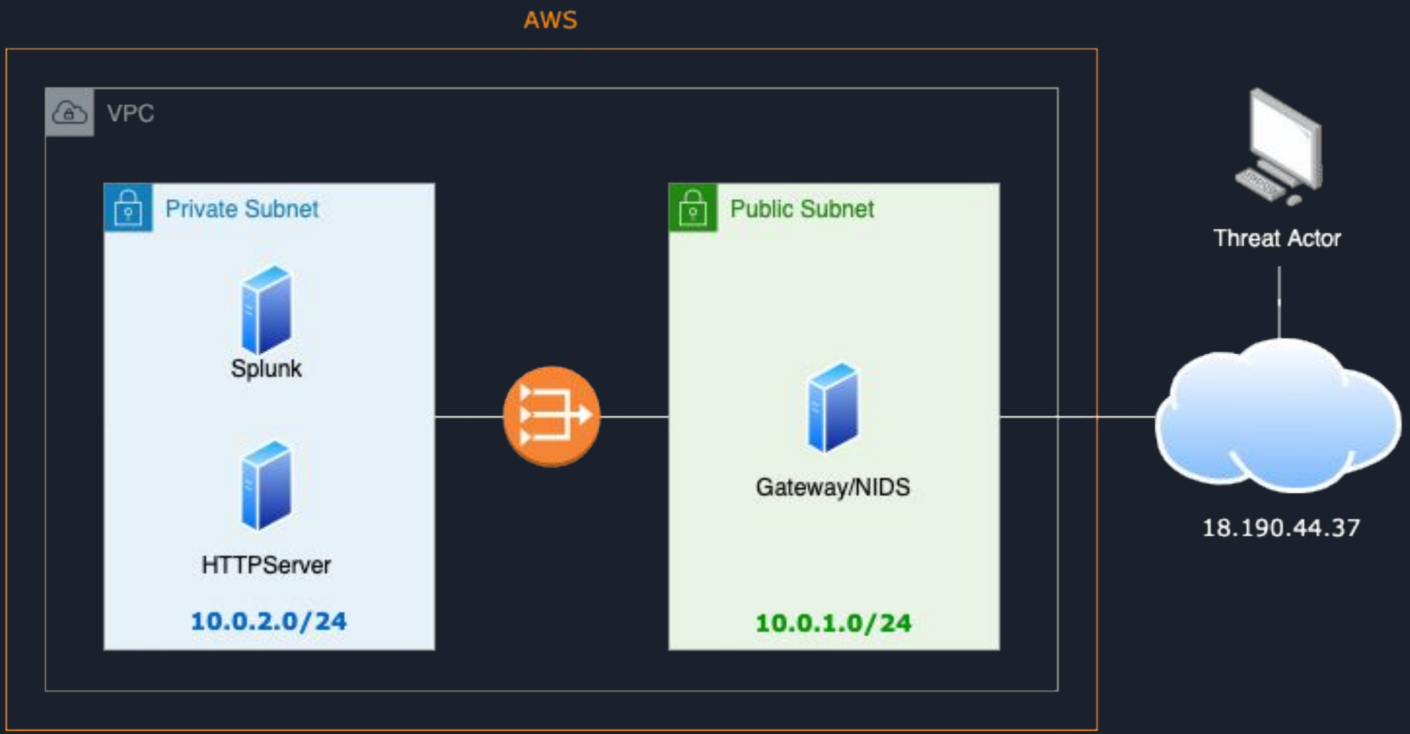
Cloud Network Intrusion Detection Simulator Presentation

CPSC 454
Fall 2021

Chris
Yu
Gage
Roberto Guerra
Louis

AWS Environment



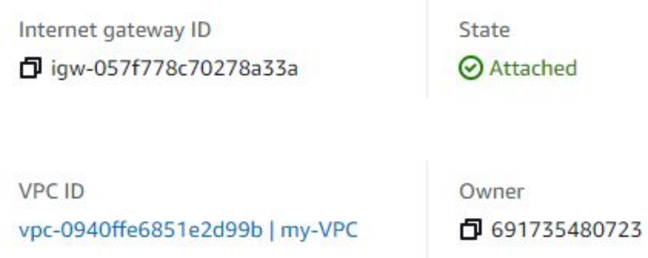
AWS VPC Setup

On AWS (Amazon Web Service) we selected the option to create a new VPC (Virtual Private Cloud)

