**Convergence** In the world of technology, convergence refers to different systems that move towards performing similar tasks. Typically, this involves tasks that were previously performed by separate systems and now have converged into a single system that performs all the tasks. For **example**, rather than carrying separate devices – like a cell phone, camera and digital organizer – each technology converges on a single device, or smartphone. Another good **example** is surfing the Internet on a high-definition TV (HDTV).

**Jitter** in IP networks is the variation in the latency on a packet flow between two systems, when some packets take longer to travel from one system to the other. ... A **jitter** buffer (or de-**jitter** buffer) can mitigate the effects of **jitter**, either in the network on a router or switch, or on a computer. The cause of jitter is that a packet gets queued or delayed somewhere in the circuit, where there was no delay or queueing for other packets. This causes a variation in latency. Jitter can be caused both by router misconfiguration and by PVC (permanent virtual circuit) misconfiguration by the carrier or provider

APT – advanced persistence treat - The intention of an APT attack is usually to monitor network activity and steal data rather than to cause damage to the network or organization. The goal of most APT attacks is to achieve and maintain ongoing access to the targeted network rather than to get in and out as quickly as possible.

To change host name of computer you need to change /etc/hosts and /etc/hostname

The symbol > is used to redirect information to a file but this overwrites everything in the file, with >> you can append the information to the file and both can be used to create files.

The | is used to redirect outputs to be processed by another command

mkdir -p d1/d2/d3/d4/d5 creates the folders one in the other, the -p options is to create parents if they do not exist

In this Debian ifconfig isn’t present so “ip a” is the command but if you install net-tools you will get it

apt – picks up the software and all dependencies on the repository and installs them on the computer, it needs to be directed to the server/repository location

1. Backup apt sourcelist file
2. Edit apt sourcelist file [/etc/apt/sources.list] (deb [trusted=1] path stretch main)
3. Update apt – apt-get update
4. Install software apt-get install “software name”
5. Install sudo
6. Add username to sudo
   1. Edit sudo – visudo
   2. Add user and privileges under root

A socket is opened everything a client communicates with the server, the socket comprises of [source ip:source port:destination ip:destination port:protocol]

netsh int ipv4 show dynamicport tcp – shows available ports to be opened, in linux it will be - Cat /proc/sys/net/ipv4/ip\_local\_port\_range

/var/www/html – here resides the default **index.html** file for **apache2** web server on linux (redirect and echo and you can change the content)

/etc/network/interfaces – is the files that has the interface configuration, edit it and change dhcp to static the enter address, netmask, gateway, network (network address), broadcast. Do a backup first

**Links** is a command line browser for linux

Use by typing the command link “website” – links [www.google.com](http://www.google.com)

Exit with q

Ssh command lets you connect to a system and the who command says how many people are connected to a computer – ssh user@computer

|  |  |  |  |
| --- | --- | --- | --- |
|  | PING VM | PING HOST | CONNECT TO INTERNET |
| NAT – DHCP | YES | YES | YES |
| NAT – Static | YES | YES | YES |
| Bridge – DHCP | YES | YES | YES |
| Host Only – DHCP | YES | YES | NO |

**Wireshark**

Filters

1. Capture filter
   1. To filter what is captured
   2. Must be applied before capturing starts
2. Display filter
   1. To filter only what I want to see from all the captured packets
   2. Can be applied anytime

**Pcap** has to be installed on any system so the NIC can receive packets that are not meant for it, in windows its winpcap, in linux is libpcap and for wireless its airpcap

**Packets**

Packet type field options

1. 0x0800 – internet protocol, version 4 (Ipv4)
2. 0x0806 – ARP
3. 0x0835 – RARP
4. 0x08DD – Ipv6

4 bytes = 1 word

½ byte = 1 nibble

Preamble 7 x 10101010 + 1 x 10101011(start frame delimiter / start of frame) - not picked up by wireshark

Destination MAC Source MAC Type IP Header (4 – version(payload of frame IPv4), 5 length of header in words) total length payload of packet source IP destination IP source port destination port

b0 c0 90 28 5b a2 50 c7 bf ba 54 f8 08 00 45 00

01 f3 05 9c 40 00 3c 06 84 08 b4 95 3d a1 c0 a8

00 82 00 50 06 82 21 a5 d4 31 b8 44 9a e4 50 18

72 0a 92 75 00 00 98 30 81 80 a4 7e 30 7c 31 0b

30 09 06 03 55 04 06 13 02 55 53 31 13 30 11 06

03 55 04 08 13 0a 57 61 73 68 69 6e 67 74 6f 6e

31 10 30 0e 06 03 55 04 07 13 07 52 65 64 6d 6f

6e 64 31 1e 30 1c 06 03 55 04 0a 13 15 4d 69 63

72 6f 73 6f 66 74 20 43 6f 72 70 6f 72 61 74 69

6f 6e 31 26 30 24 06 03 55 04 03 13 1d 4d 69 63

72 6f 73 6f 66 74 20 54 69 6d 65 2d 53 74 61 6d

70 20 50 43 41 20 32 30 31 30 02 13 33 00 00 00

cc 3a bd 3b 72 38 11 05 ba 00 00 00 00 00 cc 30

16 04 14 13 7f b8 07 3b 73 80 48 87 ab 32 61 16

6d 17 16 af 48 f0 68 30 0d 06 09 2a 86 48 86 f7

0d 01 01 0b 05 00 04 82 01 00 b7 35 ab db b4 c2

98 a1 9c 8e bd 86 07 b2 0f b0 55 1b 54 ee 35 9a

bc 56 6f e5 ab fd 5f a1 63 95 6a fb 25 09 f8 6e

60 e0 27 86 dc de f0 a6 5d 09 c1 71 1f a6 da 13

47 3e 04 22 79 f7 52 63 70 db 35 c8 df 8c 32 f5

f1 10 e2 de e9 d1 22 3c 76 95 7c 13 e8 12 0d cb

06 72 d2 eb 37 36 1a e9 8a 0b 30 5c c7 66 9b 4d

08 c1 8f 30 81 b5 ba 5b de 57 8d 5d d2 9f 8b cf

87 5d ce 61 01 17 47 43 1f f8 d0 93 a0 1e f7 a2

a6 d7 d5 b9 2b b7 ef 9f c8 3a 4a 45 ad f9 7e e2

ff 50 59 a8 35 f1 6c f2 9e a1 4f 14 65 07 17 61

8d 61 f7 4d 5b 43 45 a6 db 0d 48 10 54 8a 14 09

8e 0b 6a d9 1f 06 eb 3f 7b 3b 71 99 b6 a8 10 b7

9f cd 83 3b 22 bd 6a 5c d8 58 ad 53 3f ee 8c 06

8b fa 35 ef 13 81 9d 48 1a e2 12 9d c1 82 66 69

79 32 ea d6 14 ae f4 97 96 26 66 65 53 d5 ab b0

70 1a 43 90 49 a4 7d 44 87 7d 00 00 00 00 00 00

00

**System variable**:

* $\_ - takes the last parameter as your current parameter
* $? – show if the last command was executed successfully (0 means true) (the upper limit is 255 then it restarts from 0 meaning 256=0)
* $PATH – shows the path that the shell searches for commands to run
* $shell – shows the shell that is currently running

