**身分認證登入頁面重導向**

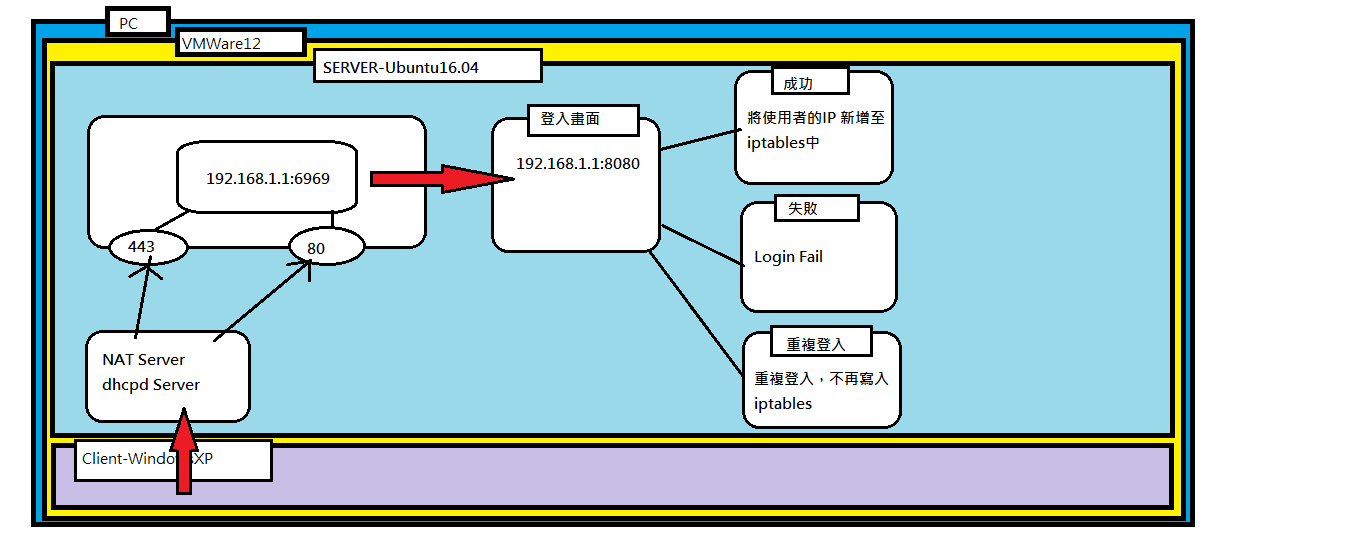
**研究方法及架構**

研究需要電腦(PC)一台、VMware12(共需要兩台虛擬機：1.Server-Ubuntu 16.04，2.Client-WindowsXP)。

本實作環境建立在 Ubuntu 16.04，在Linux系統下安裝Nat Server與dhcpd Server來分配我們想要的IP 區間給我們的Client端。接著使用登入頁面為我們的 Client做驗證，控制所有使用者的連線，如果尚未登入，所有的連線便會重導向登入頁面，如果成功了將可以正常使用網路。

假設場景：在區域網路下有大量的使用者，故使用 NAT Server+ DHCP Server來給我們的用戶端做使用，然後他們所有流量會經由這個 NAT Server根據 iptables規則處理，進入登入頁面的伺服器，並且進一步驗證使用者資訊，若是成功將可以正常使用網路。

**研究架構圖**



**實作環境**

在 Server端我們使用 DHCP Server及 NAT 服務來將我們的 IP分配給 Client端，後續分成以下三個功能建立我們的實驗環境：  
※登入頁面

※驗證身分

※重導向

一、在 VMWare12上建立一個虛擬環境(Ubuntu16.04)

二、DHCP Server安裝 + NAT 服務開啟

※打開 NAT功能

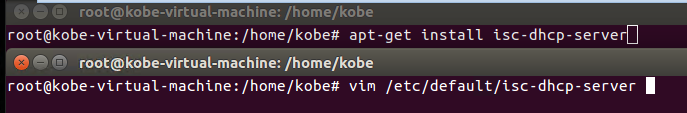
*#* vim /etc/sysctl.conf

net.ipv4.ip\_forward = 1

※DHCP Server安裝  
  
DHCP的好處是『免用戶端的設定』，在這個實驗的假設前提下，區域內電腦的數量相當多，我們需要使用 DHCP自動的分配所有的網路參數給區域網路內的任何一部電腦。

DHCP服務的配置文件為：dhcpd.conf，DHCP服務所有參數都是通過修改其文件來實現。

1.安裝 DHCP套件，並修改部分設定檔



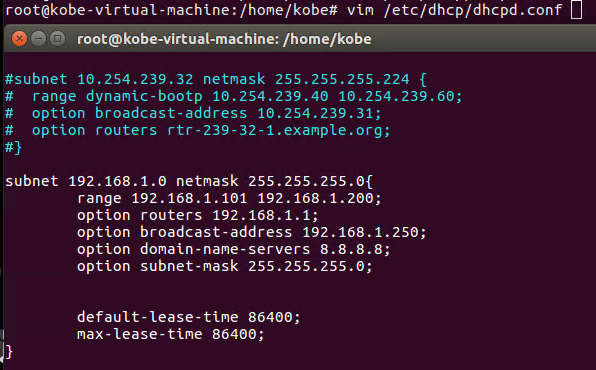
*#* apt-get install isc-dhcp-server

*#* vim /etc/default/isc-dhcp-server

修改 INTERFACES = “ens34”，為我們的第二張對內的網卡。

安裝後 dhcpd.conf是沒有做任何配置的，我們將 dhcpd.conf進行修改。

2.接下來編輯我們的 DHCP設定檔  
  
*#* vim /etc/dhcp/dhcpd.conf



subnet 192.168.1.0 netmask 255.255.255.0 {

range 192.168.1.101 192.168.1.200;

option routers 192.168.1.1;

option broadcast-address 192.168.1.250;

option domain-name-servers 8.8.8.8;

option subnet-mask 255.255.255.0;

default-lease-time 86400;

max-lease-time 86400;

}

子網段：192.168.1.0/24

IP位置配發範圍：192.168.1.101-192.168.1.200

指定預設閘道：192.168.1.1

指定廣播相關位置：192.168.1.250

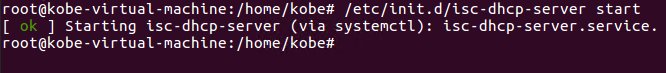
指定 DNS IP位置:8.8.8.8

指定租約更新時間：default-lease-time 86400

max-lease-time 86400

這邊我們要將租約更新時間調整為一天，我們希望每個客戶端每天都必須要重新登入，因此在 DHCP Server我們會配合iptables重設，將分配出去的 IP進行回收。