We then assigned the new VPC a non-routable IP range e.g. 10.0.0.0/16

Next we created a new internet gateway and attached it to the VPC

The internet gateway allows for communication between the VPC and the rest of the internet



AWS VPC Setup - Public and Private Subnets

We created public and private subnets within the VPC

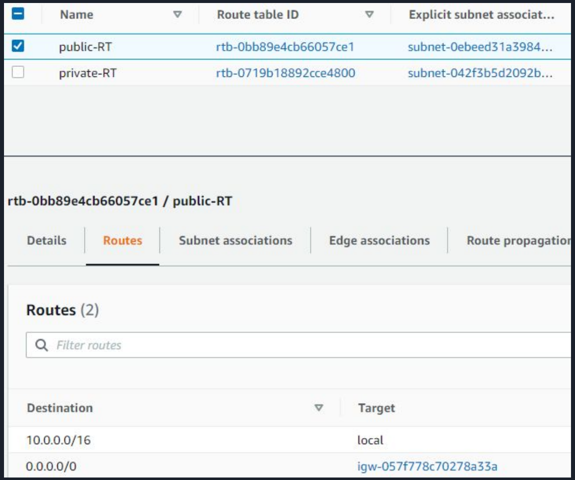
Then we assigned different IP ranges to each subnet, 10.0.1.0/24 for the public subnet and 10.0.2.0/24 for the private subnet

The public subnet connects to the internet gateway while the private subnet does not

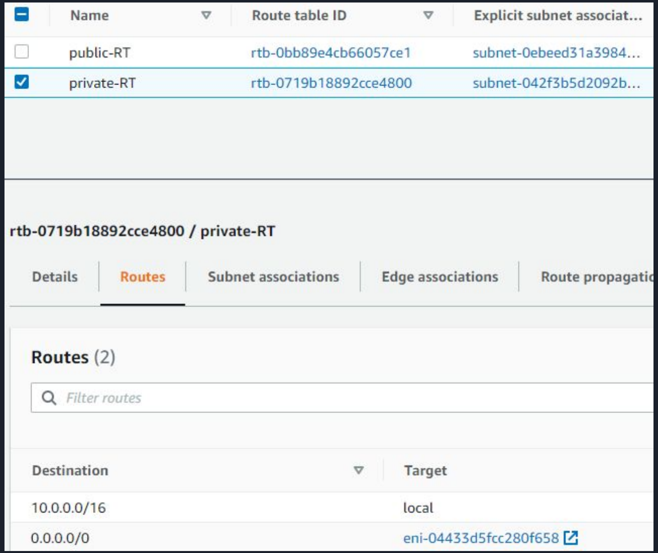
Name	Subnet ID	State	VPC	IPv4 CIDR
my-Private1	subnet-042f3b5d2092bf53c	Available	vpc-0940ffe6851e2d99b | my-...	10.0.2.0/24
my-Public1	subnet-0ebeed31a3984580c	Available	vpc-0940ffe6851e2d99b | my-...	10.0.1.0/24

AWS VPC Setup - Route Tables

We created a route table that has the internet gateway as a target and 0.0.0.0/0 as a destination, then associated this route table with the public subnet

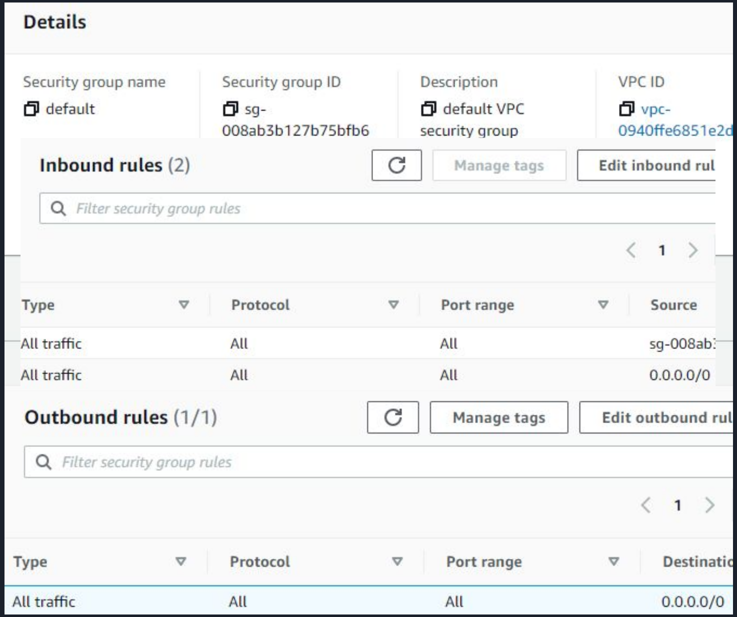


We also created another route table without the igw (Internet Gateway) as a target and associated this table with the private subnet

