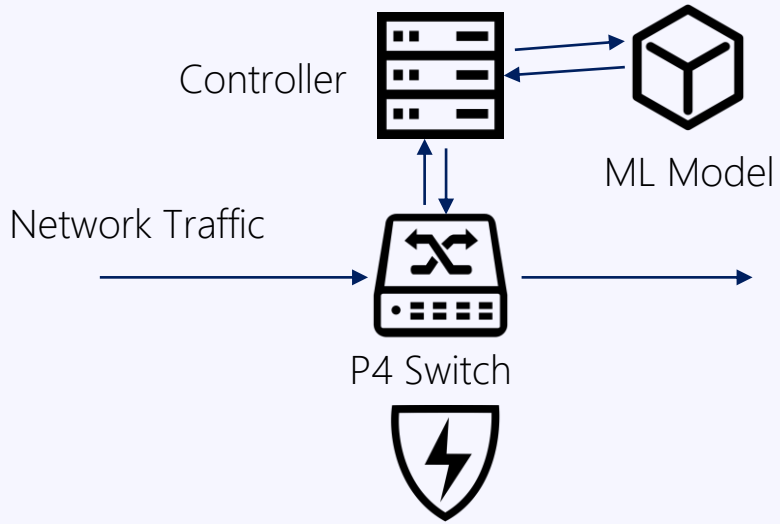
DDoS Defence in P4 Switch using Machine Learning-Based Detection and Mitigation

指導教授：張燕光
專題組員：林禹圻 江少謙

Abstract

為了應對日漸嚴重且多變的 DDoS 攻擊，基於 P4 在 Data plane 的靈活性及 Machine Learning 的準確性，我們設想若在 Switch 就提前進行防禦，可以有效減少被攻擊的影響。因此設計了一套用於 DDoS 防禦的框架，包含偵測與減緩兩個區塊，使用 P4 Switch 負責特徵擷取與減緩機制，及 Controller 負責利用機器學習模型進行攻擊偵測。



Content

1 Introduction

2 Implement

3 Experiment

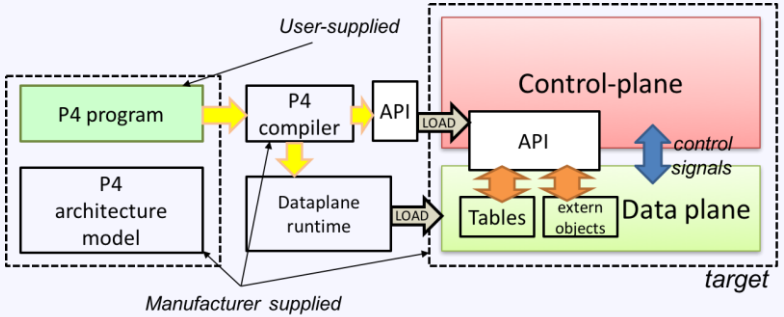
01

Introduction

P4

P4 是一種程式語言，用於控制網絡設備（如 Switch 和 Router）中的 Data Plane (快速)，是一種特定於網路領域的語言，具有許多針對網絡數據轉發進行最佳化的結構。有以下三種特性

- Target independence
- Protocol independence
- Reconfigurability

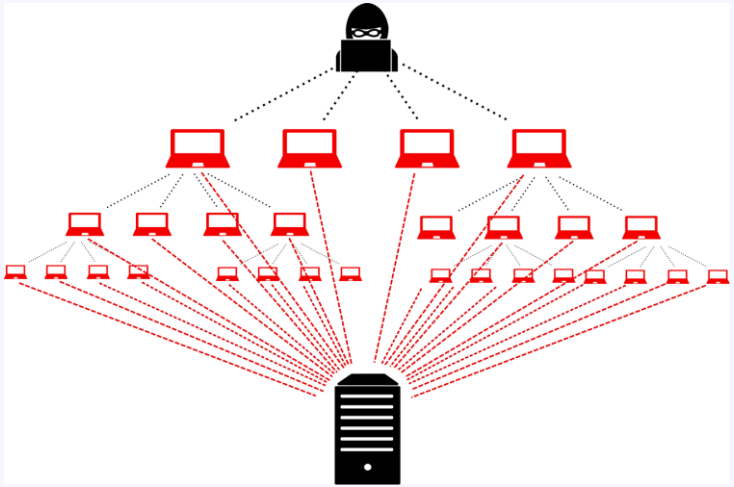


Machine learning

Machine learning (ML) 是人工智慧中以「學習」為重點的分支，其應用領域非常廣，我們在專題中使用到 ML 的監督學習，監督學習從給定的 DDoS 攻擊資料集中學習出一個模型，當新的資料到來時，可以根據這個模型預測攻擊種類。

DDoS

Distributed Denial of Service (DDoS) 攻擊，是透過惡意且分散的流量去淹沒網站或是主機等等，導致設備因為資源耗盡而無法正常使用。隨著網際網路的發達 DDoS 攻擊手法也變得越來越多元且難以防範 ，讓許多廠商與企業蒙上巨大的損失。