3.設定完成後啟動 DHCP Server服務



*#* /etc/init.d/isc-dhcp-server start

三、建立登入頁面

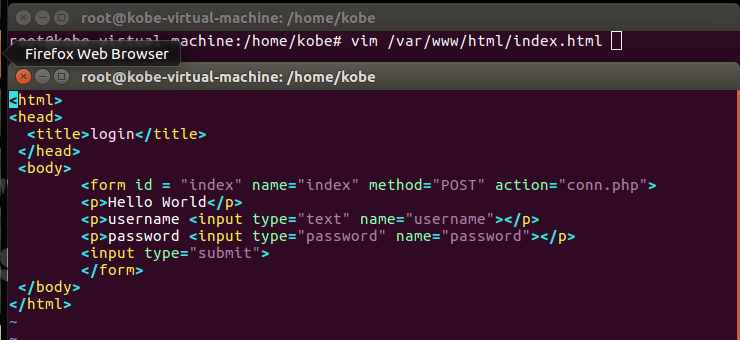
1.安裝 Apache2

我們使用 Apache2作為我們的網頁伺服器

*#* apt-get install apache2

2.進入網頁存放的資料夾 /var/www/html/ 編輯 index.html

*#* vim /var/www/html/index.html



<html>

<head>

<title>login</title>

</head>

<body>

<form id = "index" name="index" method="POST" action="conn.php">

<p>Hello World</p>

<p>username <input type="text" name="username"></p>

<p>password <input type="password" name="password"></p>

<input type="submit">

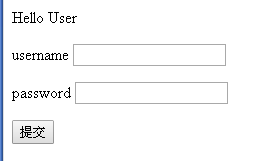
</form>

</body>

</html>

在 html檔案中，我們定義了網頁標題為 login，網頁傳輸資料方式為”POST”，action = ”conn.php”代表著嵌著 conn.php的網頁執行，處理著我們的身分驗證功能（在 四、實現驗證身分功能 部分會詳細提到）。接著定義兩個輸入格，為輸入帳號密碼的部分，並且新增一個按鈕可以將輸入的資訊提交。

我們登入網頁顯示效果如下：

  
  
3.架設資料庫

我們接下來要架設資料庫來儲存使用者的帳密，也讓伺服器能夠透過資料庫的資料做比較驗證。

*#* apt-get install mysql-server mysql-client

4.安裝 php5

PHP 適用於網頁開發並可以嵌入 HTML使用，我們要用PHP語法來寫登入後驗證的功能。

*#* apt-get install php libapache2-mod-php php-mcrypt php-mysql

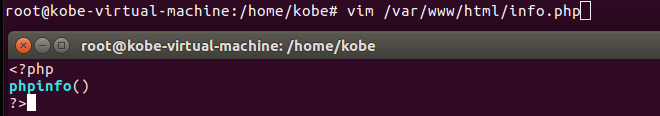
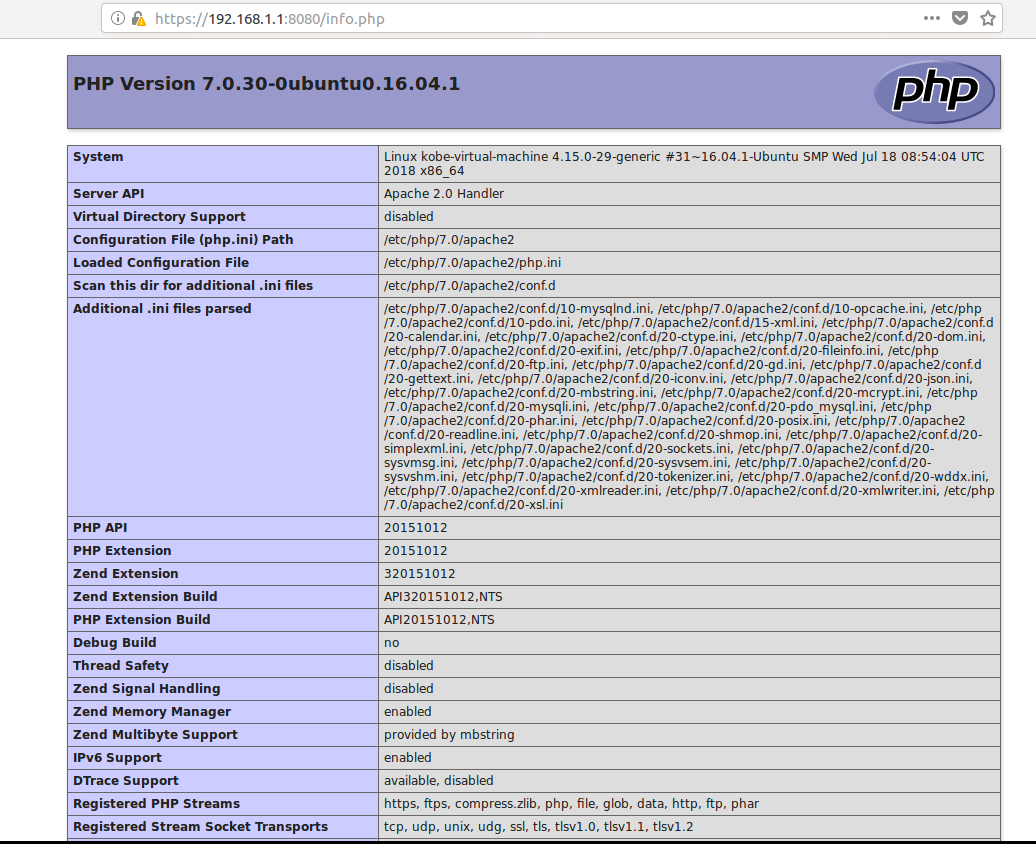
接著可以在 /var/www/html/建立一個 info.php 測試查看是否正確安裝

*#* vim /var/www/html/info.php

<?php

phpinfo()

