Waterford Institute *of* Technology

Abstract

The aim of this research project is to find out if there need IPtables rate limiting necessary on an AWS instance and if so what are the benefits over AWS security groups.

Research Report

Cloud Computing

Rate Limiting with Iptables on AWS Instance

Eoin Dalton [20070289]

Bachelor of Science (Hons) in Information Technology Management

Waterford Institute of Technology

Table of Contents

1. Introduction

In this research report I wanted to gain an understanding of the use of rate limiting with Linux iptables on an AWS instance. The questions that this report was looking to answer are:

- The need for iptables and, rate limiting on an AWS instance if you have security groups configured?
- And what are the benefits of using iptables and rate limiting over security groups?

In this report I will focus on first gaining an understanding of iptables and rate limiting. To do this, I will be do using examples of rules taken from the experiment that was carried out and some code examples found online through sources likes stack overflow.

To perform the experiment, I needed also needed an understanding of Amazon Web Services as I will be testing on an Ubuntu 18.05 LTS server up on AWS EC2 console. The experiment that I would have ideally liked to have performed, is on Kali Linux there is a way to perform a DDoS attack through one of the tools that comes built into Kali. The problem is I would need permission from AWS to carry out such tests. If I try this all packets will just be dropped before it gets to the instance and I wouldn't be able to view logs.

In the end the experiment was carried out internally. I created four instances altogether, and was then able to get them talking to each other. In section 6 you will see some of the steps involved is this stage.

In sections 7 & 8 I will compare some of the advantages of using iptables and rate limiting over security group that are provided by AWS, stating a few points for each. The report will finish with a short conclusion. You will find an appendix at the bottom with some of the scripts and information that was gathered during the my experiment.

2. Linux Iptables

"Linux Netfilter/Iptables" were create by Rusty Russell in 1998 and is all so well known for writing "IPchains". "Linux IPtables" is a command line tool that can be used to configure the Linux kernel firewall (wiki.archlinux, 2018). This is an opensource tool that allows users to packet filter, network address translation and port translation. It also can block packets based on rules configured by the users. If you look to figure 1 you will see the basic structure of iptables.

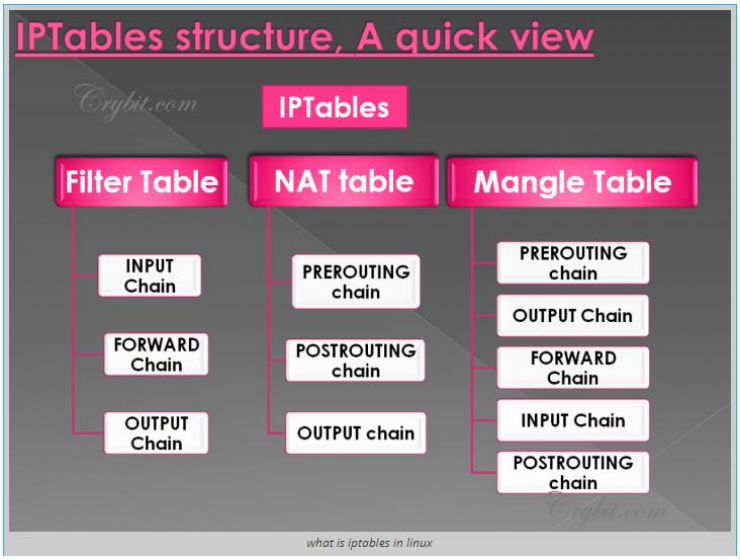

Figure 1: IPtables Structure (Ashok, 2018).

2.1 Filter Table

In figure 1 you can see the first table which the filter table. The INPUT chain deals with inbound traffic, a user filters packet based on configured rules. If you look to figure 2 these are screenshots from the experiment that was carried out. If you look to the input chain, you will see that all packets are been dropped. Then in figure 3 you will see a rule had been added to allow all HTTP/HTTPS traffic coming from port 80 and 443.

```
root@clausyd-HP-Pavilion-15-Notebook-PC:/home/clausyd# iptables -P INPUT D
ROP
root@clausyd-HP-Pavilion-15-Notebook-PC:/home/clausyd# iptables -vL
Chain INPUT (policy DROP 5 packets, 273 bytes)
 pkts bytes target     prot opt in     out     source               destin
ation

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destin
ation

Chain OUTPUT (policy ACCEPT 4 packets, 184 bytes)
 pkts bytes target     prot opt in     out     source               destin
ation
root@clausyd-HP-Pavilion-15-Notebook-PC:/home/clausyd#
```

