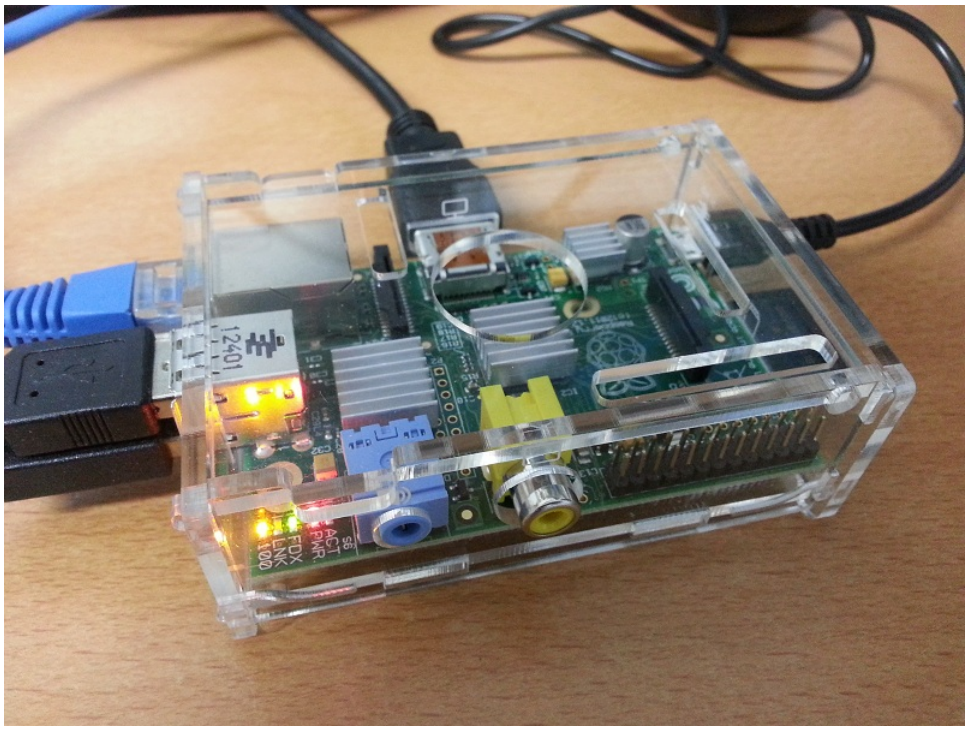


Figure 6.2 Raspberry Pi 512

6.3 POX based Firewall development

To identify malicious activities we developed a POX based Firewall that will notice any abnormal network activities in the virtual network.

The basic model can identify traffic to the specific port (80 HTTP) and destination IP (10.0.0.2). POX controller sends specific rules to the OpenFlow switch to allow or deny the specific flows based on packets and their suitability to RFC. In the shown example we check suitability to RFC of IPv4 and TCP/IP. The structure of the component follows the same pattern as the other. We store the physical addresses and ports of the OpenFlow tuples in the connection table. Very important to store switch ID for control, because each switch will have its connection table when instantiating the POXFW class. In our basic example all TCP packets with destination port 80 and destination IP address 10.0.0.2 will not be switched on the network (POXFW code in Appendix 9.3).

6.4 Testing

To test the Firewall, we run POXFW.py with POX controller under MININET environment in the installed POXFW.py in ~/pox/ext folder.

Firstly, we enabled Firewall in debug mode:

$./pox.py log.level --DEBUG firewall

We generated HTTP traffic (port 80) to IP address 10.0.0.2 by PwnPI 3.0.

In addition, we tested internal MININET tools to generate additional traffic of ping ping, iperf or pingall to verify switch behavior between virtual host's h1 and h3 (under MININET shell):

Mininet>iperf -c -p 80

Different scenario:

Mininet>iperf -c -p 25

6.5 Results

We found that HTTP (port 80) traffic to 10.0.0.2 was blocked by OpenFlow switch when SMTP (port 25) traffic was accepted and flooded to all ports.

It should be noted that, in this particular scenario, packets sent to and from h2 will be blocked as well due to they are from the blocked IP 10.0.0.2.

As result, we can see that SDN controller with OpenFlow protocol allow us to identify any abnormal behavior by configuration of the POXFW module. It means that we can use virtual networks such as MININET to control and prevent any kind of malicious activity of the connected hosts or network devices. In additional, we can investigate in practice how the OpenFlow protocol works and operates the exchange of messages between switches and the controller, and of course to understand the structure and be able to develop POX controller based products.