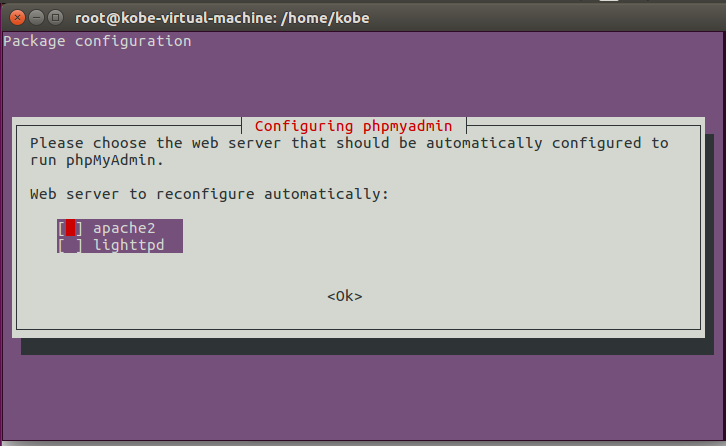
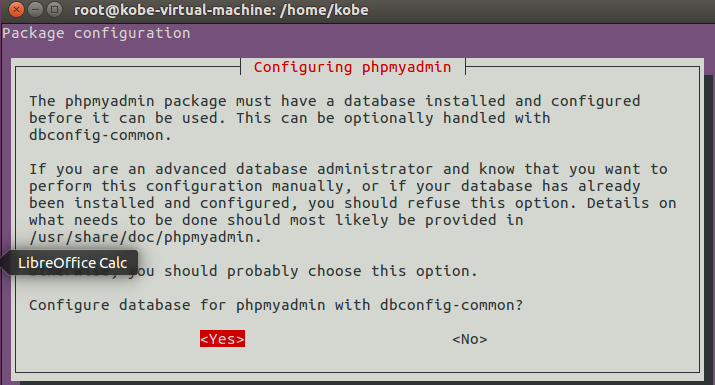
?>

  
  
5.安裝 phpmyadmin

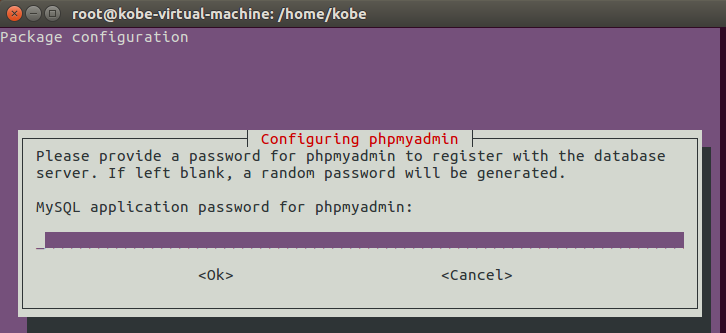
phpMyAdmin 是一個以PHP為基礎，以Web-Base方式架構在網站主機上的MySQL的資料庫管理工具，我們可用Web介面管理MySQL資料庫。

