# Hướng dẫn cài đặt OpenFlow Switch trên NetFPGA

## Cơ cấu hoạt động

Cơ cấu hoạt động của OpenFlow Switch trên NetFPGA :

Để NetFPGA hoạt trở thành OpenFlow Switch thì chỉ cần nạp file cấu hình phần cứng (bit file) đã được compile sẵn là được (openflow\_switch.bit). Tuy nhiên, để có thể hoạt động được như một Openflow switch thực thụ thì nó cần phải có một phần để làm nhiệm vụ đóng gói bản tin theo giao thức Openflow và giao tiếp với controller và phần này nằm ở phần mềm. Như vậy việc cài đặt OpenFlow Switch trên NetFPGA thực chất là cài phần mềm (driver) để giao tiếp trung gian giữa phần cứng NetFPGA và Controller.

Các bước thực hiện chính như sau :

* Cài Centos 6.5 i686
* Gắn NetFPGA lên khe PCI thứ nhất (phía dưới)
* Cài để Centos nhận NetFPGA (xuất hiện các interface của NetFPGA)
* Cài Driver (phần mềm) trung gian giao tiếp giữa controller và NetFPGA
* Chạy OpenFlow Switch
  + Reset card
  + Nạp bitfile (Openflow\_switch.bit)
  + Chạy driver giao tiếp
* Dừng OpenFlow Switch
  + Reset card
  + Kill all tiến trình chạy Driver

## Chi tiết cài đặt

*yum -y install git automake pkgconfig libtool gcc*

*wget http://ftp.gnu.org/gnu/autoconf/autoconf-2.63.tar.gz*

*tar xvzf autoconf-2.63.tar.gz*

*cd autoconf-2.63*

*./configure --prefix=/usr*

*make*

*make install*

*cd*

*git clone git://gitosis.stanford.edu/openflow.git*

*cd openflow*

*git checkout -b 1.0.0-netfpga origin/devel/tyabe/1.0.0-netfpga*

*./boot.sh*

*cd /root/openflow/hw-lib/nf2*

*wget* [*http://openflow.org/downloads/netfpga/openflow\_switch.bit.100\_3.tar.gz*](http://openflow.org/downloads/netfpga/openflow_switch.bit.100_3.tar.gz)

*# Tải bit file cấu hình phần cứng NetFPGA*

*tar xfvz openflow\_switch.bit.100\_3.tar.gz*

*cd /root/openflow*

*./configure --enable-hw-lib=nf2 --enable-hw-tables=nf2 --enable-ndebug*

*make*

*make install*

*cd*

*sudo yum -y install perl-Convert-Binary-C perl-Data-HexDump perl-Net-Pcap perl-Net-RawIP.i386 perl-Error.noarch*

*wget ftp://ftp.tw.freebsd.org/pub/ports/distfiles/Getopt-Long-2.38.tar.gz*

*tar xvf Getopt-Long-2.38.tar.gz*

*cd Getopt-Long-2.38*

*perl Makefile.PL*

*make all test*

*make install*

*cd ~/*

*cp openflow/regress/scripts/env\_vars .*

*### chu y co dau "." o cuoi*

*nano env\_vars*

*### export OF\_ROOT=/root/openflow ### thay doi duong dan~ den file openflow*

*### export PERL5LIB=${OFT\_ROOT}/lib/Perl5 ### xoa bot' phan phia sau*

*sudo /sbin/chkconfig avahi-daemon off*

*sudo /etc/rc.d/init.d/avahi-daemon stop*

*sudo /sbin/chkconfig ip6tables off*

*echo 'options ipv6 disable=1' >> /etc/modprobe.d/disable-ipv6.conf*

*nano /etc/modprobe.conf*

*### add adalias ipv6 off*

*### alias net-pf-10 off*

*nano /etc/sysconfig/network*

*## add NETWORKING\_IPV6=no*

*nano /root/openflow/regress/bin/nf2.map*

*### eth1,eth2,eth3,eth4 => eth0, eth0, eth0*

*source /root/env\_vars*

Trên đây là các bước để cài đặt phần Driver trung gian này. Bước tiếp theo là chạy Switch