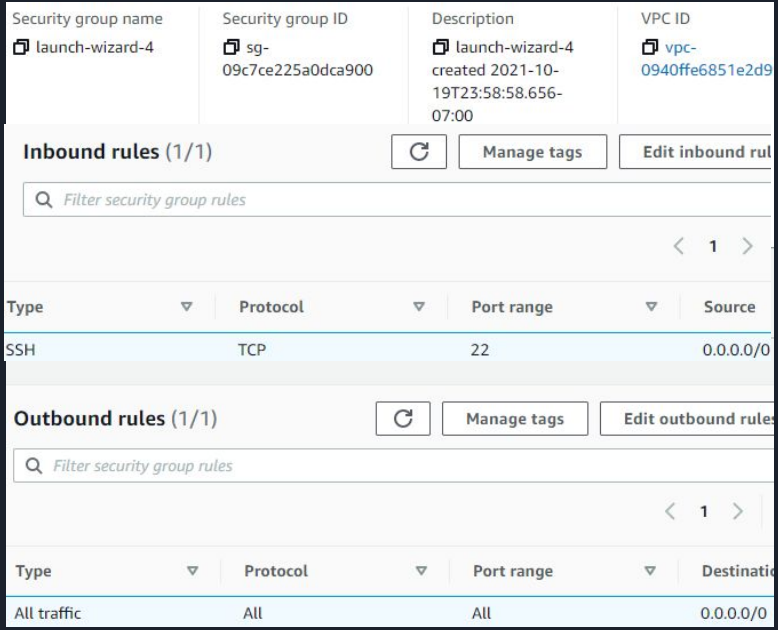


AWS VPC Setup - Security Groups

We specified allowed inbound and outbound traffic to and from the public subnet instance

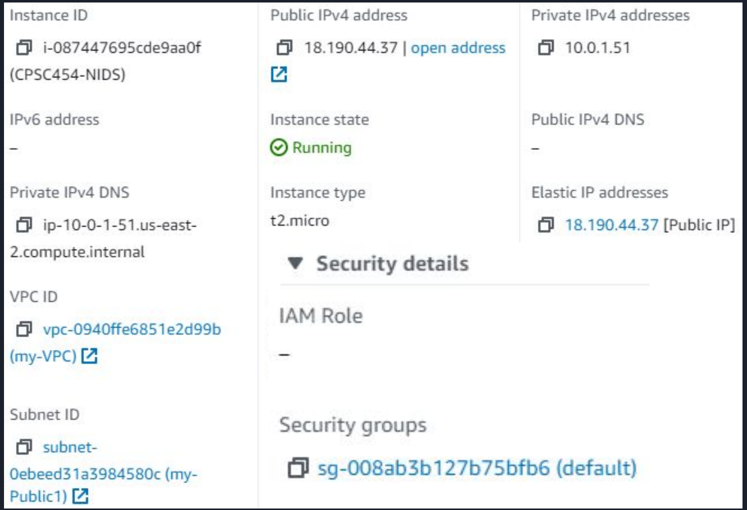


We also specified allowed inbound and outbound traffic to and from private subnet instance