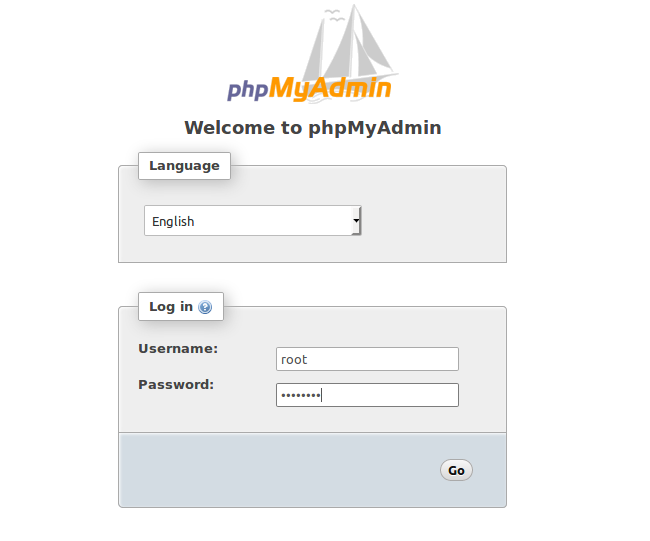
*#* apt-get install phpmyadmin

記得用空白鍵選擇我們的伺服器 apache2

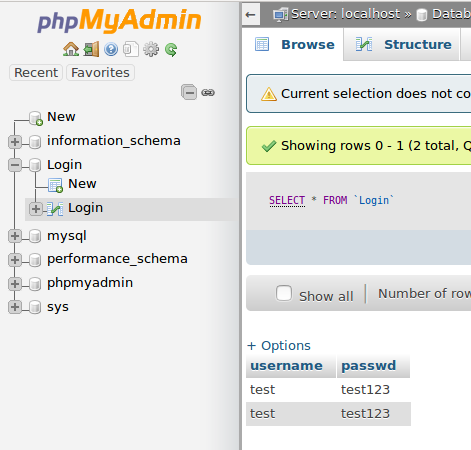
  
 選擇 <Yes>  
  


設定密碼

  
 完成之後便可以登入 phpmyadmin



新增我們的測試資料庫，測試表單，測試帳密

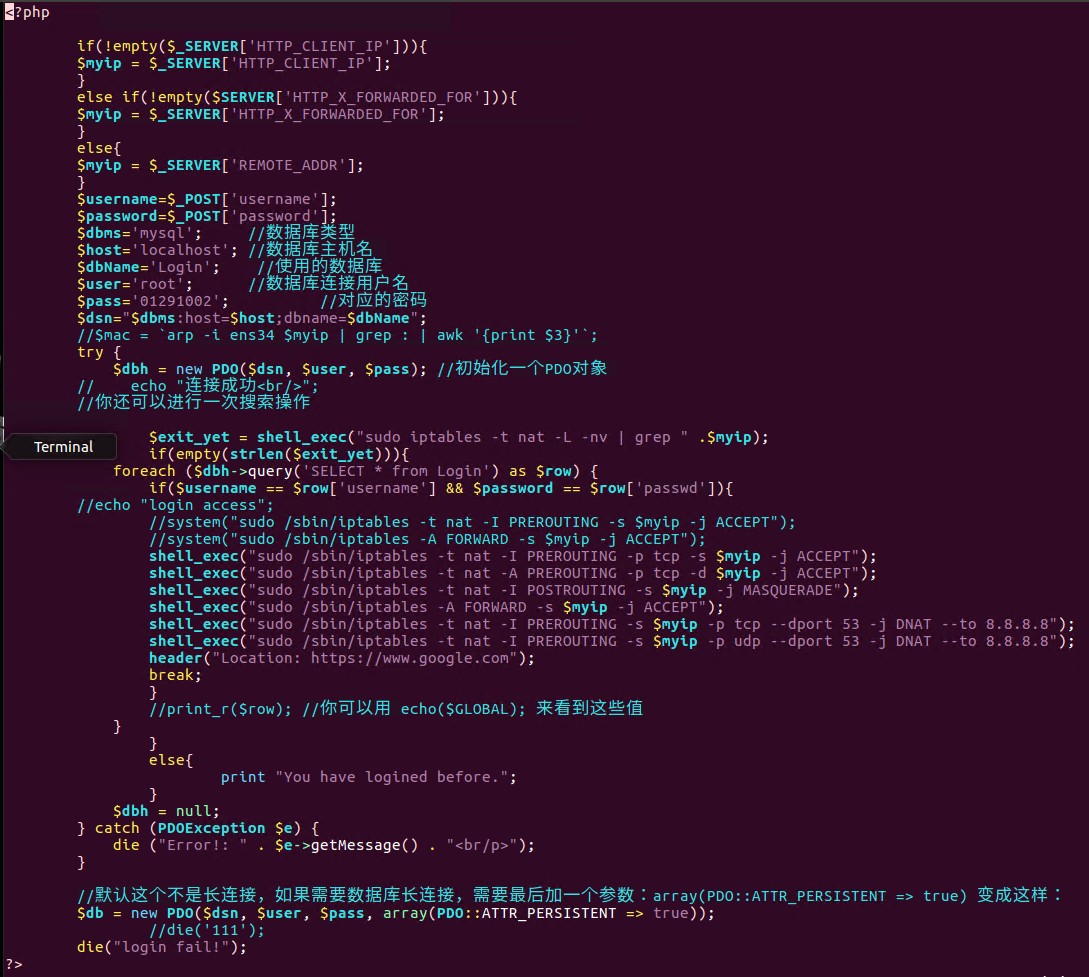


以上的設定都完成之後，我們已經做好了登入頁面的準備，接下來就進入了關鍵的驗證身分步驟。

四、實現驗證身分功能

1.在 /var/www/html中我們會新增一個 conn.php，用於執行我們的驗證身分

*#* vim /var/www/html/conn.php

  
可以看到 html用POST方法傳輸資料進 conn.php的定義

$username=$\_POST['username'];

$password=$\_POST['password'];

運用 PDO技術存取資料庫

$dbh = new PDO($dsn, $user, $pass);

針對資料庫每筆資料做搜尋查找

foreach ($dbh->query('SELECT \* from Login') as $row)

如果輸入的帳號密碼與資料庫內的資料相符，則執行網路導向的動作

if($username == $row['username'] && $password == $row['passwd']

如果輸入帳號密碼的 IP已經被新增至 iptables中，則顯示已經登入過。

$exit\_yet = shell\_exec("sudo iptables -t nat -L -nv | grep " .$myip);

if(empty(strlen($exit\_yet))){

//Do Authentication and add new iptables

}else{

print "You have logined before.";

}

驗證身分的功能已經完成，接下來要實作最重要的網路重導向的功能。  
  
五、網路重導向功能實作

這次的實驗最重要的目的：未登入的裝置，必須要經過身分認證才能使用網路。因此必須做到以下要求：

※在瀏覽器上，不管連入 http或 https網頁，都能夠正確導向登入網頁。

※除了 IP限制之外，也希望 Client在輸入網址的時候，DNS解析也會將登入網頁當作DNS 回應給 Client。

※登入過後，會將 Client的 IP新增至 iptables規則中

為了做到登入網頁可以成功接收 http與 https的請求，我們將在 Apache2伺服器開兩個port（80、443）接收流量，再將流量導入我們開啟 port8080的實際登入頁面，因此我們會在 Apache2開啟三個port的網頁窗口。而這三個 port中，我們希望 443以及8080兩個port 都是https的安全協定網頁，所以也必須要有 SSL憑證。

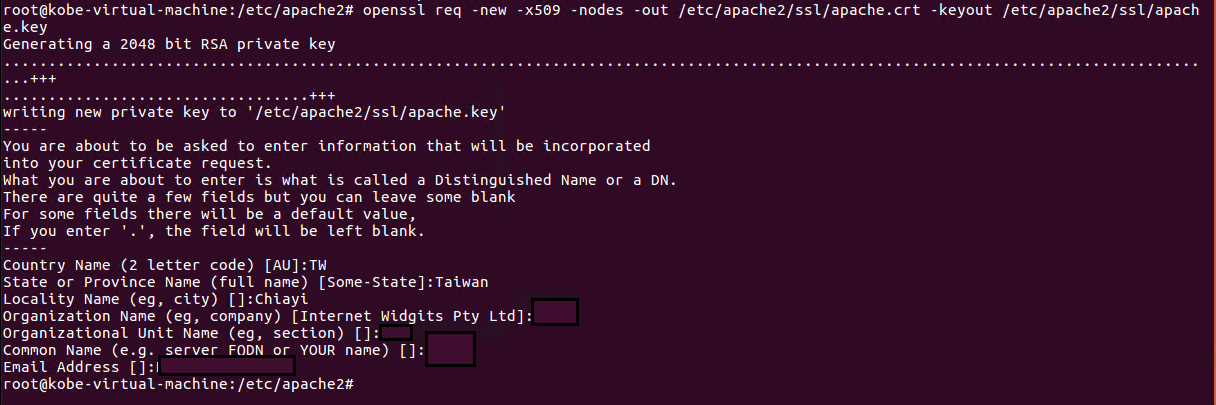
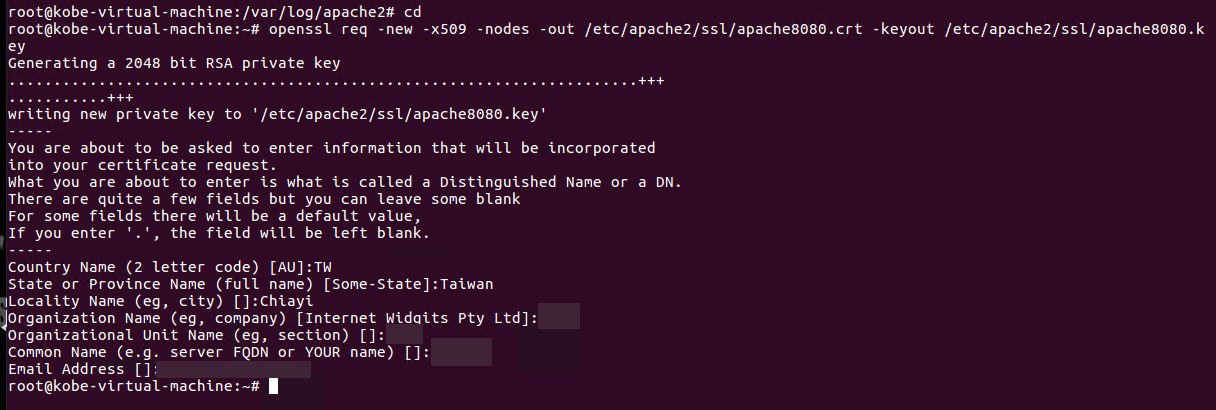
1. 首先確認產生 ssl憑證的程式已經被安裝