## Chạy OpenFlow Switch

### Chạy Openflow Switch

Để chạy Openflow Switch trên NetFPGA chúng ta cần chạy các bước như sau :

* */usr/local/sbin/cpci\_reprogram.pl -all*

### reset all netfpga card

* *nf\_download /root/openflow/hw-lib/nf2/openflow\_switch.bit*

### Nạp bitfile

* *ofdatapath punix:/var/run/test -d 000000000001 -i nf2c0,nf2c1,nf2c2,nf2c3 &*

### dpid : 00:00:00:00:00:00:00:01 .

### --detach : run in background as daemon

### chú ý lần đầu tiên sau bật máy mà chạy lệnh này sẽ bị báo ‘no such file or directory” và phải chạy lại lệnh này mới được

* *ofprotocol unix:/var/run/test tcp:192.168.1.254:6633*

### kết nối tới controller ở địa chỉ 192.168.1.254:6633

###--out-of-band cho phép chạy ẩn không bị out khi ctr C hay tat terminal

Ngoài các tùy chọn được mô tả trong các câu lệnh khai báo ở trên thì ngoài ra còn có thể có nhiều khai báo thêm ví dụ như sau :

*ofprotocol unix:/var/run/test --max-backoff=5 --log-file=/root/Desktop/log\_file.txt --fail=open --rate-limit=10000 tcp:127.0.0.1:6633*

### thời gian kết nối lại đến controller : 5s ,

### log\_file ghi ra file /root/Desktop/log\_file.txt ,

### khi controller bị mất kết nối thì hoạt động như learning switch

### --rate-limit=10000 : tăng giới hạn số lượng packet\_in/1s gửi lên controller lên 10000 thay vì mặc định 1000

### ở đây địa chỉ 127.0.0.1:6633 là địa chỉ localhost vì controller đặt ngay trên máy Centos gắn card. Địa chỉ này có thể khai báo ở đâu tùy vào địa chỉ controller ví dụ như 192.168.1.254:6633 ở trên

### Dừng Openflow Switch

Dừng OF Switch thì chỉ cần kill tất cả tiến trình phần mềm (Driver) là được.

* *killall ofprotocol*
* *killall ofdatapath*

## Công cụ quản lí Switch được cung cấp

Khi cài đặt xong Driver như trong phần 1.2 đã thực hiện thì chúng ta có một công cụ cho việc quản lí OFSwitch trên NetFPGA: “*dpctl”* cho phép thực hiện các việc như : in số flow entry, chi tiết flow entry, thông tin các port …

Bằng việc kết hợp với các vòng lặp while hoặc for ta có thể dễ dàng xem được số flow entry hoặc là in ra các file cần thiết để phân tích sau này.Ví dụ:

* Lệnh sau đây cho phép thực hiện việc in số flow entry ra 1s 1 lần:

*while :; do dpctl dump-tables unix:/var/run/test ; sleep 1; done*

* Lệnh sau đây lại cho phép 1s 1 lần in ra số flow entry vào một file dạng text đưa vào flile tương ứng với số thứ tự lần lấy ở thư mục ~/Desktop/flowcolector

*i=0*

*while [ $i -le 10 ] # while (i<10)*

*do*

*dpctl dump-flows unix:/var/run/test >>~/Desktop/flow\_colector/flowentry/$i.csv*

*sleep*

*let i++*

*echo "Day la flow table thu $i"*

*done*

Dưới đây là một số lệnh đã thử nghiệm :

