Business Continuation and Cloud Security

CA – Cloud Infrastructure Security

By Dean xxxxxxxxxxxxxx
B0009xxxxxxxxxxxxxx

Department of Informatics
School of Informatics and Engineering
Technological University xxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxx

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Digital Forensics & Cyber Security
Application Security
Mark xxxxxxxxx

10/11/2019

Plagiarism Declaration

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
DEPARTMENT OF INFORMATICS
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
LECTURER: MARK xxxxxxxxxxxxx

DECLARATION ON PLAGIARISM

I declare that the work I/We am (are) submitting for assessment by the Institute examiner(s) is entirely my (our) own work, except where the author or source has been duly referenced and attributed.

I confirm that this material has not been previously submitted for a degree or any other qualification at xxxxxxxxxxx or any other institution. I further confirm that I have read and understood the Institute policy on plagiarism in assignments and examinations (3AS08.doc) and that I am not, so far as I am aware, in breach of any of these regulations.

Signed: Dean xxxxxxxxxxxxxxxx

Date: 10/11/2019

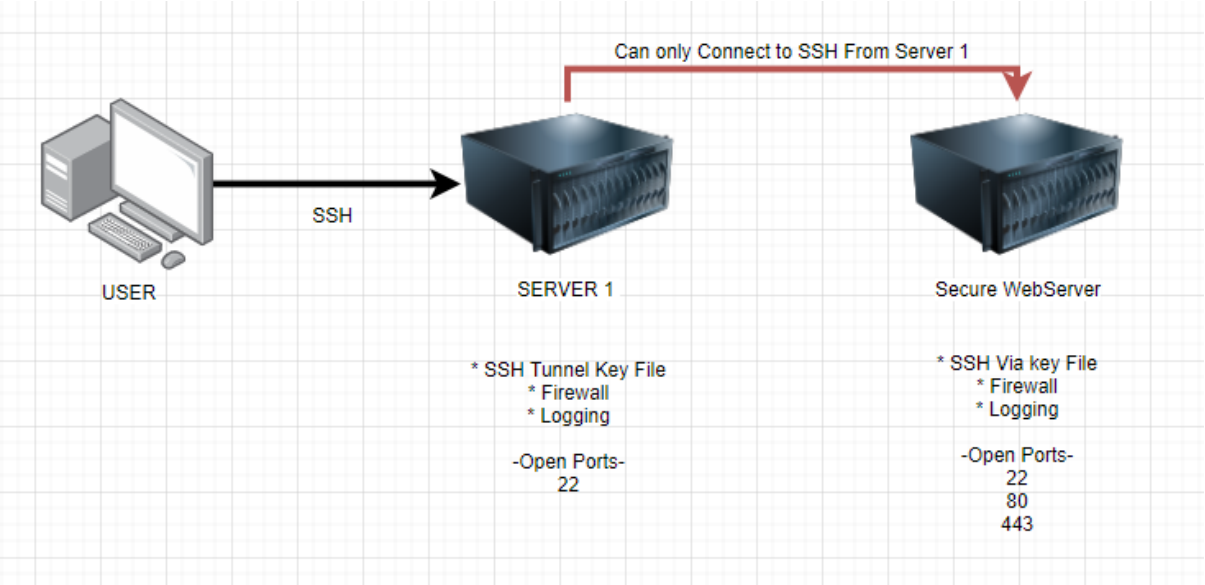
Table of Contents

Introduction

NOTE: Both the Key To access the server is and video demonstration the connection is located in the Appendix of the documentation, also under the “Connection Command” section there is a command that can be used for connection the system with ease and a little modification

The goal of this project is to build a secure cloud webserver. That is a publicly accessible Webserver that hosts a website; although publically available via the secure webserver the only way other to connect to the system would be SSH via a secondary SSH Server that is also secured and locked down to the public. The idea is to get an understanding of how to build a simple secure webserver and look at securing it in different ways.

The system would logically look like the bellow connection in the image

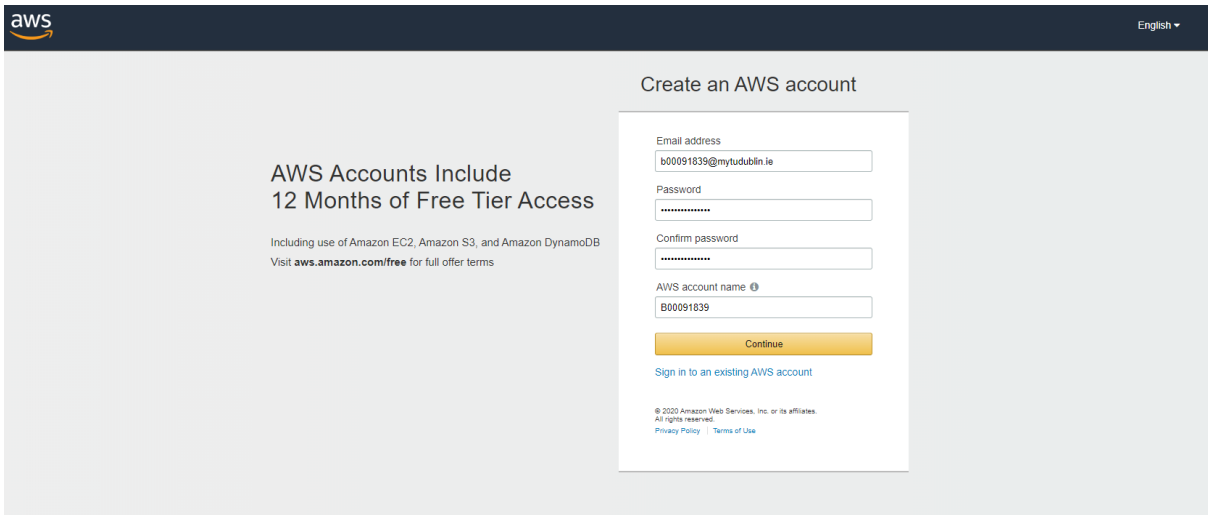


Setting up the Cloud Servers

In this section we will look at the setup of the system, which will include the configuration of the online boxes through AWS for the setup, this section will also include the configurations of the Webserver, development of a webpage and configuration of the ssh tunnel, and firewall rules.