**Tcpdump**:

-n – displays everything in number format

-D – displays the active interfaces

-i – lets you specify the interface

(protocol) – lets you filter the protocol

-c – lets you choose the capture count

-r – redirect

-p – disable promiscuous mode

host – used to filter by host, can be used with ip address or host name

if passing host and protocol the pass as string – ‘host 192.168.1.1 and icmp’

port – filters the port you want to see traffic about

portrange – for a range of port (tcpdump portrange 20-33)

dst for destination – tcpdump dst host 192.168.1.1

src – for source

net – filters traffic by network – tcpdump net 192.168.1.0/24

-xxe – see full details of 1 packet - eg. tcpdump -c 1 -xxe

-s = snap byte, restricts the bytes of the packets to whatever length I passed, also shows the upcoming bytes data type/enclosure. Eg. tcpdump -xxe -s 14 -c 1 tcp

tcpdump can use conditions;

* ! or not – negation
* && or and – concatenation
* ||or or – Alternation

Eg. rule to match and tcp traffic on port 80 with 192.168.1.254 or 192.168.1.200 as destination host

tcpdump -i eth1 ‘((tcp) and (port 80) and ((dst host 192.168.1.254) or (dst host 192.168.1.200)))’

**Firewall**

Part of a computer system designed to

* Block unauthorized communication
* While permitting authorized communications

Classification

* Based on architecture
  + Single tier – consists of WAN and LAN only
  + Two tier – consists of WAN, LAN and DMZ (De-Militarized Zone, where you put all public facing servers and services, access is controlled but not prevented)
  + Three tier / multi-tier/ N tier – different server/services have firewalls restricting communication between them and other systems

Implementation methodology

* Permit all (by default)
  + And explicitly deny what is needed
* Deny all (be default)
  + And explicitly permit what is needed

Hardware firewall

* Also known as firewall appliances (are stand-alone firewall systems)
* Consists of priority of standard OS
  + Cisco IOS
  + JunOS
  + PAN-OS (runs on paloAlto N/W H/W)
* Advantages
  + Robust and stable
* Disadvantages
  + Cost
  + Upgrade/updates are not easy compared to S/W firewall

Software firewall

* Software installed on top of your operating system
  + Iptables
  + Microsoft ISA
  + Microsoft ForeFront TMG
  + Untangle
* Advantages and disadvantages – opposite to hardware firewall

Firewall filtering techniques

* Packet filtering – operates on the first 4 layers
  + Also known as
    - Screening routers
    - Filtering routers
  + The first-generation firewall
  + Operates at the network layer or transport layer of OSI
  + Examine header information of Packet
  + Controls type dan direction of traffic
  + Can filer based on
    - Port numbers, protocol, port numbers, etc
* Application proxies – operates a on layers 5 – 7 (are an extension of packet filter firewalls)
  + Are more costly because they require more processing power so high end hardware configurations are use

Screened host firewall systems – screens all traffic, in and out, through one system/machine and any interesting data is logged in an information server, operates with two networks, if more than two networks present (adding another firewall) then it is called a screened subnet firewall

**DNS in Linux**

apt-get install bind9

/etc/resolv.conf – default dns resolution file in linux

chattr +i /etc/resolv.conf (chattr – change attribute, +i – lock file, -i – unlock file) – locks the file so its not overwritten.

Create a DNS caching server – not having its own records but saving resolved results from requests

* bind9 is the DNS server for linux – install with apt-get install bind9 dnsutils
* /etc/bind/named.conf.options – take a backup of this file
* edit file and remove comment on forwarders function and put any dns server ip address
* /etc/init.d/bind9 restart – to restart the file

1. Creating forward lookup zone

* Copy /etc/bind/db.local and open the copy (db.moa.local)
* Edit the host name and email address domain part (localhost. root.localhost – hostname.moa.gy. root.mao.gy. (email is with . not @))
* Change the serial number to reflect the date and edit number - works like a version number
* After the final SOA semicolon replace what is there with the NS record ( IN NS hostname.moa.gy. )(set inline with similar fields) – specifies name server
* Below enter (hostnamefirstpart IN A hostipaddress) (in line similar fields) – resolves nameserver hostname to its ip address. (if creating a CNAME then instead of A put CNAME and instead of ip put A name hostnamefirstpart)
* Semicolons represent comment

1. Creating reverse lookup zone

* Copy /etc/bind.db.127 rename to db.”any number(this case 192)” and edit the copy
* Edit the hostname and email same as in forward zone
* Change serial same as in forward lookup
* Enter nameserver information same as forward lookup
* Then below where the A record/ hostname resolution would go put (finalnumberinipaddress IN PTR hostname.) (cnames add as normal entries with repeated ipaddresslastnumber)(remember the . after domain names and FQDN and make sure the /etc/hosts file reflects the domain properly)

1. Link them to the main file in bind

* backup /etc/bind/named.conf.local and open the original (// is the comment in this file)
* add first zone – zone “moa.gy”{type master; file “/etc/bind/db.moa.local”;};
* create reverse zone – zone “6.168.192.in-addr.arpa”{type master; file “/etc/bind/db.192”;};
* edit the /etc/resolved.conf file to reflect the right domain information

1. restart server
2. named-checkconf – to check for errors in the files
3. named-checkzone domainname (hostname.moa.gy) /etc/bind/db.moa.local (check forward lookup zone)
4. named-checkzone reverse\_address (6.168.192.in-addr.arpa) /etc/bind/db.192 (check reverse lookup zone)
5. test resolution with host command – host hostname or host ipaddress or host FQDN

**IPTABLES**

* is the built-in firewall f linux
* Is used to configure the netfilter(kernel space module)
* Opperates in the user space modules of linux

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FILTER TABLE | NAT TABLE | MANGLE TABLE | RAW TABLE | SECURITY TABLE |
| INPUT   |  | | --- | |  | |  | |  | | PRE-ROUTING   |  | | --- | |  | |  | |  | | |  | | --- | |  | |  | |  | | |  | | --- | |  | |  | |  | | |  | | --- | |  | |  | |  | |
| OUTPUT   |  | | --- | |  | |  | |  | | POST-ROUTING   |  | | --- | |  | |  | |  | | |  | | --- | |  | |  | |  | | |  | | --- | |  | |  | |  | | |  | | --- | |  | |  | |  | |
| FORWARD   |  | | --- | |  | |  | |  | | OUTPUT   |  | | --- | |  | |  | |  | | |  | | --- | |  | |  | |  | | |  | | --- | |  | |  | |  | | |  | | --- | |  | |  | |  | |

IPTABLES -> TABLES -> CHAINS -> RULES

The view the different tables the command is -> iptables -L -t table\_name

The **filter table** is the default table.

