Assignment 2

## ***part 1:***

the non trivial password we chose for the **ubuntu server root** was an md5 hash on the word chatterl
which gave us ee7e05108dff9da88a391dbeb029ed40

the **regular user** we created for the ubuntu server:
username: httpserver
pass: 5bdf74912a51c34815f11e9a3d20b609

- to get this pass we md5 hashed the word encryption which gave us that result

- we picked this password because it cant be dictionarry attacked and a brute force would take a very years(on an average cpu atl east)

the **administrator** password we created was A13Cd3|=g|-|I_|K
the longer the password is the harder it is to guess.
It has no dictionary words and is a mix of upper case and lower case.

the **windows user** we created:
username: chatterl
pass:|_0\/3_|-|/\CK1|\|G

same logic as to why we picked the administrator password

## ***part2:***

**FTP for Linux:**
ftp was already installed on Ub  untu by default.
But we needed to install vsftpd
after installing that we set anonymous ftp to on
and forwarding to 1 so we can send traffic to windows net.ipv4.ip_forward = 1
and put the ftpcontents.pdf file into /srv/ftp

**HTTP for linux server:**
install apache2 http server from http://httpd.apache.org/download.cgi#apache24

install the following dependencies:

install and configure pcre 8.42 from https://ftp.pcre.org/pub/pcre/:

```
 ./configure --prefix=/usr  \
    --docdir=/usr/share/doc/pcre-8.43 \
    --enable-unicode-properties  \
    --enable-pcre16  \
    --enable-pcre32  \
    --enable-pcregrep-libz  \
    --enable-pcregrep-libbz2  \
    --enable-pcretest-libreadline \
    --disable-static  &&
    make  &&
    make install
```

install and configure apr  and apr-util 1.6.5 from http://apr.apache.org/download.cgi

```
    ./configure && make && make install
```

configure apache2:

```
./configure --prefix=/home/httpserver/apache2 --with-included-apr --with-pcre=/usr/local/pcre
&& make && make install
```

Create webcontent.html in server:

```
cd /home/httpserver/apache2/htdocs
nano webcontent.html
```

Configure server:

```
cd /home/httpserver/apache2/conf
/home/httpserver/apache2
sudo nano http.conf
# set document to point toward webcontent.html
set ServerRoot "/home/httpserver/apache2"
set DocumentRoot "/home/httpserver/apache2/htdocs"
set DirectoryIndex webcontent.html
change 'Listen 80' to localhost IP 'Listen 127.0.0.1:80'
```

Allow access only to httpserver account:

```
sudo chmod a+r webcontent.html
sudo chown httpserver webcontent.html
```

## FTP for windows:

To get ftp working on windows we followed the steps at https://vpsie.com/knowledge-base/how-to-setup-ftp-server-users-on-windows-2012-r2/
which in summary made us install a webserver and ftp server role service. Create a ftp user. Configure the ftp global iis settings, creating an ftp site and lastly setting up the firewall

we put the pdf file in the file explore tab for the ftp directory.

HTTP for windows:
we followed the steps at https://thesolving.com/server-room/how-to-install-and-configure-iis-on-windows-server-2012-r2/

which in summary creates an IIS and configures it.

We put the html file in the file explorer tab for the IIS

## ***PART3:***

Since the the rules get executed in order, we introduced the drop rules first followed by the accept rules. Some things to account for were that TCP connections are bidirectional and that FTP has two modes, active and passive.
The bidirectional TCP connections were handled by adding rules to both INPUT and OUTPUT for example looking at the FTP rule.
We would first set up rules for the INPUT chain to drop or allow a request to access FTP services from certain IP  10.229.100.51, 10.229.51.* ( IP address such as 10.229.51.* can be handled using “/24” notation to just match the first 24 bits, “/8” to match the first 8 bits, “16” to match the first 16 bits), then followed by setting up rules in the OUTPUT chain to give back an acknowledgement to the client (if our rules allow it) trying to establish a connection. FTP needed port 21 to establish connection, port 20 to make data transfer. Passive mode has been handled by setting up rules from port 1024 to 65535 which can be done by setting port to “1024:” .

we also added the rule sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT to account for our response time taking to long. Without it our ssh connection was timing out.

To set up rules in the Windows host, we added rules in the FORWARD chain to drop packets coming from certain ip addresses and to accept ips that re in 10.229.0.0/16 addresses followed by a use in OUTPUT chain to establish bidirectional connection. We decided to add rule to the FORWARD chain since the linux

server would filter the packet depending on the rule to either drop or forward the packet to the Window's host. Outbound traffic was blocked -A FORWARD -s 10.229.52.0/24 -j DROP
Setting up rules for HTTP and FTP in linux server were done by adding rules to INPUT and OUTPUT. FOR HTTP, port 80 was used and FTP used port 20, 21 and 1024 to 65535.

SSH was done by using port 22, added a rule to the INPUT chain using destination port as 22 since the default of OUTPUT policy is DROP, we needed to add a rule in the OUTPUT table using. To allow pings, we added a rule in INPUT chain to accept icmm-type 8 and to accept icmp-type 0 to accept in the OUTPUT chain.