*#* openssl version -a

2.接著產生憑證，我們將欲 產生的憑證放在網頁資料夾內

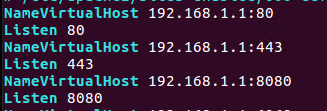
*#* mkdir /etc/apache2/ssl

*#* sudo openssl req -new -x509 -nodes -out /etc/apache2/ssl/apache.crt -keyout /etc/apache2/ssl/apache.key

  
  
 產生兩個憑證，分別用於 443port與 8080port的https安全協定網頁。

3.編輯 /etc/apache2/ports.conf，新增我們的伺服器 port

*#* vim /etc/apache2/ports.conf



NameVirtualHost 192.168.1.1:80

Listen 80

NameVirtualHost 192.168.1.1:443

Listen 443

NameVirtualHost 192.168.1.1:8080

Listen 8080

4.接著進入伺服器的網站設定新增相關資訊

*#* vim /etc/apache2/sites-enabled/000-default.conf  
  


<VirtualHost \*:80>

ServerAdmin webmaster@localhost

DocumentRoot /var/www/direct

ErrorLog ${APACHE\_LOG\_DIR}/error.log

CustomLog ${APACHE\_LOG\_DIR}/access.log combined

</VirtualHost>

<VirtualHost \*:443>

ServerAdmin webmaster@localhost

DocumentRoot /var/www/direct

ErrorLog ${APACHE\_LOG\_DIR}/error.log

CustomLog ${APACHE\_LOG\_DIR}/access.log combined

SSLEngine On

SSLCertificateFile /etc/apache2/ssl/apache.crt

SSLCertificateKeyFile /etc/apache2/ssl/apache.key

</VirtualHost>

<VirtualHost \*:8080>

ServerAdmin webmaster@localhost

DocumentRoot /var/www/html

ErrorLog ${APACHE\_LOG\_DIR}/error.log

CustomLog ${APACHE\_LOG\_DIR}/access.log combined

SSLEngine On

SSLCertificateFile /etc/apache2/ssl/apache8080.crt

SSLCertificateKeyFile /etc/apache2/ssl/apache8080.key

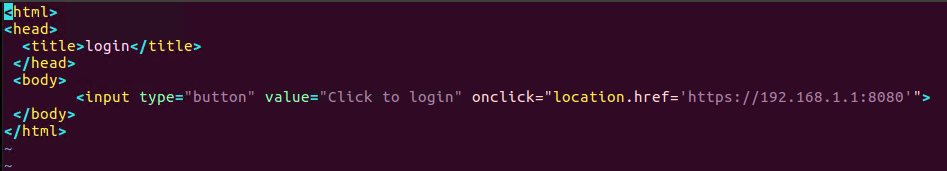
</VirtualHost>

在這個文件中，我們定義了三個port的網頁資訊，其中比較關鍵的在於DocumentRoot 的部分，這個定義為網頁的根目錄，所以只要連入這個伺服器的特定端口，都會有各自的網頁。我們希望做到的，是在 80與 443兩個port做接收流量並重導向用戶到真正的登入頁面（192.168.1.1:8080），故我們要在

/var/www/direct中建立一個 導向網頁。

5.建立導向頁面在 /var/www/direct/index.html

*#*vim /var/www/direct/index.html



<html>

<head>

<title>login</title>

</head>

<body>

<input type="button" value="Click to login" onclick="location.href='https://192.168.1.1:8080'">

</body>

</html>

確定網頁都已經新增好之後，重啟 Apache2將設定更新並套用。

6.啟用 Apache2 SSL模組

*#* sudo a2enmod ssl

*#* /etc/init.d/apache2 restart

設定了我們 http與 https能夠正常接收流量之後，我們在瀏覽器上還需要一個功能，即是 DNS request。因為身分尚未驗證通過，DNS的回應是無效的，因此我們必須要建立一個 DNS Server，回應所有的 DNS請求為登入頁面的IP，如此一來，不管是輸入 IP或是網址，都能夠正確地將用戶導向登入頁面。

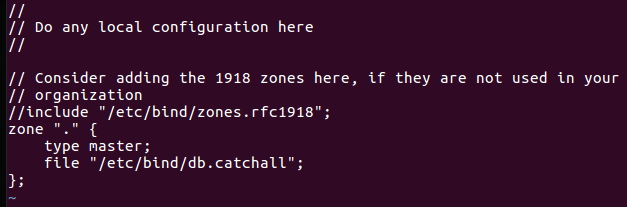
1. 首先下載 BIND 這個 DNS Server

*#* apt-get install bind9

2.定義zone，所謂zone是DNS伺服器管理時的邏輯單位

*#* cd /etc/bind/

*#* vim named.conf.local

  
zone "." {

type master;

file "/etc/bind/db.catchall";

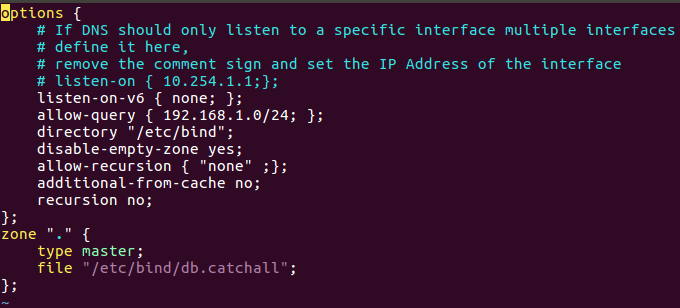
};

定義“.”，Internet DNS 最上層的位置，而 file "db.catchall"指定了其記錄檔(zone file)名稱。

3.bind 執行檔新增

Bind 的主要設定檔為 named.conf，因此Bind執行時會去讀取此執行檔。而 named.conf的功能是由於每部 name server 主要是負責某些特定 zone 中有關機器的 domain name，IP 的查詢、127.0.0.1 本機 IP 的查詢，除此之外，name server 也需要知道要怎樣去找 root name server。因此 named.conf 便會指出該 name server 管轄的網域名稱及相關檔案存取的位置及名稱。