* It provides the firewall features.
* Used for packet filtering.
* **INPUT CHAIN**
  + All packets coming to the machine, addressed to
* **OUTPUT CHAIN.**
  + All packets leaving the machine, originated from
* **FORWARD CHAIN**
  + All packets that are going through but not addressed to or has originated by me
* If a rule condition is met the remaining rules are not processed
* If no rule applies to the packet the default rule is applied
* Default policy is either accept or drop for each chain

The **NAT table** is used primarily for port forwarding, the chains are

* PRE-ROUTING
  + Used for DNAT – when packets destination is changed
* POST-ROUTING
  + Used for SNAT – when packets source is changed
* OUTPUT
  + Used if traffic is generated from the machine that the firewall is installed on

The **MANGLE table** is used to modify or alter packet, eg. give priority to one packet over the other, the chains are:

* PRE-ROUTING
* OUTPUT
* FORWARD
* INPUT
* POST-ROUTING

The **RAW table** can be used to configure the tracking through **conntrack**. The service that tracks connecctions, the chains are:

* PRE-ROUTING
* OUTPUT

The **SECURITY table** is used in system that has SELinux (security enabled linux)

SPI Firewalls check the state of a packet

**Lab –** configure a Statefull Packet inspection (SPI) Firewall that matches the following condition:

* Allow all outbound traffic
* Do not allow any inbound new connection

iptables -L – list the rules in the firewall

iptables -P – changes the default policy of a chain ( iptables -P INPUT DROP/iptables -P FORWARD DROP

), to see the verbose (more information) use -v (iptables -L -v)

iptables -A INPUT -i lo -j ACCEPT (-A = append, -i = interface, -j = jump to target, lo = loopback)

iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT (-m = match, state = loads the state module, --state = says which state to check(no space between the words) )

iptables -F – flush all the rules (does not affect the default policy)

the rules are reset on every restart so creating a script file to reconfigure

example – {

#!/bin/bash

iptables -F

iptables -P INPUT DROP

iptables -P FORWARD DROP

iptables -P OUTPUT ACCEPT

iptables -A INPUT -i lo -j ACCEPT

iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

iptables -L -v

}

Save/backup current configuration:

iptables-save > file\_path/file\_name.v4 – normally /etc/iptables/rules.v4

Restore from backup:

iptables-restore < file\_path/file\_name.v4

iptables-persistent – allows you to automatically load backed up files on computer startup

filter traffic in iptables:

* iptables -A INPUT -s xxx.xxx.xxx.xxx -j ACCEPT --s =source ip => accept from ip ( /24 or /255.255.255.0 appended works to add subnet mask)
* iptables -A INPUT iprange --src-range xxx.xxx.xxx.xxx-xxx.xxx.xxx.xxx -j ACCEPT => accept from range (change --src-range to --dst-range for destination filter)
* iptables -A INPUT -s xxx.xxx.xxx.xxx -m mac --mac-source xx:xx:xx:xx:xx:xx -j ACCEPT (filter by source MAC)
* iptables -A INPUT -p tcp --dport XXXX -j ACCEPT (for port range just use xxxx:xxxx)
* iptables -A INPUT -p tcp --destination-port XXXX -j ACCEPT (filtering by destination port number)
* iptables -A INPUT -p tcp --sport XXXX -j ACCEPT (for port range just use xxxx:xxxx)
* iptables -A INPUT -p tcp --source-port XXXX -j ACCEPT (filtering by source port number)

iptables -A INPUT -p icmp –icmp-8 -j DROP drop all incoming ping from different computers

ping works on 8 and 0 (states) 0 is return and 8 is send, so if you block the incoming 8 then your incoming/returning 0 is allowed

fail2ban – extension used in iptables to restrict access attempts:

iptables -L -n --line-numbers -> used to display lines numbers

iptables -D f2b-sshd 3 -> deletes the 3rd line from the fail2ban chain in iptables

iptables -N WHITELIST – create a custom chain

iptables -X WHITELIST – remove a custom chain

ss -ant – shows all open ports

iptables -t nat -A PREROUTING -p tcp --dport 50 -j REDIRECT –to-port 80 -> port forwarding from port 50 to 80

create custom log for iptables:

* create new chain
  + iptables -N chainname
* add rule to create log of all packets entering the chain
  + iptables -A chainname -j LOG --log-prefix “prefix to be filtered” --log-level leveloflog
* add rule to drop all packets entering the chain
  + iptables -A chaingname -j DROP
* create rule in other chains to send all packets you want logged and dropped to new chain
  + iptables -A INPUT/OUTPUT/FORWARD rules -j chainname
* create rule to send logs to custom log file
  + create and edit /etc/rsyslog.d/iptables.conf and add: ->
    - :msg, contains, "prefix to be filtered" -/var/log/customname.log (if it’s a ip then put @ infront)
    - & ~
* Restart rsyslog
  + service rsyslog restart

apt-get install xtables-addons-common libtext-csv-xs-perl => command to download package that allows use of GEO-IP in iptables

after the install – in the file /usr/lib/xtables-addon there are two files one for download and one for building

* move to the /usr/lib/xtables folder
* edit the xt\_geoip\_dl file and add a reliable source (this is the download file)
* create the /xtables folder and in that folder run the download file
* /usr/share/xt\_geoip – this is the default location that xtables looks for the database (similar to how iptables-persistent looks for the rules file)
* Then build the database files with the command -> /usr/lib/xtables-addons/xt\_geoip\_build -D /user/share/xt\_geoip/ \*.csv (call the build file and send to the xt\_geoip ditectory from all csv files)
* Add iptables rule – iptables -A INPUT -m geoip –src-cc CN -j LOG --log-level debug --log-prefix “| With love from China |” -> adds a log entry when a packet comes from china.