* *dpctl dump-tables unix:/var/run/test*

### show bang flow table

### stats\_reply (xid=0xe69c3680): flags=none type=3(table)

### 3 tables

### 0: nf2 : wild=0x00018, max= 32792, active=0

### lookup=2058812348, matched=3176889822

### 1: hash2 : wild=0x00000, max=131072, active=0

### lookup=0, matched=0

### 2: linear : wild=0x3fffff, max= 100, active=1

### lookup=0, matched=0

* *dpctl dump-flows unix:/var/run/test*

### show all flowentry

### stats\_reply (xid=0x9670430f): flags=none type=1(flow)

* *dpctl monitor unix:/var/run/test*

### print packets received from unix:/var/run/test

* *dpctl dump-ports unix:/var/run/test 1*

### print port 1 statistics

### stats\_reply (xid=0x81ea3491): flags=none type=4(port)

### 1 ports

### port 1: rx pkts=173, bytes=7439, drop=?, errs=?, frame=?, over=?, crc=?

* *dpctl dump-aggregate unix:/var/run/test*

### print aggregate flow statistics

### stats\_reply (xid=0x92e76c01): flags=none type=2(aggregate)

### packet\_count=0 byte\_count=0 flow\_count=1

* *dpctl show unix:/var/run/test*

### show basic information

### features\_reply (xid=0x80551a30): ver:0x1, dpid:11

### n\_tables:3, n\_buffers:256

### features: capabilities:0xc7, actions:0xeff

### 1(nf2c0): addr:00:4e:46:32:43:00, config: 0, state:0

### current: 1GB-FD

### advertised: COPPER

### supported: 1GB-FD

### 2(nf2c1): addr:00:4e:46:32:43:01, config: 0, state:0

### current: 1GB-FD

### advertised: COPPER

### supported: 1GB-FD

### 3(nf2c2): addr:00:4e:46:32:43:02, config: 0, state:0

### current: 1GB-FD

### advertised: COPPER

### supported: 1GB-FD

### 4(nf2c3): addr:00:4e:46:32:43:03, config: 0, state:0

### current: 1GB-FD

### advertised: COPPER

### supported: 1GB-FD

### LOCAL(tap0): addr:00:00:00:00:00:11, config: 0, state:0

### current: 10MB-FD COPPER

### get\_config\_reply (xid=0x41de9656): miss\_send\_len=128

* *dpctl dump-desc unix:/var/run/test*

### print switch description

### stats\_reply (xid=0x1455aa7e): flags=none type=0(description)

### Manufacturer: Stanford University

### Hardware: Reference Userspace Switch

### Software: 1.0.0

### Comment: security pid=5778

### Serial Num: None

* *dpctl benchmark unix:/var/run/test 1000 100000*

### check bandwidth

### Sends count echo request packets that each consist of an OpenFlow header

### plus n bytes of payload and waits for each response.

### Reports the total time required.

### This is a measure of the maximum bandwidth to vconn for round-trips of n-byte messages.

### Sending 100000 packets \* 1008 bytes (with header) = 100800000 bytes total

### Finished in 1437.6 ms (69561 packets/s) (70117447 bytes/s)

* *dpctl desc unix:/var/run/test STRING*

### set switch description

* *dpctl add-flow unix:/var/run/test in\_port=4,dl\_vlan=\*,dl\_src=\*,dl\_dst=\*,dl\_type=\*,nw\_src=\*,nw\_dst=\*,nw\_proto=\*,tp\_src=\*,tp\_dst=\*,icmp\_type=\*,icmp\_code=\*,actions=output:1*

#### add 1 flow entry vao hardware table trên NetFPGA

## Các lỗi có thể gặp

### deveice id out of bound hoac not found nf2c0

Khi thực hiện việc reset NetFPGA card (*/usr/local/netfpga/lib/scripts/cpci\_reprogram/cpci\_reprogram.pl –all*)

error : deveice id out of bound hoac not found nf2c0

Khắc phục : Nếu NetFPGA card đang cắm vào khe phía trên thì rút NetFPGA card ra cắm vào khe phía dưới. Còn nếu card đã nằm ở khe phía dưới thì rút card ra là được.

### không reset/nạp bitfile được

error : download fail

Khắc phục : tắt máy rồi rút card ra cắm lại

### “no such file or directory”

Khi thực hiện việc chạy Openflow switch thì *luôn luôn* bị có thông báo “no such file or directory” khi chạy lệnh ofdatapath. Điều này sẽ dẫn đến việc OF Switch sẽ không thể hoàn thành việc kết nối đến controller.

Khắc phục : Kill all tiến trình của driver (kill all datapath)

Sau đó chạy lại từ đầu là được

### Việc match gói tin không thực hiện trên phần cứng NetFPGA

Khi thực hiện việc cài Openflow Switch trên netfpga thì đối với phần cứng NetFPGA như đã nói thì chỉ cần nạp bitfile cấu hình là được tuy nhiên phải cần một phần driver trung gian làm nhiệm vụ đóng gói theo giao thức Openflow để giao tiếp với controller và đưa các flowentry vào phần cứng netfpga.

