Antivirus "ClamAV"

ClamAV (https://github.com/Cisco-Talos/clamav) és un software multiplataforma, gratuït i "open-source" (tot i que desenvolupat principalment per l'empresa Cisco) que permet detectar i reconèixer diferents tipus d'amenaces pel nostre sistema (com ara virus, troians, adware, rootkits i malware en general) gràcies a mantenir actualitzada a diari una gran base de dades pròpia on aquests elements maliciosos s'hi troben llistats

NOTA: ClamAV és capaç de detectar fins i tot les amenaces que estan contingudes dins d'arxius contenidors (com són els de tipus "zip", "tar", "rar", "7z", etc) o que estan incrustades en forma de "macros" en l'interior d'arxius genèrics (com són els de tipus "docx", "xlsx", "pdf", etc

Existeixen diversos mecanismes per detectar malware. Entre ells, l'anomenat **anàlisi estàtic**, el qual consisteix en examinar la mostra de malware sense arribar a executar-lo; és a dir, representa un conjunt de tècniques que pretenen obtenir tota la informació possible del fitxer estudiat a partir de les seves característiques intrínseques, com ara la seva mida, el seu tipus de fitxer, el seu hash MD5/SHA1/SHA256 (que serveix com a identificador únic del fitxer inspeccionat particular, identificador que pot haver estat ja marcat com a "sospitós" en bases de dades de malware) i/o, sobre tot, el valor binari o de cadena d'alguna (o algunes) seccions concretes del seu contingut (les anomenades "signatures", que serveixen per corroborar la càrrega maliciosa que hi pot haver en alguna zona de l'interior del fitxer inspeccionat particular). Aquest tipus d'anàlisi és el que solen fer els "antivirus" com el ClamAV.

NOTA:Fins i tot, amb les eines adients, es pot arribar a "desassamblar" el codi del fitxer estudiat per arribar a conèixer així les instruccions originals del programa i, per tant, saber de primera mà què fa i com funciona

NOTA: No obstant, els autors de malware sovint intenten dificultar la feina dels analistes amb tot un seguit de contratècniques que els antivirus i similars han de saber reconèixer i neutralitzar, com per exemple:

- Ofuscació : operació de dificultar el màxim possible que el contingut textual d'un binari pugui ser llegida. En el món del desenvolupament de programari maliciós és vital ocultar cadenes significatives com solen ser URLs, claus de registre de Windows, etc i er aconseguir-ho, en molts casos s'utilitzen estàndards criptogràfics.
- "Binding" : operació d'unir el programari maliciós a una altra aplicació legítima
- Ús de "crypters" i "empaquetadors": són eines i tècniques que s'utilitzen per xifrar/comprimir/empaqueta un programari maliciós i evitar així que l'antivirus vegi el seu interior.

D'altra banda, a més de l'anàlisi estàtic, per detectar malware una altra tècnica habitual és fer l'**anàlisi dinàmic**; en aquest cas, el que s'examina és el comportament del binari en execució, observant tots els artifactes que hi pugui generar (a més de al disc, a la memòria RAM també!), els diferents processos que posi en marxa, les connexions i el tràfic de xarxa que estableixi, etc. Per fer aquest anàlisi ja es requereixen eines més complexes, especialitzades i multidisciplinars que, a més, han d'estar ben protegides dins d'un laboratori aïllat per evitar que l'execució del malware pugui infectar més màquines de l'estrictament dedicada al seu estudi (és per això que se solen anomena "sandbox").

Per a què ClamAV pugui, doncs detectar, identificar, analitzar i classificar malware present al sistema, aquest programa incorpora de sèrie una base de dades oficial de "**signatures**" (és a dir, tant de **hashes de fitxers** públicament reconeguts com a maliciosos com de **seqüències de valors hexadecimals** concrets també reconeguts públicament com a maliciosos -els quals es poden correspondre a cadenes de text, o simplement a tires binàries- i que poden aparèixer en alguna part del contingut de fitxers qualsevol). Aquestes signatures (les quals poden venir indicades amb comodins o altres modificadors, i venir acompanyades d'altres elements específics com metadades, etc...tal com estudiarem de seguida) ClamAV les utilitza per comparar-les amb les signatures trobades en aquell precís moment durant l'escaneig als fitxers inspeccionats de la nostra màquina, per així trobar alguna coincidència maliciosa, si n'hi hagués.