Registering for AWS Free-Tier

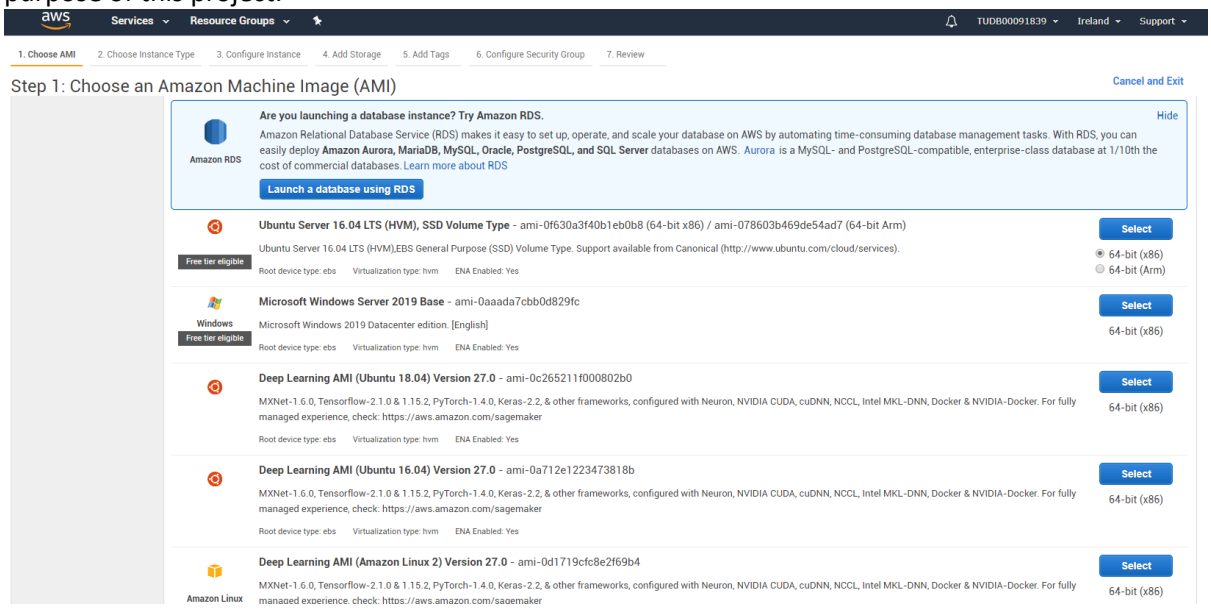
After advisement from my lecturer, it was found that Amazon Web Services “AWS” free tier would be sufficient for this project. The AWS allows for a user to build a pre-configured service or start up a virtual machine with multiple types of operating systems. The server to register for an AWS account is available through the following link https://portal.aws.amazon.com/billing/signup#/start



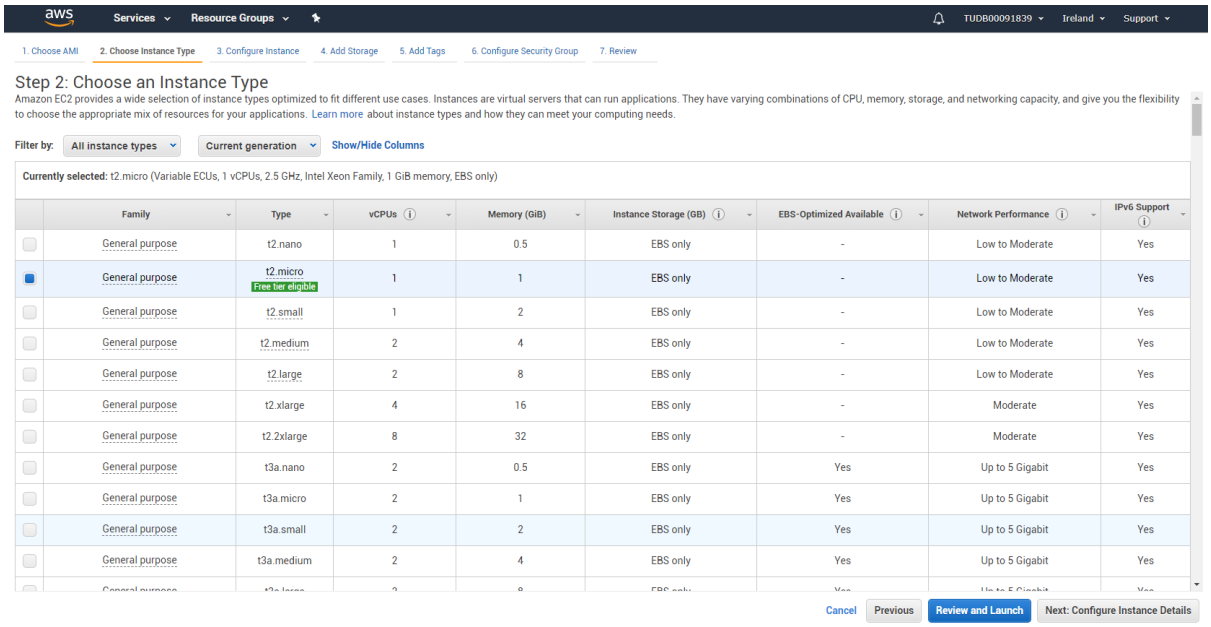
After filling in personal details it then pushes you to verify via credit card number to verify you are a real person joining the service, at which it charges your card 1 US Dollar to confirm it is a real card.

Setup of AWS Service

Here we will look at the setup options available to both systems, the best options for this system was to use an Amazon Virtual Machine Image as they are pre-built images of what is needed for the purpose of this project.

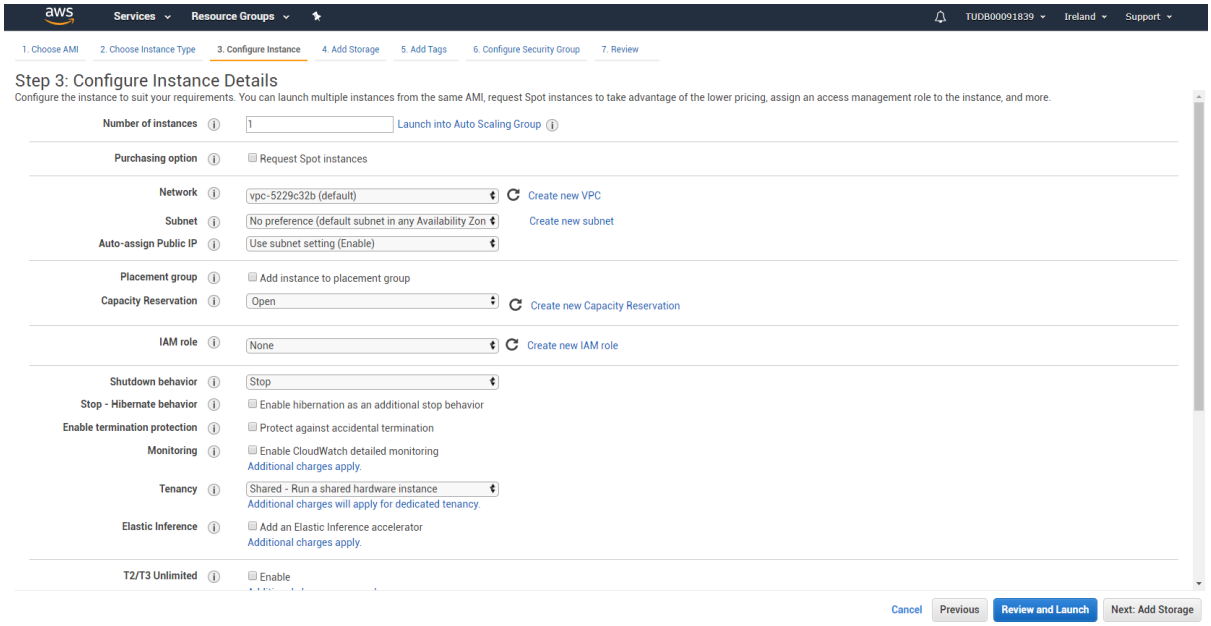


The option that I felt was best for this assignment was to user an Ubuntu Server 16.04, this is a system I know well and well come with some of the software I needed pre-configured on the system and ready to go.