**systemctl status “service name” – to check if a service is running**

**VM LAB SETUP**

Set the machine in VMWare as represented to simulate a real config

* LAN machine
  + Ip address – 192.168.4.2
  + Netmask – 255.255.255.0
  + Gateway – 192.168.4.254
  + Network – 192.168.4.0
  + Broadcast – 192.168.4.255
* Middle machine
  + LAN NIC
    - Ip address – 192.168.4.254
    - Netmask – 255.255.255.0
    - Network – 192.168.4.0
    - Broadcast – 192.168.4.255
  + WAN NIC
    - Ip address – 192.168.5.254
    - Netmask – 255.255.255.0
    - Gateway – 192.168.5.2
    - Network – 192.168.5.0
    - Broadcast – 192.168.5.255
* WAN machine (internet)
  + Ip address – 192.168.5.2
  + Netmask – 255.255.255.0
  + Gateway – 192.168.5.254
  + Network – 192.168.5.0
  + broadcast – 192.168.5.254

**SQUID**

Proxy server (forward proxy)

cp /etc/squid/squid.conf{,.backup} -> creates a backup file squid.conf.backup

grep “^#” filename -> shows all lines in the file that begins with #

grep -v “^#” filename -> shows all lines in the file that does not begin with #

grep . filename -> shows all lines in a file that does not being with and empty space

grep -v “^#” filename | grep . -> shows all the files in the file that does not start with # not a blank space

in the /etc/squid/squid.conf file after reducing the lines you can add your networks

acl netname src 192.168.0.0/24 -> defines the network you are working with

ensure squid runs on http\_port 3128

ensure that above the http\_access deny all that your network is allowed -> http\_access allow netname

restart squid service -> systemclt restart squid

/var/log/squid/access.log is the log file

To filter/block traffic:

* add acl listname url\_regex “/etc/squid/filename.lst” -> the list name is anything and the file extension is just to remind what is it
* add before the rule for allowing the traffic from that network -> http\_access deny listname
* create file and add keywords in all forms that you wish to filter
* restart squid

network forwarding from one nic to the other

* check the proc file
  + cat /proc/sys/net/ipv4/ip\_forward
* turn on the proc file
  + echo 1 > /proc/sys/net/ipv4/ip\_forward

configure nat for snat

this is in a script file

#!/bin/bash

iptables -F

iptables -t nat -F

iptables -P INPUT DROP

iptables -P FORWARD DROP

iptables -P OUTPUT DROP

iptables -A INPUT -p tcp --dport 22 -j ACCEPT

iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT (these two allows putty to keep working)

echo 1 > /proc/sys/net/ipv4/ip\_forward

iptables -A FORWARD -i ens33 -o ens37 -j ACCEPT (sends traffic out)

iptables -A FORWARD -i ens37 -o ens33 -m state --state RELATED,ESTABLISHED -j ACCEPT (allows traffic to come back)

iptables -t nat -A POSTROUTING -o ens37 -j MASQUERADE (masks the ip address)

HACKERTYPER.NET

HPING3 – Read up

One function is packet crafting

**Intrusion Detection Systems (SNORT)**

* snort -W – shows all interfaces
* snort -i intereface\_number – select which interface to capture on
* snort -d – shows the application layer data
* snort -e – shows the link layer data
* -c – configuration file
* -l – log file
* -A – allert – where will the information appear
* -T – test

Dir /b – bare format of file names in windows cmd

eg. snort -i 1 -dev – shows application and link layer data of packets from interface 1, the v means verbose

when extracting if “browser.ie.rules” is not present in the rules folder do not continue, turn off antivirus and redo the extraction and make sure to overwrite without prompt

snort in Packet Capture mode

* capture traffic on the interface
* command - snort

snort in IDS mode

* snort -i 1 -c c:\snort\etc\snort.conf -l c:\snort\log -A console (rerun after every error fix)

backup config file

ERROR 1 – ipvar

Solution:

* Replace all instances of ipvar in the conf file with var
* Save

ERROR 2 – snort\_dynamicpreprocessor

Solution:

* Open conf file and find dynamicpreprocessor
* Change path from linux path the windows path (lib folder)
* Add path for each file in the directory (remove the directory word from the file since they are files not directory)
* Save

ERROR 3 – snort\_dynamicengine

Solution:

* Just under the dynamicpreprocessor part change the file extension from so to dll
* Remove lib from the start of the name
* Change from linux to windows path
* Save

ERROR 4 – snort\_dynamicrules

Solution:

* Create the folder in lib (snort\_dynamicrules)
* Change the path to windows format in conf file (under dynamicengine)
* Save

ERROR 5 – normalize\_ip4

Solution:

* Search in conf file for normalize\_ip4
* Comment out the active lines
* Save

ERROR 6 – white\_list.rules -> black\_list.rules

Solution:

* Go to the rules folder
* Create simple txt file and change name and extension to white\_list.rules
* Create also black\_list.rules
* Save

Look for pid

Extra things to change (around line 100):

* Change to absolute windows path, the paths for
  + var RULE\_PATH ../rules (C:\Snort\rules)
  + var SO\_RULE\_PATH ../so\_rules (C:\Snort\so\_rules)
  + var PREPROC\_RULE\_PATH ../preproc\_rules (C:\Snort\preproc\_rules)

A few lines down change:

* var WHITE\_LIST\_PATH ../rules
* var BLACK\_LIST\_PATH ../rules

to

* var WHITE\_LIST\_PATH ..\rules
* var BLACK\_LIST\_PATH ..\rules

then change:

* whitelist $WHITE\_LIST\_PATH/white\_list\_rules,\
* blacklist $BLACK\_LIST\_PATH/black\_list\_rules

to

* whitelist $WHITE\_LIST\_PATH\white\_list\_rules,\
* blacklist $BLACK\_LIST\_PATH\black\_list\_rules

go to the last section where the rules section is and:

* find - include $RULE\_PATH/blacklist.rules
* change to - include $RULE\_PATH/black\_list.rules

SAVE

Run the command snort -i 1 -c c:\snort.conf -l c:\Snort\log -T to test the system

In notepad++

* open new page
* type –> alert ip any any -> any any (msg:”IP Packets Detected”; sid: 1000000;)
* ‘in order’ send an alert if an IP packet is detected with any source IP and any source port to any destination IP and any destination port and show the message (start from the sid 1000000 because we have to create all our own rules so start the sequence from this number)
* Save in the rules folder, make sure the extension is as .rules
* Go to conf file, at the bottom where the rules are, add a line with the direction to the rule you created to be included

To output to a syslog server:

* add in the .conf file under syslog
  + output alert\_syslog: host=ip\_of\_syslog\_server:port(514), LOG\_AUTH LOG\_ALERT
* run command
  + snort -I -c c:\snort\etc\snort.conf -l c:\\snort\log -s

**Snort on Linux**

Install command (includes dependency check):

* sudo apt-get install -y bison flex gcc libdnet libdumbnet-dev libluajit-5.1-dev libnghttp2-dev libpcap-dev libpcre3-dev libssl-dev make openssl zlib1g-dev

download and install daq (data acquisition library)

* download source code of the software
  + sudo mkdir /usr/src/snort\_src
  + cd $\_ -> go the the directory you just created
  + sudo wget url\_of\_daq
  + sudo tar -zxf file\_name -> extract the file (z to unzip, x to extract, f to say which file and v before the f is verbose to show the process)
  + enter directory created

after downloading software source code form developer to install you must

* configure – do a check for perquisites
* make – compilation
* make install – installing the program to the OS
* (some software may have extra steps, it’s up to the developer)
  + Run configuration file ./configure
  + make
  + sudo make install

Install snort

* go back to the daq file location and download snort from there (-q runs it in quiet mode)
  + extract snort similar to the daq file (sudo mkdir /usr/src/snort\_src/)
* enter the file location
* ./configure --enable-sourcefire -> configure snort to work with sourcefire
* make
* make install

possible errors

* sudo snort -V
  + will not be able to load config file
* sudo ldconfig – fix the error
* sudo snort -V – should work now

REMEMBER NEVER TO RUN MAJOR APPLICATIONS AS ROOT!!!

