Simulacre Prova Final UF2 Seguretat

PKI

L'objectiu del següent exercici és implementar un servidor HTTP**S** mitjançant una clau privada ("server.key") i un certificat de servidor ("server.crt") associat als noms "*.pepito.com" i "pepito.com", el qual estarà signat per la clau privada d'una CA pròpia ("ca.key"); lògicament, el certificat arrel d'aquesta CA ("ca.crt") caldrà integrar-lo als clients per a què hi confiin.

En lloc d'utilitzar la suite OpenSSL, ja coneguda, per generar els artefactes anteriors ("ca.crt", "ca.key", "server.crt" i "server.key"), en el següent exercici es demanarà fer servir una comanda especialitzada justament en aquestes tasques que no hem vist fins ara: *Step* (https://smallstep.com/cli); al llarg de l'enunciat s'anirà indicant com utilitzar-la.

> NOTA: L'ajuda bàsica de per a què serveix i com funciona aquest programa es troba a https://smallstep.com/docs/step-cli/basic-crypto-operations i la seva referència de comandes es troba a https://smallstep.com/docs/step-cli/reference/certificate/create
> NOTA: Existeix una altra comanda anomenada *step-ca* però està molt més especialitzada en la gestió d'una CA i no ens caldrà utilitzar-la. En tot cas, la seva documentació es pot trobar a https://smallstep.com/docs/step-ca i a https://smallstep.com/docs/step-ca/getting-started
> NOTA: Existeixen altres eines, a banda d'step, que simplifiquen molt la creació de certificats, CSRs, etc. En podem destacar el formulari web https://certificatetools.com i el servei online complet https://pkiaas.io No les veurem, però.

1.-PREVI: En aquest exercici caldrà que utilitzis dues màquines virtuals (amb un sistema operatiu de tipus "Server") que es puguin fer "ping" entre sí (per simplificar l'exercici, es recomana que les tarjes de xarxa d'ambdues màquines estiguin configurades en el mode "adaptador pont"): una màquina farà de servidor HTTPS i l'altra màquina farà de client mitjançant la comanda *curl* A la màquina servidora, doncs, instal·la el paquet "apache2" (a Ubuntu) o els paquets "httpd" i "mod_ssl" (a Fedora)

> NOTA: En qualsevol cas, també caldrà assegurar-te que el tallafocs estigui apagat (*sudo systemctl stop ufw* en Ubuntu, *sudo systemctl stop firewalld* en Fedora)
> NOTA: Si estàs a Fedora, hauràs d'executar a més la comanda *sudo setsebool -P httpd_read_user_content 1* (per tal d'indicar al framework de seguretat SELinux que permeti a lApache llegir el certificats+claus)

a) En una màquina qualsevol, executa les següents comandes per instal·lar la comanda *step*:

*A Ubuntu: *wget https://dl.smallstep.com/cli/docs-ca-install/latest/step-cli_amd64.deb*
sudo dpkg -i step-cli_amd64.deb

> NOTA: Per desinstal·lar el programa, caldria fer *sudo dpkg -r step-cli* i tot seguit *rm -rf $HOME/.step*

* A Fedora: *wget https://dl.smallstep.com/cli/docs-ca-install/latest/step-cli_amd64.rpm*
sudo rpm -i step-cli_amd64.rpm

> NOTA: Per desinstal·lar el programa, caldria fer *sudo dnf remove step-cli* i tot seguit *rm -rf $HOME/.step*

Tot seguit, executa-hi la comanda següent, la qual crea un certificat autosignat anomenat "ca.crt" (juntament amb una clau privada anomenada "ca,key").

step certificate create --profile root-ca "MyCA" ca.crt ca.key

> NOTA: Aquest CRT no el farem servir com un certificat de servidor estàndar sinó que funcionarà, tal com es pot veure pel valor del seu paràmetre "--profile"- com a certificat arrel (per així poder actuar com una CA privada i generar per tant certificats pels nostres diferents servidors, com per exemple el servidor HTTPS); és per això que no caldrà que tingui cap camp "SAN" ja que no el vincularem a cap nom DNS (el nom "MyCA" representa el "Common Name" que tot certificat ha de tenir)

Finalment, executa-hi la comanda següent, la qual crea un certificat signat per "ca.key" anomenat "serverhttps.crt" (juntament amb una clau privada anomenada "serverhttps.key") vàlid per un any i associat als noms DNS "pepito.com", "www.pepito.com" i "cloud.pepito.com"

step certificate create --profile leaf "pepito.com" --san "www.pepito.com" –san "cloud.pepito.com" serverhttps.crt serverhttps.key --ca ca.crt --ca-key ca.key --not-after=8760h --no-password --insecure

NOTA: Els paràmetres *--no-password --insecure* serveixen per no haver d'assignar cap passphrase a la clau privada creada
NOTA: Es pot comprovar tots els detalls del certificat recentment creat amb la comanda *step certificate inspect server.crt*
NOTA: La comanda anterior genera directament el certificat signat per la CA però també es podria fer el mateix procés explícitament en dues passes: primer crear el CSR (simplement afegint-hi a la comanda *step certificate create* anterior el paràmetre *--csr* i esborrant els paràmetres *--ca* i *--ca-key*) i posteriorment signar-lo amb els fitxers "ca.crt" i "ca.key" mitjançant la comanda *step certificate sign*

b) A la màquina que farà de servidor HTTPS, realitza els passos necessaris per oferir via aquest protocol un "VirtualBox" accessible via "www.pepito.com". Concretament:

1.- Crea un arxiu anomenat "pepito.conf" dins de la carpeta "/etc/httpd/conf.d" (a Fedora) o "/etc/apache2/sites-available" (a Ubuntu) amb el següent contingut (on "serverhttps.crt" i "privatehttps.key" són el certificat i clau privada del servidor, creats al darrer pas de l'apartat anterior, i que hauràs de copiar a la ubicació indicada)

```
<VirtualHost *:443>
ServerName www.pepito.com
DocumentRoot /var/www/pepito
SSLEngine on
SSLCertificateFile /etc/ssl/serverhttps.crt
SSLCertificateKeyFile /etc/ssl/privatehttps.key
</VirtualHost>
```

2.- Assigna a "privatehttps.key" els permisos 640 i el propietari "root" i a "serverhttps.crt" els permisos 644 i el propietari "root" també

3.- Crea (com a root) la carpeta "/var/www/pepe" i crea-hi allà dins un arxiu anomenat "pag.html" amb el contingut següent: *<html><body>Hola, sóc Pepito</body></html>*

4.- (Només a Ubuntu): executa les comandes *sudo a2enmod ssl* i *sudo a2ensite pepito*

5.- Reinicia la configuració del servei (*sudo systemctl reload apache2* o *sudo systemctl reload httpd*)

c) A la màquina client, fes que es resolgui el nom "www.pepito.com" a la IP de la màquina servidora (recorda que la manera ràpida i fàcil de fer-ho és editant convenientment el seu arxiu "/etc/hosts") i tot seguit prova d'executar les següents comandes. Raona el perquè del resultat obtingut en cadascuna.