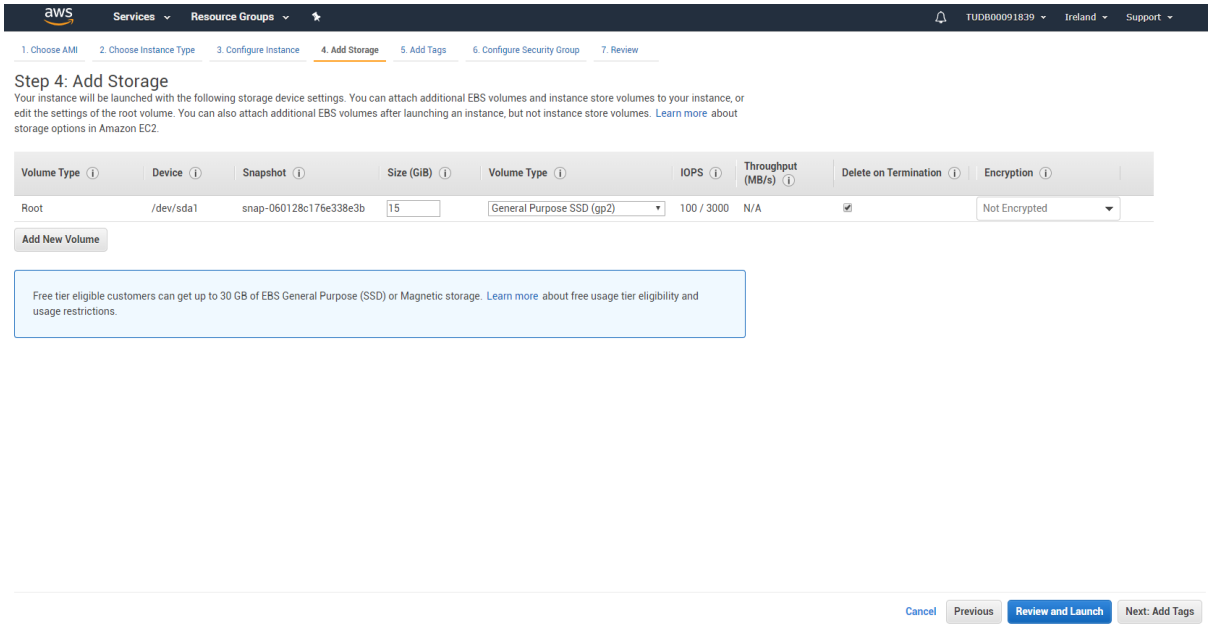


Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes
General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes
General purpose	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
General purpose	t3a.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes
General purpose	t3a.micro	2	1	EBS only	Yes	Up to 5 Gigabit	Yes
General purpose	t3a.small	2	2	EBS only	Yes	Up to 5 Gigabit	Yes
General purpose	t3a.medium	2	4	EBS only	Yes	Up to 5 Gigabit	Yes

The general specification's should be what is needed for the project to be workable and be able to function well for the intended use.



There default credentials that where provided are what will be needed for the configuration, so no changes where needed for the building on the system, in the configuration instance details menu. To make sure there is enough room for the OS, Application, log files and Website on the required system, with Linux being such a lightweight system I felt that have 15gbs of storage would be plenty for the purposes of the assignment.



The next stage after the above was to configure security for Virtual Machines, For the purpose of the project on both systems; I decided to use the one key file called **CIS_Server_1.pem** this will allow the same key to be used on both systems, across one login making it easier not to mix up the keys for each place, as they are being used.

Firewall Rules

The configuration for the system so it would allow the required that to be transmitted only such as only being able to SSH from the SSHserver to the Webserver and only have the Web traffic available to the public when viewing the webpage

AWS offer their own version of a Firewall and rulesets to disable incoming and outgoing connections such as simple GUI that can be found under Amazon Security Group that will allow you to set the network rules

SSH Server

Here we will look at the inbound and outbound rules for the SSHserver on the AWS firewall

Inbound

Bellow you can see the inbound rules on the SSH server showing that any use can SSH to the first box

sg-029ac06da49a3c3ef - CloudInfrastructureSecurity

Details | Inbound rules | Outbound rules | Tags

Inbound rules — Edit inbound rules

Type	Protocol	Port range	Source	Description - optional
SSH	TCP	22	0.0.0.0/0	SSH Server / Tunnel

Outbound

Bellow you can see the outbound rules for security; This is so the server cannot be used to SSH to any other system that it is not intended for and It can only be used to SSH to the Webserver it was setup for

sg-029ac06da49a3c3ef - CloudInfrastructureSecurity

Details | Inbound rules | Outbound rules | Tags

Outbound rules — Edit outbound rules

Type	Protocol	Port range	Destination	Description - optional
SSH	TCP	22	172.31.21.67/32	Only SSH out

Webserver

Here we will look at the inbound and outbound rules for the WebServer on the AWS firewall

Inbound

Bellow we can see the inbound rules, as required by the assignment both HTTP and HTTPS ports are available from a remote system, here you can see all traffic from both IPv4 and IPv6 is allowed to access the content on port 80 and 443.

Here you can also see that there is a dedicated source address for the SSH meaning that only connections from the specified Internal IP address that is listed in the table will be able to connect to the system via SSH. This IP belongs to the SSHserver, this means that in order for someone to SSH to this server they would need to first connect to the SSHserver.

Inbound rules | Outbound rules | Tags

Inbound rules Edit inbound rules

Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	0.0.0.0/0	Port 80
HTTP	TCP	80	::/0	Port 80
SSH	TCP	22	172.31.25.203/32	Conenction from Server 1
HTTPS	TCP	443	0.0.0.0/0	Port 443
HTTPS	TCP	443	::/0	Port 443

Outbound

In the bellow image you can see the rules for the outbound rules of the WebServer, these rules allow all traffic from both HTTP and HTTPs to be sent to all users who request content on both IPv4 and IPv6, but like the inbound rules there destination port for the SSH is bound to the SSHserver, meaning that if there is a possibility of this Site being exploit that it can only be used to access the SSHserver and not be used to attempt to SSH to any other systems

Inbound rules | Outbound rules | Tags

Outbound rules Edit outbound rules

Type	Protocol	Port range	Destination	Description - optional
HTTP	TCP	80	0.0.0.0/0	-
HTTP	TCP	80	::/0	-
SSH	TCP	22	172.31.25.203/32	-
HTTPS	TCP	443	0.0.0.0/0	-
HTTPS	TCP	443	::/0	-

SSH Server Security

The SSH server needs to be able to receives a connection from the user, and then can be used to connect to the Webserver when needed.
To connect To the SSHserver we can used the following command,

ssh -i "C:\Users\Dean\Documents\College 2020 Documents\Cloud Security CA\CIS_Server_1.pem" **ubuntu@ec2-52-208-12-75.eu-west-1.compute.amazonaws.com**

The above commands uses the CIS_Server_1.pem key file to allow connection to the server fast and securely without the need for usernames and passwords to be inputted by the user.

the first step is to lock down the entire system, this we can use the external firewall on AWS that can be seen in a past section of the paper, but this can also be done on the system with a firewall called UFW, by default UFW is installed on Amazon AWS systems and has SSH open so that it will allow connection from SSH Clients Running the command **sudo netstat -ntlp | grep LISTEN** we can see what ports are open on the firewall and listening for traffic

```
ubuntu@ip-172-31-25-203:~$ sudo netstat -ntlp | grep LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1332/sshd

tcp6       0      0 :::22                   :::*                    LISTEN      1332/sshd

ubuntu@ip-172-31-25-203:~$
```

As can be seen in the above image the only port open listening for Traffic is 22, which is for the SSH connection that is needed for the project.