NOTA: Cal dir que ClamAV permet també definir signatures personalitzades, les quals es poden afegir de forma local a les ja definides en la base de dades oficial, obtinguda gratuïtament dels servidors de signatures de ClamAV disponibles a Internet (o d'altres orígens fiables).

NOTA: Les signatures també es poden combinar entre sí (amb associacions del tipus "i", "o", "no", etc) per tal de construir signatures més complexes (anomenades "lògiques")

NOTA: Els valors "coneguts" de malware ja existent s'han conegut gràcies a haver realitzar un anàlisi dinàmic (és a dir, a haver estudiat el comportament del malware executat dins d'un entorn controlat) amb l'ajuda d'eines com Radare2/Ghidra i/o Wireshark, entre altres (a aquest anàlisi a vegades també se li anomena "heurístic" perquè investiga el comportament del programa). Com ja hem dit, ClamAV només fa anàlisi estàtic (és a dir, estudia l'estructura interna del malware sense executar-lo), així que molts cops el "mèrit" no està tant en escriure la signatura en sí sinó en l'anàlisi previ per trobar els patrons a indicar-ne.

Per instal·lar ClamAV haurem d'executar les següents comandes:

sudo apt install clamav clamav-daemon clamav-freshclam (A Ubuntu)
sudo dnf install clamav clamd clamav-update (A Fedora)

ClamAV té una arquitectura interna de tipus client-servidor. En aquest sentit, cal saber que el paquet "clamav" de Fedora inclou, a més de les comandes de terminal que de seguida veurem, totes les definicions dels diferents serveis Systemd de ClamAV funcionant com a client, però el servei *clamd* específicament (és a dir, "la part servidora") és proporcionat per separat, en el paquet "clamd". A Ubuntu/Debian, en canvi, el paquet "clamav" només inclou comandes de terminal mentre que tots els serveis (tant els de client com el de servidor) estan disponibles en el paquet "clamav-daemon"

D'altra banda, el paquet "clamav-data" (instal·lat automàticament com a dependència del paquet "clamav", tant a Ubuntu/Debian com a Fedora) conté la base de dades inicial de signatures de malware (és a dir, el conjunt de hashes de fitxers i de patrons binaris o de cadena que identifiquen inequívocament cada peça de software maliciós conegut) sobre la que ClamAV intentarà trobar-hi coincidències amb els fitxers presents al sistema quan faci els escanejos del disc; aquesta base de dades s'actualitza regularment i de forma automàtica (comunicant-se amb un servidor central ubicat a Internet i proporcionat per Cisco gratuïtament) gràcies a una tasca programada que, atenció, estarà present al sistema només si s'instal·la el paquet "clamav-update" (a Fedora) o "clamav-freshclam" (a Ubuntu/Debian)

> NOTA: Si es vol saber com es faria una instal·lació de ClamAV a partir del codi font (en sistemes Debian), un bon tutorial és https://kifarunix.com/install-clamav-on-debian-11
> NOTA: Un altre projecte anti-malware per Linux interessant (però una mica abandonat) és "Linux Malware Detect" o "MalDet" (https://github.com/rfxn/linux-malware-detect)

Un cop instal·lat ClamAV, el primer que hauríem de fer és descarregar del servidor central d'Internet la base de dades de signatures de malware més actualitzada que hi hagi en aquest precís moment (ja que el paquet "clamav-data" instal·lat de sèrie és segur que no estarà al dia). Depenent de la distribució de Linux triada, això s'aconsegueix de formes diferents:

* A Fedora, per realitzar l'actualització manual de la base de dades de signatures hem d'executar cada cop la següent comanda: *sudo freshclam* però en el cas d'haver instal·lat el paquet "clamav-update", aquesta actualització aconseguirem que sigui automàtica (per defecte, cada dues hores) si iniciem i activem el servei homònim, així: *sudo systemctl --now enable clamav-freshclam*

* En Ubuntu, en canvi, en instal·lar-se el paquet "clamav-freshclam" (el qual, de fet, si no s'instal·la explícitament és instal·lat automàticament com a dependència del paquet "clamav"), a diferència del que passava a Fedora, el servei homònim és posat en marxa automàticament (i és activat per propers reinicis del sistema també!). Per tant, com que degut a l'execució immediata del servei "clamav-freshclam" l'actualització de les signatures es fa sempre just en instal·lar ClamAV (i, a partir d'aquí, cada dues hores), en principi no caldrà fer mai l'actualització manualment (executant la comanda *sudo freshclam* explícitament) ni ens haurem de preocupar d'aquest tema.