* *curl https://www.pepito.com/pag.html*

* *curl -k https://www.pepito.com/pag.html*

* *curl --cacert ca.crt https://www.pepito.com/pag.html*
(on "ca.crt" és el certificat arrel creat al primer apartat d'aquest exercici)

* *curl --ca-native https://www.pepito.com/pag.html*
(després d'haver integrat "ca.crt" dins del "trust store" general del sistema; això ho pots fer simplement amb la comanda *sudo step certificate install ca.crt* o bé, més manualment, copiant el fitxer "ca.crt" a la carpeta "/usr/local/share/ca-certificates" -a Ubuntu- o "/etc/pki/ca-trust/extracted" -a Fedora- i tot seguit executant la comanda *sudo update-ca-certificates* -a Ubuntu- o *sudo update-ca-trust* -a Fedora-)

TOR

2.-a) Arrenca una màquina virtual VirtualBox (Ubuntu o Fedora, Server o Desktop, és igual) amb la tarja de xarxa en mode "adaptador pont" i instal.la-hi el paquet "tor". Configura, a l'arxiu "/etc/tor/torrc", la seva directiva *SOCKSPort* per a què en lloc d´escoltar a la IP "loopback" escolti a la IP de la tarja de xarxa i tot seguit posa en marxa el dimoni.

b) Arrenca una segona màquina virtual amb la tarja de xarxa en mode "adaptador pont" i instal.la-hi el paquet "torsocks". Configura'l per a què es connecti al servidor Tor remot que està funcionant a l´altra màquina virtual. Finalment executa la comanda *curl https://check.torproject.org/api/ip* i tot seguit *torsocks curl https://check.torproject.org/api/ip* ¿Quina diferència hi ha entre el resultat de la primera i la segona?

bII) Si tornes a executar de nou la mateixa comanda *torsocks* de l'apartat anterior, després d'haver reiniciat el servei tor, ¿la IP obtinguda en aquesta segona execució és la mateixa que la que vas obtenir a l'apartat anterior? Per què?

bIII) ¿Quina diferència hi ha entre executar *curl -x socks5://ip.serv.Tor:9050 www.hola.com* i *curl –x socks5h://ip.serv.Tor:9050 www.hola.com* ? En aquest sentit, ¿l'ús de *torsocks* és prescindible si fem servir *curl* d'alguna de les maneres anteriors?

c) Configura ara el servidor *tor* de la primera màquina virtual per a què només utilitzi nodes d'entrada d'Alemanya i nodes de sortida de Portugal. Després de reiniciar-lo, obre el navegador Firefox de la màquina real, configura'l per a què utilitzi com a proxy SOCKS el teu servidor *tor* i vés a https://check.torproject.org ¿De quin país és la IP que uses ara? (ho pots comprovar consultant-la a webs com https://geoip.com)

d) ¿Quines directives hauries d'indicar al fitxer de configuració de Tor (no cal que ho facis, només explica-les) per convertir el teu servidor tor en un "exit relay"? ¿I en un "bridge" (sense "pluggable transport")?

> NOTA: Les instruccions oficials per implementar un "bridge" amb el PT "Obfs4" incorporat es poden consultar a https://community.torproject.org/relay/setup/bridge/fedora (si el sistema utilitzat és Fedora) o https://community.torproject.org/relay/setup/bridge/debian-ubuntu (si el sistema utilitzat és Ubuntu). Una altra guia similar però més completa és https://sigvids.gitlab.io/create-tor-private-obfs4-bridges.html

3.-a) Instal.la un servidor Apache2 a la mateixa màquina virtual on tens funcionant el servidor *tor* de l'exercici anterior. Fes que l'Apache2 només escolti a través de la IP loopback (això ho pots fer indicant la directiva *Listen 127.0.0.1:80* en lloc d'on estigui present la directiva Listen en la configuració actual del servidor...a Ubuntu és a l'arxiu "/etc/apache2/ports.conf" i a Fedora a l'arxiu "/etc/httpd/conf/httpd.conf").

b) Edita les directives adients de l'arxiu "/etc/tor/torrc" per a fer que el servidor Apache2 estigui disponible públicament al port 80 dins de la xarxa Tor com a servei "onion",

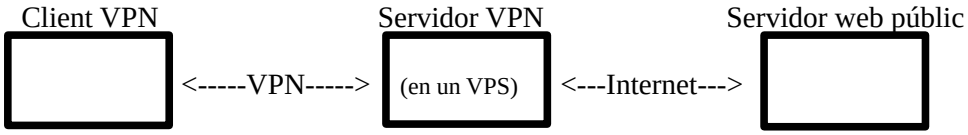
c) Accedeix des de la màquina "client" (bé amb el "Tor Browser" o bé mitjançant un curl "torificat") al domini ".onion" del teu servidor web autogenerat a l'apartat anterior per confirmar que la pàgina per defecte de l'Apache es troba visible. Comprova també que des d'un navegador estàndar sense fer servir la xarxa Tor aquest domini ".onion" no porta enlloc.

4.-Segueix els passos indicats a https://community.torproject.org/relay/setup/guard/debian-ubuntu (si la teva màquina virtual de treball és Ubuntu) o https://community.torproject.org/relay/setup/guard/fedora (si la teva màquina virtual de treball és Fedora) per implementar un relay d'entrada o intermig. Comprova amb l'eina "Nyx" (i les seves múltiples pantalles: d'"events", de "connexions", etc) que efecticament aquest node estigui treballant correctament.

VPN WIREGUARD

5.-PREVI: Arrenca una màquina virtual VirtualBox (Ubuntu o Fedora, és igual, però millor Server per no consumir massa recursos) i instal.la-hi el paquet "wireguard-tools" i, si calgués, també el paquet "nftables" i "curl". Apaga-la i, en el cas de fer servir VirtualBox, clona-la (de forma "enllaçada" i reinicialitzant la MAC de la tarja de xarxa) un cop, i tot seguit, clona-la de nou un altre cop (també de forma "enllaçada i reinicialitzant la MAC de la tarja de xarxa). Hauràs de tenir finalment, doncs, tres màquines amb el mateix software instal·lat.