Checking the **/var/log** folder we can see that the system is logging interactions with the server which includes login attempts which can be used for tracking attack attempts.

```
ubuntu@ip-172-31-21-67:/$ cd /var/log/
ubuntu@ip-172-31-21-67:/var/log$ ls
alternatives.log  apache2  auth.log  cloud-init.log         dist-upgrade  fsck      lastlog      lxd     syslog.1            wtmp
amazon            apt      btmp      cloud-init-output.log  dpkg.log      kern.log  letsencrypt  syslog  unattended-upgrades
ubuntu@ip-172-31-21-67:/var/log$
```

Web Server Security

This will only allow connection from the preset IP address
ssh -i "C:\Users\Dean\Documents\College 2020 Documents\Cloud Security CA\CIS_Server_1.pem" **ubuntu@ec2-34-244-49-203.eu-west-1.compute.amazonaws.com**

the first step is to lock down the entire system, this we can use the external firewall on AWS that can be seen in a past section of the paper, but this can also be done on the system with a firewall called UFW, by default UFW is installed on Amazon AWS systems and has SSH open so that it will allow connection from SSH Clients. Running the command **sudo netstat -ntlp | grep LISTEN** we can see what ports are open on the firewall and listening for traffic

```
ubuntu@ip-172-31-21-67:~$ sudo netstat -ntlp | grep LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1323/sshd

tcp6       0      0 :::22                   :::*                    LISTEN      1323/sshd

ubuntu@ip-172-31-21-67:~$
```

As can be seen in the above image the only port open listening for Traffic is 22, which is for the SSH connection that is needed for the project. On this system we need to open two other ports as required to be able to access the HTTP and HTTPS for the webserver this is done with the command **sudo ufw allow 80** and **sudo ufw allow 443** and can be seen them in the photo bellow.

```
ubuntu@ip-172-31-21-67:~$ sudo ufw allow 80
Rules updated
Rules updated (v6)
ubuntu@ip-172-31-21-67:~$ sudo ufw allow 443
Rules updated
Rules updated (v6)
ubuntu@ip-172-31-21-67:~$
```

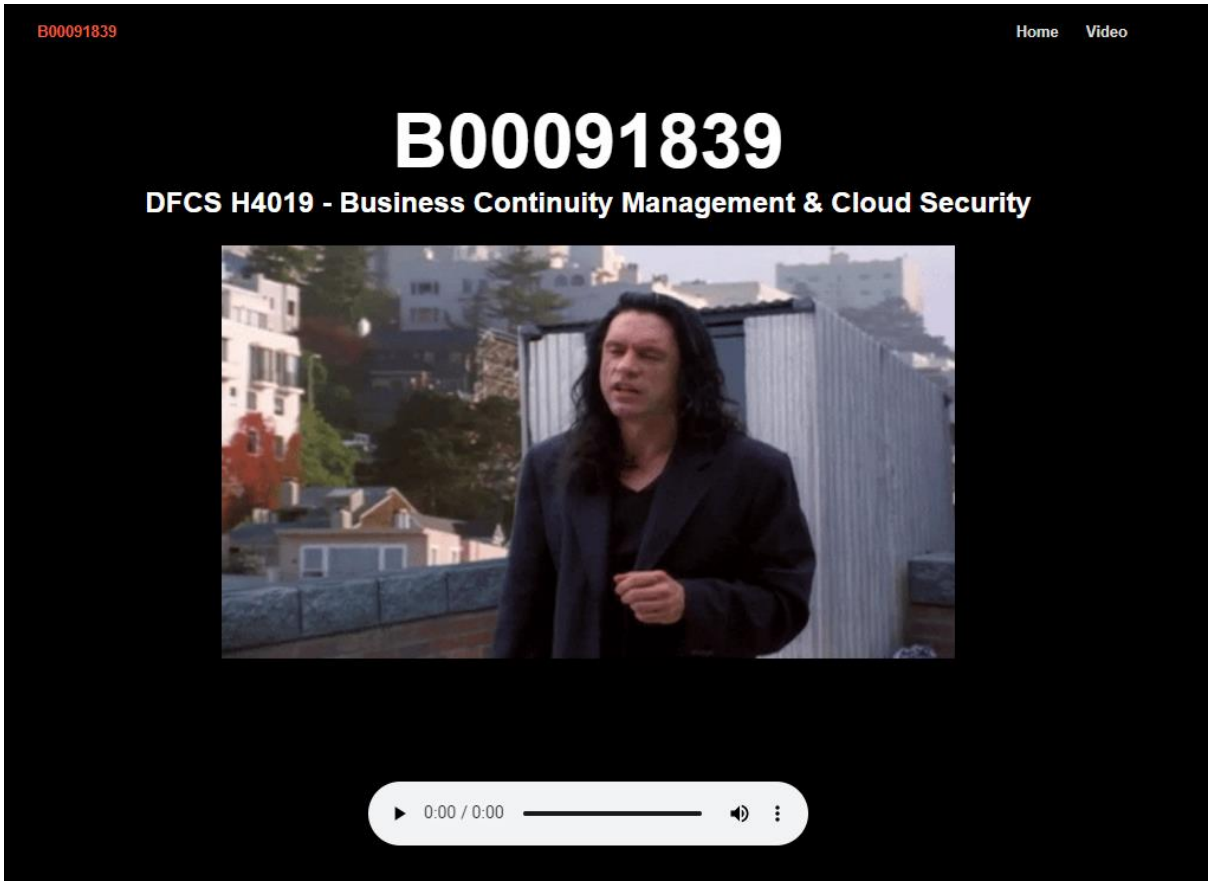
For the purpose of the webserver I decided to use apache2 as this is a service that I know well and can easily be installed and configured on the WebServer as can be seen bellow with the command **sudo apt-get install apache2**

```
ubuntu@ip-172-31-21-67:~$ sudo apt-get install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3
  libaprutil1-ldap liblua5.1-0 ssl-cert
Suggested packages:
  www-browser apache2-doc apache2-suexec-pristine | apache2-suexec-custom openssl-blacklist
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.1-0 ssl-cert
0 upgraded, 10 newly installed, 0 to remove and 40 not upgraded.
Need to get 1,562 kB of archives.
After this operation, 6,438 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

When the Apache2 server is configured and installed, the next step was to upload the website to the Webserver these files are located /var/www/html