7. Business model

We know that POX controller got Apache License version 2.0, therefore we can identify an option for Open Source business model from developer's point of view.

Potential company could develop additional modules such as provided in the article with different functionality. POX community will define general policy and rules, and then we can develop an efficient modular model, and then get our products optimized and efficient.

We can identify that the business model will be based on low skill diversity and high modularity. POX has to improve core modules to be able to adopt new modules from Open Source community.

We can say, that now is the Bootstrapping stage (Weiss 2012) – we can use existing Open Source in our projects to get better product, for example GUI, or develop new modules such as Firewall.

In addition, we can develop better POX controller and create a new Open Source community where external innovators will contribute to our product, and we will provide support or customization services.

We can identify SDN Open Source community as a group (vertical pyramid) of other Open Source communities with a wide range of contributors. Every community in this model contributes each other.

We found in SDN and OpenFlow technologies great business opportunity and academic value.

8. Conclusion

Nowadays, computer networks are very complicated and have expensive elements. Existing modern network security solutions for Zero-Day attack identification are focused on a single host that could be analyzed in some virtual environment. Our model provides cheap and efficient virtual solution with easy to deploy and flexible for programming platform. Very important point is that POX now has Apache License version 2.0 agreements and could be used in wide range of network products. It means that we have an opportunity to develop Open Source products and use well known business models for the further business development.

Further development of the provided idea could be development of the self-study firewall that could dynamically identify abnormal activities and conventional traffic. Another major part is GUI development both for POX controller and for modules such as POXFW.

The major network virtualization issue is to develop an abstraction layer on the physical network infrastructure. This technology, used in projects such as

GENI, proposes a new way of organizing computer networks using virtual machines to represent virtual switches in the network. However, this type of virtualization faces the challenge of achieving an efficient implementation in the data plane switches. Today networks have hardware optimization of packet forwarding that reached significant advantage in performance on software solutions. In this context, Software Defined Networking (SDN) could be an alternative for the network virtualization model development. SDN networks offer a new way to virtualize the network, not just with independent elements of the virtual network, but with the partitioning of address spaces of connection tables.

This abstract vision of computer network and implicit Open Source capabilities can be offered to every potential integrator who needs to implement a new service on the existing physical network. It is great Open Source business opportunity for quick-thinking entrepreneurs.

9. References

1. Aster * x: Load-Balancing as a Network Primitive, http://www.stanford.edu/ ~ nikhilh/pubs/handigol-acld10.pdf
2. ElasticTree: Saving Energy in Data Center Networks, http://static.usenix.org/event/nsdi10/tech/full_papers/heller.pd
3. Ethane Projects Home, http://yuba.stanford.edu/ethane/
4. FlowVisor: A Network Virtualization Layer http://www.openflow.org/downloads/technicalreports/openflow-tr-2009-1-flowvisor.pdf
5. Ganguly, D., 2012. Network and Application Security. Fundamentals and Practices. Taylor & Francis Group, LLC
6. GENI. Global Environment for Network Innovations, 2012. http://www.geni.net/
7. Lammle, T., 2011. CCNA Study Guide. Sybex
8. Goldberg I., Wagner D., Thomas R., and Brewer E., 1996. A Secure Environment for Untrusted Helper Applications (Confining the Wily Hacker)"
9. Li, X., Ji, Z. and Hu, M., 2005. Stateful Inspection Firewall Session Table Processing. International Journal of Information Technology, Vol. 11 No. 2
10. OpenFlow: Enabling Innovation in Campus Networks, www.openflow.org / Documents / OpenFlow-WP-latest.pdf
11. Open Networking Summit2012 schedule, http://opennetsummit.org/speakers.html
12. OpenFlow Switch Specification v1.3.0, https://www.opennetworking.org/images/stories/downloads/specification/openflow-spec-v1.3.0.pdf
13. OpenFlow v1.0.0 source code, http://openflowswitch.org/downloads/openflow-1.0.0.tar.gz
14. Piper, S., 2013. Big Data Security for Dummies. Solera Networks
15. Reichenberg, N., 2013. The Firewall: From Past to Present... and Beyond. Security Week. April 11, 2013
16. Sane Projects Home, http://yuba.stanford.edu/sane/