El que es pretén en aquest exercici és realitzar aquesta configuració:



Edita la primera màquina virtual per a què tingui dues tarjes de xarxa, la primera (*enp0s3*) en mode "Xarxa interna" que anomenarem "VPN" i la segona (*enp0s8*) en mode "Xarxa interna" també però que anomenarem "Internet". Edita la segona màquina virtual per a què tingui la seva única tarja de xarxa en el mode "Xarxa interna" anomenada "VPN". Edita la tercera màquina virtual per a què tingui la seva única tarja de xarxa en el mode "Xarxa interna" anomenada "Internet"

a) Implementa primer simplement la connectivitat entre el client VPN i el servidor HTTP (que es troben en xarxes diferents) fent servir el servidor VPN com a porta d'enllaç (que pertany a les dues xarxes). És a dir, l'objectiu d'aquest apartat és només que es facin "ping" entre el client VPN i el servidor HTTP, sense cap VPN encara. Per aconseguir això, realitza els següents passos:

1.-Arrenca la màquina que farà de servidor VPN i assigna (de la manera que vulguis) la IP 8.0.0.1/8 a la seva tarja *enp0s3* i la IP 16.16.0.1/16 a la seva tarja *enp0s8* (o *enp2s0*);

2.- Encara al servidor VPN, activa-hi l´"IP-Forward" tal com ja s'ha vist als exercicis de classe

NOTA: No establirem cap regla NAT perquè estem suposant que totes les IPs involucrades són públiques (la del "router" del client VPN, la del servidor VPN perquè està ubicat en un VPS contractat i la del servidor HTTP perquè és un servidor públic d'Internet)

3.- Arrenca la màquina que farà de client VPN i assigna (de la manera que vulguis) la IP 8.0.0.2/8 a la seva tarja *enp0s3*

4.- Arrenca la màquina que farà de servidor HTTP i assigna (de la manera que vulguis) la IP 16.16.0.2/16 a la seva tarja *enp0s3*

5.- Fes que la porta d'enllaç del client VPN sigui la IP de la tarja *enp0s3* del servidor VPN tal com ja s'ha vist als exercicis de classe. Igualment, fes que la porta d'enllaç del servidor HTTP sigui la IP de la tarja *enp0s8* del servidor VPN

b) Ara ja es pot implementar el túnel VPN pròpiament dit entre la màquina client i el servidor. Per aconseguir això, realitza els següents passos:

1.- Al servidor VPN crea les seves claus pública i privada, tal com indica la teoria de classe. Igualment, al client VPN crea igualment les seves claus pública i privada.

2.- Edita al servidor VPN l'arxiu "/etc/wireguard/wg0.conf" adientment per a què estigui disponible a la VPN amb la IP 192.168.3.1 (el port d'escolta pot ser qualsevol) i només tingui un "peer" reconegut amb la IP 192.168.3.2/32 (el qual representa el client VPN). Activa aquesta configuració per a què la nova tajra *wg0* sigui operativa.

3.- Edita al client VPN l'arxiu "/etc/wireguard/wg0.conf" adientment per a què estigui disponible a la VPN amb la IP 192.168.3.2 i només tingui un "peer" reconegut, que serà el servidor VPN. Pel túnnel VPN has d'indicar que vols que s'hi transmeti tot el tràfic generat pel client, tingui el destí que tingui. Activa aquesta configuració per a què la nova tarja *wg0* sigui operativa (si tens Gnome funcionant, això ho pots fer directament anant a l'apartat "VPN" de la secció de "Xarxa" del panell de control de Gnome)

c) Fes les següents comprovacions per confirmar que el túnnel VPN s'ha realitzat correctament:

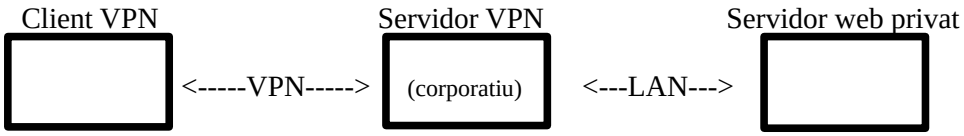
***** Comprova que la tarja *wg0* del client pot fer "ping" a la del servidor executant (a la màquina client) la comanda *ping -I 192.168.3.2 192.168.3.1*.

***** Comprova que també pugui fer "ping" al servidor HTTP executant *ping -I 192.168.3.2 16.16.0.2*

* Comprova que, de fet, no cal ni indicar que la tarja d'origen sigui "wg0" executant *ping 16.16.0.2*

* Executa a la màquina servidor HTTP la comanda *python -m http.server 4321* Tot seguit, comprova al client VPN que pots accedir a la pàgina inicial d´aquest servidor HTTP executant la comanda *curl http://16.16.0.2:4321*

6.-PREVI: El que es pretén en aquest exercici és realitzar aquesta configuració (amb les mateixes tres màquines utilitzades a l'exercici anterior):



a) Implementa primer simplement la connectivitat entre el client VPN i el servidor HTTP (que es troben en xarxes diferents i ara el servidor HTTP està en una xarxa privada) fent servir el servidor VPN com a porta d'enllaç (el qual pertany a les dues xarxes). És a dir, l'objectiu d'aquest apartat és només que es facin "ping" entre el client VPN i el servidor HTTP, sense cap VPN encara. Per aconseguir això, realitza els següents passos:

1.- Manté la mateixa adreça IP pel client VPN que la que tenia a l'exercici anterior (és a dir, 8.0.0.2/8), i la de la tarja *enp0s3* del servidor VPN també (és a dir, 8.0.0.1/8), però per ser una mica més realista en l'escenari, canvia les IPs de la tarja *enp0s8* del servidor VPN i la de la tarja *enp0s3* del servidor HTTP per a que siguin IPs privades. En concret, per a què siguin les IPs *172.16.0.1/16* i *172.16.0.2/16* , respectivament

2.- Assegura´t que l'"IP-Forwarding" es mantingui activat al servidor VPN

3.- Assegura't que la porta d'enllaç del client VPN continui essent la IP de la tarja *enp0s3* del servidor VPN . Igualment, fes que la porta d'enllaç del servidor HTTP (ara intern de la LAN) sigui la (nova) IP de la tarja *enp0s8* del servidor VPN

4.- Configura el tallafocs del sistema Nftables del servidor VPN per tal que incorpori les regles adients (tal com ja s'han estudiat als exercicis de classe) que permetran que aquest servidor pugui aplicar tant el SNAT (concretament, la regla "masquerade" dins de la cadena "postrouting") com el DNAT (concretament, la regla homònima de la cadena "prerouting") a la tarja que està connectada al món exterior (en aquest cas, doncs, *enp0s3*).

Abans de passar al següent apartat, comprova que el SNAT i el DNAT funcionen posant en marxa al servidor HTTP la comanda *python -m http.server 4321* i tot seguit comprovant en el client VPN que la comanda *curl http://8.0.0.1* pot accedir a la pàgina web inicial d'aquest servidor HTTP intern.

b) Ara ja es pot implementar el túnel VPN pròpiament dit entre la màquina client i el servidor. Per aconseguir això, realitza els següents passos:

1.- Al servidor VPN sustitueix la regla DNAT del tallafocs per a què afecti no a la tarja *enp0s3* (com hauràs fet a l'apartat anterior) sinó a la tarja *wg0*, i reinicia de nou el servei Nftables.