Figure 2: Drop all INPUT traffic.

```
root@clausyd-HP-Pavilion-15-Notebook-PC:/home/clausyd# iptables -A INPUT -p tcp --sport 8
0 -j ACCEPT
root@clausyd-HP-Pavilion-15-Notebook-PC:/home/clausyd# iptables -A INPUT -p tcp --sport 4
43 -j ACCEPT
root@clausyd-HP-Pavilion-15-Notebook-PC:/home/clausyd# iptables -vL
Chain INPUT (policy DROP 15 packets, 1009 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere
tcp spt:http
    2   104 ACCEPT     tcp  --  any    any     anywhere             anywhere
tcp spt:https

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 17 packets, 1113 bytes)
 pkts bytes target     prot opt in     out     source               destination
root@clausyd-HP-Pavilion-15-Notebook-PC:/home/clausyd#
```

Figure 3: HTTP & HTTPS traffic allowed.

The next CHAIN in figure 1 looks at the forward chain. This deals with forwarding packet on the network. If you look at figure 4 you will see a simple diagram to shows the process of each chain. You can see the forward chain is just looking at the packets to be forwarded to another endpoint on the network. This is useful if you have more the on machine on the network. If the packet does not correspond with the rule it could be dropped.

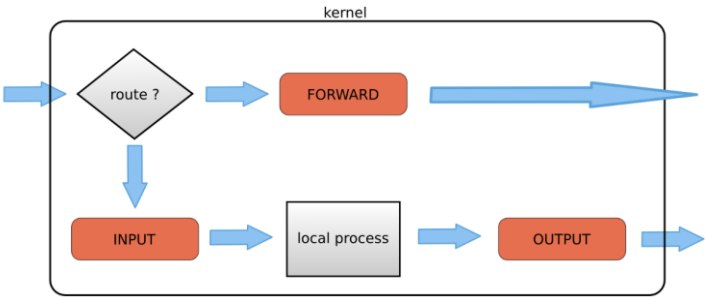

Figure 4: Example of CHAINS (Cobbaut, 2015).

The last CHAIN in the filtering table from figure 1 looks at outbound traffic. IPtables allows you to limit ports, DROP/ALLOW traffic going to websites, IP addresses. If you look to figure five you will see where connections going to Moodle has been blocked.

```
Chain OUTPUT (policy ACCEPT 1079 packets, 734K bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DROP       all  --  any    any     wit-moodle-a10-pw.heanet.ie  anywhere

   57  3420 DROP       tcp  --  any    any     anywhere             wit-moodle-a10-pw.hea
net.ie
```

Figure 5: Block out bound connection to Moodle.

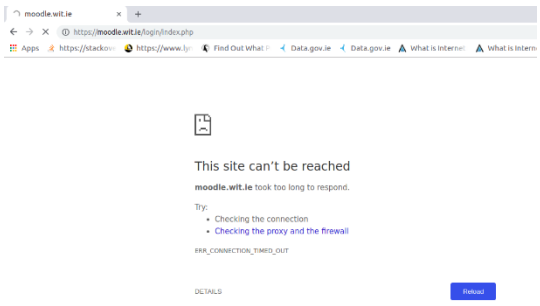

Figure 6: Moodle no Connection

2.2 Network Address Translation (NAT) Table

NAT deals with changing the network address information in the "Internet Protocol (IP) datagram packet headers while they are in transit across a traffic routing device" (Ashok, 2018). What happens here, let's say on you home network we have a public IP the speaks to the world, then inside your home we have a private network. NAT deal with the routing of packets on a subneted network.

- PREROUTING Chain – this translates the network address in the packet headed before routing takes place.
- POSTROUTING Chain - this translates the network address in the packet headed after routing takes place.
- OUTPUT Chain – looks at the packet after it was altered POST or PRE-routing.

2.3 Mangle Table

Mangle table allows the altering for the packet. It allows a user to queue the traffic then based on a service you can have precedence over other services. A user can alter the type of services field in the packet to allow file transfer protocol (FTP) packets come first over a service like HTTP traffic (Bandel, 2000). A user can alter the time to live value (TTL) or any packet data depending on what they need.

- PREROUTING Chain: used to alter the packet before routing takes place.
- OUTPUT Chain: alter the packet on outbound traffic.
- FORWARD Chain: alter the packet of traffic that is been forwarded to another machine.
- INPUT Chain: alter the packet on inbound traffic.
- POSTROUNTING Chain: used to alter the packet after routing takes place.

4. Rate Limiting

What rate limiting looks at is setting limits of the amount of incoming, outgoing and forwarded packets. If the user sets a rule to limit to 50 packets per second it will match any connections to has 50 packets per second, the user can state what they want to do with the packets DROP, ALLOW. Having this feature allows user to protect against potential DDoS attacks. iptables allow rate limiting for SSH, HTTP, HTTPS and ICMP. If an IP repeatedly break rate limits, then the can be blacklisted.