```
ubuntu@ip-172-31-21-67:/var/www/html$ ls
a4R06RZ_460sv.mp4  index.html            Memes.html        skrollr.min.js  Website.css
images             jquery-1.12.4.min.js  oh-hi-mark.mp3    Video.css
index.css          Memes.css             scrollspy.min.js  Video.html
ubuntu@ip-172-31-21-67:/var/www/html$
```

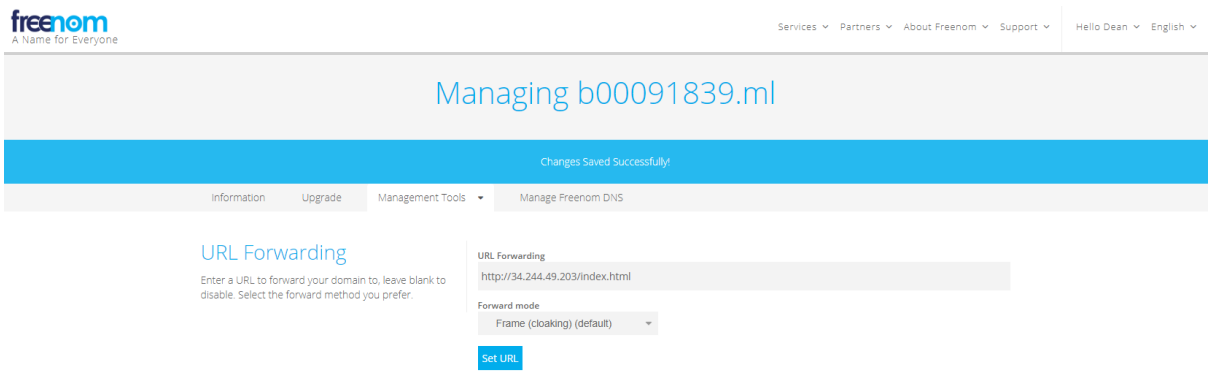
Checking the open ports on at the IP address we can that the website is up and working and can be requested by any user who requests it.



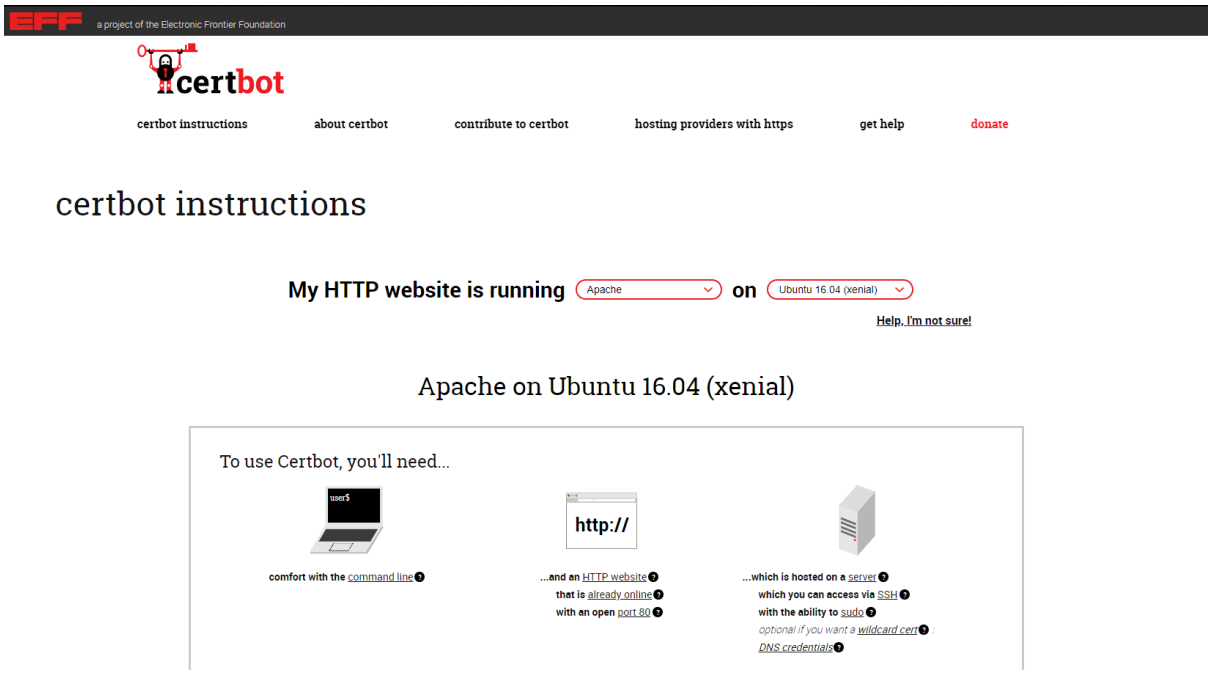
The simple code used to build this website can be seen bellow it uses simple HTML, a simple CSS file and some simple JavaScript, this all can be seen in the code section bellow under the Website section

The next step was to get a domain, for this we can use Dot.TK a company offering free domain names and URL forwarding, setting up is simple and straight forward, for the project I choose to use my student number and .ml domain.

The address is as follows www.b00091839.ml. In the bellow image you can see the configuration of URL cloaking to hide the IP address and redirect all traffic to the domain to the website address.



The next step was to configure SSL on the system; the simplest approach was to use Certbot and Lets Encrypt to install a free certificate.



Using certbot is easy as running the script with the command show in the above photo; we can see the console output and details in the image bellow

```
ubuntu@ip-172-31-21-67: /
Setting up python3-certbot-apache (0.31.0-1+ubuntu16.04.1+certbot+1) ...
Setting up python-certbot-apache (0.31.0-1+ubuntu16.04.1+certbot+1) ...
Setting up python3-icu (1.9.2-2build1) ...
Processing triggers for libc-bin (2.23-0ubuntu11) ...
ubuntu@ip-172-31-21-67:/$ sudo certbot --apache
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator apache, Installer apache
Enter email address (used for urgent renewal and security notices) (Enter 'c' to
cancel): b00091839@mytudublin.ie
Starting new HTTPS connection (1): acme-v02.api.letsencrypt.org

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf. You must
agree in order to register with the ACME server at
https://acme-v02.api.letsencrypt.org/directory
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(A)gree/(C)ancel: A

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Would you be willing to share your email address with the Electronic Frontier
Foundation, a founding partner of the Let's Encrypt project and the non-profit
organization that develops Certbot? We'd like to send you email about our work
encrypting the web, EFF news, campaigns, and ways to support digital freedom.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: Y
Starting new HTTPS connection (1): supporters.eff.org
No names were found in your configuration files. Please enter in your domain
name(s) (comma and/or space separated)  (Enter 'c' to cancel): b00091839.ml
Obtaining a new certificate
Performing the following challenges:
http-01 challenge for b00091839.ml
Enabled Apache rewrite module
Waiting for verification...
```

Using certbot caused issues with dot.tk's DNS service causing the certification system to fail every time, when attempting to get a certificate. When other student where asked they were found to be having similar issues with the setup process of install a certificate with dot.tk. The issue can be seen in the following console output

```
IMPORTANT NOTES:
 - The following errors were reported by the server:

   Domain: b00091839.ml
   Type:   connection
   Detail: Fetching
   http://b00091839.ml/.well-known/acme-challenge/sev3abe0sz5SE_k7Lwsq
   Timeout after connect (your server may be slow or overloaded)