En tot cas, aquesta base de dades descarregada es guardarà a la carpeta "/var/lib/clamav" i estarà formada per tres fitxers d'un format comprimit i signat digitalment anomenat "CVD" (de "ClamAV Virus Database")...:

"**daily.cvd**" : Conté les signatures MD5/SHA256 dels fitxers maliciosos més recents (s'actualitza diàriament)
"**main.cvd**" : Conté les signatures MD5/SHA256 que prèviament hi eren al fitxer "daily.cvd" i que tenen un risc baix de falsos positius
"**bytecode.cvd**" : Conté el conjunt de patrons binaris maliciosos compilats

...els quals poden ser, si cal, inspeccionats i manipulats internament amb una comanda específica (i avançada) anomenada *sigtool* (per exemple, per saber, entre altres dades estadístiques, la quantitat de signatures incloses en algun dels fitxers anteriors – com ara "main.cvd"-, es pot executar la comanda *sigtool -i /var/lib/clamav/main.cvd* ; per veure'n més possibilitats, consulta *man sigtool* i també la documentació oficial online del programa, aquí: https://docs.clamav.net/manual/Signatures.html)

D'altra banda, la comanda *freshclam* (així com el servei *clamav-freshclam*, el qual en essència executa aquesta comanda en segon pla) té associat com a fitxer de configuració el fitxer "**/etc/freshclam.conf**". D'entre totes les directives que allà hi són presents, en podem destacar les següents (pots trobar més informació a *man freshclam.conf*)

DatabaseMirror : Indica el nom DNS del servidor central d'Internet d'on s'obtindran les actualitzacions de la base de dades de signatures. Per defecte és "database.clamav.net". D'altra banda, es poden afegir URLs "ad-hoc" (de tipus "http://", "https://" o "file://") apuntant a altres fitxers de signatures alternatius amb la directiva *DatabaseCustomURL*
Checks : Indica el nombre de comprovacions que realitzarà diàriament el servei *clamav-freshclam* contra el servidor central d'Internet especificat a la directiva anterior per tal de mantenir permanentment actualitzada la base de dades de signatures local. Per defecte la freqüència de comprovacions és de dues hores.
DatabaseDirectory : Indica la ruta de la carpeta on es guardarà la base de dades de signatures. Per defecte és "/var/lib/clamav"

ClamAV és un software que actua sota demanda. És a dir, no és com altres antivirus que es mantenen en execució contínuament escanejant els eventuals fitxers que hi puguin aparèixer al disc, sinó que només s'executa quan l'usuari li demana, i només per fer un escaneig del contingut d'una carpeta (o d'un fitxer) especificada en concret (una excepció a aquest funcionament és el dimoni *clamonacc*, però aquest és un nivell més avançat d'ús del programa). En aquest sentit, a ClamAV el podem fer servir bàsicament de dues maneres (sempre fent servir com a referència la base de dades de signatures actualitzada en el punt anterior):

Mode "standalone"

Podem realitzar directament un escaneig del fitxer o de la carpeta indicada (opcionalment, de forma recursiva) executant la comanda següent (cal tenir en compte, atenció, que depenent de si es necessiten permisos de lectura en el/s fitxer/s o carpeta/es indicats, aquesta comanda s'haurà d'executar com administrador o no): *clamscan /ruta/fitxerOcarpeta*

A la comanda anterior se li poden afegir diverses opcions (a escriure entre el nom de la comanda i la ruta), d'entre les quals en podem destacar les següents (per conéixer-ne més, consulta *man clamscan*):

-r : Quan s'indica una carpeta a escanejar (o vàries, si s'indiquen comodins), aquest paràmetre fa que l'escaneig es faci de forma recursiva (és a dir, que es faci no només pel contingut inmediat de la carpeta en qüestió sinó també pel contingut de totes les subcarpetes que hi puguin haver a dins d'ella).

-i : Mostra a la pantalla les rutes dels arxius infectats. Per defecte, si no s'indica aquest paràmetre (o el paràmetre *-o* , el qual fa que no es mostrin els missatges de fitxers "OK"), es mostren les rutes de tots els fitxers escanejats, una per una. Al final es mostra (a no ser que s'indiqui *--no-summary*) un resum estadístic final de tot l'escaneig.