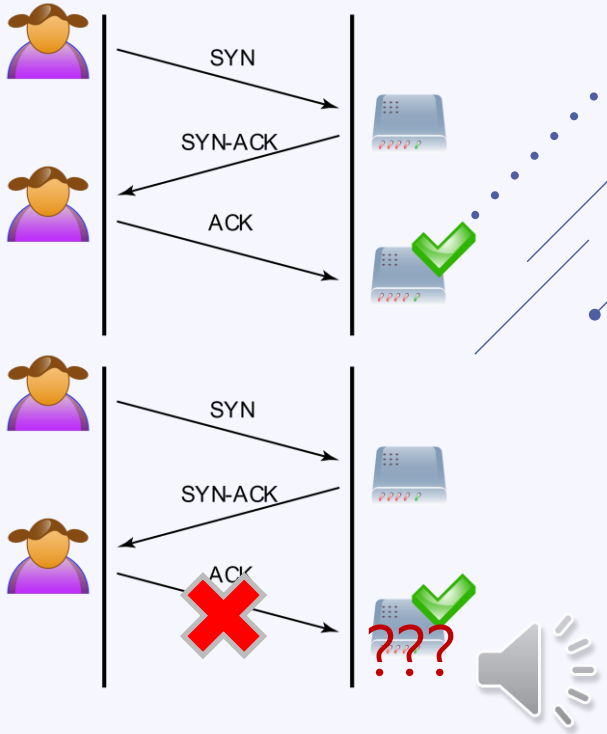
Traffic Type

1. SYN flood：

上圖是正常TCP 的三次握手。

下圖則是當 SYN flood 攻擊發生時，使用者會傳送大量的 SYN 給伺服器，而當伺服器回答 SYN ACK 後，並不會做任何回應，這樣會導致伺服器大量的空間以及時間都會被占用。

。



Traffic Type

2. UDP flood：

伺服器接收到 UDP 封包後，會交給負責的應用程式，接著回應處理狀況。而當應用程式收到過多的 UDP 封包，會導致服務無法正常提供。

3. ICMP flood：

在 ICMP flood 時攻擊者會傳送大量的 ICMP reply ，導致目標為了處理這些封包而受到影響，且大量的 ICMP reply 和 ICMP echo Request 也會導致 link 的 bandwidth 被塞滿。

Traffic Type

4. TCP flag flood：

通過連續向目標發送全部flag都打開的 TCP 封包，當系統遇到這類在正常 TCP 傳輸中不會見到的 flag 設定，若 OS 設定不良，可能會有未知的重大影響。這種攻擊因為全部flag都打開可以做為其他種攻擊的掩護，干擾防火牆的判斷。

在此次專題，我們將在正常 TCP 傳輸中不會見到的 flag 設定統稱為 Fault flags，並用其去判斷是否遭受 TCP flag flood 攻擊。

5. Benign：

未遭受攻擊的合法流量，研究指出正常流量中 TCP : UDP 大約為 1:3。

02

Implement

Feature

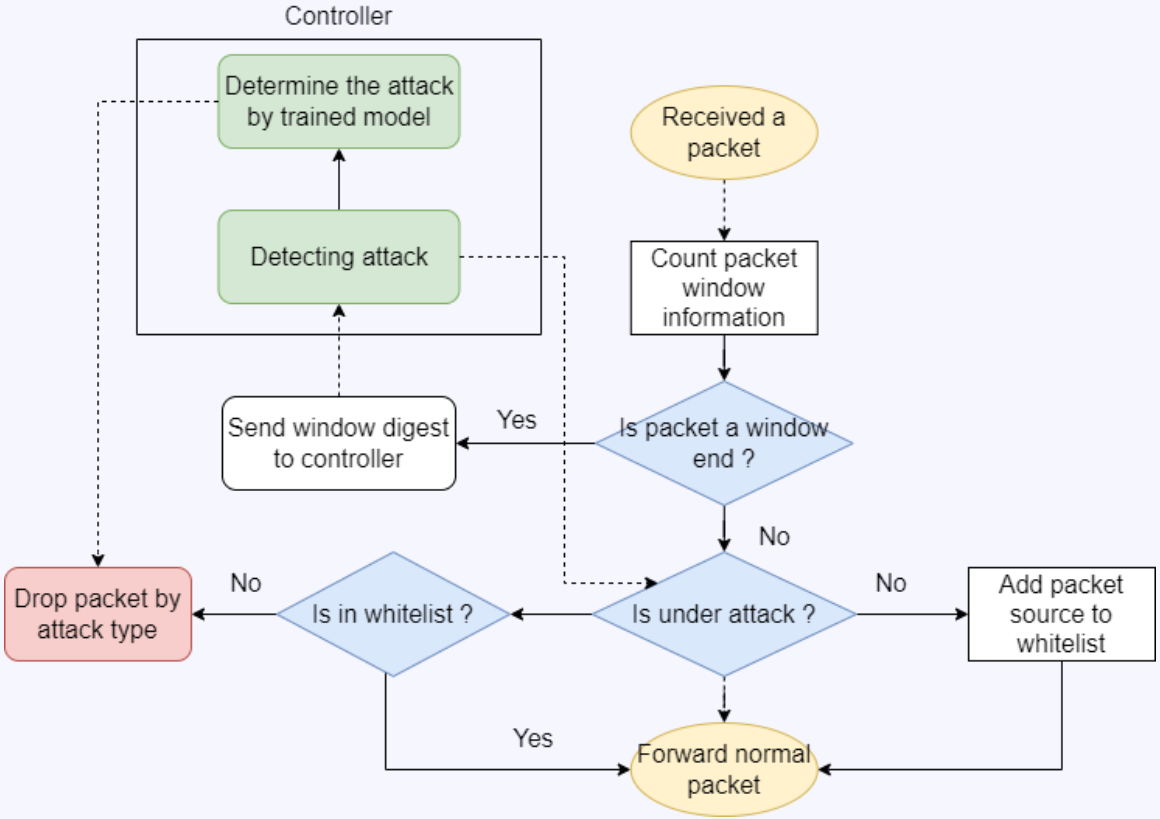
訓練資料集方面，使用 CIC 2019 DDoS 的 pcap 檔做為我們的攻擊資料集，而正常資料集則是自行透過 Wireshark 收集 pcap。以0.1秒為單位，將時間內的所有封包整理成1組特徵，特徵如下圖。