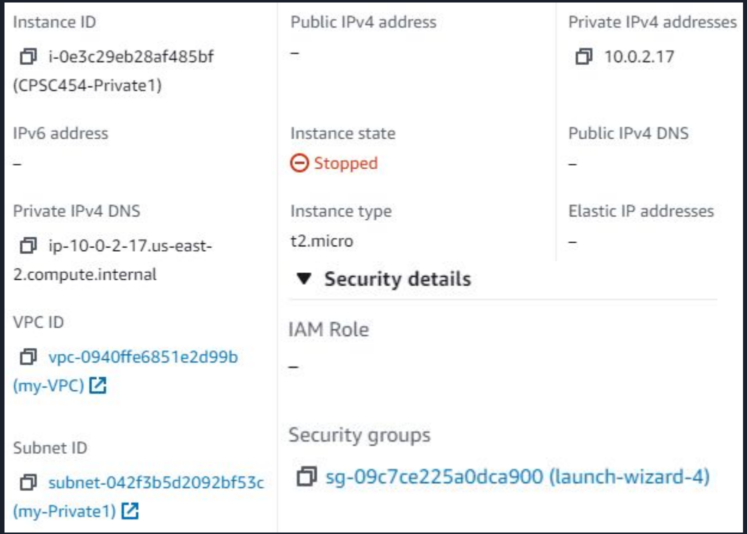


AWS VPC Setup - EC2 Instance

We created an EC2 instance for the public subnet



We also created an EC2 instance for the private subnet



AWS VPC Setup - NIDS (Snort)

In the ECS2 instance for the public subnet we made a temp directory

```
mkdir ~/snort_src && cd ~/snort_src
```

Then we downloaded and installed the Data Acquisition library (DAQ) used by Snort

Finally, we downloaded and installed Snort

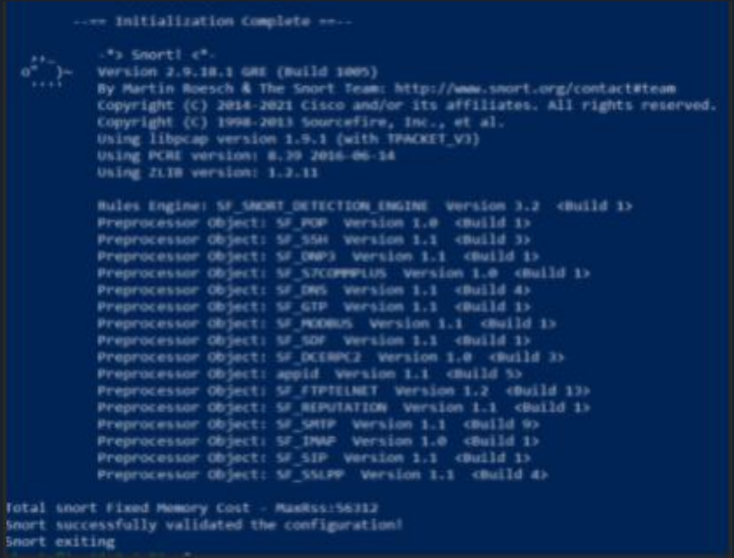
```
wget https://www.snort.org/downloads/snort/snort-2.9.16.tar.gz
```

```
./configure --enable-sourcefire && make && sudo make install
```

Snort Configuration

We modified the snort.conf file with: **sudo snort -T -c /etc/snort/snort.conf**

We then validated by testing our configuration as shown in Fig 5: **sudo snort -T -c /etc/snort/snort.conf**



Snort Rules

```
alert tcp any any -> any 22 (msg:"SSH detected"; threshold: type both, track by_src, count 5, seconds 30; sid:001; rev:001;)
alert tcp any any -> any 23 (msg:"Possible NMAP scan detected";threshold: type both, track by_src, count 1, seconds 30; sid:002; rev:001;)
alert tcp any any -> any 22 (msg:"SSH Brute Force Detected";  threshold: type threshold, track by_src, count 30, seconds 5; sid:003; rev:001;)
alert tcp any any -> any 8000 (msg:"Splunk Access"; threshold: type both, track by_src, count 1, seconds 60; sid:004; rev:001;)
alert icmp any any -> any 8080 (msg:"ICMP flood"; sid:005; rev:1; classtype:icmp-event; threshold: type both, track by_dst, count 500, seconds 3;)
alert tcp any any -> any 8080 (msg:"Possible DoS Attack Type: SYN flood"; flow:stateless; sid:006; threshold: type both, track by_dst, count 20, seconds 10;)
```

Snort utilizes a series of rules that help define malicious network activity and uses those rules to find packets that match against them and generates alerts for users.