Another way rate limiting can be useful is to set a limit on the amount of connection from a single IP address. If you look to figure 7 below, this was a script found at Programmers Blog (Blog, 2017). This is a bash script where they are saying that an IP can have 100 connections and it looks like is 100 second. They set a variable to DROP and variables for time-period and IP count and IPtables to use later in the rules.

There are four iptables rules at the bottom. They are all looking at inbound traffic, the protocol is TCP and the destination port is 80 and 443 which is HTTP and HTTPS traffic. It will be transmitted on Ethernet 0. -m is load the module state. They then use the variables time and IP count if the rules match that criteria the action is to drop the packet. This is a simple script that will filter all web traffic and its written is six lines of code

```
#!/bin/bash
IPT=/sbin/iptables
# Max connection in seconds
TIME_PERIOD=100
# Max connections per IP
BLOCKCOUNT=100

# default action can be DROP or REJECT
DACTION="DROP"
$IPT -A INPUT -p tcp --dport 80 -i eth0 -m state --state NEW -m recent --set
$IPT -A INPUT -p tcp --dport 80 -i eth0 -m state --state NEW -m recent --update --seconds $TIME_PERIOD --hitcount $BLOCKCOUNT -j $DACTION

$IPT -A INPUT -p tcp --dport 443 -i eth0 -m state --state NEW -m recent --set
$IPT -A INPUT -p tcp --dport 443 -i eth0 -m state --state NEW -m recent --update --seconds $TIME_PERIOD --hitcount $BLOCKCOUNT -j $DACTION
```

Figure 7: Rate Limit time and connection amount (Blog, 2017).

In figure 8 you will see an example on the to rate limit for SSH connections. What you will see is the first command the are using -N LOGDROP which is changing the rule so that all the drops are logged for examination. The -I is stands for insert into the inbound chains this rule. They are stating that the protocol is TCP over port 22 which is the port for SSH. It's over an ethernet connection. It then states that over a time limit of 60 seconds their can be four failed connection attempts and thing over that will be dropped and logged. The can protect us against brute force attempts.

```
/sbin/iptables -N LOGDROP
/sbin/iptables -A LOGDROP -j LOG
/sbin/iptables -A LOGDROP -j DROP
iptables -I INPUT -p tcp --dport 22 -i eth0 -m state --state NEW -m recent --set
iptables -I INPUT -p tcp --dport 22 -i eth0 -m state --state NEW -m recent  --update --seconds 60 --hitcount 4 -j LOGDROP
```

Figure 8: Rate Limit SSH connections (Huckaby, 2010).

For the experiment is the project, it looked at rate limit on ping requests. If you have an army of zombie bots that all start to pinging at the one-time, rate limit on ICMP can protect against this. If you look at figure 8 you will see the rule that was set. It is rate limiting on inbound ICMP (ping). This is limits packets to one second the limit burst one states that is can have one packet match before the one second limit works. The first command excepts the inbound pings, then the second command kicks in and state that any packets matching this limit and the burst then send it to the log file as told by the LOG command. The prefix commands all you to add message to the log files. The next command then will drop the connection and the last command is allowing outbound pings. These commands will run from top to bottom sequence.

```
iptables -A INPUT -p icmp -m limit --limit 1/s --limit-burst 1 -j ACCEPT

iptables -A INPUT -p icmp -m limit --limit 1/s --limit-burst 1 -j LOG --log-prefix PING-
DROP:

iptables -A INPUT -p icmp -j DROP

iptables -A OUTPUT -p icmp -j ACCEPT
```

Figure 9: Rules used in experiment (FrostyFire, 2004).

If you look to figure 10 you will see that not packets have been lost, this was before the rules were set. In figure 11 you can see significant packet loss. If you look at the terminal in figure 11 on the bottom right, you will see nearly half of the packets have been dropped. If you look to figure 12 at the bottom of page seven you will see all the packets the have been dropped. There is uses full information in here such SRC IP address and MAC addresses of attacking machine. This can be used for blacklist and can be automated with some bash scripts.