   To fix these errors, please make sure that your domain name was
   entered correctly and the DNS A/AAAA record(s) for that domain
   contain(s) the right IP address. Additionally, please check that
   your computer has a publicly routable IP address and that no
   firewalls are preventing the server from communicating with the
   client. If you're using the webroot plugin, you should also verify
   that you are serving files from the webroot path you provided.
 - Your account credentials have been saved in your Certbot
   configuration directory at /etc/letsencrypt. You should make a
   secure backup of this folder now. This configuration directory will
   also contain certificates and private keys obtained by Certbot so
   making regular backups of this folder is ideal.
```

This issue is with the way dot.tk forwards the domain to. The way in which it works is that it mirrors the webpage to the URL, but let's encrypt needs to see the directories of the website but that is unavailable with dot.tk

To repair this issue, I made the decision to set up a Self-Signed SSL using OpenSSL on the system. The following commands where used to achieve this
To generate a SSL certificate
openssl genrsa -des3 -passout pass:x -out CA.key 2048

The next was to move it to the location
sudo mkdir /etc/httpd/httpscertificate

This will extract the private key to the https certifcates
openssl rsa -passin pass:x -in ca.key –out /etc/httpd/httpscertificate/cloudca.key

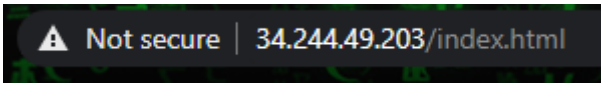
This will create a CSR file "Certifcate Signing Request"
openssl req -new -key /etc/httpd/httpscertificate/ cloudca.key -out /etc/httpd/httpscertificate/cloudsecCA.csr
The above command allows you to enter personal information that will allow you to identity the certificate

The following command will be used to create the .crt file from the CSR file
openssl x509 -req -days 365 -in /etc/httpd/httpscertificate/cloudsecCA.csr -signkey /etc/httpd/httpscertificate/cloudca.key -out /etc/httpd/httpscertificate/singedcloudsecca.crt

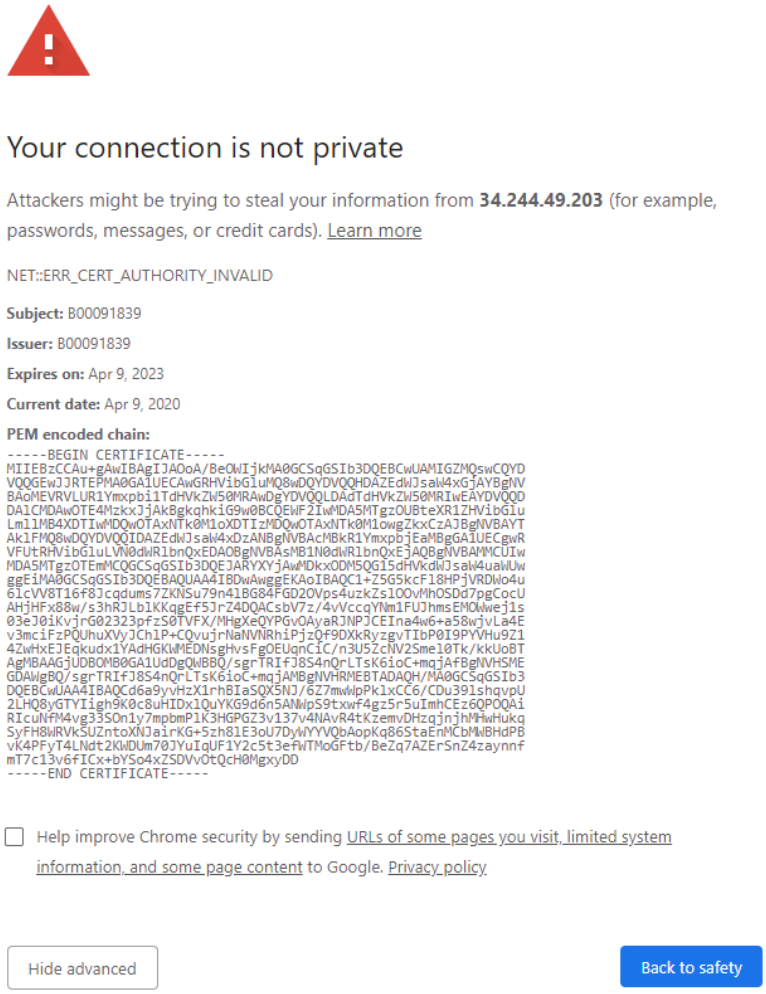
The following commands are used to get Apache to use the certificate
sudo yum install mod_ssl
sudo nano /etc/httpd/conf.d/ssl.conf
ADD THE FOLLOWING
SSLCertificateFile
SSL CertificateKeyFile
sudo apachectl restart

The certificate should be accepted now

The certificate is installed and usable, but due to it not being hosted by a Trusted Certificate Authority it is not trusted and there for comes back as Not Secure as can be seen bellow.



Viewing the certificate that is hosted on the website we can see the custom certificate installed on the system, you can also see here the browser does not trust the certificate, and the reasoning is for this is marked as **NET::ERR_CERT_AUTHORITY_INVALID**, a quick google search of this error shows that this is due to it not being a certificate issued by a trusted source.



In the above you can see the issue as discussed before, the option to get around this would be to use a paid domain name with a DNS service capable of handling a certificate being used on their system. Bellow you can see the error output from certbot as it tries to add a certificate to the domain b00091839.ml

Checking the **/var/log** folder we can see that the system is logging interactions with the server which includes login attempts which can be used for tracking attack attempts.