### Loop

Có thể xảy ra 2 trường hợp chính xảy ra loop :

Th1 : đấu dây loop các port khiến cho khi có gói broadcast thì sẽ chạy loop khiến cho NetFPGA bị treo

Th2 : Xảy ra khi đẩy gói với tốc độ quá cao. Trường hợp này hay gặp với việc đẩy gói tcpsyn.

Biểu hiện :

* Bắt trên Wireshark thấy số gói trên các interface tăng lên đến vài trăm nghìn hoặc 1 2 triệu gói
* Sử dụng lệnh Dump table như trình bày ở 1.4 thấy giữa 2 lần sử dụng lệnh dump table số lượng match và lookup tăng lên đột biến đến vài trăm nghìn gói .. trong khi không ping thông

Khắc phục : Reset card và chạy lại từ đầu

### Máy tính không thể kết nối đến NetFPGA Card

Trường hợp này xảy ra phần lớn do việc cổng mạng của máy tính cắm vào NetFPGA card chỉ hỗ trợ tốc độ tối đa là 100Mbps trong khi NetFPGA chỉ hỗ trợ đúng một tốc độ là 1000Mbps nên khi cắm vào 2 đầu không thể thống nhất chế độ hoạt động dẫn đến down. Công cụ để kiểm tra : ethtool

* *ethtool [INTERFACE]*

Công cụ trên cho phép in ra các thông tin cua interface như các tốc dộ hoạt động : 10, 100, 1000 Mbps .Nếu interface chỉ thấy tốc độ tối đa hỗ trợ là 100 Mbps thì sẽ không thể kết nối được đến NetFPGA. Và muốn kết nối đến NetFPGA phải đi qua một Switch có hỗ trợ tốc độ 1Gbps.

# Hướng dẫn cài đặt công cụ cần thiết

## 2.1. Wireshark.

Đây là một trong những công cụ hữu hiệu cho việc bắt và phân tích gói tin . Công cụ này không chỉ phục vụ tốt cho việc nghiên cứu mạng cơ bản mà còn mạng SDN sử dụng giao thức Openflow. Việc cài Wireshark có thể được tự động cài khi cài mininet và khi đó nó được tự động cài thêm các phần cần thiết để đọc được gói tin Openflow. Tuy nhiên đối với máy đã cài Wireshark bằng câu lệnh apt-get install wireshark thì không thể cài được theo cách này nữa và thay vì sẽ đọc được các gói tin Openflow thì nó chỉ đọc được đến TCP. Cách dưới đây cho phép cài bản wireshark cao nhất cho phép đọc được bản tin Openflow đến version1.4

### Hướng dẫn cài Wireshark trên Ubuntu

* Cách 1 : Compile Source code
* Bước 1 : tải Source code : Lên trang chủ của Wireshark
* Giải nén Source code và cài các gói phụ thuộc

sudo apt-get build-dep wireshark

sudo apt-get install qt5-default

sudo apt-get install libssl-dev

sudo apt-get install libgtk-3-dev

* + Bước 3 : vào thư mục chứa source code wireshark và thực hiện việc compile

cd wireshark

./autogen.sh

./compile –with-ssl enable-setcap-install

Câu lệnh compile phía trên chỉ là một trong các tùy chọn compile wireshark cung cấp

* Cách 2 : Add PPA và sử dụng apt-get

sudo add-apt-repository ppa:dreibh/ppa

sudo apt-get update

sudo apt-get install wireshark

### Hướng dẫn cài Wireshark trên Centos

# Tcpreplay

* Tcpreplay (TCP (Transmission Control Protocol) Replay) là một mã nguồn mở và các dự án phần mềm miễn phí được thực hiện trong C và được thiết kế để hoạt động như một công cụ dòng lệnh đó bao gồm một số tiện ích cho hệ điều hành \* NIX, cho phép người sử dụng để kiểm tra một loạt các thiết bị mạng bằng cách sử dụng giao thông bị bắt qua các thư viện libpcap.
* Chương trình được viết bằng ngôn ngữ lập trình C và nó là hoàn toàn tương thích với nhiều bản phân phối GNU / Linux, cũng như các hệ thống UNIX khác giống như Solaris, BSD và Mac OSX.
* Chương trình đi kèm sẵn với một số công cụ CLI, bao gồm tcpprep, tcprewrite, tcpreplay và tcpbridge. Chúng cho phép phân loại lưu lượng mạng như máy chủ và máy khách, phát lại giao thông trở lại vào mạng, viết lại TCP / IP và lớp 2, 3 và 4 tiêu đề, cũng như làm cầu nối hai phân đoạn mạng.