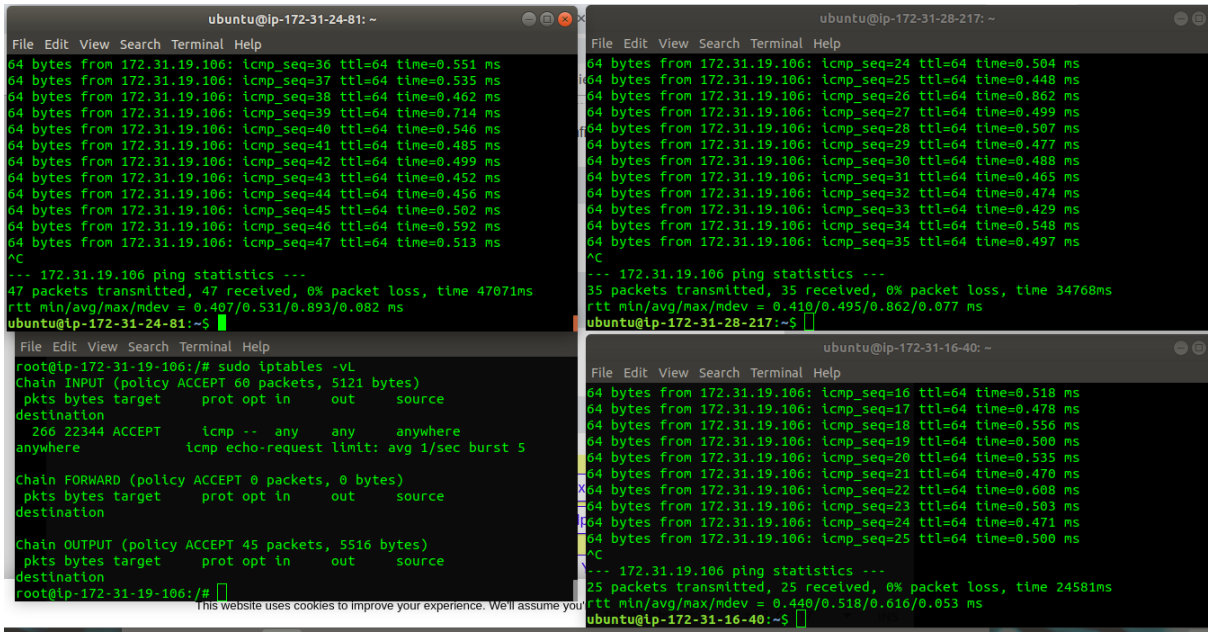

Figure 10: 0% packet loss.