*#* vim named.conf



options {

listen-on-v6 { none; };

allow-query { 192.168.1.0/24; };

directory "/etc/bind";

disable-empty-zone yes;

allow-recursion { "none"; };

additional-from-cache no;

recursion no;

};

zone "." {

type master;

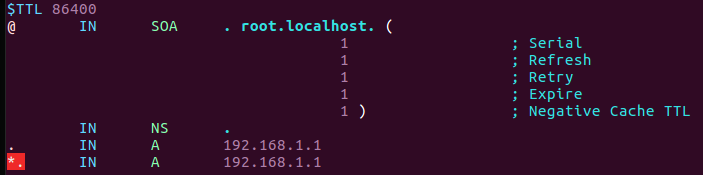
file "/etc/bind/db.catchall";

};

在文件中我們設定只有子網域 192.168.1.0/24的 IP才會被 bind接受查詢。定義“.”，Internet DNS 最上層的位置，而 file "db.catchall"指定了其記錄檔(zone file)名稱。記錄檔都存放在 /etc/bind 這個目錄下面，也就是上面 options 指定的 directory。

4.新增記錄檔(zone file)

*#* vim db.catchall



$TTL 86400

@ IN SOA . root.localhost. (

1 ; Serial

1 ; Refresh

1 ; Retry

1 ; Expire

1 ) ; Negative Cache TTL

IN NS .

. IN A 192.168.1.1

\*. IN A 192.168.1.1

區域檔案是由多個記錄組成的，每一個記錄稱為資源記錄(Resource Record，簡稱RR)。當在設定DNS名稱解析、反向解析及其他的管理目的時，需要使用不同類型的RR。

SOA

Start Of Authority，這種 record 放在 zone file 一開始的地方，每一個記錄檔只能有一個 SOA，而且一定是檔案中第一個“記錄”，它描述這個 zone 負責的 name server，version number…等資料，以及當 slave server 要備份這個 zone 時的一些參數。 緊接在 SOA 後面指定了這個區域的授權主機和管理者的信箱，然後是用括號括起來的 5 個參數

※Serial：代表這個 zone file 的版本，每當 zone file 內容有變動，name server 管理者就應該增加這個號碼，因為 slave 會將這個號碼與其 copy 的那份比對以便決定是否要再 copy 一次 (即進行 zone transfer)。

※Refresh：slave server 每隔這段時間(秒)，就會檢查 master server 上的 serial number。

※Retry：當 slave server 無法和 master 進行 serial check時，要每隔幾秒 retry 一次。

※Expire：當時間超過 Expire 所定的秒數而 slave server 都無法和 master 取得連絡，那麼 slave 會刪除自己的這份 copy。

※Negative：代表這個 zone file 中所有 record 的內定的 TTL 值，也就是其它的 DNS server cache 這筆 record 時，最長不應該超過這個時間。

NS

name server，用來指定操作的DNS伺服器主機名稱，需注意的是不可以IP位址表示。

A

address，將DNS網域名稱對應到IPv4的32位元位址。

5.啟動 BIND

*#* chown -R bind:bind /etc/bind

*#* /etc/init.d/bind9 start

6.註解掉 dnsmasq

*#* vim /etc/NetworkManager/NetworkManager.conf

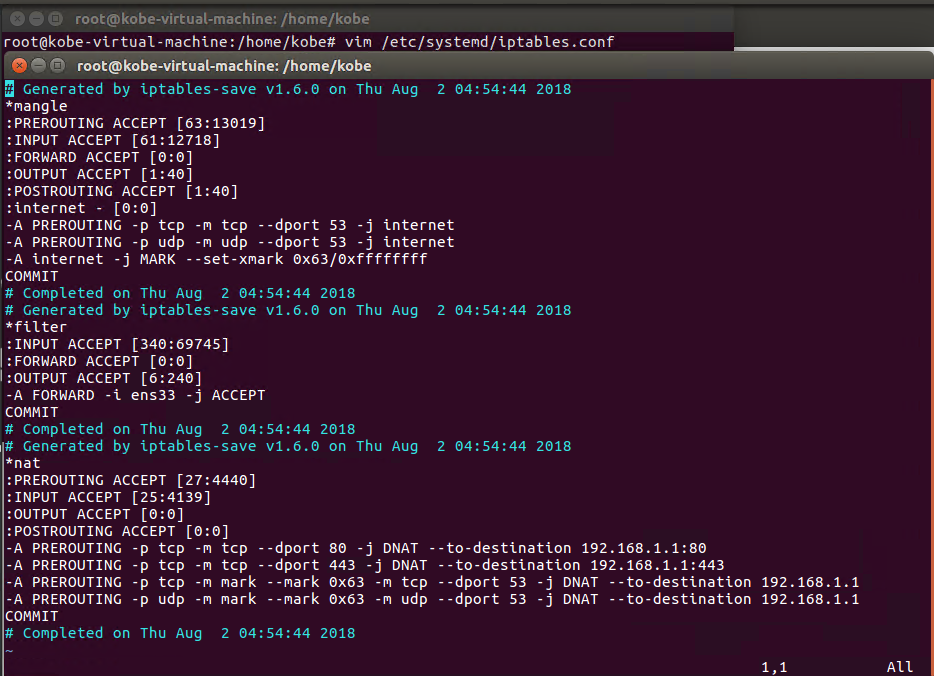
dns=dnsmasq -> #dns=dnsmasq

為了避免 DNS查找時經由系統預設dnsmasq被啟用，故註解掉。

7.重新啟動Network-manager

*#* service network-manager restart

四、防火牆設定



*#* vim /etc/systemd/iptables.conf

預設的狀況下，防火牆並不會自啟動，因此我們將檔案存在 /etc/systemd/iptables.conf中，將規則寫入檔案中，並利用指令將檔案中的防火牆規則讀入並應用。

預設的狀況下，iptables至少有三個表格，其中我們最需要用到的是管理後端主機的 nat表，主要在進行來源與目的之 IP或 port的轉換。以及mangle表，主要功能是根據規則修改數據包的一些標誌位，以便其他規則或程序可以利用這種標誌對數據包進行過濾或策略路由。為了給特定的數據包做上標誌，需要使用mangle表，由於需要在路由選擇前做標誌，因此應該使用PREROUTING鏈