Nói tóm lại Tcpreplay là một trong các công cụ cho phép phát lại một file pcap hoặc pcapng. Các file này được sinh ra bằng các công cụ bắt gói như tcpdump hay wireshark…

Tcpreplay sẽ cố gắng sinh ra các gói tin theo đúng như trong file pcap ghi lại.

Câu lệnh sử dụng :

* *Tcpreplay -i [INTERFACE] [FILE\_PCAP]*

Cách hoạt động chung : chỉ ra interface sẽ đẩy gói qua, chỉ tên file sẽ sử dụng cho việc phát gói tin.

Trên đây mới chỉ là tùy chọn cơ bản, và tcpreplay sẽ cố gắng sinh gói tin theo đúng như thời gian và tốc độ phát gói được ghi lại trong file pcap. Ngoài ra chúng ta có thể có thêm nhiều tùy chọn khác :

*--pps [packet per second]* : muốn phát với bao nhiêu gói trên giây

*--topspeed* : Cho phép đẩy gói với tốc độ tối đa mà không quan tâm đến thời gian giữa các gói

*--loop=10* : phát lại file 10 lần

*--loop=0* : phát lại mãi mãi đến khi ctrl C break ra

*--multiplier=0.5* : cho phép phát lại với tốc độ = ½ tốc độ được ghi lại trong file

Ngoài ra khi thực hiện phát gói sẽ gặp phải trường hợp trong file không chứa địa chỉ mac nguồn và mac đích khiến cho việc phát gói không thể phát theo đúng gói tin được ghi lại (toàn gói Ethernet II). Để thực hiện việc phát gói những file này thì phải thêm mac nguồn mac đích cho file trước. Và bằng cách tương tự chúng ta có thể thay đổi nhiều thông tin khác.

*tcprewrite --dlt=enet --enet-dmac=d0:27:88:ab:7e:0b --enet-smac=14:da:e9:cb:12:02 --infile=testddos7.pcap --outfile=testdd\_fil.pcap --pnat=192.168.1.250:192.168.1.200*

Câu lệnh trên cho phép sửa/ thêm địa chỉ mác nguồn thành 14:da:e9:cb:12:02 và địa chỉ mac đích là d0:27:88:ab:7e:0b , ngoài ra chuyển tất cả địa chỉ 192.168.1.250 thành địa chỉ 192.168.1.200.

* **Hướng dẫn cài đặt trên Ubuntu**
  + Vào terminal cài thư viện cần thiết:

*sudo apt-get install build-essential libpcap-dev*

* Download tại đây: [*https://github.com/appneta/tcpreplay/releases/download/v4.1.1/tcpreplay-4.1.1.tar.gz*](https://github.com/appneta/tcpreplay/releases/download/v4.1.1/tcpreplay-4.1.1.tar.gz)
* Giải nén và cd tới thư mục tcpreplay-4.1.1 vừa tải về rồi chạy lệnh:

*./configure*

*make*

*sudo make install*

* **Hướng dẫn sử dụng**
  + Câu lệnh sửa đổi địa chỉ mac trong file pcap:

*tcprewrite --dlt=enet --enet-dmac=00:9C:02:1D:F8:86 --enet-smac=ff:ff:ff:ff:ff:ff --infile=filterddos.pcap --outfile=filterddos\_mac\_broadcast.pcap ## add mac broadcast*

* Thay đổi địa chỉ IP đích trong file pcap:

*tcprewrite --infile=filterddos\_mac\_broadcast.pcap --outfile=filterddos\_mac\_broadcast\_ip\_250.pcap --pnat=71.126.222.64:192.168.1.250--cachefile=input.cache*