Feature	Description
Packet size	平均封包大小(bytes)
Packet interval	平均兩封包之間的間隔時間(s)
TCP ratio	TCP封包佔所有封包比例
UDP ratio	UDP封包佔所有封包比例
ICMP ratio	ICMP封包佔所有封包比例
Other Protocol ratio	其他協定之封包佔所有封包比例
SYN/TCP	所有TCP封包內只有SYN flag的封包比例
FaultFlags/TCP	所有TCP封包內Fault flag的封包比例

Model

ML 模型方面，因特徵足夠具代表性，沒有使用深層模型的必要 (ex RNN)，而淺層模型，考慮到要配合 P4 即時處理的特性，無法做正規化，不考慮 KNN、SVM 等與距離相關的模型，所以選擇基於 Decision Tree 的 RF (Random Decision Forests)模型。

Architecture and Workflow



P4Runtime

P4Runtime 是基於 Protobuf 以及 gRPC 框架上的協定，用於 Controller 與 P4 Switch 之間的溝通。在這裡我們使用 P4Runtime 搭配 P4 的 digest 將 P4 Switch 統計到的數據傳送給 Controller ，以及 Controller 將模型判斷結果傳回給 P4 Switch 。

```
struct digest_t {
    bit<32> packet_count;
    bit<32> packet_size;
    bit<32> icmp_count;
    bit<32> tcp_count;
    bit<32> udp_count;
    bit<32> other_count;
    bit<32> syn_count;
    bit<32> flags_count;
    time_t interval;
}

digest_t info;
digest<digest_t>(0, info);
```

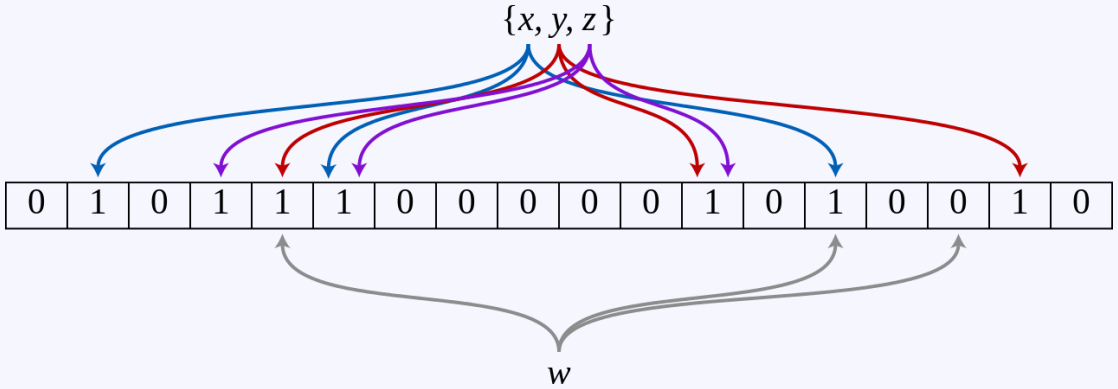
P4 Switch

```
def ReadDigest(self, digest_id=None, list_id = None, dry_run=False):
    if digest_id and list_id:
        request = p4runtime_pb2.StreamMessageRequest()
        request.digest_ack.digest_id = digest_id
        request.digest_ack.list_id = list_id
        self.requests_stream.put(request)
    if dry_run:
        print("P4Runtime Read:", request)
    else:
        for response in self.stream_msg_resp:
            if response.WhichOneof('update') == 'digest':
                yield response.digest
```