POSTCONFIGURATION->

getent – check existence of group or password, possibly more things CHECK UP ON THIS

Create group and user to run snort

* sudo groupadd snort
* sudo useradd snort -r -s /usr/sbin/nologin -c SNORT\_IDS -g snort (-r is system, -s and no login says cannot log in by shell, -c is comment for if you want to view why the user was created and -g is to say which group to add user too)

create needed folders

* sudo mkdir -p /etc/snort/rules
* sudo mkdir /var/log/snort
* sudo mkdir /usr/local/lib/snort\_dynamicrules

copy the configuration files to the required location

* sudo cp path\_where\_snort\_was\_extracted(/usr/src/snort\_src/folder/etc/)/\*.conf\* destination\_path(/etc/snort)
* sudo cp path\_where\_snort\_was\_extracted(/usr/src/snort\_src/folder/etc/)/\*.map destination\_path(/etc/snort)

create needed files

* sudo touch /etc/snort/rules/{white\_list,black\_list,local}.rules

change permission to(and) then ownership of the snort directory and everything inside

* sudo chmod -R 5775 /etc/snort (-R means recursive)
* sudo chmod -R 5775 /var/log/snort
* sudo chmod -R 5775 /usr/local/lib/snort\_dynamicrules

then

* sudo chown -R snort:snort /etc/snort (-R means recursive)
* sudo chown -R snort:snort /var/log/snort
* sudo chown -R snort:snort /usr/local/lib/snort\_dynamicrules

sudo ls -ld /etc/snort(or any of the folder names) – check the status of the changes

edit snort.conf file

* backup file (sudo cp /etc/snort{,.backup})
* change the lines
  + var RULE\_PATH ../snort/rules to /etc/snort/rules
  + var SO\_RULE\_PATH ../so\_rues to /etc/snort/so\_rules
  + var PREPROC\_RULE\_PATH ../preproc\_rules to /etc/snort/preproc\_rules
  + var WHITE\_LIST\_PATH ../rules to /etc/snort/rules
  + var BLACK\_LIST\_PATH ../rules to /etc/snort/rules
* remove or comment out the all the rule paths in step 7 (include $RULE\_PATH things)
  + keep these
    - include $RULE\_PATH/local.rules
    - include $RULE\_PATH/white\_list.rules
    - include $RULE\_PATH/black\_list.rules

run snort test

* sudo snort -T -c etc/snort/snort.conf

create alert rule in local.rules

run snort

* sudo snort -i ens33 -u snort -g snort -c /etc/snort/snort.conf -A console

**Database SQL (MariaDB)**

apt-get install mariadb-server

* uses unix\_socket plugin authentication
* 3306 is the port number used (ss -ant to see the running ports and services state)
* Commands are not case sensitive but best practice is the use capitals
* Table names are case sensitive
* Each command should end with a ;

Commands:

* mysql -u root to connect to the server
* SHOW DATABASES; - used to show the databases present
* CREATE DATABASE database \_name; to create a new database but if the database exists it gives an error, to avoid this use CREATE DATABASE IF NOT EXISTS database\_name;
* SELECT DATABASE(); to see the current database
* USE database\_name; to switch/use a database
* SHOW TABLES; to show tables in current database
* CREATE TABLE table\_name(table content); to create a table
* SHOW COLUMNS FROM database\_name; or DESC database\_name; show the structure of a database

users

|  |  |  |  |
| --- | --- | --- | --- |
| Id | integer | 10 | Primary key |
| Username | varchar | 30 | Unique |
| Password | varchar | 32 |  |

CREATE TABLE users (

id int(10) NOT NULL AUTO\_INCREMENT,

username varchar(30) NOT NULL,

password varchar(32) NOT NULL,

PRIMARY KEY (id),

UNIQUE (username)

);

Other commands:

* SELECT \* FROM database\_name; show all content of the table
  + SELECT field1, field2, field3 FROM table\_name; shows information of these fields from the table
  + SELECT field1 AS ‘TEMPNAME1’, filed2 AS ‘TEMPNAME2’, field3 AS ‘TEMPNAME3’ FROM table\_name; shows content from these filed with temporary names
* Select with conditions
  + SELECT field1, field2 FROM table WHERE field1=criteria;
* To add values:
  + INSERT INTO database\_name VALUES (values\_in\_order\_of\_columns);
  + INSERT INTO database\_name (id, password, username) VALUES (id\_vlaue, ‘password\_value’, ‘username\_value’);
  + INSERT INTO database\_name (password, username) VALUES (‘password\_value’, ‘username\_value’);
* Edit existing data:
  + UPDATE table\_name SET field\_name=value WHERE field\_name=criteria;
* Edit existing table:
  + ALTER TABLE table\_name ADD COLUMN new\_field\_name field\_type(size) NOT NULL(if needed) AFTER field\_to\_be\_before\_this\_one (default is at the end);
* exit to exit() no ;
* Making backup:
  + exit the db\_server
  + mysqldump --user root my\_db >my\_db.sql
* Deleting table data
  + DELETE FROM table\_name WHERE field\_name=criteria;
* Delete columns
  + ALTER TABLE table\_name DROP COLUMN column\_name;
* Delete table:
  + DROP TABLE table\_name;
* Delete database:
  + DROP DATABASE database\_name;
* Restore from backup
  + First create new database (name doesn’t have to be the same)
  + Exit db\_server
  + mysql -u root -p new\_database\_name < backup\_name.sql => -u is user and -p is means password but in this version no password is there
  + when prompted for password use
    - mysql -u root

**Barnyard2**

* open source interpreter for snort unified2 binary files
* why use barnyard
  + allows snort to write to disk in an efficient manner and leaving the task of parsing binary data into various formats to a separate process that will not cause snort to miss network traffic

Download command for prerequisite

* apt-get install mariadb-server mariadb-client libmariadbclient18 libmariadbclient-dev libmariadbd-dev autoconf libtool unzip dos2unix default-libmysqlclient-dev

Configuration

* in /etc/snort/snort.conf go to the unified2 part of section 6
* follow the example and put
  + output unified2: filename snort.u2, limit 128

Download and install barnyard

* cd dl/
* wget -q url for barnyard.zip
* unzip barnyard2-master.zip > /dev/null
* enter barnyard folder
* run command
  + ./autogen.sh
    - This creates the configure file
* Create softlink(shortcut) to dumdnet.h file
  + sudo ln -s /usr/include/dumbnet.h /usr/include/dnet.h
* link database to barnyard
  + sudo ./configure --with-mysql --with-mysql-libraries=/usr/lib/x86\_64-linux-gnu/
    - make sure the checking for mysql =yes
  + sudo make
  + sudo make install
    - which barnyar2 to confirm