2.- Al servidor VPN no cal que canvïis res de la configuració escrita a l´arxiu "/etc/wireguard/wg0.conf", només cal que activis de nou la tarja *wg0* i ja està. Igualment, al client VPN no cal que tampoc canvïis res de la configuració escrita al seu arxiu "/etc/wireguard/wg0.conf": només cal que activis la seva tarja *wg0* i prou

c) Fes les següents comprovacions per confirmar que el túnnel VPN s'ha realitzat correctament:

***** Comprova que pots accedir, a través del servidor VPN (ara sí, VPN de veritat) a la pàgina web inicial del servidor HTTP intern executant al client VPN la comanda *curl http://192.168.3.1:4321*

NOTA: Una altra pràctica interessant seria implementar la comunicació via VPN de dos servidors interns de respectives LANs. No obstant, per fer això necessitaríem quatre màquines virtuals: les dels dos servidors interns i les dels dos servidors VPN (que en aquest cas farien a la vegada de client VPN de l'altre servidor VPN respectiu) i no ho farem. En tot cas, als següents enllaços s'explica prou bé com implementar aquesta configuració, la qual, insistim, consisteix bàsicament en implementar el tunel VPN entre dos servidors VPN on cadascun farà de client VPN de l'altre (deixant de banda els servidors interns, que no tindran constància de l'existència d'aquest túnel): https://sio2sio2.github.io/doc-linux/07.serre/04.vpn/02.wireguard/01.conf.html#sede-sede i https://staaldraad.github.io/2017/04/17/nat-to-nat-with-wireguard

TALLAFOCS NFTABLES

7.- Suposant que tenim una màquina hipotètica amb el servei Nftables configurat per llegir (només) els següents scripts cada cop que es posa en marxa, respon les següents preguntes:

Script "base.nft"

```
#!/usr/sbin/nft -f
flush ruleset
add table inet filter
add chain inet filter input { type filter hook input priority filter; policy drop;}
add chain inet filter forward { type filter hook forward priority filter; policy drop;}
add chain inet filter output { type filter hook output priority filter; policy drop;}
add rule inet filter input iifname lo accept
add rule inet filter output oifname lo accept
```

Script "regles.nft"

```
include "base.nft"
define ips_bones = 1.0.0.0/8
add rule inet filter input ip saddr $ips_bones log prefix \"Són de confiança\" accept
add rule inet filter input icmp type {echo-request, echo-reply} accept
add rule inet filter input tcp dport 22 ct state new accept
add rule inet filter input udp sport 53 accept
add rule inet filter output udp dport 53 accept
add rule inet filter output ct state established,related accept
```

a) ¿Aquest equip pot rebre "pings"? Si és que sí, els podrà rebre provinents de qualsevol lloc? ¿I pot enviar "pings"? Si és que sí, els podrà enviar a qualsevol lloc?

b) ¿Aquest equip pot navegar (és a dir, accedir als ports 80 i 443 de màquines externes)? Si és que sí, pot accedir a qualsevol servidor web extern?

c) ¿Quin/s servei/s dedueixes que esta/n funcionant en aquest equip?

d) ¿Per què és necessària la línia ... *output ct state established,related accept* ?¿Afecta a paquets UDP?

e) Si en aquest equip posés en marxa un servidor qualsevol escoltant al port 7777, ¿quines màquines remotes hi podrien accedir?

f) Suposant que en aquesta màquina ara executes les comandes *sudo nft flush ruleset && sudo nft -f fitxer.nft* (on el contingut de "fitxer.nft" és el mostrat a continuació), i que tens en marxa un servidor DHCP en aquesta màquina, ¿quin sentit tindria afegir-hi aquesta regla (on "x:x:x:x:x:x" i "y:y:y:y:y:y" representen direccions MACs concretes d'altres màquines)?: *sudo nft add rule ip filter input ether saddr { x:x:x:x:x:x , y:y:y;y:y:y } counter accept*

```
table ip filter {
        chain input {  type filter hook input priority filter; policy drop;
        }
        chain forward {  type filter hook forward priority filter; policy accept;
        }
        chain output {  type filter hook output priority filter; policy accept;
        }
}
```

8.- En aquest exercici hauràs d'utilitzar una màquina virtual VirtualBox qualsevol que tingui la seva tarja de xarxa en mode "adaptador pont" i on, a més, hauràs de posar en marxa un servidor Netcat escoltant al port 4444. Implementa-hi les regles Nftables necessàries per tal d'aconseguir el següent (els apartats a) i b) són independents!):

> NOTA: Recorda que per posar en marxa un servidor Netcat de forma permanent pots fer servir la comanda *ncat* que ve dins del paquet "nmap" així: *ncat -k -l -p 4444* o, si només es vol que s'escolti per una IP concreta i no totes les de la màquina, així *ncat -k -l adreça.ip.a.utilitzar 4444*

a) Aconsegueix que només les peticions (i respostes) de tipus "ping" (és a dir, paquets ICMP tipus 8 i 0), DNS (és a dir, paquets UDP al/del port 53) i HTTP/S (és a dir, paquets TCP als/dels ports 80 i 443) funcionin però cap més tràfic pugui sortir ni entrar del/al sistema. ¿Com ho comprovaries?

b) Aconsegueix que només s'hi pugui accedir al servidor Netcat des de la màquina real i cap altra. ¿Com ho comprovaries (recorda que la sintaxi del client Netcat en aquest cas seria *ncat adreça.ip.servidor 4444*)?

9.-Instal·la el següent programa en una màquina virtual qualsevol i menciona com a mínim tres funcionalitats seves que consideris interessants: https://github.com/safing/portmaster

SSH

10.-Arrenca una màquina virtual VirtualBox qualsevol però que tingui una tarja de xarxa en mode adaptador pont i instal.la-hi (i habilita'l, per a que s'inicïi automàticament a cada reinici) un servidor SSH. Crea, en aquesta màquina, un usuari anomenat "pepito". A partir d'aquí:

a) Configura aquest servidor SSH (emprant un fitxer anomenat "01-meu.conf" creat dins de la carpeta "/etc/ssh/sshd_config.d" per a no modificar el fitxer de configuració "sshd_config" per res) per a què:

* Escolti al port 2222
* Mostri un missatge genèric per tothom abans de loguejar-se contingut a l'arxiu "/opt/benvinguda"
* Quan detecti que es connecta l'usuari "pepito", que executi automàticament la comanda *ls -l*

b) Configura, tant aquest servidor com el client SSH que tindràs a la teva màquina real (o en una altra màquina virtual, si vols) per a què implementin l'autenticació per claus per l'usuari "pepito"

c) Configura el client SSH de la màquina real (emprant l'arxiu "~/.ssh/config" propi del teu usuari) per a què:

* En el cas (només!) de connectar-se al servidor anterior...
    ... es connecti automàticament al port 2222
    ... utilitzi l'usuari "pepito" sempre per defecte
    ... es pugui escriure un àlies anomenat "miserver" associat a la IP del servidor
* En el cas de connectar-se a qualsevol altre servidor..
    ... s'executi sempre en l'ordinador local la comanda *ls %d*

d) Utilitza la comanda *scp* per tal de copiar una carpeta qualsevol de la màquina client (la real) dins de la carpeta "/home/pepito" de la màquina servidora. Comprova-ho. Esborra tot seguit tota aquesta carpeta en local i, finalment, copia de nou amb *scp* la carpeta acabada de pujar al servidor SSH a la ubicació del client on era originalment.