Controller

Whitelist

使用 Whitelist 紀錄未被攻擊時正常流量的 IP ，在遭受攻擊時根據 Whitelist 讓封包通過，保證原本的連線不受攻擊影響。
Whitelist 使用 Bloom Filter 實作，根據封包的資訊計算多組位址將其寫入或從 Bloom Filter 讀出。



Mitigation Mechanism

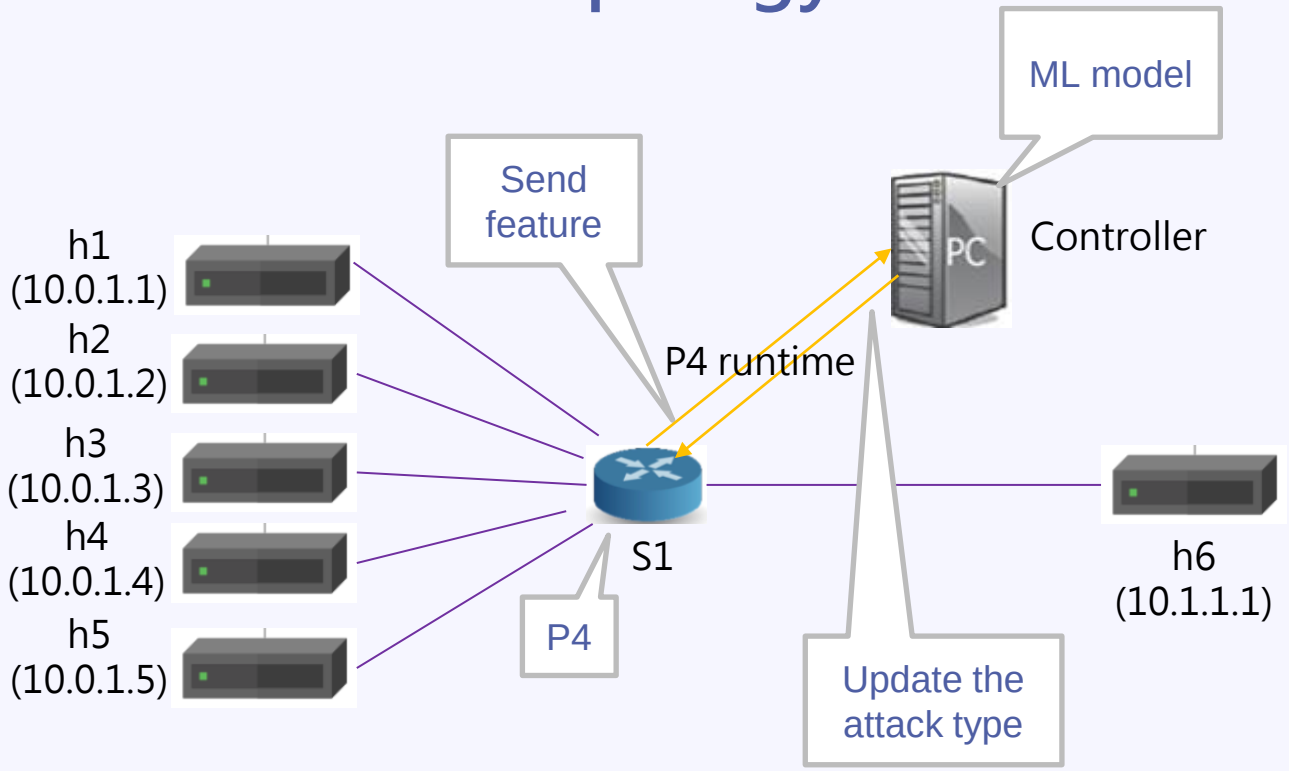
P4 Switch 根據 Controller 傳來的結果，針對不同種類的攻擊使用不同的減緩機制，以減少正常封包誤判的情況。

- SYN Flood : Drop 掉帶有 SYN Flag 的封包。
- TCP Flags : Drop 掉 TCP Flags 異常的封包。
- ICMP Flood : Drop 掉 ICMP Protocol 的封包。
- UDP Flood : Drop 掉 UDP Protocol 且數量大於 Threshold 的封包。