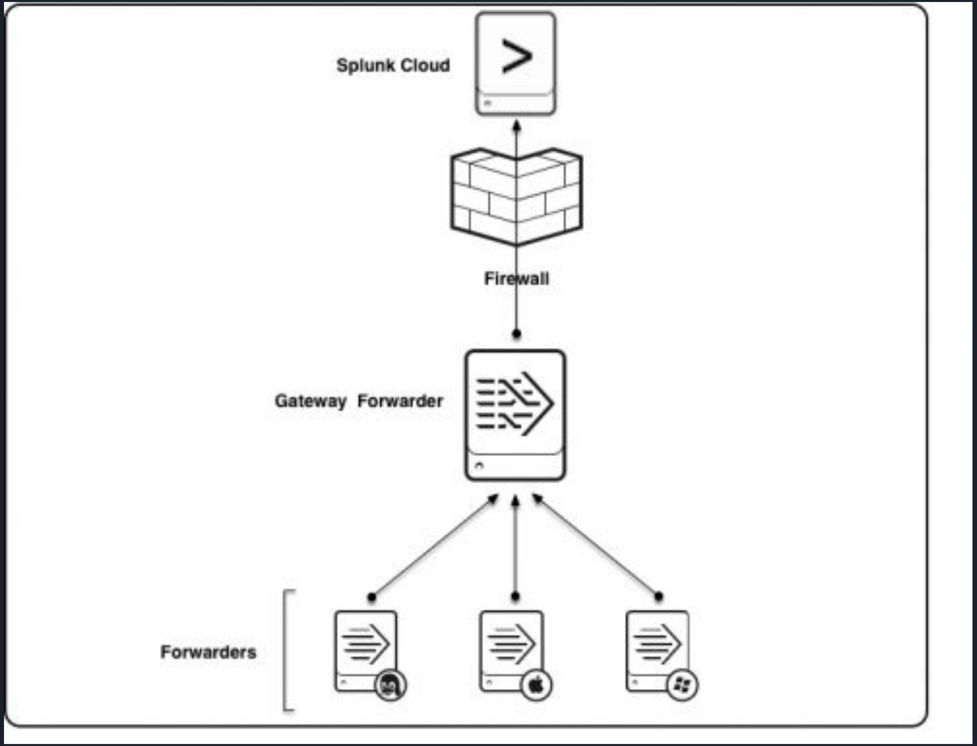
<action> <protocol> <source ip> <source port> -> <destination ip> <destination port> (rule options)

Splunk (SIEM)

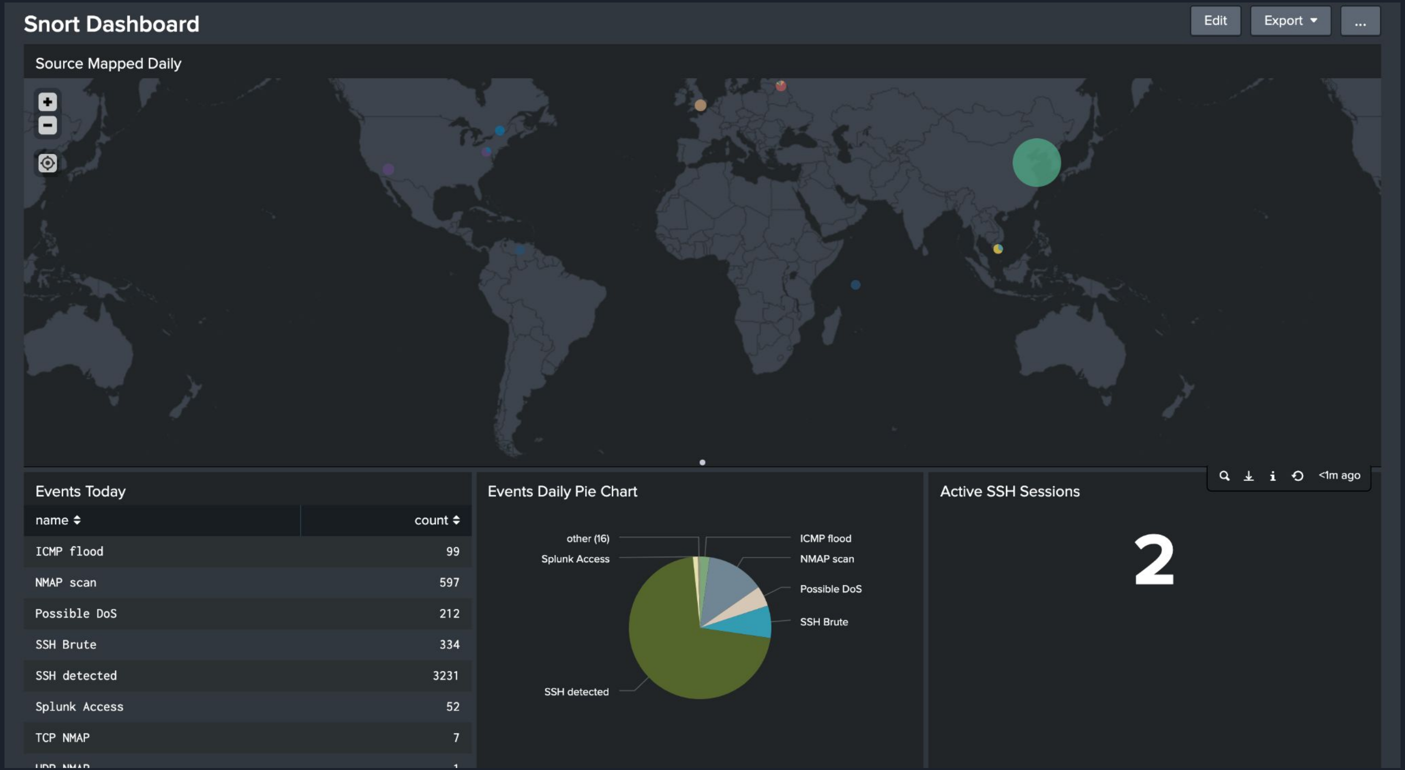
We have included a Security Information and Event Management system, called *Splunk*, to analyze traffic in and out of the cloud. Splunk is also able to capture, index, and correlate real time data into a searchable container.