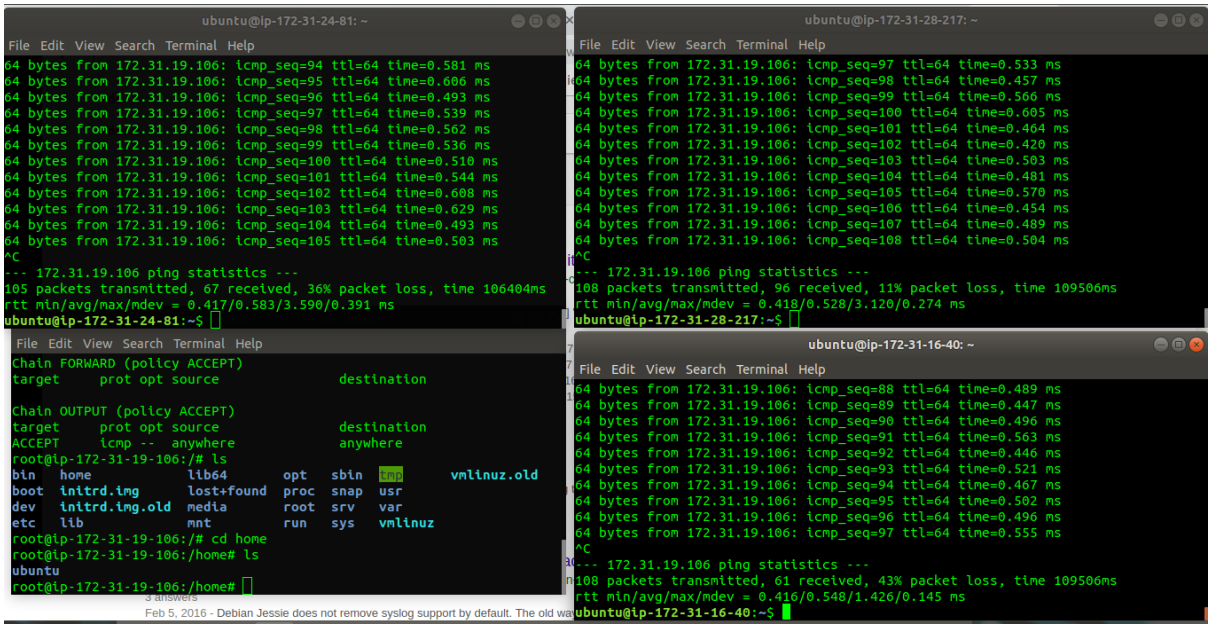

Figure 11: After rate limiting pack loss is significant.

```
root@ip-172-31-19-106:/var# tail /var/log/kern.log
Oct 27 18:56:40 ip-172-31-19-106 kernel: [17271.587060] PING-DROP:IN=eth0 OUT= MAC=06:25:86:3e:b6:5e:06:79:75:57:ec:92:08:00 SRC=172.31.16.40 DST=172.
31.19.106 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=1852 DF PROTO=ICMP TYPE=8 CODE=0 ID=1612 SEQ=99
Oct 27 18:56:41 ip-172-31-19-106 kernel: [17272.611107] PING-DROP:IN=eth0 OUT= MAC=06:25:86:3e:b6:5e:06:79:75:57:ec:92:08:00 SRC=172.31.16.40 DST=172.
31.19.106 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=1940 DF PROTO=ICMP TYPE=8 CODE=0 ID=1612 SEQ=100
Oct 27 18:56:42 ip-172-31-19-106 kernel: [17273.635156] PING-DROP:IN=eth0 OUT= MAC=06:25:86:3e:b6:5e:06:79:75:57:ec:92:08:00 SRC=172.31.16.40 DST=172.
31.19.106 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=2163 DF PROTO=ICMP TYPE=8 CODE=0 ID=1612 SEQ=101
Oct 27 18:56:43 ip-172-31-19-106 kernel: [17274.659227] PING-DROP:IN=eth0 OUT= MAC=06:25:86:3e:b6:5e:06:79:75:57:ec:92:08:00 SRC=172.31.16.40 DST=172.
31.19.106 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=2347 DF PROTO=ICMP TYPE=8 CODE=0 ID=1612 SEQ=102
Oct 27 18:56:44 ip-172-31-19-106 kernel: [17275.683211] PING-DROP:IN=eth0 OUT= MAC=06:25:86:3e:b6:5e:06:79:75:57:ec:92:08:00 SRC=172.31.16.40 DST=172.
31.19.106 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=2597 DF PROTO=ICMP TYPE=8 CODE=0 ID=1612 SEQ=103
Oct 27 18:56:45 ip-172-31-19-106 kernel: [17276.707289] PING-DROP:IN=eth0 OUT= MAC=06:25:86:3e:b6:5e:06:79:75:57:ec:92:08:00 SRC=172.31.16.40 DST=172.
31.19.106 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=2599 DF PROTO=ICMP TYPE=8 CODE=0 ID=1612 SEQ=104
Oct 27 18:56:46 ip-172-31-19-106 kernel: [17277.731320] PING-DROP:IN=eth0 OUT= MAC=06:25:86:3e:b6:5e:06:79:75:57:ec:92:08:00 SRC=172.31.16.40 DST=172.
31.19.106 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=2694 DF PROTO=ICMP TYPE=8 CODE=0 ID=1612 SEQ=105
Oct 27 18:56:47 ip-172-31-19-106 kernel: [17278.755445] PING-DROP:IN=eth0 OUT= MAC=06:25:86:3e:b6:5e:06:79:75:57:ec:92:08:00 SRC=172.31.16.40 DST=172.
31.19.106 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=2938 DF PROTO=ICMP TYPE=8 CODE=0 ID=1612 SEQ=106
Oct 27 18:56:48 ip-172-31-19-106 kernel: [17279.779491] PING-DROP:IN=eth0 OUT= MAC=06:25:86:3e:b6:5e:06:79:75:57:ec:92:08:00 SRC=172.31.16.40 DST=172.
31.19.106 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=3134 DF PROTO=ICMP TYPE=8 CODE=0 ID=1612 SEQ=107
Oct 27 18:56:49 ip-172-31-19-106 kernel: [17280.803542] PING-DROP:IN=eth0 OUT= MAC=06:25:86:3e:b6:5e:06:79:75:57:ec:92:08:00 SRC=172.31.16.40 DST=172.
31.19.106 LEN=84 TOS=0x00 PREC=0x00 TTL=64 ID=3143 DF PROTO=ICMP TYPE=8 CODE=0 ID=1612 SEQ=108
root@ip-172-31-19-106:/var#
```

Figure 12: Dropped packets.

5. Amazon Web Services (AWS)

AWS offers cloud-based services for all types of businesses. From small to big and new to old it can cater for all. Some of the main services it has to offer is computing power, storage, infrastructure, networking. If a company want's, they could host their hole organisation in a virtualised setting. This would involve combining most of AWS's services. The business would need compute to power for all the different endpoints. They would need infrastructure to host the virtual network. Networking would be needed to connect all endpoints (with virtual switches). Storage would be necessary to store on information and this could be stored on AWS databases. That is the power of AWS.