All hosts of the group's internal network was given complete access to the group's hosts by adding rules to INPUT, OUTPUT and FORWARD chain to accept packet from 10.229.4.0/24. The default to refuse all other inbound traffic was set by doing "-P INPUT DROP". Logging had been set up by creating a new chain named LOGGING, then sending all the other incoming messages that and writing them into the log then dropping them

links we used for this part:

https://unix.stackexchange.com/questions/93554/iptables-to-allow-incoming-ftp

https://www.digitalocean.com/community/tutorials/iptables-essentials-common-firewall-rules-and-commands

a list of our rules follows

```
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-N LOGGING
#block those specified from accessing our Ubuntu machine
-A INPUT -s 10.229.100.51/32 -p tcp -m tcp --dport 80 -m conntrack --ctstate NEW,ESTABLISHED -j DROP
-A INPUT -s 10.229.51.0/24 -p tcp -m tcp --dport 80 -m conntrack --ctstate NEW,ESTABLISHED -j DROP
-A INPUT -s 10.229.2.0/24 -p tcp -m tcp --dport 80 -m conntrack --ctstate NEW,ESTABLISHED -j DROP
-A INPUT -s 10.229.100.51/32 -p tcp -m tcp --dport 21 -m conntrack --ctstate NEW,ESTABLISHED -j DROP
-A INPUT -s 10.229.51.0/24 -p tcp -m tcp --dport 21 -m conntrack --ctstate NEW,ESTABLISHED -j DROP
-A INPUT -s 10.229.2.0/24 -p tcp -m tcp --dport 21 -m conntrack --ctstate NEW,ESTABLISHED -j DROP
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
# let those specified into our Ubuntu machine
-A INPUT -s 10.229.0.0/16 -p tcp -m tcp --dport 21 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
-A INPUT -s 10.229.0.0/16 -p tcp -m tcp --dport 20 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
```

```
-A INPUT -s 10.229.0.0/16 -p tcp -m tcp --sport 1024:65535 --dport 1024:65535 -m conntrack --ctstate ESTABLISHED -j ACCEPT

-A INPUT -s 10.229.0.0/16 -p tcp -m tcp --dport 80 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT

-A INPUT -p tcp -m tcp --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT

-A INPUT -p tcp -m tcp --sport 22 -m conntrack --ctstate ESTABLISHED -j ACCEPT

-A INPUT -p icmp -m icmp --icmp-type 8 -j ACCEPT

-A INPUT -j LOGGING

-A INPUT -s 10.229.4.0/24 -j ACCEPT

#block specified individuals form accessing our windows machine

-A FORWARD -s 10.229.100.51/32 -p tcp -m tcp --dport 80 -m conntrack --ctstate NEW,ESTABLISHED -j DROP

-A FORWARD -s 10.229.51.0/24 -p tcp -m tcp --dport 80 -m conntrack --ctstate NEW,ESTABLISHED -j DROP

-A FORWARD -s 10.229.6.0/24 -p tcp -m tcp --dport 80 -m conntrack --ctstate NEW,ESTABLISHED -j DROP

-A FORWARD -s 10.229.100.51/32 -p tcp -m tcp --dport 21 -m conntrack --ctstate NEW,ESTABLISHED -j DROP

-A FORWARD -s 10.229.51.0/24 -p tcp -m tcp --dport 21 -m conntrack --ctstate NEW,ESTABLISHED -j DROP

-A FORWARD -s 10.229.6.0/24 -p tcp -m tcp --dport 21 -m conntrack --ctstate NEW,ESTABLISHED -j DROP

#allow everyone who didn’t get dropped in

-A FORWARD -s 10.229.0.0/16 -p tcp -m tcp --dport 21 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT

-A FORWARD -s 10.229.0.0/16 -p tcp -m tcp --dport 20 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT

-A FORWARD -s 10.229.0.0/16 -p tcp -m tcp --sport 1024:65535 --dport 1024:65535 -m conntrack --ctstate ESTABLISHED -j ACCEPT

-A FORWARD -s 10.229.0.0/16 -p tcp -m tcp --dport 80 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT

-A FORWARD -s 10.229.4.0/24 -j ACCEPT

# we should no be allowd to access any servecies from 100.51 3.* and 52.*

-A FORWARD -s 10.229.100.52/32 -j DROP

-A FORWARD -s 10.229.52.0/24 -j DROP

-A FORWARD -s 10.229.3.0/24 -j DROP

#send replies back

-A OUTPUT -p icmp -m icmp --icmp-type 0 -j ACCEPT

-A OUTPUT -p tcp -m tcp --dport 21 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT

-A OUTPUT -s 10.229.0.0/16 -p tcp -m tcp --dport 20 -m conntrack --ctstate ESTABLISHED -j ACCEPT

-A OUTPUT -s 10.229.0.0/16 -p tcp -m tcp --sport 1024:65535 --dport 1024:65535 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT

-A OUTPUT -s 10.229.0.0/16 -p tcp -m tcp --sport 80 -m conntrack --ctstate ESTABLISHED -j ACCEPT

-A OUTPUT -s 10.229.4.0/24 -j ACCEPT
```

```
-A OUTPUT -p tcp -m tcp --sport 22 -m conntrack --ctstate ESTABLISHED -j ACCEPT

-A OUTPUT -p tcp -m tcp --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT

-A OUTPUT -p icmp -m icmp --icmp-type 0 -j ACCEPT

-A OUTPUT -p icmp -m icmp --icmp-type 8 -j ACCEPT

#logging rules to log all dropped packets

-A LOGGING -m limit --limit 2/sec -j LOG --log-prefix "IPTables-Dropped: "

-A LOGGING -j DROP

# this is the default rule

-P INPUT DROP
```