11.-Implementa en una màquina virtual VirtualBox qualsevol un servidor SOCKS a partir d'un servidor SSH. Tot seguit implementa un túnnel des de la teva màquina real a aquest servidor SOCKS fent servir el client SSH de la teva màquina real. Finalment, configura el navegador de la teva màquina real per a què utilitzi aquest túnnel.

ALTRES TÚNELS

La llibreria "Global Socket Tookit" (https://www.gsocket.io), implementada en diversos programes que de seguida estudiarem, permet establir una connexió TCP a través d'Internet entre dos sistemes funcionant darrera d'un tallafocs NAT (és a dir, amb adreces IP privades!). Per aconseguir això utilitza com a nus d'"empalmament") la xarxa gratuïta "Global Socket Relay" (GSRN), la qual connecta entre sí les màquines que coneixen el mateix secret (en lloc de fer servir ni l'adreça IP ni el número de port de cadascú). Aquest secret s'utilitza per derivar-ne identificadors de sessió TLS temporals, de forma que el secret no abandona mai cada màquina i la GSRN només veu trànsit xifrat (aconseguint així que la connexió entre els dos extrems sigui confidencial i segura).

12.-Arrenca dues màquines virtuals VirtualBox qualssevol però que tinguin la seva respectiva tarja de xarxa en mode NAT (això és important!!) i segueix els següents passos per instal·lar el Global Socket Toolkit a cadascuna:

* A Ubuntu: Executa *sudo apt install gsocket*
* A Fedora: Executa *sudo dnf install git gcc make autoconf automake openssl-devel*
    bash -c "$(curl -fsSL https://gsocket.io/install.sh)"
    cd gsocket && sudo make install

a) Executa les següents comandes i digues què fan i per a què serveixen els paràmetres indicats (com a ajuda pots consultar *man gs-netcat*):

A la màquina A: *gs-netcat -li*
A la màquina B: *gs-netcat -i* i tot seguit executar qualsevol comanda com per exemple: *echo 'hola';id; exit*

NOTA: Una altra forma d'aconseguir quelcom similar però en aquest cas de forma no interactiva seria fer per exemple:
A la màquina A: *gs-netcat -l -e "echo 'hola';id; exit"*
A la màquina B: *gs-netcat*

b) Executa les següents comandes i digues què fan i per a què serveixen els paràmetres indicats (com a ajuda pots consultar *man gs-netcat*):

A la màquina A: *gs-netcat -s unsecret -lr > unfitxer.txt*
A la màquina B: *gs-netcat -s unsecret < unfitxer.txt*

> NOTA: Una altra comanda del GST més especialitzada en el trànsit de fitxers és *blitz*, utilitzada d'aquesta manera:
> A la màquina A (la que rebrà els fitxers): *blitz -l*
> A la màquina B (la que els enviarà): *blitz /usr/share/* /etc/**

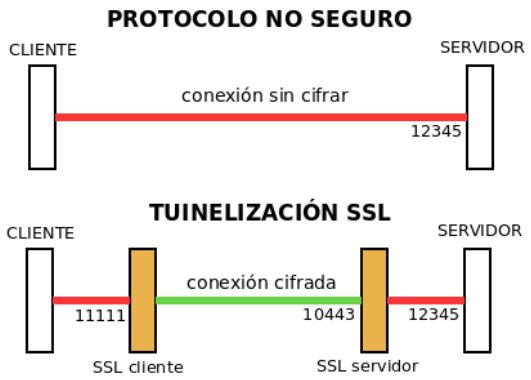
c) Executa les següents comandes i digues què fan i per a què serveixen els paràmetres indicats (com a ajuda pots consultar *man gs-netcat*). Important: a "Màquina A" (tot i que podria ser qualsevol altra màquina, però per no haver de tenir tres màquiens virtuals en marxa) hauràs de tenir, en aquest apartat, a més, un servidor SSH en marxa (tot i que podria ser de qualsevol altre tipus):

A la màquina A: *gs-netcat -l -d 127.0.0.1 -p 22*
A la màquina B: *gs-netcat -p 2222* i, en un altre terminal, *ssh -p 2222 usuari@127.0.0.1*

> Pista: Entre màquina A i B s'estableix un "túnel" a través de la GSRN gràcies a les comandes *gs-netcat* respectives. A la màquina A, aquesta comanda *gs-netcat* el que fa és reenviar tot el que li arribi al port local 22 (on se suposa que hi ha un servidor SSH en marxa). A la màquina B, aquesta comanda *gs-netcat* obre un port local, que servirà com a punt d'entrada al túnel, el qual és utilitzat per la comanda client ssh. D'aquesta manera, el client ssh connecta a través del túnel amb el servidor SSH de l'altre extrem (fent servir un usuari existent en aquest servidor, òbviament). En altres paraules: aquest és un exemple pràctic de com accedir des d'Internet (a través de la GSRN) a un servidor que només està escoltant realment de forma local (o com a molt, dins d'una xarxa privada)

> NOTA: El GST incorpora més comandes interessants a més de les vistes en aquest exercici (com *gs-mount* o *gs-sftp*). Totes elles tenen la seva corresponent pàgina del manual
> NOTA: D'altra banda, altres paràmetres interessants de la comanda *gs-netcat* són *-S* o *-D* (ambdós combinats amb *-l*)
> NOTA: També existeix una versió del GST mínima que només inclou *gs-netcat* i ja està, directament descarregable de https://www.gsocket.io/deploy
> NOTA: El codi per implementar un servidor "relay" de la GSRN es troba a https://github.com/hackerschoice/gsocket-relay
> NOTA: En el cas d'haver instal·lat GST amb la comanda *sudo make install*, per desinstal·lar-la caldrà fer (des de dins de la mateixa carpeta "gsocket"), la comanda *sudo make uninstall*

Amb *Stunnel* (https://www.stunnel.org) es pot implementar un túnel TLS entre dos extrems que a priori es comunicarien de forma insegura. Com es mostra al gràfic següent, la idea és interposar un client-servidor TLS "com un bolet" entre els client-servidor insegurs que es volen comunicar. Els dos clients estan executant-se a la mateixa màquina i els dos servidors en una altra distinta. D'aquesta manera, el trànsit de xarxa circula entre les dues màquines de forma xifrada. Tècnicament, el cliente final connecta amb el cliente TLS a través d'un port local (per exemple, el 11111) i és aquest port des d'on es contacta a través de la xarxa amb el servidor TLS de l'altre extrem (per exemple, al puerto 10443), el qual, al seu torn, connecta amb el servidor final a través d'un altre port local seu (per exemple, el 12345).