```
ubuntu@ip-172-31-21-67:/var/log$ ls
alternatives.log  apache2  auth.log  cloud-init.log         dist-upgrade  fsck      lastlog      lxd     syslog.1            wtmp
amazon            apt      btmp      cloud-init-output.log  dpkg.log      kern.log  letsencrypt  syslog  unattended-upgrades
ubuntu@ip-172-31-21-67:/var/log$
```

Connection Command

The bellow command will allow for connection to the WebServer via SSHServer using the private key, command uses the certificate 2nd system to connect to the webserver.

ssh ubuntu@ec2-52-208-12-75.eu-west-1.compute.amazonaws.com -i "C:\Users\Dean\Documents\College 2020 Documents\Cloud Security CA\CIS_Server_1.pem" -t ssh -i CIS_Server_1.pem ubuntu@ec2-34-244-49-203.eu-west-1.compute.amazonaws.com

Brakedown of the above command
ssh = standard command for using ssh

-i = this argument allows for connection from a key file

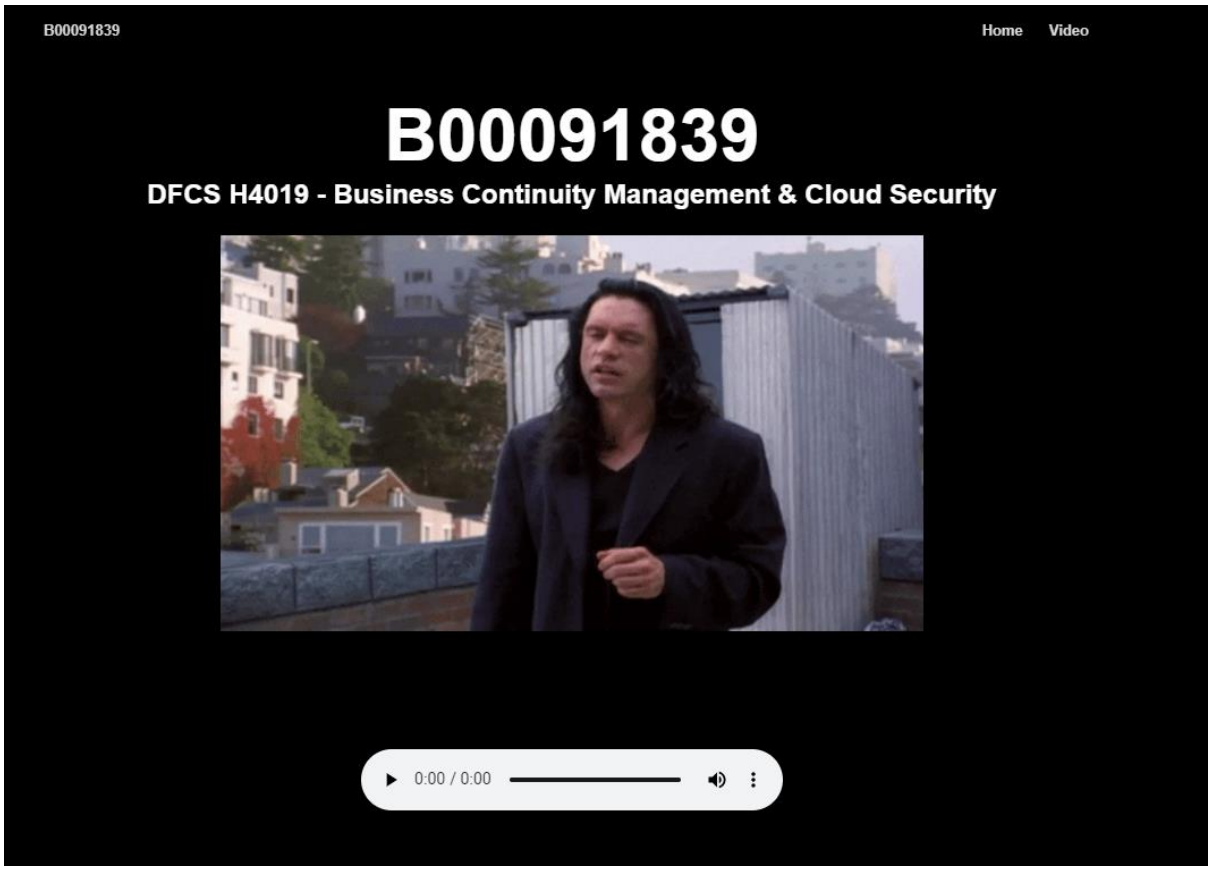
-R = This is the port bind argument and it works [YOUR PORT]:[DESTINATION]:[THEIR PORT] this allows for a connection to be remote

ubuntu@ec2-52-208-12-75.eu-west-1.compute.amazonaws.com = this is the address for the SSH server to be connected to remotely

ubuntu@ec2-34-244-49-203.eu-west-1.compute.amazonaws.com = this is the address for the webserver at which we want to connect to

The Website

The was no requirement for specifications of the website so I decided to throw a simple website together that will show the website is up and working, the view of this webpage and the code for the webpage can be found bellow.



```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>B00091839</title>
<meta name="author" content="Dean O'Neill B00091839">
<link href="Website.css" rel="stylesheet">
<link href="index.css" rel="stylesheet">
<script src="jquery-1.12.4.min.js"></script>
<script src="skrollr.min.js"></script>
<script src="scrollspy.min.js"></script>
<script>
$(document).ready(function()
{
  function skrollrInit()
  {
    skrollr.init({forceHeight: false, mobileCheck: function() { return false; }, smoothScrolling: false});
  }
  skrollrInit();
});
</script>
</head>
<body data-spy="scroll">
<div id="container">
<div id="wb_Image1" style="position:absolute;left:175px;top:205px;width:620px;height:349px;z-index:2;">
<img src="images/giphasdasdasdagagasfy.gif" id="Image1" alt=""></div>
<div id="wb_Text1" style="position:absolute;left:94px;top:79px;width:782px;height:98px;text-align:center;z-
index:3;">
<span style="color:#FFFFFF;font-family:Arial;font-
size:64px;"><strong>B00091839<br></strong></span><span style="color:#FFFFFF;font-family:Arial;font-
size:24px;"><strong>DFCS H4019 - Business Continuity Management & Cloud Security</strong></span></div>
<div id="wb_MediaPlayer1" style="position:absolute;left:299px;top:612px;width:372px;height:100px;z-
index:4;">
<audio src="oh-hi-mark.mp3" id="MediaPlayer1" autoplay controls preload="auto">
</audio>
</div>
<div id="PageHeader1" style="position:fixed;overflow:hidden;text-
align:center;left:0;top:0;right:0;height:50px;z-index:7777;" data-top="background:rgba(33, 33, 33, 0.0);" data-
-50-top="background:rgba(33, 33, 33, 1.0);">
<div id="PageHeader1_Container" style="width:970px;position:relative;margin-left:auto;margin-
right:auto;text-align:left;">
<div id="wb_CssMenu3" style="position:absolute;left:607px;top:5px;width:339px;height:40px;z-index:0;">
<ul role="menubar">
<li class="firstmain"><a role="menuitem" href="./index.html" target="_self">Home</a>
</li>
<li><a role="menuitem" href="./Video.html" target="_self">Video</a>
</li>
</ul>
</div>
<div id="wb_CssMenu4" style="position:absolute;left:8px;top:5px;width:161px;height:41px;z-index:1;">
<ul role="menubar">
<li class="firstmain"><a role="menuitem" href="./index.html" target="_self">B00091839</a>
</li>
</ul>
</div>
</body>
</html>
```

Testing

In this section we will look at testing the system, this will be in the form of an NMAP scan to check if the ports are open, in the testing of the Webserver since the rules are set it will be tested via an SSH attempt from another system not the SSH

Testing Function

The first we test the function of the connection between the SSHserver and the Webserver, the details of this connection can be found in the section connection command section

```
C:\Users\Dean>ssh ubuntu@ec2-52-208-12-75.eu-west-1.compute.amazonaws.com -i "C:\Users\Dean\Documents\College 2020 Documents\Cloud Sec
urity CA\CIS_Server_1.pem" -t ssh -i CIS_Server_1.pem ubuntu@ec2-34-244-49-203.eu-west-1.compute.amazonaws.com
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-1101-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Kubernetes 1.18 GA is now available! See https://microk8s.io for docs or
   install it with:

     sudo snap install microk8s --channel=1.18 --classic

 * Multipass 1.1 adds proxy support for developers behind enterprise
   firewalls. Rapid prototyping for cloud operations just got easier.

     https://multipass.run/

23 packages can be updated.
0 updates are security updates.

New release '18.04.4 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


*** System restart required ***
Last login: Thu Apr  9 23:26:19 2020 from 172.31.25.203
ubuntu@ip-172-31-21-67:~$ exit
logout
Connection to ec2-34-244-49-203.eu-west-1.compute.amazonaws.com closed.
Connection to ec2-52-208-12-75.eu-west-1.compute.amazonaws.com closed.