* Phát lưu lượng:

*sudo tcpreplay –i eth0 testddos.pcap*

# Caida data set

<https://www.caida.org/data/passive/ddos-20070804_dataset.xml>

Bộ dữ liệu này chứa traffic traces khoảng hơn một giờ vụ tấn công từ chối dịch vụ DDoS vào ngày 4 tháng 8 năm 2007. Loại tấn công DDoS này nhằm ngăn chặn truy cập vào các máy chủ mục tiêu bằng cách tiêu thụ tài nguyên máy tính trên máy chủ và tiêu thụ tất cả băng thông của mạng kết nối với máy chủ đó.

Tổng tích thước của bộ dữ liệu là 21 GB được ghi lại trong khoảng một giờ (20:50:80 UTC đến 21:56:16 UTC). Cuộc tấn công bắt đầu lúc 21:13, khi đó lưu lượng mạng tăng lên nhanh chóng trong chỉ trong vòng vài phút từ khoảng 200 kbits/s tới khoảng 80 Mbits/s. Traffic trace trong khoảng một giờ được chia thành các file nhỏ trong khoảng 5 phút và được lưu trữ ở định dạng pcap.

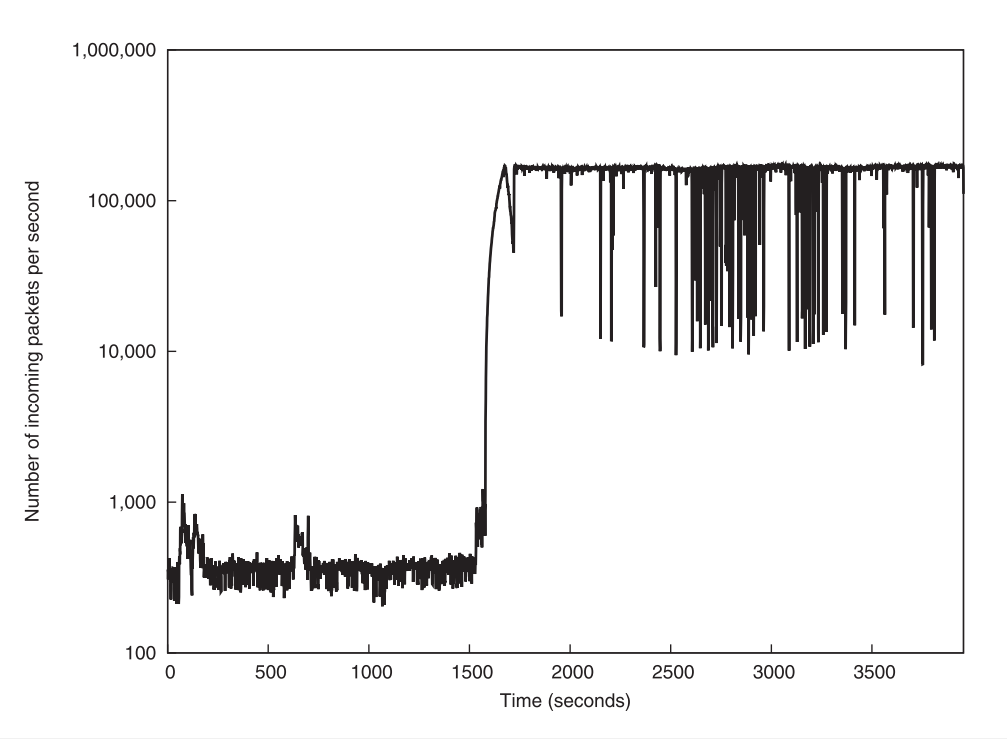
Ở đây chúng ta chú ý tới thư mục to-victim, trong này chỉ chứa chứa file traces của traffic một chiều tấn công tới máy nạn nhân. Còn lại các file trong thư mục from-victim chứa file traces của lưu lượng từ máy nạn nhân responses to the attack thì ta tạm thời chưa quan tâm tới. Một số thông tin của dữ liệu mà ta sử dụng về ip nguồn, tốc độ gói lúc cao điểm… được cho trong bảng 4.4 sau.