13.-Arrenca dues màquines virtuals VirtualBox qualssevol però que tinguin la seva respectiva tarja de xarxa en mode "adaptador pont". A partir d'aquí:

a) Crea, amb la comanda *openssl req -x509* ... (ja coneguda d'exercicis de classe) una clau privada i un certificat autosignat (amb un "Common Name" qualsevol i sense cap "Subject Alternative Name", no li cal)

> NOTA: Ambdós fitxers (clau i certificat) han de tenir com a propietari l'usuari "root" i permisos 600 i 640 respectivament

b) Posa en marxa el següent servei insegur en la màquina que farà de "servidora", el qual estarà escoltant només per la interfície "loopback" (en el port 12345): *ncat -k -l 127.0.0.1 12345*

> NOTA: Si tens dubtes sobre com utilitzar el programa ncat anterior, consulta la "NOTA" de l'exercici 8

c) Instal·la la Stunnel en aquesta màquina servidora (fent *sudo apt install stunnel4* a Ubuntu o *sudo dnf install stunnel* a Fedora). Tot seguit, configura'l creant l'arxiu "/etc/stunnel/stunnel.conf" amb el següent contingut:

[netcat-ssl]
cert = /ruta/certificat
key = /ruta/clauprivada
accept = IP.MAQUINA.SERVIDORA:10443
connect = 127.0.0.1:12345

> NOTA: Aquesta configuració el que fa és posar a escoltar Stunnel com a servidor TLS en el port 10443 de la IP pública de la màquina i trasllada la informació (en clar ja) al servei que estigui escoltant al port 12345 de la interfície local

d) Instal·la la Stunnel en la màquina client i configura'l creant l'arxiu "/etc/stunnel/stunnel.conf" amb contingut:

[netcat-ssl]
client = yes
accept = 127.0.0.1:11111
connect = IP.MAQUINA.SERVIDORA:10443

> NOTA: Per defecte, el client Stunnel accepta certificats autosignats o sense validar per cap CA!. Si això no es vol, caldria afegir les següents línies a la seva configuració:
> *CAfile=/ruta/ca.crt*
> *checkHost=nomServidor*
> *verifyChain=yes*

e) Executa a la màquina client la comanda *ncat 127.0.0.1 11111* prova d'escriure-hi alguna cosa. ¿Què passa?

> NOTA: Tot i que la comunicació es duu a terme perfectament, el fet que el netcat servidor sempre connecti amb el stunnel local fa que per a ell totes les connexions siguin locals i desconeix, per tant, quina és la IP del client amb què s'està comunicant (en el client passa el mateix, encara que en aquest cas és menys important). Per pal·liar això, la part servidor de stunnel s'hauria d'executar com un proxy transparent amb la línia *transparent=source*
> NOTA: A https://sio2sio2.github.io/doc-linux/98.apendice/01.cryto/03.aplicaciones/04.ssl.html#tunelizacion i també a https://etherarp.net/securing-connections-with-stunnel/index.html teniu més informació sobre com i per a què usar Stunnel

EXTRA: DNS SEGUR

Tal com s'explica a https://dnsprivacy.org/the_problem (i de forma més tècnica, a https://datatracker.ietf.org/doc/html/rfc7626), el protocol DNS és molt sensible a la falta de privacitat perquè tothom qui accedeixi al tràfic DNS pot veure a quins dominis hi estem accedint en qualsevol moment (i fins i tot, manipular les peticions i/o respostes!) ja que es transmet en clar. Per solucionar aquest problema, s'han dissenyat diferents estàndards que pretenen dotar de confidencialitat, autenticitat i integritat al protocol DNS gràcies a l'us de la criptografia; un d'ells és DNS-over-TLS (**DoT**, https://tools.ietf.org/html/rfc7858, on els servidors DNS hauran d'escoltar al port 853 TCP) i un altre és el DNS-over-HTTPS (**DoH**, https://tools.ietf.org/html/rfc8484 , on els servidors DNS hauran d'escoltar al port 443 TCP, com si fos un servidor HTTPS -que, de fet, ho és-)

NOTA: Un article tècnic molt interessant sobre DoH és https://daniel.haxx.se/blog/2018/06/03/inside-firefoxs-doh-engine
NOTA: Un article tècnic que desenvolupa les diferències entre DoT i un altre protocol molt interessant que proporciona (només) integritat al DNS anomenat **DNSSEC** és https://blog.apnic.net/2018/08/20/dnssec-and-dns-over-tls
NOTA: Malauradament, la capçalera SNI als missatges HTTPS també revela el nom del lloc web contactat per l'usuari, de manera que proporciona un canal de fuites similar per al trànsit web com les consultes DNS. Afortunadament, hi ha treball en curs al grup de treball TLS de l'IETF per xifrar el missatge "Client Hello" incloent el SNI, l'anomenat "TLS Encrypted Client Hello" o "ECH". Per saber-ne més, llegiu https://blog.cloudflare.com/announcing-encrypted-client-hello

14.-PREVI: El protocol DoH, tal com ja sabem dels paràgrafs anteriors, fa servir l'encriptació proporcionada "de sèrie" per HTTPS per utilitzar-lo com a canal a través del qual enviar les peticions DNS necessàries (i rebre'n les respostes). Està pensat per ser usat en navegadors, que ja fan servir HTTPS per defecte.

Cal tenir en compte, però, no tots els servidors DNS admeten aquest tipus de canal per establir comunicació amb els clients; és per això que els navegadors incorporen una llista (petita) de servidors DNS coneguts compatibles amb el protocol DoH, d'entre els quals es pot escollir el que desitgem utilitzar. Cal tenir llavors en compte, però, que el navegador en qüestió no farà servir pas els servidors DNS definits a nivell de sistema sinó els indicats a la seva configuració particular, de forma independent a la resta de programes del sistema!.

a) A la teva màquina real vés al menú "Preferències->General" del Firefox i allà pulsa el botó "Paràmetres de xarxa" que apareix al final de tot; al quadre que apareix activa l'opció "Habilita DNS sobre HTTPS" i al desplegable associat titulat "Utilitza el proveïdor", indica el valor "Personalitzat"; finalment, a la caixa de text que apareix titulada "Personalitzat" escriu la següent url: *https://doh.familyshield.opendns.com/dns-query* Amb això estàs indicant que vols que Firefox faci servir com a servidor DNS el servidor 208.67.220.123. Per tant, a partir d'ara, tot i que la comanda *resolvectl dns* continui mostrant els servidors DNS configurats al sistema general, el Firefox farà servir el servidor DoH que té ell configurat en particular, el qual "de regal", per cert, té banejats un conjunt de pàgines perilloses per la canalla (porno, de cases d'apostes, etc). De fet, pots provar d'anar per exemple a "www.betway.com" i hauries de veure la pàgina de "censura" de l'OpenDNS.