--remove : Elimina els fitxers que s'hagin trobat infectats (per defecte no es toquen)

--move /ruta/carpeta : Indica la carpeta on es mouran els fitxers que s'hagin trobat infectats (si no s'indica aquest paràmetre, no es mouran). També existeix el paràmetre *--copy /ruta/carpeta*, que fa una còpia.

--include "expr.reg" : Indica una expressió regular que es confrontarà amb els noms dels fitxers ubicats sota la carpeta indicada per escanejar; només els fitxers el nom del qual hi coincideixi s'escanejaran, la resta no. També existeix el paràmetre contrari, *--exclude "expr.reg"* el qual que fa que els fitxers que tinguin un nom concordant amb l'expressió regular indicada no siguin escanejats. Per exemple, *clamscan --include ".*\.txt$" -r /opt* només escaneja els fitxers sota la carpeta "/opt" (i subcarpetes) que tinguin l'extensió ".txt".

-d /ruta/fitxerOCarpeta : Indica explícitament un fitxer que es farà servir com a base de dades de signatures durant l'escaneig, o bé una carpeta (on podran estar ubicats múltiples fitxers). Per defecte s'utilitza la carpeta "/var/lib/clamav".

-l /ruta/fitxer.log : Guarda en el fitxer indicat el resum estadístic de l'escaneig.

-f /ruta/fitxer : En lloc d'indicar directament el/s fitxer/s o carpeta/es a escanejar com a paràmetres de la comanda, es pot utilitzar aquest paràmetre, el qual serveix per indicar un fitxer el contingut del qual serà la llista de rutes de fitxers o carpetes que seran escanejades (una per línia).

Mode client-servidor

Aquest mode és molt utilitzat per escanejar virus als missatges de correu, on el servidor de correu en aquest cas actuaria com a "client ClamAV" en sol·licitar al "servidor ClamAV" l'escaneig dels missatges que rep. En concret, per posar en marxa el "servidor ClamAV", només cal posar en marxa el dimoni pertinent, així:

sudo systemctl --now enable clamav-daemon (A Ubuntu)
sudo systemctl --now enable clamd@scan (A Fedora)

Aquest dimoni no escaneja res en temps real (ni de forma programada) sinó que només ho fa quan és disparat mitjançant la comanda client següent (i només per realitzar la inspecció del/s fitxer/s o carpeta/es indicat/s en dita comanda client): *clamdscan /ruta/fitxerOCarpeta*

Fent-ho així, la comanda client només aporta el fitxer/carpeta al servidor, que és qui realitza l'escaneig pròpiament dit; en aquest sentit, el servidor podria estar funcionant a la mateixa màquina local que el client o bé remotament; si fos aquest el cas, el client faria una "pujada" del/s fitxer/s en qüestió al servidor, el servidor faria tot seguit l'escaneig pròpiament dit i finalment retornaria el resultat d'aquest al client.

A la comanda client anterior se li poden afegir diverses opcions (a escriure entre el nom de la comanda i la ruta), d'entre les quals en podem destacar les següents (per conéixer-ne més, consulta *man clamdscan*, però cal tenir en compte que la majoria dels paràmetres de *clamdscan* no tenen efecte pràctic sobre la manera de com fer l'escaneig perquè aquest és realitzat realment pel dimoni *clamd@scan/clamav-daemon*, que té la seva pròpia configuració):

--fdpass : S'utilitza si el servidor escolta localment. Serveix per passar al servidor l'usuari (i els permissos associats) amb què s'ha executat la comanda *clamdscan* per tal de què el servidor utilitzi per fer l'escaneig aquest usuari i no pas el seu usuari per defecte (indicat a la directiva *User=* del seu arxiu de configuració, com veurem de seguida). Això sol ser necessari per a què el servidor pugui accedir a les mateixes carpetes/fitxers a les què pot accedir l'usuari que executa la comanda client.

-l /ruta/fitxer.log : Guarda en el fitxer indicat el resum estadístic de l'escaneig.

-f /ruta/fitxer : En lloc d'indicar directament el/s fitxer/s o carpeta/es a escanejar com a paràmetres de la comanda, es pot utilitzar aquest paràmetre, el qual serveix per indicar un fitxer el contingut del qual serà la llista de rutes de fitxers o carpetes que seran escanejades (una per línia).