Bảng 4.1: Các thông số của file CAIDA dataset DDoS 2007 [15]

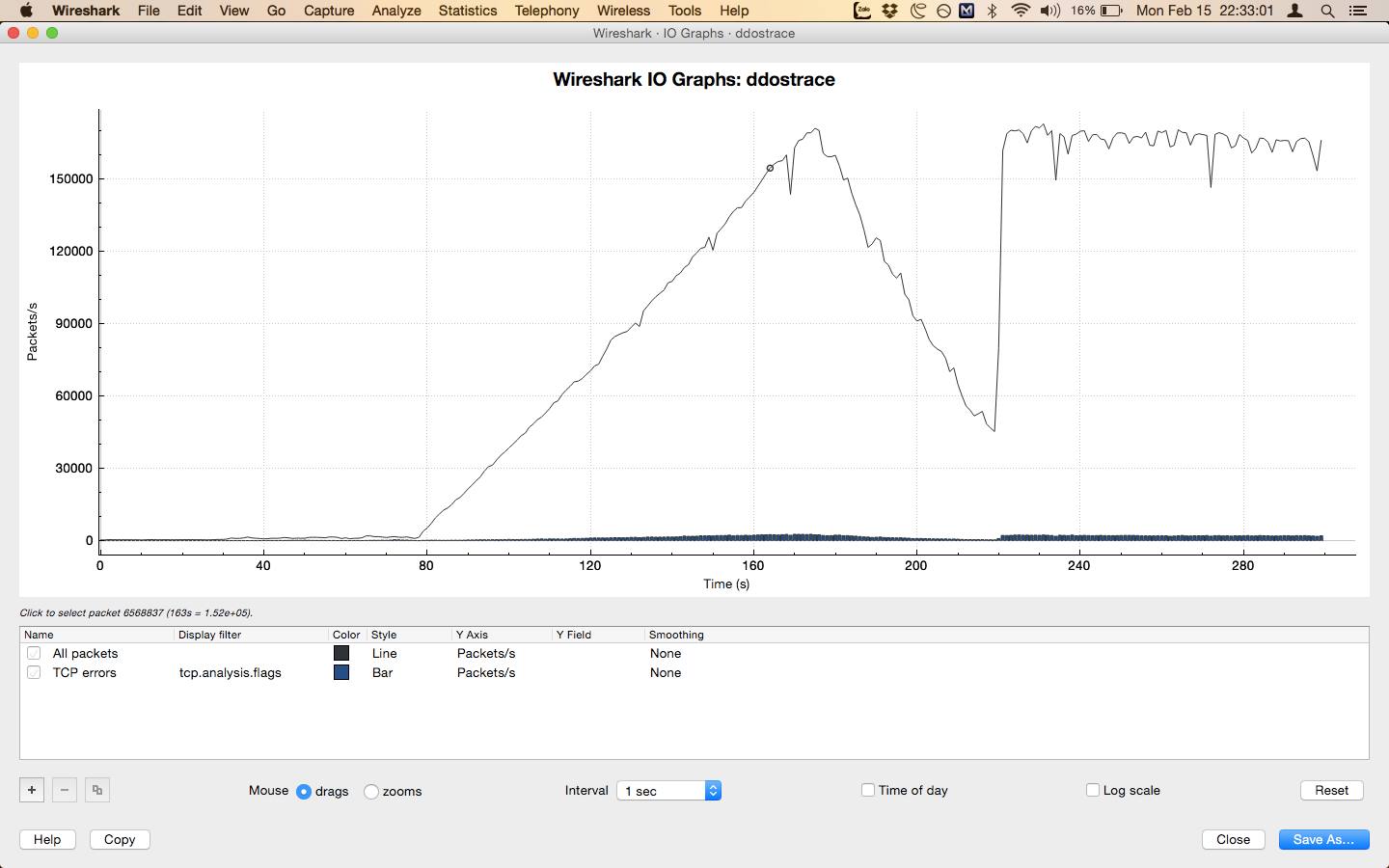
|  |  |
| --- | --- |
| Parameters | CAIDA dataset |
| Thời gian cùa toàn bộ file | 66 min |
| Loại tấn công | Ping Flood |
| Loại gói tin chủ yếu | ICMP |
| Tổng số gói tin | 359,655,765 |
| Tổng số địa chỉ nguồn | 9311 |
| Ngưỡng tốc độ gói tin đến | 175,010 packets per second |