NOTA: Si es volgués utilitzar el protocol DoT, en canvi, caldria activar-lo per totes les aplicacions del sistema de forma global ja que afecta a totes elles. En concret, a la majoria de sistemes Linux això es fa configurant el resolver DNS Systemd-resolved (al seu arxiu de configuració "/etc/systemd/resolved.conf") però no entrarem en més detalls. Cal saber, però, que no tots els servidors DNS públics admeten l'ús de DoT, així que sovint al final s'acaben realitzant consultes DNS estàndards (i, per tant, no segures)

b) Canvia el servidor DoH configurat al Firefox per a què sigui qualsevol altre dels indicats a https://dnsprivacy.org/public_resolvers o també a https://www.privacyguides.org/en/dns ¿Trobes alguna diferència quan tornes a anar a "www.betway.com"? ¿Per què?

NOTA: Un altre servidor DoH que pots provar és l'indicat a https://my.nextdns.io/5fde1b/setup

Un mètode per navegar més segur consisteix en usar llistes negres que inclouen llocs maliciosos que realitzen "**tracking**" -també anomenat "**profiling**"- (és a dir, un seguiment dels nostres costums de navegació per vendre aquesta informació a agències de publicitat) o fins i tot que serveixen "**malware**" (és a dir, programes que infecten el nostre sistema, els quals, segons el tipus, poden deixar inutilitzat aquest o bé fer-lo servir com a "zombi" per atacs remots o bé poden robar dades sensibles, o espiar les accions de l'usuari, etc) o fan "**phishing**" (és a dir, suplantar un web oficial per un altre impostor i així poder, per exemple, robar dades personals com ara nº de tarja de crèdit, introduïdes allà per usuaris que no descobreixen el parany).

Si no es vol fer servir software de tercers (software antimalware, plugins "adblockers" al navegador, etc) es pot implementar un sistema de "blacklisting" artesà simplement fent servir l'arxiu "/etc/hosts" del nostre sistema simplement assignant a cada nom de lloc maliciós una IP inabastable (com 127.0.0.1 o 0.0.0.0). L'inconvenient d'aquest sistema és que no té granularitat (és a dir, només es poden bloquejar domini complets) i que és necessari actualitzar "a mà" regularment el fitxer (o bé utilitzar un script que automatitzi el procés agafant les actualitzacions des de la font o fonts escollides). D'altra banda, es té l'avantatge de no dependre d'un servei extern i es té el control absolut de les entrades a la llista; tampoc hi ha un retard perceptible a l'hora d'accedir a llocs mitjançant navegador encara que el fitxer hosts tingui milions d'entrades.

NOTA: Una altra opció seria fer, com ja s'ha vist a l'exercici anterior, consultes a servidors DNS que incloguin aquestes característiques de protecció (els quals redireccionen a una advertència explicativa quan bloquegen l'accés a un lloc i/o permeten aplicar filtres de continguts personalitzats); d'aquesta manera no haurem de mantenir la "llista negra" (però es té menys control). Alguns exemples són:

*Comodo Secure DNS: 8.26.56.26 i 8.20.247.20
*Dyn Internet Guide: 216.146.35.35 i 216.146.36.36
*FoolDNS: 87.118.111.215 i 213.187.11.62
*NextDNS: 45.90.28.201 i 45.90.30.201
*GreenTeam Internet: 81.218.119.11 i 209.88.198.133
*OpenDNS: 208.67.222.222 i 208.67.220.220. Per més: 208.67.222.123 i 208.67.220.123
*Norton ConnectSafe: 199.85.126.10 i 199.85.127.10.
Per pornografia: 199.85.126.20 199.85.127.20. Per més: 199.85.126.30 i 199.85.127.30

Hi ha una llista comparativa de funcionalitats entre aquests serveis molt útil a https://avoidthehack.com/best-dns-privacy

15.-A una màquina virtual VirtualBox qualsevol (Ubuntu o Fedora, tant se val) que tingui almenys una tarja de xarxa en mode "adaptador pont", descarrega't algun d'aquests arxius "hosts" prefabricats amb molts llocs "sospitosos"...:

https://someonewhocares.org/hosts/zero/hosts
https://winhelp2002.mvps.org/hosts.txt
http://pgl.yoyo.org/as/serverlist.php?showintro=0;hostformat=hosts (obtingut de https://pgl.yoyo.org/adservers)
https://github.com/mitchellkrogza/ultimate.hosts.blacklist
https://github.com/StevenBlack/hosts

NOTA: A https://filterlists.com pots trobar una "metallista" de moltes llistes de dominis potencialment perillosos (aquestes llistes poden venir en diversos formats, entre els quals el format de l'arxiu "hosts", per aprofitar-les en diversos programes d'"ad-block, tallafocs, etc). Més "metallistes" són https://blocklist-tools.developerdan.com/blocklists o https://firebog.net

...i escull un domini de la llista per escriure'l al navegador (que no tingui configurat cap proxy); observa si pots accedir-hi. ¿Què passa llavors si substitueixes l'arxiu "/etc/hosts" oficial del sistema pel fitxer descarregat: pots continuar accedint?

AdGuardHome (https://adguard.com/en/adguard-home/overview.html) és un proxy DNS que resol els dominis relacionats amb publicitat a la IP 0.0.0.0, amb la qual cosa aconsegueix treure tota la publicitat de la nostra xarxa. És a dir, és com si fos un gran arxiu "/etc/hosts" com els estudiats a l'exercici anterior però compartit per tots els ordinadors de la LAN. Així ho expliquen a la seva web:

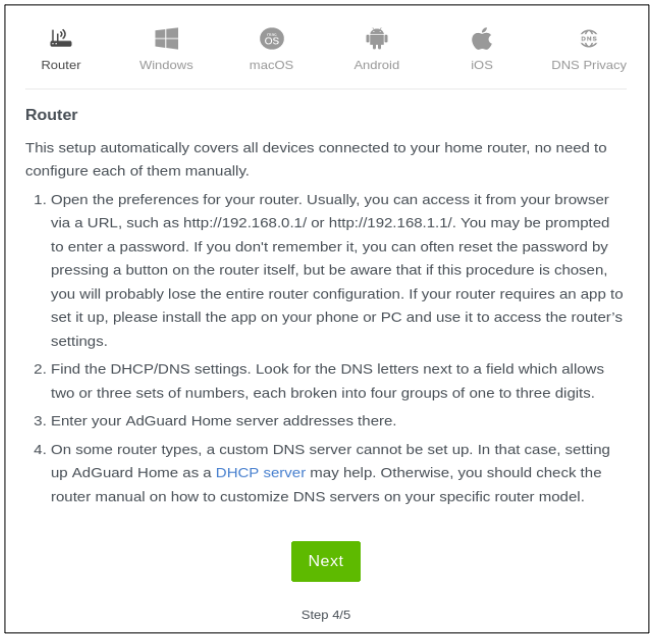
AdGuard Home is much like your personal browser ad blocker, but rather than being a plug-in on your favourite web browser, AdGuard Home is a fully-fledged server application which runs on a separate machine somewhere on your network (or perhaps even on a VPS you own). Its primary goal is to provide your entire network with a mechanism to actively block certain requests that websites you visit make – in this case, requests for adverts, malware, or various other malicious things. AdGuard Home is effectively a DNS proxy, whereby it acts as your network's primary DNS nameserver, filters requests, then relays the requests that satisfy its filters to an "upstream" DNS nameserver, ***which does the real DNS resolution***

AdGuard per defecte fa ús de vàries llistes públiques de dominis relacionats amb publicitat en general i d'altres relacionats amb malware, però també podem afegir les nostres pròpies llistes, o fins i tot les nostres pròpies expressions regulars per a què, per exemple, es bloquegin totes les pàgines web relacionades amb "porn". També es poden redireccionar directament peticions fetes a un domini cap a un altre (és a dir, fer "DNS Spoofing") en comptes de simplement derivar-les al servidor DNS "superior".

NOTA: Existeix un software semblant (i, de fet, més famós) anomenat Pi-Hole (https://pi-hole.net)

16.-Descarrega en una màquina virtual VirtualBox qualsevol amb la tarja de xarxa en mode adaptador pont, el paquet AdGuardHome adient (consulta https://github.com/AdguardTeam/AdGuardHome/wiki/Getting-Started per saber com fer-ho) i executa en un terminal el binari obtingut per tal d'accedir (remotament via web) a l'assistent inicial de configuració (tal com ja se t'indicarà a pantalla). Un cop dins de l'assistent, indica a totes les seves pantalles els valors que tu vulguis excepte a la primera, on has d'indicar que tant el servidor HTTP com el servidor DNS integrats en AdGuardHome escoltin en totes les adreces IPs (per aconseguir això últim hauràs d'aturar primer el servei "systemd-resolved" del teu sistema).

a) Una de les pantalles de l'assistent inicial de configuració de AdGuardHome és la següent. Explica amb les teves pròpies paraules el que s'hi explica en la captura següent.



b) Un cop finalitzat l'assistent, vés a la pantalla inicial del panell web d'AdGuardHome. ¿Què indiquen les estadístiques "Top clients", "Top queried domains" i "Top blocked domains"? ¿Què vol dir el valor "Average processing time"?

c) Arrenca una segona màquina virtual VirtualBox qualsevol (però que tingui entorn gràfic!) també amb la seva tarja de xarxa en mode adaptador pont i configura de forma temporal amb la comanda *sudo resolvectl dns enp0s3 ip.primera.maq.virtual* el servidor DNS que emprarà la teva tarja enp0s3 per a què aquest sigui el servidor AdGuardHome funcionant a la primera màquina virtual. Comprova-ho observant la sortida de les comandes *resolvectl dns* i *resolvectl query elpuig.xeill.net*_

A l'apartat "Filters"-> "DNS blocklists" del panell web d'AdGuardHome apareix un conjunt de diverses llistes de dominis DNS sospitosos (per bloquejar-los). Aquestes llistes (que es poden afegir o eliminar del conjunt segons el que necessitem) normalment estaran referenciades amb una URL externa per tal de mantenir-se actualitzades en tot moment. Poden estar escrites en dos formats: el del clàssic arxiu "hosts" (com és el cas del filtre que ja ve incorporat, tot i que no actualitzat per defecte, anomenat "AdAway") i un de més complex basat en regles anomenades "AdBlock" (perquè el software que primer les va implementar es deia així).

d) Un cop llegit el paràgraf blau anterior, activa totes les llistes possibles de dominis sospitosos que AdGuardHome ofereixi (n'hi ha moltes!) i, de pas, actualitza-les. A partir d'aquí, obre el navegador de segona màquina virtual i visita des d'allà la pàgina "www.elpais.com" o "www.hola.com" o qualsevol altra web que se t'acudeixi ¿Què veus de diferent del normal i per què? D'altra banda, visita la pàgina "gecko.me" o "liadm.com" ¿Què veus aquí i per què?

Per si les llistes anteriors no fossin prou, també existeix la possibilitat d'afegir llistes pròpies de destins sospitosos creades per nosaltres mateixos

e) Vés ara a l'apartat "Filters"-> "Custom filtering rules" de la configuració d'AdGuardHome i a partir de llegir l'ajuda que se't mostra en la pròpia pàgina (i els enllaços indicats a la "NOTA") esbrina com prohibir la navegació al domini "microsoft.com" (i tots els seus possibles subdominis). Fes-ho i comprova-ho des del navegador de segona màquina virtual

NOTA: La sintaxi de les regles de filtratge de tipus "AdBlock" que es poden afegir a l'apartat anterior es pot consultar https://github.com/AdguardTeam/AdGuardHome/wiki/Hosts-Blocklists També són interessants els articles https://kb.adguard.com/en/general/how-to-create-your-own-ad-filters i https://kb.adguard.com/en/general/dns-filtering-syntax

A vegades, però, és més fàcil directament bloquejar un determinat "servei" (Facebook, Youtube, etc) sense tenir en compte el domini concret al què s'estigui anant ("facebook.es", "facebook.com", etc). Aquesta possibilitat és molt interessant perquè hi ha serveis que són accessibles des de molts dominis diferents i si els volguéssim especificar "a mà" segur que ens oblidaríem d'algun

f) Vés ara a l'apartat "Filters"-> "Blocked services" de la configuració d'AdGuardHome i bloca Facebook. Comprova que des del navegador de segona màquina virtual no pots navegar ni a "www.facebook.com" ni a "www.facebook.es" ni a "facebook.com", per exemple.

Una altra opció interessant de l'"AdGuardHome" és la possibilitat de modificar les respostes DNS a determinades peticions per tal de redireccionar el client a un destí que no es correspon al destí realment demanat. Això és el que se'n diu "DNS Spoofing" i és la base dels atacs de "phishing".

g) Vés ara a l'apartat "Filters"-> "DNS rewrites" de la configuració d'AdGuardHome i fes que totes les peticions a "*.marca.com" es redirigeixin a "elpuig.xeill.net". Comprova què passa en intentar visitar des del navegador de segona màquina virtual la pàgina "www.marca.com" (o qualsevol subdomini relacionat)