Post install

* make sure you are in the extracted folder
  + sudo cp etc/barnyard2.conf /etc/snort/ (make sure the first one is not /etc)
  + sudo mkdir /var/log/barnyard2
  + sudo chown snort.snort /var/log/barnyard2/
  + sudo touch /var/log/snort/barnyard2.waldo
  + sudo chown snort.snort /var/log/snort/barnyard2.waldo

Configure database server/service

* sudo mysql -u root (log into mysql as root)
* USE mysql (use this database)
* UPDATE user set plugin=’’ where User=’root’; (do not run multiple times and it should only match and change one entry)
* flush privileges; (to apply)
* exit

restart service

* sudo systemctl restart mariadb.service

open installation wizard

* sudo mysql\_secure\_installation
  + follow instructions in the wizard
    - everything yes and don’t forget the password

create database

* sudo mysql -u root -p
* CREATE DATABASE IF NOT EXISTS snort;
* USE snort;
* Import structure
  + source /home/’username’/’barnyard\_extract location’/schemas/create\_mysql
* SHOW TABLES; - to check to make sure

Configure database

* Create user snort and give permissions on snort database
  + CREATE USER ‘snort’@’localhost’ IDENTIFIED BY ‘any\_password’;
  + grant create, insert, select, delete, update on snort.\* to ‘snort’@’localhost’;

edit barnyard2 config to connect barnyard to sql server

* edit file /etc/snort/barnyard2.conf
  + add the line
    - output database: log, mysql, user=snort password=db\_user\_password dbname=snort host=localhost
* secure config file
  + sudo chmod o-r /etc/snort/barnyard2.conf (with numbers its 640 instead of 0-r)

generate the SID file

* create 3 dummy rules in local.conf
* download sidmap perl file
  + wget -q url for create-sidmap.pl
* convert file from windows to linux format
  + dos2unix create-sidmap.pl
* give execute permission
  + chmod 755 create-sidmap.pl
* pick up all the rules files and map the sid
  + ./create-sidmap.pl /etc/snort/rules > /etc/snort/sid-msg.map (then redirect the output to a map file but we don’t permission to so a script file is used to overcome to error)
    - sudo sh -c ‘./create-sidmap.pl /etc/snort/rules/ > /etc/snort/sid-msg.map’

if you want to work with barnyard (today onwards) add rev:number(1 or 2 ….n); to identify the version eg (alert ip 192.168.0.1 any -> any any (msg: “from base”; sid: 1000001; rev 1;))

* start snort
* in /var/log/snort look for the snort.u2 file

run snort

* snort -q -c /etc/snort/
* snort.conf -I ens33 -u snort -g snort

run barnyard

* make sure the user and group are set as snort
  + chown snort.snort /var/log/snort/snort.u2
* sudo barnyard2 -c /etc/snort/barnyard2.conf -d /var/log/snort -f snort.u2 -w /var/log/snort/barnyard2.waldo -g snort -u snort
  + -c - config file
  + -d – directory for binary file
  + -f - binary file
  + -w – waldo file

**BASE** – Basic Analysis and Security Engine

Doesn’t run on Debian 9

Configure SQL for remote user access

* Change the open 3306 port that’s open on loopback/localhost to exterant interface
  + Backup and open /etc/mysql/mariadb.conf.d/50-server.cnf
  + Search for the bind-address setting and change to 0.0.0.0 and save
  + Restart service
    - sudo systemctl restart mariadb.service
    - ss -ant to check the status of the ports
* allow remote user to access
  + log into maria db
  + GRANT create, insert, select, delete, update ON snort.\* TO ‘snort’@’ip\_of\_base\_conputer’ IDENTIFIED BY ‘snort\_db\_user\_password’;
* On base machine
  + Make sure the hostname is right
  + Make sure you have the right repo and instead of stretch its jessie also no need ofr [trusted=1]
  + Install maria db client
    - Apt-get install mariadb-client
  + Connect to the database
    - Myspl -usnort -p’db\_user\_password’ -hdb\_server\_ip
      * No space after -u -p and -h and password goes in quotes
* Install and configure base and php5
  + Install php5
    - apt-get install apache2 php5 p7zip libapache2-mod-php5 php5-mcrypt php5-mysqlnd libphp-adodb unzip
  + download base
    - mkdir downloads
    - cd downloads/
    - wget -q url\_for\_base
    - tar -zxf base\_file\_name
    - move extracted folder to web accessible location
      * mv base\_folder\_name /var/www/html/base/
    - change to /var/www/html/
    - change owner of base folder to apache user
      * chown -R www-data:www-data base/
    - backup and edit the php.ini file
      * /etc/php5/apache2/php.ini
      * Search for error reporting (second occurance)
        + Change to error\_reporting = E\_ALL & ~E\_NOTICE
      * Save and exit
    - Restart apache
      * Service apache2 restart
    - Browser base in browser host\_ip/base
      * Everything should be in green
      * Click continue
      * Enter ath to ADODB
        + /usr/share/php/adodb
      * Fill form
        + Database name – snort
        + Database host – ip of server
        + Port leave as default if you haven’t changed the port
      * Continue through the steps, you can choose if you want authentication

What are the differences between windows ping and linux ping – test in wireshark

AND about itype - iptables

alert icmp any any -> any any (msg: “Windows Ping”; itype:8; content:”abcdefghijklmnop”; sid:1000002; rev:1)

alert icmp any any -> any any (msg: “Linux Ping”; itype:8; content:”|10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F|”; sid:1000003; rev:1)

**Virtual Private Network VPN**

The connection of two private (trusted) networks over an untrusted network

Used to extend a private network

Benefits

* Extends geographical connectivity
* Improves security
* Reduces operational costs versus a traditional WAN
* Improves productivity

VPN characteristics

* Traffic is encrypted
* Remote site is authenticated
* Multiple protocol support
* Connection is point to point

Types of VPN

* Based on connection type
  + Site-to site VPN
  + Dialup / Client VPN

Tunnel

* A means of forwarding data across a network from one node to another, as if the two nodes were directly connected.
* How it is achieved
  + Encapsulation / encapsulating the data

VPN functions/services

* Authentication
* Access control
* Confidentiality
* Data integrity

VPN protocols

* Internet protocol security (IPsec) – commonly used
* Transport Layer Security (SSL/TLS)
* Datagram Transport Layer Security (DTLS)
* Microsoft Point-to-Point Encryption (MPPE)
* Microsoft Secure Socket Tunneling Protocol (SSTP)
* Multi Path VPN (MVPN)
* Secure Shell (SSH) VPN
* OpenVPN – commonly used
* Point-to-Point Tunneling Protocol (PPTP)
* Layer 2 Tunneling Protocol (L2TP)
* Secure Socket Tunneling Protocol (SSTP)
* Layer 2 Forwarding Protocol (L2F)
* SOCKS – is not used as much as the ones above.