。

※PREROUTING：在進行路由判斷之前所要進行的規則。

※POSTROUTING：在進行路由判斷之後所要進行的規則。

iptables 參數說明：

-A：新增一條規則，該規則增加在最後面。

-i：設定『封包進入』的網路卡介面

-o：設定『封包流出』的網路卡介面

-t：指定規則在預作用的那個規則表上

-p：封包的協定

-s：來源封包的 IP或 Network(網域)

--sport：來源封包的 port號碼

-d：目標主機的 IP或 Network(網域)

--dport：目標主機的 port號碼

-j：動作，可以接以下幾種動作

ACCEPT：接受該封包

DROP：丟棄封包

LOG：將封包的資訊記錄下來

MASQUERADE：地址偽裝，作用是從服務器的網卡上，自動獲取當前ip地址來做NAT

底下則運用了上述規則，定義了我們所需要的 iptables設定：

1. 允許外網網卡的通過

*#* iptables -A FORWARD -i ens33 -j ACCEPT

1. 將 80與 443port的流量導向到指定的伺服器

*#*iptables -A PREROUTING -p tcp -m tcp --dport 80 -j DNAT --to-destination 192.168.1.1:80

*#*iptables -A PREROUTING -p tcp -m tcp --dport 443 -j DNAT --to-destination 192.168.1.1:443

1. 將 DNS解析過程結果導向指定的伺服器

*#* iptables -N internet -t mangle

*#* iptables -t nat -A PREROUTING -p tcp -m mark --mark 0x63 -m tcp --dport 53 -j DNAT --to 192.168.1.1

*#* iptables -t nat -A PREROUTING -p udp -m mark --mark 0x63 -m udp --dport 53 -j DNAT --to 192.168.1.1

*#* iptables -t mangle -A PREROUTING -p tcp -m tcp --dport 53 -j internet

*#* iptables -t mangle -A PREROUTING -p udp -m udp --dport 53 -j internet

*#* iptables -t mangle -A internet -j MARK --set-xmark 0x63/0xffffffff

1. 將登入確認的 IP加入防火牆規則，寫在登入網頁的php檔案中，如果驗證成功則執行以下防火牆規則。

sudo /sbin/iptables -t nat -I PREROUTING -p tcp -s $myip -j ACCEPT

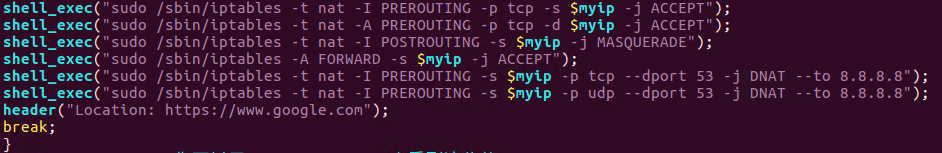
sudo /sbin/iptables -t nat -A PREROUTING -p tcp -d $myip -j ACCEPT

sudo /sbin/iptables -t nat -I POSTROUTING -s $myip -j MASQUERADE

sudo /sbin/iptables -A FORWARD -s $myip -j ACCEPT

sudo /sbin/iptables -t nat -I PREROUTING -s $myip -p tcp --dport 53 -j DNAT --to 8.8.8.8

sudo /sbin/iptables -t nat -I PREROUTING -s $myip -p udp --dport 53 -j DNAT --to 8.8.8.8



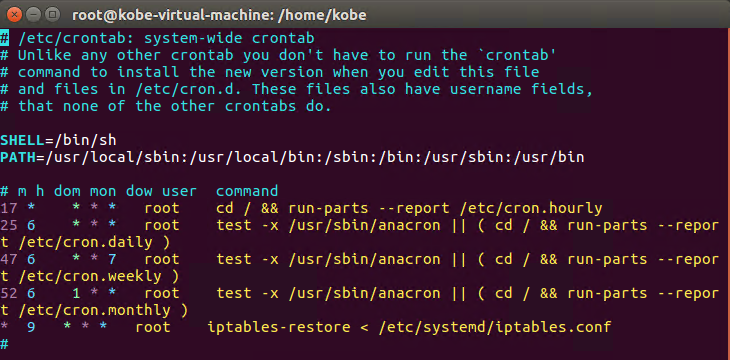
*#* iptables-save > /etc/systemd/iptables.conf

※這指令很重要，這是永久保存我們防火牆規則，並且在重開機後不會重設，而我們也希望所有的 Client每天都必須要重新登入，因此我們必須要將 iptables規則重設為原本的狀態，故我們需要一個定時執行程式。

我們使用 crontab來做到每日執行的排程

crontab 是 Linux 內建的機制，可以根據設置的時間參數來執行例行性的工作排程。顯示出前五項參數應該要帶進去的數字。依序是分鐘, 小時, 日期, 月份, 星期, command

#vim /etc/crontab



\* 9 \* \* \* root iptables-restore < /etc/systemd/iptables.conf

表示每天的9點鐘會執行後面的指令：藉由讀取 /etc/systemd/iptables.conf來回復最原始的防火牆規則。

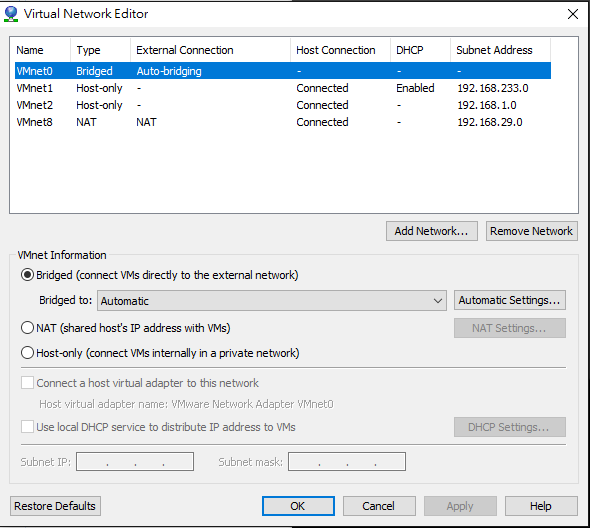
**實驗驗證**

在這邊我們需要驗證以下兩件事：

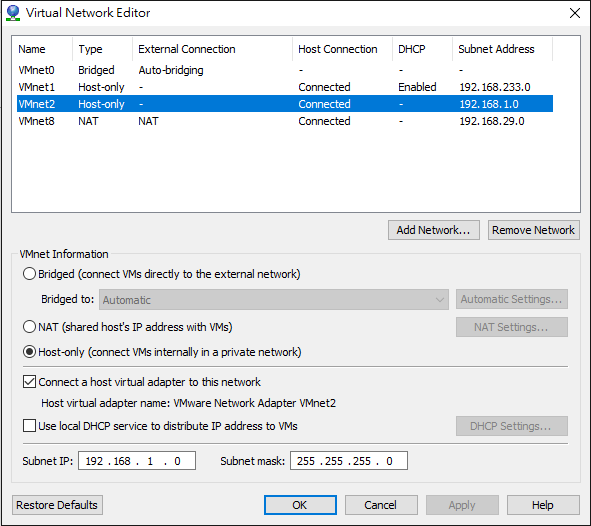
※驗證 Client 端在登入後是否可以正常連線

我們在虛擬機內有兩個作業系統，一為我們的 Ubuntu16.04，作為我們的 Server端，另一則為我們的 Windows，作為我們的 Client端。我們必須將兩者連線，才能達到實驗目標。在我們的虛擬機當中，我們可以透過 Virtual Network Editor來新增我們的虛擬網路，達到我們的連線目的，透過設定將兩台虛擬主機的網卡可以連線在一起。