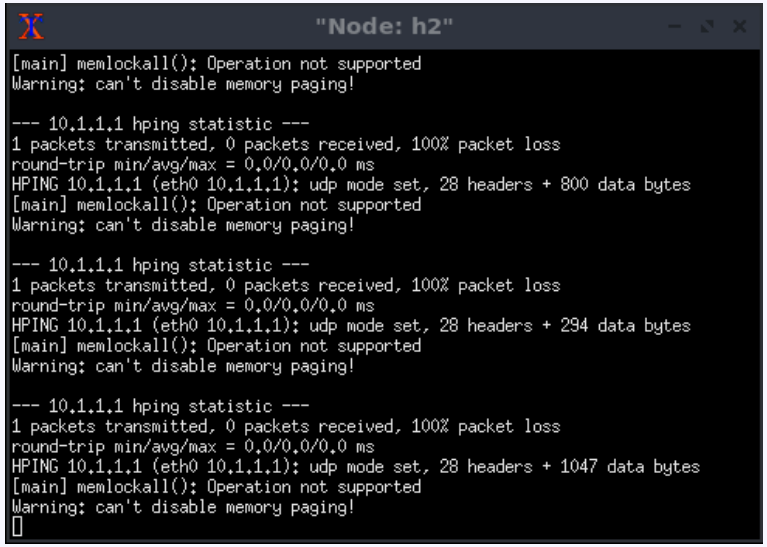
03

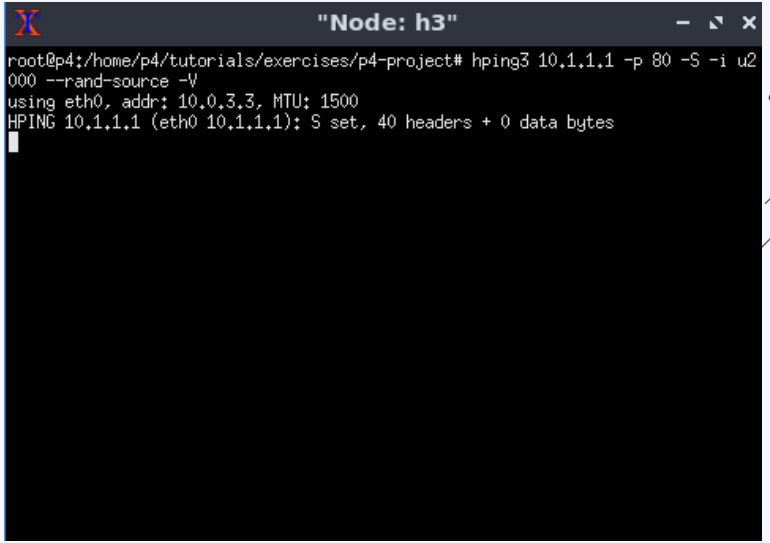
Experiment

Topology

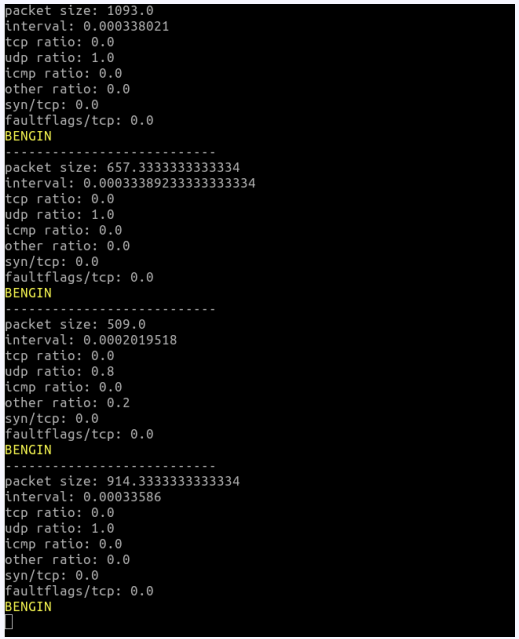

ML model
Send feature
PC
Controller
h1 (10.0.1.1)
h2 (10.0.1.2)
h3 (10.0.1.3)
h4 (10.0.1.4)
h5 (10.0.1.5)
P4 runtime
S1
P4
h6 (10.1.1.1)
Update the attack type