17. SDN Standards: What and Whatnot, http://opennetsummit.org/talks/ONS2012/pitt-tue-standards.pdf
18. Skoudis, E., Liston, T. 2009. Counter Hack Reloaded. Second Edition. Prentice Hall
19. Software Defined Networking: The New Norm for Networks, https://www.opennetworking.org/images/stories/downloads/white-papers/wp-sdn-newnorm.pdf
20. Stanford Clean Slate project website, http://cleanslate.stanford.edu/
21. Technology Review About 2009 Top Ten selection of cutting-edge technology, http://www.technologyreview.com/article/412194/tr10-software-defined-networking/
22. The OpenFlow Switch Specification. Available at http://OpenFlowSwitch.org.

10. Appendix Section

10.1 Install Mininet

You will need to download these files individually.

The files include virtualization software, a SSH-capable terminal, an X server, and the VM image.

The tutorial image is distributed as a compressed VirtualBox image (vdi). VirtualBox enables you to run a virtual machine inside a physical machine, and is free and available for Windows, Mac and Linux. You can export the VirtualBox image to vmdk format and use it with VMWare using the instructions below.

The following instructions assume the use of VirtualBox, but the instructions should apply regardless of virtual software after you complete the initial setup.

Download Files

You'll need to download the files corresponding to your OS, plus the tutorial VM.

Start now with downloading a compressed VM image:

- Virtual Machine Image (OVF format, 64-bit, Mininet 2.0)

Important: For this VM image, the user name is 'mininet' with password 'mininet'.

The OVF format can be imported into VirtualBox, VMware, or other popular virtualization programs.

If this does not work for you, you can also try our older VM image that uses Mininet 1.0:

- VirtualBox VM Image (zipped VM image, 32-bit, Mininet 1.0)

Important: For this VM image, the user name is 'openflow' with password 'openflow'.

You will also need virtualization software, an X server, and an ssh-capable terminal emulator:

OS Type	OS Version	Virtualization Software	X Server	Terminal
Windows	7+	VirtualBox	Xming	PuTTY
Windows	XP	VirtualBox	Xming	PuTTY
Mac	OS X 10.7-10.8 Lion/Mountain Lion	VirtualBox	download and install XQuartz	Terminal.app (built in)
Mac	OS X 10.5-10.6 Leopard/Snow Leopard	VirtualBox	X11 (install from OS X main system DVD, preferred), or download XQuartz	Terminal.app (built in)
Linux	Ubuntu 10.04+	VirtualBox	X server already installed	gnome terminal + SSH built in

Install and Verify

After you have downloaded the appropriate software and VM images, make sure that each column item (X server, Virtualization software, and SSH terminal) is installed and working for your platform, and that the VM image loads and runs correctly for your configuration.

Setup Virtual Machine

Import Virtual Machine Image

If you downloaded the .ovf image,

- **Start up VirtualBox, then select File>Import Appliance and select the .ovf image that you downloaded.** You may also be able to simply double-click the .ovf file to open it up in your installed virtualization program.
- **Next, press the "Import" button.** This step will take a while - the unpacked image is about 3 GB.
- **Continue with Finish VM Setup below.**

Note: If you downloaded the older .zip archive instead of the .ovf image, setup requires a few more steps:

OS X Users: Double-click the virtual machine image to extract it.

Linux Users (and OS X users who prefer the command line): from a terminal, unzip the virtual machine image, e.g.:

```
$ unzip OpenFlowTutorial-101311.zip
```

(If you downloaded a different VM image, make sure you use its correct file name.)

Windows Users: unzip the image in Windows Explorer.

For the .zip archive, you need to set up a new VirtualBox VM. Open VirtualBox.

- Select New
- Press Continue in the next prompt.
- Name your VM OpenFlowTutorial, Operating System Linux, Version Ubuntu. Click Continue.
- Set the memory at 512MB and click Continue
- At that point VB should ask you to Create a new hard disk, or use the existing one. Select "Use existing hard disk".
- Click the icon to select the hard disk. This will open the Virtual Media Manager Window.
- Press Add, and find the extracted OpenFlowTutorial*.vdi from the previous steps. Click Select and then Continue.
- Your VM installation is complete. Press Done.

Finish VM Setup

You will need to complete one more step before you are done with the VM setup.