以下根據我們的需求做設定：

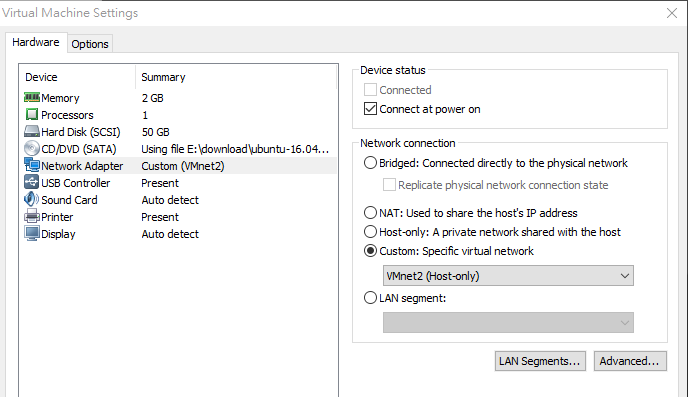
1. 打開 VMWare12-> Edit-> Virtual Network Editor-> Change Setting

2.Add Network-> Select a Network to add：VMnet2

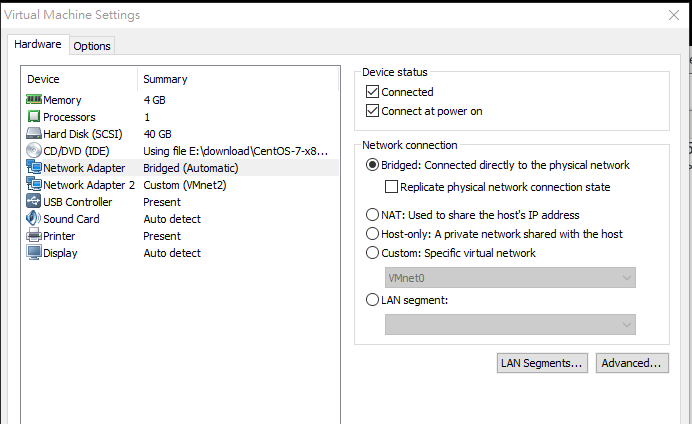


※將 Use local DHCP service to distribute IP address to VMs取消勾選

※將我們的 Subnet IP根據我們 DHCP Server 所訂定的規則，將子網域定在192.168.1.0/24的區間

3.接著將我們 Windows的虛擬機網卡設成經由 VMNet2連線

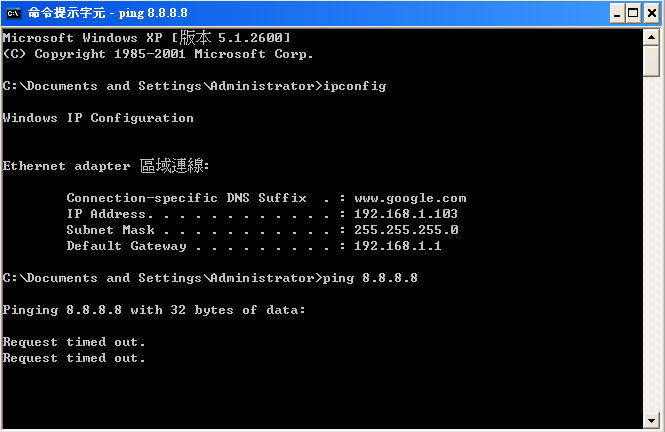
4.Ubuntu16.04 網卡會有兩張：



一張網卡對外藉由 Bridged方式拿到一個實體IP，可以透過這個網卡對外網進行連線的動作。

一張網卡對內將他設成 VMnet2連線，這張網卡即是對內所有的區域網路，接收所有區域網路的網路封包傳送。

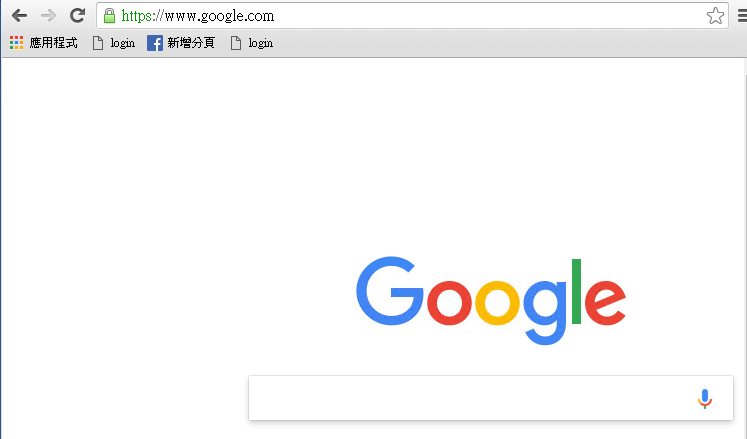
1. 進入 Windows之後，可以看到自己已經被分配到 IP但是無法上網



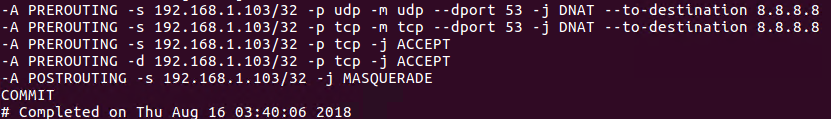
1. 經由瀏覽器不管輸入任何 IP或是網址都會正確導向登入頁面



7.輸入帳密之後，會被重導向到google的首頁

  
8.在 Server端可以透過 iptables知道這台機器的 IP已經被寫入防火牆規則

*#*/sbin/iptables-save



9.若是已經登入過，則會有警告提示