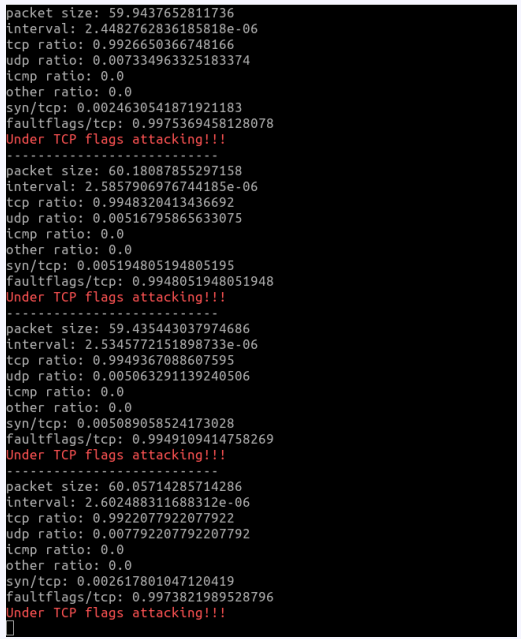
Hosts

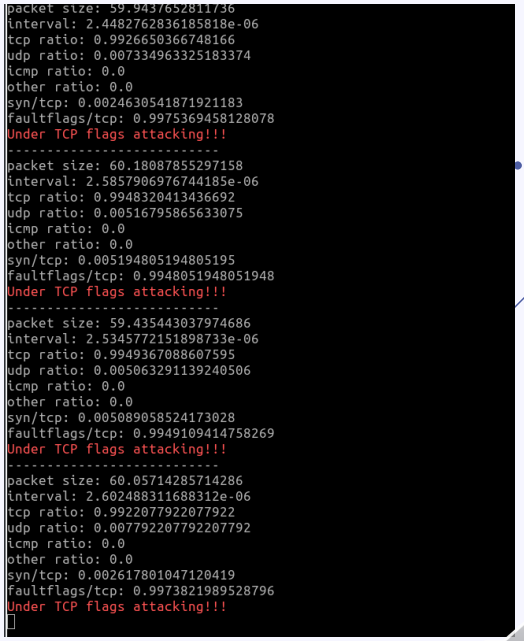

"Node: h2"
[main] memlockall(): Operation not supported
Warning: can't disable memory paging!
--- 10.1.1.1 hping statistic ---
1 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
HPING 10.1.1.1 (eth0 10.1.1.1): udp mode set, 28 headers + 800 data bytes
[main] memlockall(): Operation not supported
Warning: can't disable memory paging!
--- 10.1.1.1 hping statistic ---
1 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
HPING 10.1.1.1 (eth0 10.1.1.1): udp mode set, 28 headers + 294 data bytes
[main] memlockall(): Operation not supported
Warning: can't disable memory paging!
--- 10.1.1.1 hping statistic ---
1 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
HPING 10.1.1.1 (eth0 10.1.1.1): udp mode set, 28 headers + 1047 data bytes
[main] memlockall(): Operation not supported
Warning: can't disable memory paging!


"Node: h3"
root@p4:/home/p4/tutorials/exercises/p4-project# hping3 10.1.1.1 -p 80 -S -i u2
000 --rand-source -V
using eth0, addr: 10.0.3.3, MTU: 1500
HPING 10.1.1.1 (eth0 10.1.1.1): S set, 40 headers + 0 data bytes

Controller


packet size: 1093.0
interval: 0.000338021
tcp ratio: 0.0
udp ratio: 1.0
icmp ratio: 0.0
other ratio: 0.0
syn/tcp: 0.0
faultflags/tcp: 0.0
BENGIN
---
packet size: 657.3333333333334
interval: 0.00033389233333333334
tcp ratio: 0.0
udp ratio: 1.0
icmp ratio: 0.0
other ratio: 0.0
syn/tcp: 0.0
faultflags/tcp: 0.0
BENGIN
---
packet size: 509.0
interval: 0.0002019518
tcp ratio: 0.0
udp ratio: 0.8
icmp ratio: 0.0
other ratio: 0.2
syn/tcp: 0.0
faultflags/tcp: 0.0
BENGIN
---
packet size: 914.3333333333334
interval: 0.00033586
tcp ratio: 0.0
udp ratio: 1.0
icmp ratio: 0.0
other ratio: 0.0
syn/tcp: 0.0
faultflags/tcp: 0.0
BENGIN


packet size: 59.9437652811736
interval: 2.4482762836185818e-06
tcp ratio: 0.9926650366748166
udp ratio: 0.007334963325183374
icmp ratio: 0.0
other ratio: 0.0
syn/tcp: 0.0024630541871921183
faultflags/tcp: 0.9975369458128078
Under TCP flags attacking!!!
---
packet size: 60.18087855297158
interval: 2.5857906976744185e-06
tcp ratio: 0.9948320413436692
udp ratio: 0.00516795865633075
icmp ratio: 0.0
other ratio: 0.0
syn/tcp: 0.005194805194805195
faultflags/tcp: 0.9948051948051948
Under TCP flags attacking!!!
---
packet size: 59.435443037974686
interval: 2.5345772151898733e-06
tcp ratio: 0.9949367088607595
udp ratio: 0.005063291139240506
icmp ratio: 0.0
other ratio: 0.0
syn/tcp: 0.005089058524173028
faultflags/tcp: 0.9949109414758269
Under TCP flags attacking!!!
---
packet size: 60.05714285714286
interval: 2.602488311688312e-06
tcp ratio: 0.9922077922077922
udp ratio: 0.007792207792207792
icmp ratio: 0.0
other ratio: 0.0
syn/tcp: 0.002617801047120419
faultflags/tcp: 0.9973821989528796
Under TCP flags attacking!!!


packet size: 59.9437652811736
interval: 2.4482762836185818e-06
tcp ratio: 0.9926650366748166
udp ratio: 0.007334963325183374
icmp ratio: 0.0
other ratio: 0.0
syn/tcp: 0.0024630541871921183
faultflags/tcp: 0.9975369458128078
Under TCP flags attacking!!!
---
packet size: 60.18087855297158
interval: 2.5857906976744185e-06
tcp ratio: 0.9948320413436692
udp ratio: 0.00516795865633075
icmp ratio: 0.0
other ratio: 0.0
syn/tcp: 0.005194805194805195
faultflags/tcp: 0.9948051948051948
Under TCP flags attacking!!!
---
packet size: 59.435443037974686
interval: 2.5345772151898733e-06
tcp ratio: 0.9949367088607595
udp ratio: 0.005063291139240506
icmp ratio: 0.0
other ratio: 0.0
syn/tcp: 0.005089058524173028
faultflags/tcp: 0.9949109414758269
Under TCP flags attacking!!!
---
packet size: 60.05714285714286
interval: 2.602488311688312e-06
tcp ratio: 0.9922077922077922
udp ratio: 0.007792207792207792
icmp ratio: 0.0
other ratio: 0.0
syn/tcp: 0.002617801047120419
faultflags/tcp: 0.9973821989528796
Under TCP flags attacking!!!