6. Creating AWS Instances that can Communicate

The reason we want AWS instance communicating is, so we can preform the practical element of this report. The report was looking a rate limiting on an AWS instance. What I done was to set up four Ubuntu 18.04 servers. One server would be the target and have the rate limit rules configured and the three other severs would send ping requests to the target server. First, we needed all machines to communicate with each other. As these machines are created in isolation they can't communicate.

After doing some searching on the internet on how to get instances talking, one article was found. (Admin, 2017) stated that you could make them communicate be configuring their security groups. All you do is add security group ID for machine to the three other machines and the same for all machines. If you look at figure 12 below , this is UbuntuVM3 as you will see I allowed ICMP, TCP, HTTP and HTTPS traffic from all the rest of the instances. SSH is set by default to allow me to ssh into each device.

Figure 13: AWS Security Group rules

Once the security groups were configured the next task I done was to write four bash scripts for the SSH connection from my personal Ubuntu machine. I wrote the scripts for my own convenience. If you see figure 14 is the simple script. I install the openssh client on my machine I could then connect directly for command prompt. In figure 14 you will see the path to where my private key is. The next part is my public domain in AWS. I used chmod 400 to project the .pem file where the public and private key were stored. In figure 15 I am just showing each file once I made them executable.

```
#!/bin/bash
ssh -i /home/clausyd/OneDrive/Amazon/My_Key_Pair.pem ubuntu@ec2-54-246-250-0.eu-west-1.compute.amazonaws.com
```

Figure 14: Script to connect to instances

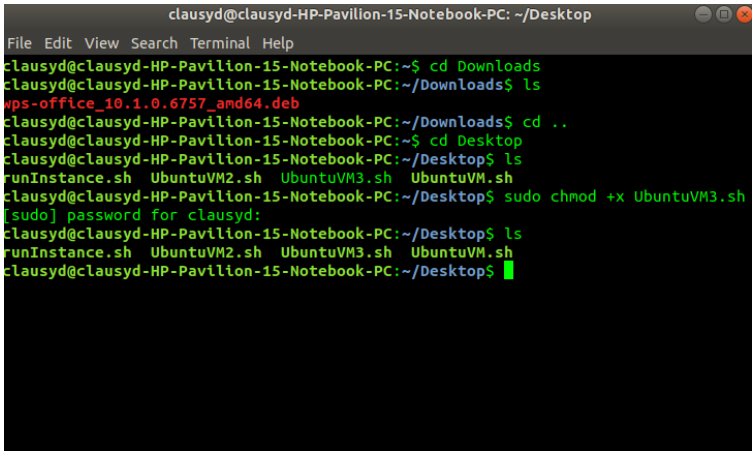

Figure 15:. Making scripts executable.

Once I could SSH into all the instances, I then was testing ping connection between devices. If you look at figure 15 you will see successful pings from three UbuntuVM's to the target device. The next stage involved configuring rate limit rules and performing tests this is found in section 4 of the document.