Select your VM and go to the Settings Tab. Go to Network->Adapter 2. Select the "Enable adapter" box, and attach it to "host-only network".(Sidenote: on a new VirtualBox installation you may not have any "host-only network" configured yet. To have one select File menu/Preferences/Network and "Add host-only network" button with default settings. Then you can try the attach.) This will allow you to easily access your VM through your host machine.

At that point you should be ready to start your VM. Press the "Start" arrow icon or double-click your VM within the VirtualBox window.

In the VM console window, log in with the user name and password for your VM.

Note that this user is a sudoer, so you can execute commands with root permissions by typing `sudo command`, where *command* is the command you wish to execute with root permission.

Choose Preferred Editor

Nano, Vim, Emacs, and Gedit come installed on the OpenFlowTutorial VM. Brief instructions for each:

Nano: You can immediately modify a file. When you're done, hit 'ctrl-x', then say 'Yes' to the prompt, to save and quit.

Vim: to modify a file, type 'i' to enter Insert mode, then use the arrow keys to navigate and edit. When you're done, hit 'esc', type ':wq', then press enter, to save and quit.

Highly recommended for the NOX tutorial: add the following to ~/.vimrc in the VM:

```
set tabstop=4 set expandtab
```

Emacs: you can immediately modify a file. When you're done, hit 'ctrl-x', 'ctrl-s', then hit 'ctrl-x', 'ctrl-c' to exit.

Gedit: a graphical text editor, no instructions needed.

Eclipse: Eclipse and its dependencies would require about 500MB extra space on the VM image, so it's not shipped by default. If you have Eclipse installed on the host VM, using the Remote Systems Explorer can be a convenient way to access and modify text files on the VM, with many of the advantages of Eclipse, such as syntax highlighting.

If you have another preferred text editor, feel free to install it now:

```
$ sudo apt-get install <editor>
```

Command Prompt Notes

In this tutorial, commands are shown along with a command prompt to indicate what subsystem they are intended for . For example,

```
$ ls
```

indicates that the `ls` command should be typed at a Unix (e.g. Linux or OS X) command prompt (which generally ends in `$` if you are a regular user or `#` if you are root.

Other prompts used in this tutorial include

```
mininet>
```

for commands entered in the Mininet console and

```
C:>
```

for code entered into a Windows command window.

Set Up Network Access

The tutorial VM is shipped without a desktop environment, to reduce its size. All the exercises will be done through X forwarding, where programs display graphics through an X server running on the host OS.

To start up the X forwarding, you'll first need to find the guest IP address.

VirtualBox

If you are running VirtualBox, you should make sure your VM has **two network interfaces**. One should be a **NAT interface** that it can use to access the Internet, and the other should be a **host-only interface** to enable it to communicate with the host machine. For example, your NAT interface could be eth0 and have a 10.x IP address, and your host-only interface could be eth1 and have a 192.168.x IP address. You should ssh into the **host-only interface**at its associated IP address. Both interfaces should be configured using DHCP. If they are not already configured, you may have to run dhclient on each of them, as described below.

Access VM via SSH

In this step, you'll verify that you can connect from the host PC (your laptop) to the guest VM (OpenFlowTutorial) via SSH.

From the virtual machine console, log in to the VM, then enter:

```
$ ifconfig -a
```

You should see three interfaces(eth0, eth1, lo), Both eth0 and eth1 should have IP address assigned. If this is not the case, type

```
$ sudo dhclient ethX
```

Replacing ethX with the name of a downed interfaces; sometimes the eth ports appear as eth2 or eth3, you can fix this by editing /etc/udev/rules.d/70-persistent-net.rules and removing the existing configuration lines.

Note the IP address (probably the 192.168 one) for the **host-only** network; you'll need it later. Next, log in, which will depend on your OS.

Mac OS X and Linux

Open a terminal (Terminal.app in Mac, Gnome terminal in Ubuntu, etc). In that terminal, run:

```
$ ssh -X [user]@[Guest IP Here]
```

Replace [user] with the correct user name for your VM image.

Replace [Guest IP Here] with the IP you just noted. If ssh does not connect, make sure that you can ping the IP address you are connecting to.

Enter the password for your VM image. Next, try starting up an X terminal using

```
$ xterm
```

and a new terminal window should appear. If you have succeeded, you are done with the basic setup. Close the xterm. If you get a 'xterm: DISPLAY is not set error', verify your X server installation from above.