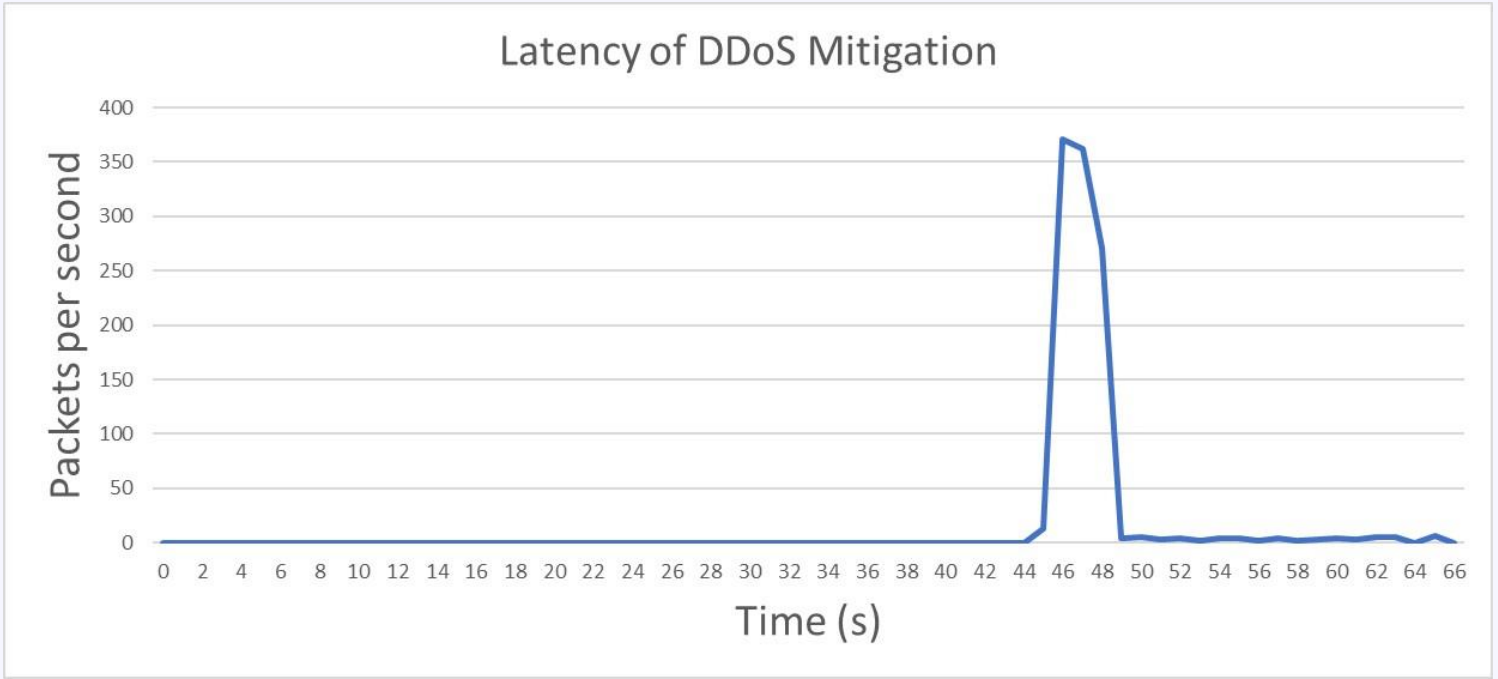
Experiment Setup

使用 P4 程式並利用 BMv2 模擬，使用hping3去模擬攻擊。

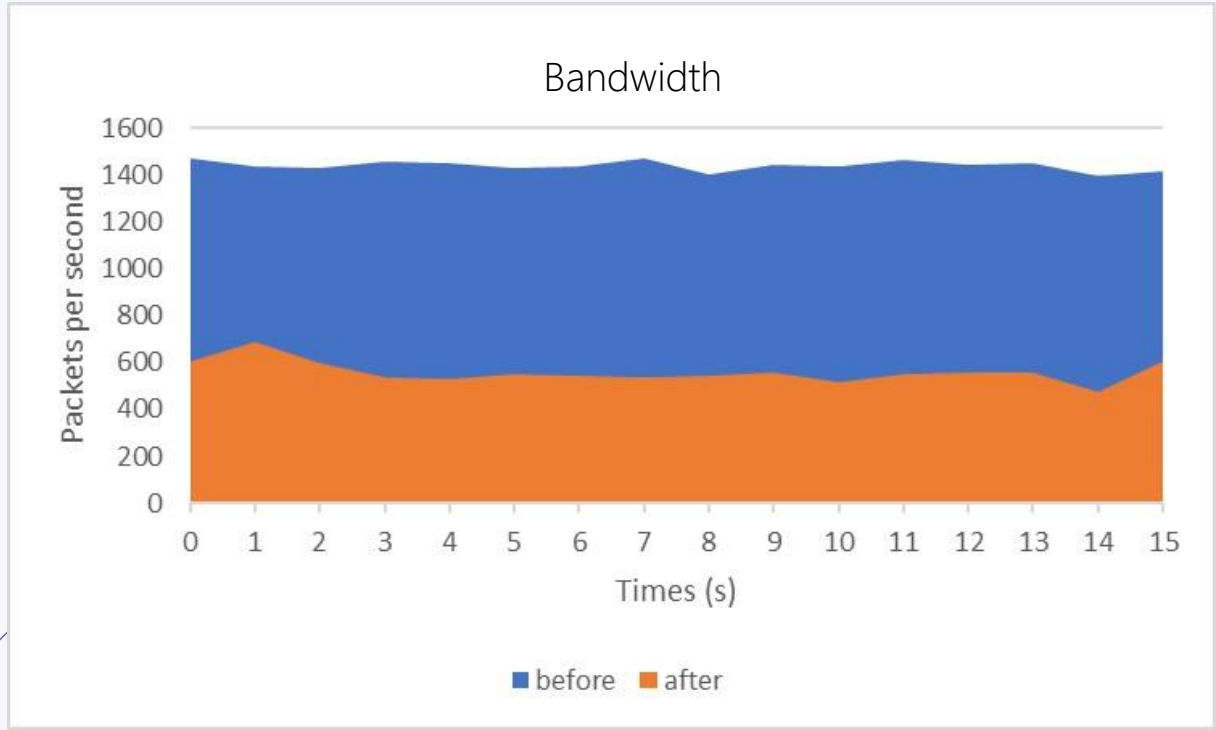
環境設置 ：

1. 系統剛開始配置正常流量TCP : UDP = 1 : 3。
2. 利用hping3快速的以random source的方式傳送封包實現DDoS。
3. 攻擊時會持續嘗試新連線 TCP : UDP = 1 : 3，新連線的流量的比例約佔原本的正常流量的3成。

Result and Discussion



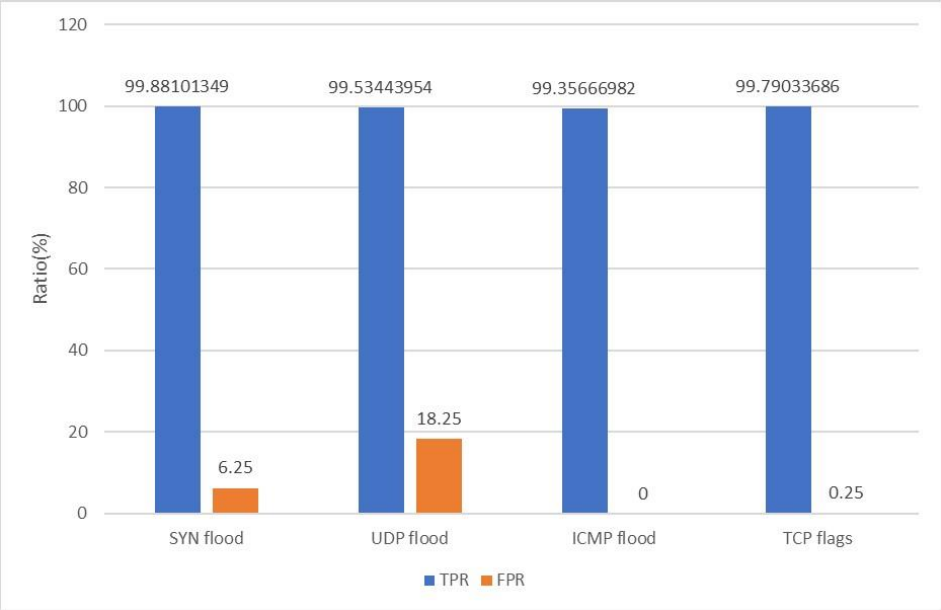
Result and Discussion



Result and Discussion

透過下兩個指標，紀錄當遭受不同攻擊時的結果

- True Positives Ratio (TPR) ：Drop掉的攻擊封包 / 總共傳送的攻擊封包
- False Positives Ratio (FPR) ：Drop掉的正常封包 / 總共傳送的正常封包



Improvement

實驗結果顯示，針對不同的攻擊可以有效的阻擋及讓正常的流量通過，所以我們認為改進可以從效能方面下手。有以下兩點

- Throughput : 若減少 Bandwidth 的損失可以有更好的 Throughput ，讓我們設計的框架在現實環境更適用。
- Latency : 若是減少延遲，可以提升遇到攻擊時的反應速度。

Thanks for listening