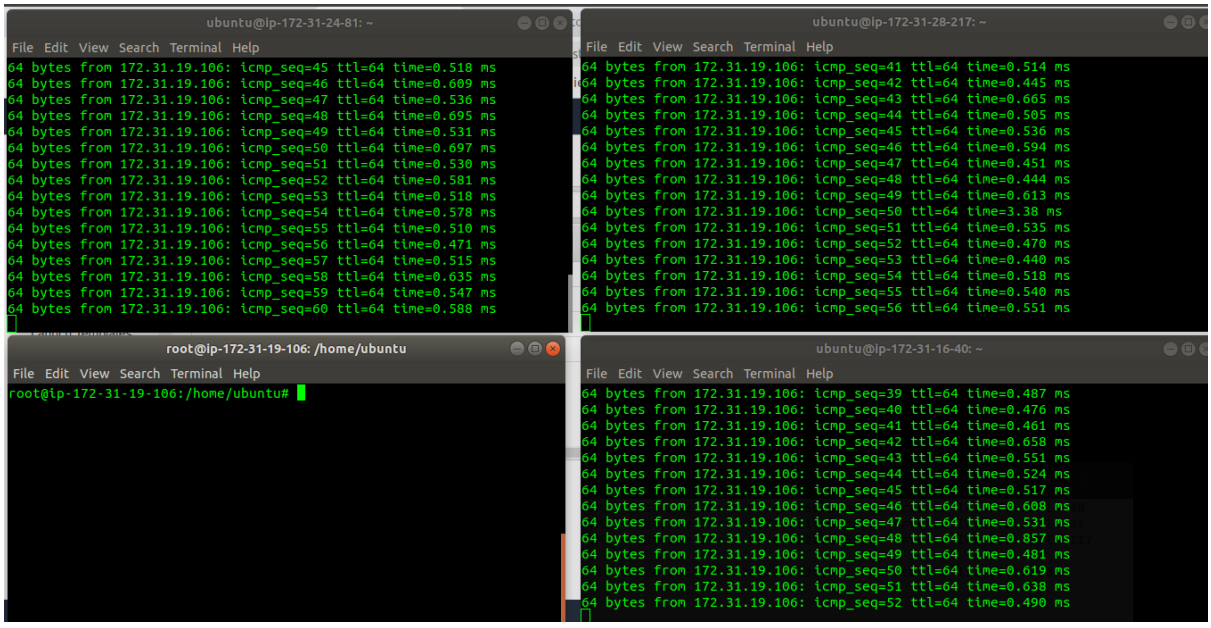

Figure 16: The UbuntuVM's pinging target:

7. Limitations of Amazon Security Groups

Some of the limitation of the AWS security groups:

- 60 rule limits for INPUT & OUTPUT traffic.
- No Rate Limiting on Packets

In the AWS security groups there is a limit to the number of rules allowed. This is set at 60 and can be adjusted to 100 in total. This is a relativity low number. When configuring my security group contained 14 rules to allow four allow four different protocols from three different devices. If you are filtering by an IP or port a user would not be long using the 60 rules.

AWS security group don't cater for rate limiting packets. You can specify rules on protocol, ports , source and destination IP addresses. This is basic a basic firewall. However, AWS prevents DDoS attacks is with the use of Amazon shield standard. The AWS security rules will always take precedence over Iptables rules. This means that if a large number of packets are sent to the sever AWS was handle it. But you server will still receive a certain amount of packet before AWS shield will start to drop the packets.

Amazon do have a solution, but users must pay for this service, called Amazon shield advanced. This will allow users to "write customized rules to mitigate sophisticated application layer attacks" and it all so allows user to "set up rules proactively to automatically block bad traffic, or respond to incidents as they occur" (Amazon, 2018).

8. Benefits of IPtables and Rate Limiting over AWS Security Groups

- The benefits of using Iptables and rate limiting is the fact that AWS security group don't allow that level of configuration.
- The logs from such attacks can be useful for understanding what has happened and for blocking malicious IP addresses.
- Its free open source tool.
- You can set rule to be proactive dropping packet that match criteria of the rule.

Security group will allow filtering on protocol, ports , source and destination IP addresses. Rate limiting in iptables will allow user to limit through time and amount of connection of packets. In the example for the experiment we had a time frame of one second with the limit, and number of packets to one. So, if a user sends more the one packet in the space of one second and extra packets will be dropped. You then could add the IP to a blacklist. If you want this type of functionality in AWS, you must pay for Amazon shield advanced. The logs that are collected in Iptables has useful information such an IP addresses and MAC addresses, ports and protocols. This is useful for blacklist and general monitoring.

10. Conclusions

For this research document we were asked to research a cloud technology. After this the first search I done on Google was to search what are cloud technologies. As we had not cloud module up to this point, I have no ideas on what the technologies are let alone what I could research. After a talk with Richard and Jimmy Mc Gibney it was decided that I would research Linux iptables rate limiting.

First, I performed my experiment as I find it easier to write about something if I understand how it works. Then I started researching the topic. I tried to explain what the sections are about with the screenshots from my experiment but also with extracts of rules taken from online.

The conclusions I reached after the research for this project is that the use of iptables and rate limiting on an AWS instances depends on the user's preferences and money. IPtables and rate limiting is free to be used on Linux instances. If you want the same functionality from an AWS service, the user must pay for Amazon shield advanced. The security groups in AWS does not have the functionality to rate limit packets. AWS will help prevent DDoS attack with their free Amazon shield standard, but this will drop all packets before the reach you instance but the instance will still receive a certain number of packets.

The use of rate limiting can be useful for ssh connection as you can filer the connection by IP and the amount of connection tries. It can also be used as a proactive way of preventing DDoS and if you match criteria in a rule, such as the number of packets from a certain source, anything over that limit can be dropped and the IP can be added to a blacklist.

In conclusion if you are looking for a free way of setting firewall rules and limits packets to you instance Linux iptables and rate limiting is the way to go. If money is not problem, then just use Amazon shield advanced they can even write the rules for you.

Bibliography

Admin, 2017. *linuxroutes.com.* [Online]
Available at: https://linuxroutes.com/make-two-ec2-instances-connect-aws/
[Accessed 27 10 2018].