Windows

In order to use X11 applications such as xterm and wireshark, the Xming server must be running, and you must make an ssh connection with X11 forwarding enabled.

First, start Xming (e.g. by double-clicking its icon.) No window will appear, but if you wish you can verify that it is running by looking for its process in Windows' task manger.

Second, make an ssh connection with X11 forwarding enabled.

If you start up puTTY as a GUI application, you can connect by entering your VM's IP address and enabling X11 forwarding.

To enable X11 forwarding from puTTY's GUI, click puTTY->Connection->SSH->X11, then click on Forwarding->"Enable X11 Forwarding", as shown below:

You can also run putty (with the -X option for X11 forwarding) from the Windows command line:
Open a terminal: click the Windows 'Start' button, 'run', then enter 'cmd'.

Change to the directory where you saved putty.

```
C:> cd <dir>
```

Run:

```
C:> putty.exe -X openflow@[Guest IP Here]
```

Replace [Guest IP Here] with the IP you just noted.

If putty cannot connect, try pinging the VM's IP address to make sure you are connecting to the correct interface.

```
C:> ping [Guest IP Here]
```

Once the ssh connection succeeds or a terminal window for the VM pops up, log in to the VM. Now, type:

```
$ xterm -sb 500
```

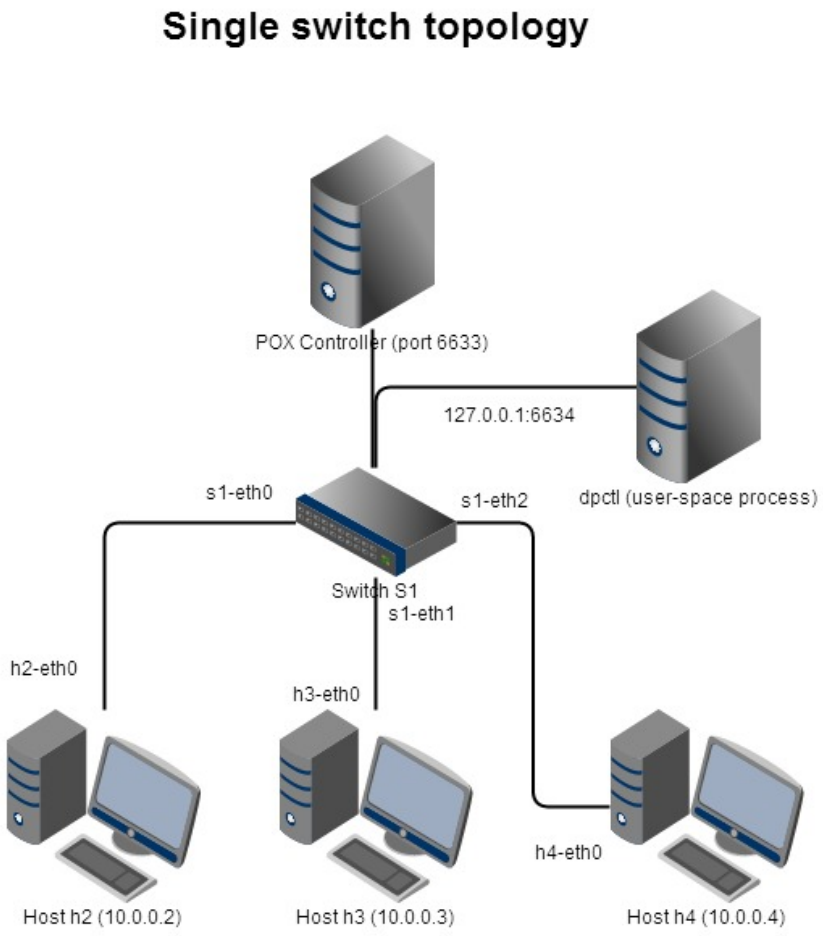
to start an X terminal (the -sb 500 is optional but gives 500 lines of scrollback.)

A white terminal window should appear. If you have succeeded, you are done with the basic setup. Close the xterm.

If the xterm window does not appear, or if you get an error like "xterm: DISPLAY is not set," make sure that Xming is running in Windows and that you have correctly enabled X11 forwarding.