Biểu đồ trong Hình 4.6 thể hiện số lượng gói tin đến trong một giây theo dữ liệu CAIDA. Luu lượng một chiều ở đây cho thấy sự khác biệt rõ rệt, đầu tiên là giai đoạn lưu lượng bình thường, số gói chung bình trong giai đoạn này chỉ khoảng 385 packets trên giây cho tới khoảng giây thứ 1500s thì bắt đầu có tấn công, số lượng gói tin tăng đột biến lên trung bình khoảng 125,705 packets trên giây.

Đợt tấn công DDoS được ghi lại là kiểu tấn công làm cạn kiệt băng thông bằng cách gửi một lượng lớn gói tin ICMP đến mục tiêu và khiến hệ thống dịch vụ của mục tiêu bị hết khả năng về băng thông, bị chậm lại, hệ thống bị treo hay đạt đến trạng thái hoạt động bão hòa.



**Hình 4. 1 : Biểu đồ lưu lượng đến của CAIDA DDoS attack dataset [10]**

1. 

**Hình 4. 2: Lưu lượng CAIDA DDoS dataset 5 phút bắt đầu tấn xuất hiện tấn công [10]**

# Cài công cụ phát gói giả lập tấn công DDoS

## 6.1. bonesi

*git clone* [*https://github.com/Markus-Go/bonesi.git*](https://github.com/Markus-Go/bonesi.git)

*cd bonesi*

*./configure*

*Make*

*Make install*

# Triển khai tesbed

Sơ đồ test bed



Như trên hình có thể thấy muốn kết nối các thành phần cần phải gán địa chỉ IP cho chúng. Việc gán Ip nên để tĩnh để dễ quy hoạch và quản lí. Ở đây , do việc đặt địa chỉ ip cho openflow switch thực chất là gán địa chỉ cho cổng mạng của máy centos gắn card. Máy chạy controller pox như trên hình là một máy khác nằm độc lập với Openflow Switch. Tuy nhiên, chúng ta hoàn toàn có thể chạy controller pox ngay trên máy centos gắn card (địa chỉ controller pox thay vì khai báo 192.168.1.254:6633 thì trở thành 127.0.0.1:6633). Các cổng của NetFPGA không cần đặt IP (mặc dù chúng có thể). Việc đặt IP cho các cổng NetFPGA có thể sẽ khiến mạng không thể thông được.

Việc đặt IP như trên đây chỉ là một cách và không nhất thiết phải theo. Cốt yếu là dải mạng của Switch kết nối lên Controller và dải mạng các thiết bị kết nối đến các cổng của netfpga nên để khác nhau( ở đây là 192.168.1.0/24 và 10.0.0.0/24 )

Sau khi cắm cáp mạng kết nối các thành phần và đặt IP cho chúng theo như sơ đồ trên thì bắt đầu chạy hệ thống

* Kiểm tra kết nối giữa Switch và Controller
  + *Ping 192.168.1.254*
* Chạy Controller
* ./pox.py openflow.spanning\_tree openflow.discovery forwarding.l2\_learning
* Chạy Openflow Switch
  + */usr/local/sbin/cpci\_reprogram.pl -all*

### reset all netfpga card

* + *nf\_download /root/openflow/hw-lib/nf2/openflow\_switch.bit*

### Nạp bitfile

* + *ofdatapath punix:/var/run/test -d 000000000001 -i nf2c0,nf2c1,nf2c2,nf2c3 &*

### dpid : 00:00:00:00:00:00:00:01 .

### --detach : run in background as daemon

* + *ofprotocol unix:/var/run/test tcp:192.168.1.254:6633*

### connect to controller

* Ping kiểm tra xem mạng đã thông hay chưa
  + Từ các máy kết nối với Switch ở dưới ping với nhau để xem xem đã thông nhau hay chưa *ping 10.0.0.3*
  + Trên máy chạy Openflow switch kiểm tra xem có flowentry hay chưa

Sử dụng lệnh dump table như trong mục 1.4

* Bắt đầu phát gói