Amazon, 2018. *Amazon.* [Online]
Available at: https://aws.amazon.com/shield/
[Accessed 29 10 2018].

Ashok, A., 2018. *crybit.com.* [Online]
Available at: https://crybit.com/what-is-iptables-in-linux/
[Accessed 28 10 2018].

Bandel, D., 2000. *www.informit.com.* [Online]
Available at: http://www.informit.com/articles/article.aspx?p=19626
[Accessed 28 10 2018].

Blog, P., 2017. *Programster's Blog.* [Online]
Available at: https://blog.programster.org/rate-limit-requests-with-iptables
[Accessed 29 10 2018].

Cobbaut, P., 2015. *linux-training.be.* [Online]
Available at: http://linux-training.be/networking/ch14.html
[Accessed 28 10 2018].

FrostyFire, 2004. *Stack OverFlow.* [Online]
Available at: https://stackoverflow.com/questions/23377706/iptables-prevent-flooding
[Accessed 27 10 2018].

Huckaby, J., 2010. *rackaid.* [Online]
Available at: https://www.rackaid.com/blog/how-to-block-ssh-brute-force-attacks/
[Accessed 29 10 2018].

wiki.archlinux, 2018. *wiki.archlinux.* [Online]
Available at: https://wiki.archlinux.org/index.php/iptables
[Accessed 28 10 2018].

11. Appendices

11.1 AWS Instance Details

Group ID: sg-0aa9ae9808eae698d (UnbuntuVM1) Instance ID: i-0f2a17350b1ead4e9 Private IP: 172.31.16.40

Group ID: sg-092fbd9ec724d12b3 (UnbuntuVM2) Instance ID: i-0071a3e14d8dc8879 Private IP: 172.31.24.81

Group ID: sg-06e2bc62f9cf425d2 (UnbuntuVM3) Instance ID: i-0fa64322a23b57674 Private IP: 172.31.28.217

Group ID: sg-05eec9c74f7e93b38 (target) Instance ID: i-0a23a2d75fd309b21 Private IP: 172.31.19.106

11.2 Rate Limit Commands

iptables -A INPUT -p icmp -m limit --limit 1/s --limit-burst 1 -j ACCEPT

iptables -A INPUT -p icmp -m limit --limit 1/s --limit-burst 1 -j LOG --log-prefix PING-DROP:

iptables -A INPUT -p icmp -j DROP

iptables -A OUTPUT -p icmp -j ACCEPT

11.3 Scripts to connect to Instances

Target Server

#!/bin/bash

ssh -i /home/clausyd/OneDrive/Amazon/My_Key_Pair.pem ubuntu@ec2-34-244-22-220.eu-west-1.compute.amazonaws.com

UbuntuVM

#!/bin/bash

ssh -i /home/clausyd/OneDrive/Amazon/My_Key_Pair.pem ubuntu@ec2-54-246-250-0.eu-west-1.compute.amazonaws.com

UbuntuVM2

#!/bin/bash

ssh -i /home/clausyd/OneDrive/Amazon/My_Key_Pair.pem ubuntu@ec2-54-154-33-11.eu-west-1.compute.amazonaws.com

UbuntuVM3

#!/bin/bash

ssh -i /home/clausyd/OneDrive/Amazon/My_Key_Pair.pem ubuntu@ec2-52-19-161-206.eu-west-1.compute.amazonaws.com

11.4 Process Report

Process Report

I have worked on my research project over the midterm as I wanted it completed so I can concentrate on the rest of the project to the semester. If what I have needs a rework, please advise.

Project Structure

The structure of my reports is as follows, first introduction. The I will speak about iptables and rate limiting. To explain these two sections I will be using screenshots from the experiment that I performed. I then give a paragraph to what AWS is about the services they offer. In the next section I will explain how I got all Amazon instances communicating to each other again with screen shot for the experiment. The next two sections I will look at the advantages of using iptables and rate limiting have over Amazon security groups and what amazon offer as alternatives two rate limiting. This section I will speak about Amazon shield standard and advanced and the limitations of the AWS security groups. The report will also consist of introduction, conclusion and appendix. In the appendix I put any scripts, code and information about my instances on AWS.

Project Experiment

For the experiment I want for flood the sever with packets. If I tried this from the outside, Amazon will just drop all the packets before the reach the instance and I wouldn't be able to see the logs that are generated. I decided to set up four instances in the EC2 console. I tested the rate limit on the ICMP protocol. I was able to get the four instances communicating by altering their security group. I first sent pings from the four instances with no rule set and could see no packet loss. I then set the rule and could see significant packet loss. If the experiment needs more work, please advise.

Figure 17: Process report