Splunk (cont.)

We have also implemented a Splunk Forwarder, which can forward Snort log files to a port of our Splunk instance and create dashboards on Splunk.



SIEM Dashboard



SIEM Dashboard (cont.)

Time	Event
12/6/21 11:31:49.862 PM	[**] [1:1:1] SSH detected [**] [Priority: 0] 12/06-23:31:49.862105 218.92.0.204:29607 -> 10.0.1.51:22 TCP TTL:30 TOS:0x0 ID:18617 IpLen:20 DgmLen:116 DF ***AP*** Seq: 0xAC329FF2 Ack: 0xF6B87E1C Win: 0x106 TcpLen: 32 Show all 6 lines host = ip-10-0-1-51	source = /var/log/snort/alert.full	sourcetype = snort_alert_full

indexed snort logs on Splunk

Search String

```
index="snort" source="/var/log/snort/alert.full" "SSH detected" | dedup src_ip |
    stats count by src_ip | eventstats sum(count) as total | table total
```

creating unique searches

Active SSH Sessions

2

dashboard display of SSH sessions

Intrusion Detection Simulations

Our project includes scripts that contains various network tools to detect and prevent the following attacks:

- SSH Brute Force attacks (Hydra)
- DDoS attacks (hping3)
  - ICMP Flooding
  - SYN Flooding
- Nmap Scan Detection
  - TCP Scan
  - UDP Scan

```python
import time, subprocess

while True:

    #TCP NMAP scan
    a = subprocess.Popen('nmap -sC -sV -Pn 18.190.44.37', shell=True)
    time.sleep(10)

    #UDP NMAP scan
    b = subprocess.Popen('nmap -sU 18.190.44.37', shell=True)
    time.sleep(10)

    #SSH Brute Force
    c = subprocess.Popen('hydra -l user -P /usr/share/wordlists/dirb/others/best110.txt ssh://18.190.44.37', shell=True)
    time.sleep(10)

    time.sleep(600)
```

script running nmap scans and hydra in a loop

Intrusion Detection Simulations (cont.)

```python
import os
import signal
import time
import subprocess

while True:
    #ICMP Flooding
    icm = subprocess.Popen("hping3 -1 -p 8080 --flood 18.190.44.37", shell=True)
    time.sleep(1)

    #SYN Flooding
    syn = subprocess.Popen("hping3 -S --flood -p 8080 18.190.44.37", shell=True)
    time.sleep(1)

    os.killpg(os.getpgid(icm.pid), signal.SIGTERM)
    os.killpg(os.getpgid(syn.pid), signal.SIGTERM)
```

script running hping3 commands on demand