Start Network

The network you'll use for the first exercise includes 3 hosts and a switch (and, eventually, an OpenFlow controller, but we'll get to that later)



To create this network in the VM, in an SSH terminal, enter:

```
$ sudo mn --topo single,3 --mac --switch ovsk --controller remote
```

This tells Mininet to start up a 3-host, single-(openvSwitch-based)switch topology, set the MAC address of each host equal to its IP, and point to a remote controller which defaults to the localhost.

Here's what Mininet just did:

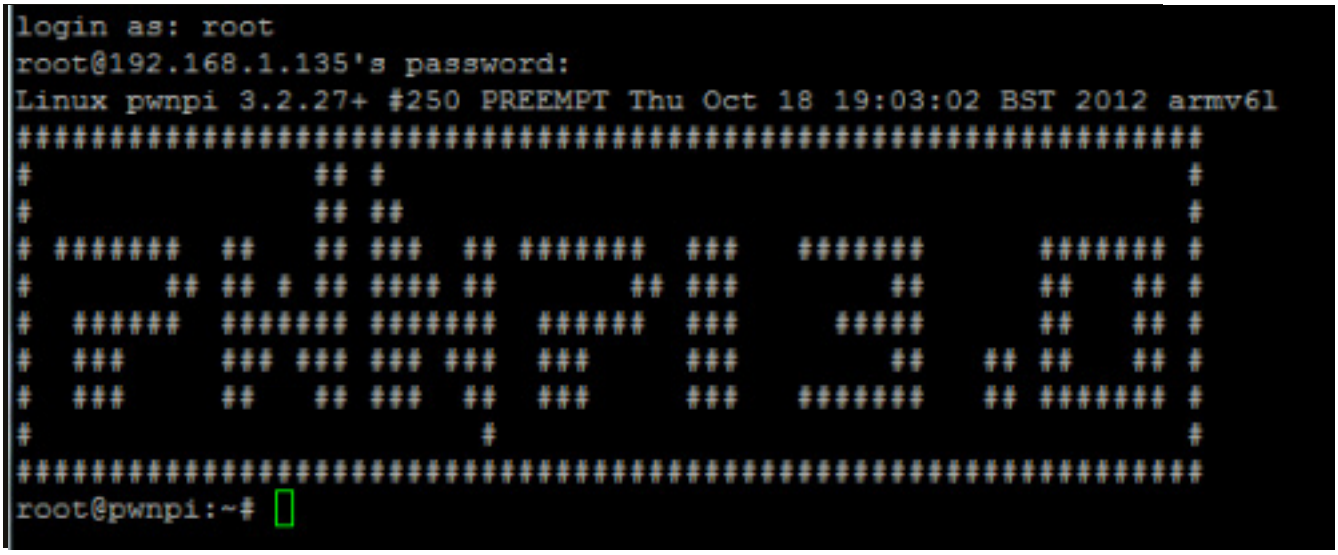
- Created 3 virtual hosts, each with a separate IP address.
- Created a single OpenFlow software switch in the kernel with 3 ports.
- Connected each virtual host to the switch with a virtual ethernet cable.
- Set the MAC address of each host equal to its IP.
- Configure the OpenFlow switch to connect to a remote controller.

10.2 Installing PwnPi on Rasberry Pi

To install this tool we need Rasberry Pi 512 model B with SD card 8GB or larger and Windows System to install PwnPi image onto the SD card.

Step-by-step instructions:

1. Download the PwnPi image form RasberryPi from http://sourceforge.net/projects/pwnpi/files/pwnpi-3.0.img.7z/download
2. Install the image to SD card by http://sourceforge.net/projects/win32diskimager/
3. Connect SD card with PwnPi into the SD slot on RasberryPi. Connect video, Ethernet cables, keyboard and mouse and boot.
4. Now we can run PwnPi with root credentials: login: root password: toor