C:\Users\Dean>
```

In the response we can see we are connected to the Webserver, and upon giving the command exit we can see the connection to both systems are no closed.

Testing of the SSHserver

The Public IP Address of the SSHserver is **52.208.12.75** the command used for testing the SSHserver was as follows **nmap -p 1-65535 -T4 -A -v 52.208.12.75**

```
Nmap Output   Ports / Hosts   Topology   Host Details   Scans

nmap -p 1-65535 -T4 -A -v 52.208.12.75                    Details

Starting Nmap 7.80 ( https://nmap.org ) at 2020-04-10 15:41 GMT Daylight Time
NSE: Loaded 151 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 15:41
Completed NSE at 15:41, 0.00s elapsed
Initiating NSE at 15:41
Completed NSE at 15:41, 0.00s elapsed
Initiating NSE at 15:41
Completed NSE at 15:41, 0.00s elapsed
Initiating Ping Scan at 15:41
Scanning 52.208.12.75 [4 ports]
Completed Ping Scan at 15:41, 3.09s elapsed (1 total hosts)
Nmap scan report for 52.208.12.75 [host down]
NSE: Script Post-scanning.
Initiating NSE at 15:41
Completed NSE at 15:41, 0.00s elapsed
Initiating NSE at 15:41
Completed NSE at 15:41, 0.00s elapsed
Initiating NSE at 15:41
Completed NSE at 15:41, 0.00s elapsed
Read data files from: C:\Program Files (x86)\Nmap
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 5.60 seconds
           Raw packets sent: 8 (304B) | Rcvd: 0 (0B)
```

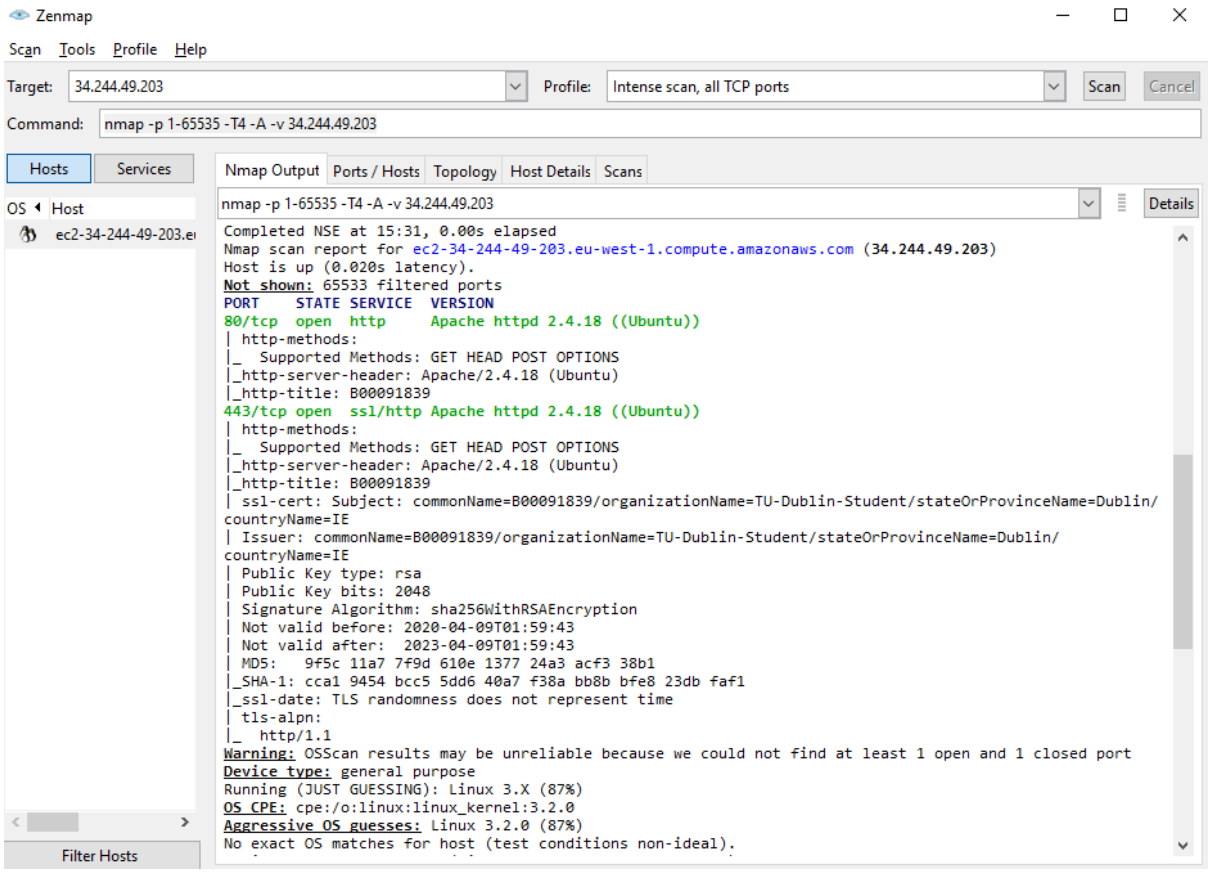
Scanning the SSHserver doesn't not show any ports open, but is still connectable from any location via SSH, this can be seen bellow as tested with a login to the server

```
C:\Users\Dean>ssh ubuntu@ec2-52-208-12-75.eu-west-1.compute.amazonaws.com -i "C:\Users\Dean\Documents\College 2020 Documents\Cloud Sec
urity CA\CIS_Server_1.pem"
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-1105-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage


16 packages can be updated.
0 updates are security updates.


Last login: Fri Apr 10 14:20:15 2020 from 95.44.6.112
ubuntu@ip-172-31-25-203:~$
```

Testing of the WebServer

The Public IP Address of the Webserver is **34.244.49.203** the command used for testing the Webserver was as follows **nmap -p 1-65535 -T4 -A -v 34.244.49.203**

The results of this scan that can be below the only open ports are port 80 and port 443; using nmap we can also see the SSL certificate information



Attempting to SSH to the Webserver, using the bellow command shows that the connection didn't work, and since we know it works as a connection for the SSHserver we can assume that this works.

```
C:\Users\Dean>ssh ubuntu@ec2-34-244-49-203.eu-west-1.compute.amazonaws.com  -i "C:\Users\Dean\Documents\College 2020 Documents\Cloud S
ecurity CA\CIS_Server_1.pem"
ssh: connect to host ec2-34-244-49-203.eu-west-1.compute.amazonaws.com port 22: Connection timed out

C:\Users\Dean>
```

Appendix

The Private Key located in the **CIS_Server_1.pem** file for testing purposes only can be seen bellow, this key can be used for connection to the systems, as needed

```
-----BEGIN RSA PRIVATE KEY-----
MXXXXXXXXXXXXXXXXXdsh+fIG7mJ57St7WMesymf9T12ss44ldWF0cUB+XXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXmF2PnfhznzddFMUVuNtbnc2Ewz5y
eT2O7yHEpE+bYXLv7h3U8+gaE2p9I9m6nexH
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xAN7ITFuRtbBPpu13rNe86ouVoGVUTjmuDEps5WwlnpQ0bLd9W0a3GdlhL32BlrJsItW+fGz4yjl
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXX+
HgITjIn+xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx+Fh
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX+5Rt6
+muhR8NJqEc8qywGIAvM5FVWkxYgfdG9tl1dU0fHATiI+/iWxxxxxxxxxxxxxxxxxxxxxxVOUB
-----END RSA PRIVATE KEY-----
```

Video link

As requested in the assignment, the video bellow hosted on YouTube showing logging into the system and modifying a section of the HTML code on the website