Types of VPN

* Trusted VPN – when the ISP is configuring and securing the VPN
  + Private IP networks circuits leased from the internet providers
    - Layer 2 VPN
      * ATM circuits,
      * Frame-relay circuits and
      * Transport of Layer 2 frames over Multiprotocol Label Switching (MPLS).
    - Layer 3 VPN
      * Multiprotocol Label Switching (MPLS) with constrained distribution of routing information through BGP. (RFC 4364)\*
* Secure VPN – when you are using the protocols, configuration and encryption yourself
* Hybrid VPN – using secure in a trusted network

Debian

* Tcpdump openvpn and apache2

Create file server.conf (linux server file can be anywhere)

* proto udp
* port 1194
* dev tun (tunnel device)
* auth none
* cipher none
* ifconfig 10.8.0.1 10.8.0.2 (local address then external address)
* verb 3 (3 is debug/log level verbose)

sudo openvpn --config server.conf(full path of file)

in windows, install openvpn (best option, create config file in the program filles/openvpn/opvpn client/core)

* file name client.conf
  + remote “server ip”
  + proto udp
  + port 1194
  + dev tun
  + auth none
  + cipher none
  + ifconfig 10.8.0.2 10.8.0.1 (local address then external address)
  + verb 3

open cmd as administrator

* cd to the core folder
* openvpn.exe --config client.conf

IPSec (uses HMAC)

* IPSec consists of the following
  + Encapsulating Security Payload (ESP)
    - Data Integrity
    - Encryption
    - Authentication
    - Anti Replay
    - Protocol number 50
  + Authentication Header (AH)
    - Data Integrity
    - Authentication
    - Anti Replay
    - Protocol number 51
  + Internet Key Exchange (IKE)
    - Used by IPSec to exchange encryption keys dynamically
    - Establishes Security Association (SA) between two devices
      * Internet Security Association and Key Management Protocol (ISAKMP)
        + Used by IKE to define how Security Associations are set up
        + Used to define direct connections between two hosts that are using IPSec
    - Works in two modes
      * Main mode
        + 6 packets sent
        + More secure
      * Aggressive mode
        + 3 packets sent
        + Less secure
        + Faster

IPSec has two modes

* Transport mode
  + AH – IP Header -> AH -> TCP Header + Payload
  + ESP – IP Header -> ESP -> TCP Header + Payload
* Tunnel Mode
  + AH – IP (separate from original) -> AH -> Original IP Header -> TCP Header + Payload
  + ESP – IP (separate from original) -> ESP -> Original IP Header -> TCP Header + Payload

Security Association (SA) consists of

* Information about the communications endpoints – eg. public IP address
* Whether AH or ESP are being used with IPSec
* The secret key / algorithm being used

Configure IPSec

Comp1

* Change hostname
* Configure APT
* Install packages
  + apt-get install strongswan tcpdump
* configure files
  + nano /etc/ipsec.secrets
  + add at the end
    - current\_pc\_ip other\_pc\_ip : PSK “secret\_key”
  + nano /etc/ipsec.conf
  + add at the end
    - include /etc/ipsec.d/\*.conf
  + create file
  + nano /etc/ipsec.d/comp1-comp2.conf
    - conn comp1-comp2
      * type=transport
      * authby=secret
      * left=curent\_pc\_ip
      * right=other\_pc\_ip
      * pfs=yes (pfs means Perfect Forward Secrecy)
      * auto=start
  + restart service
    - systemctl restart ipsec

Comp2

* Change hostname
* Configure APT
* Install packages
  + apt-get install strongswan tcpdump
* configure files
  + nano /etc/ipsec.secrets
  + add at the end
    - current\_pc\_ip other\_pc\_ip : PSK “secret\_key”
  + nano /etc/ipsec.conf
  + add at the end
    - include /etc/ipsec.d/\*.conf
  + create file
  + nano /etc/ipsec.d/ comp2-comp1.conf
    - conn comp2-comp1
      * type=transport
      * authby=secret
      * left=curent\_pc\_ip
      * right=other\_pc\_ip
      * pfs=yes (pfs means Perfect Forward Secrecy)
      * auto=route
  + restart service
    - systemctl restart ipsec

Configure self-signed certificate on apache

* install openssl and apache2
  + apt-get install openssl apache2
* activate ssl on apache
  + a2enmod ssl
  + a2ensite default-ssl
  + service apache2 reload
* create ssl certificate directory
  + mkdir /etc/apache2/ssl
* create certificate
  + openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.crt
  + enter details
    - for common name use this computer IP or FQDN
* protect folder as is contains the private key
  + chmod 600 /etc/apache2/ssl/\*
* add path for certificate and key to conf file
  + backup file
  + nano /etc/apache2/sites-enabled/default-ssl.conf
    - set ServerName “what you put as common name in certificate details”
    - set SSLCertificateFile as path the file (/etc/apache2/ssl/apache.crt)
    - set SSLCertificateKeyFile as path the file (/etc/apache2/ssl/apache.key)
* to check in command line
  + openssl s\_client -connect “IP or FQDN”:PORT (443)

PPTP VPN Setup

* works on port number 1723
  + check if its open –> ss -ant | grep 1723
* install pptpd
* add DNS Server
  + nano /etc/ppp/pptpd-options
    - change ms-dns servers to the correct DNS Servers
* add vpn user account
  + nano /etc/ppp/chap-secrets (add line at the end)
    - username pptpd password ip\_address(can be \*) (must use tab not spaces)
* edit conf file
  + nano /etc/pptpd.conf
    - add local and remote IPs based on examples
      * localIP being the local adapter/server’s IP
      * and remoteIP is the range of IPs that the users will be able to use

PPTP Client on Debian

* apt-get install pptp-linux
* create connection
  + pptpsetup --create tunnel\_name --server pptp\_server\_ip --username user\_name --password pass\_word
* edit the tunnel file
  + nano /etc/ppp/peers/tunnel\_name
    - below noauth type
      * require-mppe
* connect to server
  + pon tunnel\_name debug dump logfd 2 nodetach

Squid configuration (reverse proxy)

* /etc/squid/squid.conf
* http\_port listening\_port accel defaultsite=FQDN\_or\_site\_IP vhost (virtual host setup and change port to 80)
* cache\_peer ip\_address\_of\_peer parent port\_number 0 no-query originserver name=rule\_name (forward traffic coming from port 80)
* acl user\_group\_name dstdomain site\_IP (at the end of the acl block)
* http\_access allow user\_group\_name (put first in the http\_access block)
* cache\_peer\_access rule\_name allow user\_group\_name (in the cache\_peer\_access block)
* cache\_peer\_access rule\_name deny all
* restart squid