10.3 POXFW code (Python)

```
############################################################
# OpenFlow POXFW                                           #
# We catch communication to the specific destination port  #
# and specific destination IP address                      #
############################################################

from pox.core import core
from pox.lib.util import dpidToStr
from pox.lib.revent import *
import pox.openflow.libopenflow_01 as of
import pox.lib.packet as pkt

log = core.getLogger( )
DST_IP_ADDRRESS = "10.0.0.2"
DST_PORT = 80

class POXFW ( object ):

        def __init__(self,connection):
        #Init function for POX controller and OpenFlow switch and POXFW rules
                self.connection=connection
                connection.addListeners(self)
                self.mac_to_port={}
                self.install_POXFW_rules( )


        def install_POXFW_rules(self):
        #Define a basic FW rules of accept or drop packets to the specific port.

                log.debug("First rule implementation $s, send packets with DST port to the controller"
%(dpidToStr(self.connection.dpid)))

                #create flow
                msg = of.ofp_flow_mod()
                #message to switch to get the flow and catch by FW rule when DST port is equal to the wanted
                msg.match = of.ofp_match (dl_type = pkt.ethernet.IP_TYPE, nw_proto =
pkt.ipv4.TCP_PROTOCOL, tp_dst = DST_PORT)
                #Get a packet with DST port: DST_PORT
                msg.actions.append(of.ofp_action_output(port = of.OFPP_CONTROLLER))
                #Send the command to the controller
                self.connection.send (msg)

                log.debug("Second rule implementation %s, IP packets that are sent to" %DST_IP_ADDRRESS
%(dpidToStr(self.connection.dpid)))

                #create flow
                msg = of.ofp_flow_mod()
                #message to switch to get the flow and catch by FW rule when DST IP is equal to the wanted
                msg.match = of.ofp_match(dl_type = pkt.ethernet.IP_TYPE, nw_dst = DST_IP_ADDRRESS)
                #Get a packet with DST IP is equal to the wanted
                msg.actions.append(of.ofp_action_output(port = of.OFPP_CONTROLLER))
                #Send the command to the controller
                self.connection.send(msg)


        def act_like_switch(self,packet_in,packet):
        #Define switch behavior
                #Assign MAC addresses to ports
                self.mac_to_port[str(packet.src)] = packet_in.in_port
```

```
                if str(packet.dst) in self.mac_to_port:
                        port = self.mac_to_port[str(packet.dst)]
                        log.debug("Send packet to port %d"%(port))
                        #create flow
                        msg = of.ofp_flow_mod()
                        msg.match = of . ofp_match . from_packet ( packet )

                        #Create flow and control that traffic from the same flow is going to the correct port
                        log.debug("Create flow %s.%i of type %d" %(packet.dst,port,packet.type))
                        msg.idle_timeout = 10
                        msg.hard_timeout = 30

                        if packet_in.buffer_id !=-1 and packet_in.buffer_id is not None:
                                msg.buffer_id = packet_in.buffer_id
                        else:
                                if packet_in.data is None:
                                        return
                                        msg.data = packet_in.data
                                        msg.actions.append(of.ofp_action_output(port = port))
                                        self.connection.send(msg)
                                else:
                                        port = of.OFPP_FLOOD
                                        log.debug("Flooding to all ports")
                                        msg = of.ofp_packet_out()
                                        msg.in_port = packet_in.in_port

                        if packet_in.buffer_id !=-1 and packet_in.buffer_id is not None:
                                msg.buffer_id = packet_in.buffer_id
                        else :
                                if packet_in.data is None:
                                return
                                msg.data = packet_in.data
                                action = of.ofp_action_output(port = port)
                                msg.actions.append(action)
                                self.connection.send(msg)

        def _handle_PacketIn(self,event):
                #We handle packets from the network that are coming through the switch
                log.debug("Packet on the interface %s",dpidToStr(event.dpid))
                packet_eth = event.parsed

                if not packet_eth.parsed :
                        log.warning("Corrupted packet")
                        return

                #Checks suitability of the packets to RFC of IPv4 TCP and generate related messages
                if packet_eth.type == pkt.ethernet.IP_TYPE:
                        packet_ip = packet_eth.payload

                if packet_ip.dstip == DST_IP_ADDRRESS:
                        log.warning("Packet DST is" % DST_IP_ADDRRESS)
                        return

                if packet_ip.protocol == pkt.ipv4.TCP_PROTOCOL:
                        packet_tcp = packet_ip.payload

                if packet_tcp.dstport == DST_PORT:
                        log.warning("Packet DST port is" %DST_PORT)
                        return

                packet_in = event . ofp
                self.act_like_switch(packet_in,packet_eth)
```

```
def launch ( ):

def enable_switch(event):
        POXFW(event.connection)
        core.openflow.addListenerByName("Started connection",start_switch)
```