**NGINX**

Components and deifne in available and switched on when copied to enabled

* install
  + apt-get nginx (creates a web server)
* delete the entry from sites-enabled
  + rm /etc/nginx/sites-enabled\*
* cd /etc/nginx/sites-available
* create and edit the file rp.conf
  + server {
    - listen 80;
      * location / {
        + proxy\_pass <http://192.168.106.138>; (IP of desired destination)
      * }
  + }
* Create link to rp.conf in sites-enabled
  + ln -l /etc/nginx/sites-available/np.conf /etc/nginx/sites-enabled/
* restart service and check port
  + sysemctl restart nginx
  + netstat -tulpn | grep :80

**OpenVPN with EasyRSA on Debian**

Build the CA

* download and extract the Easy-RSA file
  + wget -q url/filename(easyrsa-unix…)
  + tar -xf filename
  + cd folder\_name of extracted content
  + locate the vars.example file
* edit variables file
  + cp vars.example vars
  + nano vars (close to the end edit the certificate vaules)
    - uncomment and enter all the correct certificate values
* generate the PKI structure
  + ./easyrsa init-pki
* generate the private key and certificate
  + ./easyrsa build-ca nopass (nopass means do not encrypt with password)

Generating the VPN Server Certificate, key and Encryption Files

* Download and extract the easy-rsa file
  + wget -q url/filename(easyrsa-unix…)
  + tar -xf filename
  + cd folder\_name of extracted content
* create pki folder
  + ./easyrsa init-pki
  + ./easyrsa gen-req server\_name nopass
* Install openvpn and create key and request
  + apt-get install openvpn
  + cp pki/private/keyfilename.key /etc/openvpn/
  + send request copy to CA
    - scp pki/reqs/requestfilename.req uesr@CA\_IP:/tmp
* import request (on CA)
  + ./easyrsa import-req /tmp/requestfilename.req common\_name\_of\_requester
* Sign and issue request
  + ./easyrsa sign-req server FQDN\_of\_vpnserver(also should be common name)
* Send certificates (rootca and vpnserver) to vpnserver
  + scp pki.issued/vpnserverfilename.crt user@vpnserverip:/tmp
  + scp pki/ca.crt user@vpnserverip:/tmp
* copy certificates the openvpn folder
  + cp /tmp/ vpnserverfilename.crt /etc/openvpn
  + cp /tmp/ca.crt /etc/openvpn
* generate the deffie hellmen (used for key exchange)
  + ./easyrsa gen-dh
* Generate the HMAC signature
  + openvpn --genkey --secret filename1.key
* Copy files to openvpn directory
  + cp filename1.key /etc/openvpn
  + cp pki/dh.pem /etc/openvpn

Generating a Client Certificate and Key Pair

* cd (go to the home directory of the user)
* mkdir -p ~/client-configs/keys
* chmod -R 700 client-configs/
* cd easyrsa\_folder
* Generate the request
  + ./easyrsa gen-req clientcommonname nopass
  + cp pki/private/ clientkeyfile.key ~/client-configs/keys/

Send request file to CA

* scp pki/reqs/clientrequestfile.req user@caip:/tmp/

import request on CA

* ./easyrsa import-req /temp/clientrequestfile.req clientcommonname

Sign the request

* ./easyrsa sign-req client FQDN\_of\_client

Send client certificate to vpn server

* scp pki/issued/ clientcommonname.crt user@vpnserverIP:/tmp/
* cp /tmp/ clientcommonname.crt ~/client-configs/keys/
* cp filename1.key ~/client-configs/keys
* cp /etc/openvpn/ca.crt ~/client-configs/keys/

Configure openvpn service

* cd (go to the home directory of the user)
* copy openvpn sample configuration file
  + cp /usr/share/doc/openvpn/examples/sample-config-files/server.conf.gz /etc/openvpn
  + gzip -d /etc/openvpn/server.conf.gz
* edit the configuration file
  + nano /etc/openvpn/server.conf
    - search for tls-auth and add below
      * key-direction 0
    - add just below under cipher AES line
      * auth SHA256
    - go down to the deffie hellman parameters and change the file name
      * dh dh\_filename.pem (dh.pem)
    - below look for user and group and uncomment the lines
      * user nobody
      * group nogroup
    - find the “ca ca.crt” part and add below
      * ca ca.crt
      * cert vpnservercertificate.cert
      * key vpnserverkeyfile.key

starting and enabling the openvpn service

* systemctl start openvpn@server (server.conf)

creating the client configuration infrastructure

* mkdir -p ~/client-configs/files
* cp /usr/share/doc/examples/sample-config-files/client.conf ~/client-configs/clientconffilename.conf
* edit the client config file
  + nano ~/client-config/clientconffilename
    - uncomment user and group
    - specify the server ip
      * remote “vpn server ip”
    - got to filepaths
      * comment out the three files
        + #ca ca.crt and the others
    - Go to tls-auth and comment it out
      * #tls-auth
    - Go to the cipher line and add below
      * auth SHA256
    - At the end of the file add
      * key-direction 1
* Create the ovpn file
  + Create a file that has the following content
    - clientconffile.conf content
    - <ca> ca.crt content </ca>
    - <cert> vpnservercertfile.crt content </cert>
    - <key> vpnserverkeyfile.key content </cert>
    - <tls-auth> filename1.key (hmac signature file) content </tls-auth>
  + cat clientconffile.conf >> files/clientfqdnfilename.ovpn
  + echo’<ca>’ >> files/clientfqdnfilename.ovpn
  + cat keys/ca.crt >> files/clientfqdnfilename.ovpn
  + echo’</ca>’ >> files/clientfqdnfilename.ovpn
  + echo’<cert>’ >> files/clientfqdnfilename.ovpn
  + cat keys/vpnservercertfile.crt >> files/clientfqdnfilename.ovpn
  + echo’</cert>’ >> files/clientfqdnfilename.ovpn
  + echo’<key>’ >> files/clientfqdnfilename.ovpn
  + cat keys/vpnserverkeyfile.key >> files/clientfqdnfilename.ovpn
  + echo’</key>’ >> files/clientfqdnfilename.ovpn
  + echo’<tls-auth>’ >> files/clientfqdnfilename.ovpn
  + cat keys/filename1.key >> files/clientfqdnfilename.ovpn
  + echo’</tls-auth>’ >> files/clientfqdnfilename.ovpn

Client configuration

* on windows (.net framework 4.0)
  + install the secure openvpn software
  + copy the clientfqdnfilename.ovpn file into the config folder in the openvpn programs folder
  + connect the interface
* on linux
  + copy the clientfqdnfilename.ovpn to the machine
  + just above the start of the <ca> block add
    - script-security 2
    - up /etc/openvpn/update-resolve-conf
    - down /etc/openvpn/update-resolve-conf
    - run the connection
      * openvpn --config clientfqdnfilename.ovpn