ClamAV, funcionant en el mode client-servidor, utilitza diversos fitxers de configuració que podrem editar segons les nostres necessitats (la comanda *clamscan*, del mode "standalone", en canvi, no té fitxer de configuració perquè funciona només a través dels paràmetres que se l'indiquin des de la línia de comandes).

Concretament, el fitxer "**/etc/clamav/clamd.conf**" (a Ubuntu) o "**/etc/clamd.d/scan.conf**" (a Fedora) inclou directives relacionades amb el funcionament del dimoni resident *clamd@scan/clamav-daemon*. Alguna de les més destacables són les següents (pots trobar més informació a *man clamd.conf*):

NOTA: El nom (sense extensió) del fitxer de configuració del dimoni resident a Fedora ("scan.conf") ha de ser el mateix que el valor indicat després de l'arroba quan s'invoca dit dimoni (així, "clamd@scan"). Això és perquè el valor escrit després de l'arroba indica el nom del fitxer de configuració que farà servir el dimoni, de forma que es podria tenir diversos dimonis ClamAV funcionant a la vegada, cadascun amb una configuració diferent.

DatabaseDirectory : Indica la ruta de la carpeta on s'anirà a buscar la base de dades de signatures (ha de coincidir amb la ruta indicada a l'opció homònima a l'arxiu "freshclam.conf")
ExcludePath : Indica una expressió regular que concordarà amb els noms de fitxers o carpetes que no seran escanejades
MaxDirectoryRecursion : Indica el nº màxim de subcarpetes que s'escanejaran recursivament
CrossFilesystems : Indica si s'escanejaran (o no) carpetes o fitxers ubicats en altres sistemes de fitxers diferents de la ruta inicial indicada
VirusEvent: Indica la comanda que s'executarà automàticament quan es trobi un virus. Aquí podem indicar, per exemple, l'execució d'alguna comanda per enviar emails o algun altre tipus d'alerta
LocalSocket : Indica el nom del "socket" (pot ser una cadena qualsevol) que s'utilitzarà per rebre peticions locals (si no s'indica aquesta directiva, el servei no escoltarà peticions locals)
TCPSocket : Indica el número de port que s'utilitzarà per rebre peticions remotes (si no s'indica aquesta directiva, el servei no escoltarà peticions provinents de la xarxa). Per defecte el servei es posarà a escoltar en totes les adreces IP disponibles del sistema; si es vol restringir a una IP en concret, caldrà afegir la directiva *TCPAddr* (on s'hi haurà d'indicar l'adreça IP triada en qüestió)
User : Indica l'usuari amb els privilegis del qual el dimoni s'executarà. Per defecte és "clamscan"
Example : La presència d'aquesta línia fa que el dimoni *clamd/clamav-daemon* no s'inicïi

D'altra banda, el fitxer que s'especifiqui amb el paràmetre ***-c*** de la comanda *clamdscan* (i també de la comanda *clamdtop*) bàsicament serveix per indicar, en el cas que el servidor *clamd@scan/clamav-daemon* funcioni de forma remota, l'adreça IP i port d'escolta d'aquest, per tal que el client *clamdscan* s'hi pugui connectar (si el servidor funciona de forma local aquest fitxer no és necessari). Concretament, les directives importants són:

TCPAddr : Indica l'adreça IP del servidor Clamd on s'hi connectarà
TCPSocket : Indica el nombre de port del servidor Clamd on s'hi connectarà

EXERCICIS:

0.-a) Descarrega't, en una màquina virtual qualsevol, l'antivirus ClamAV dels repositoris oficials (d'Ubuntu o Fedora, segons el sistema on hi estiguis treballant). Seguidament, executa des d'un terminal la comanda *sudo freshclam* per actualitzar les signatures dels virus.

aII) Inicia el servei "clamav-freshclam". D'aquesta manera el sistema automàticament anirà comprovant de forma periòdica si hi ha disponibles actualitzacions de la base de dades de signatures llestes per descarregar

El fitxer EICAR és un fitxer (definit en un estàndard a nivell europeu) dissenyat per provar el comportament i resposta dels programes antivirus. La idea és permetre provar el funcionament del software antivirus sense haver d'utilitzar un vertader virus informàtic (el qual podria causar danys al sistema en no respondre l'antivirus correctament). En realitat el fitxer EICAR només és un arxiu de text amb extensió "com" i el següent contingut (tot i que es pot descarregar directament d'aquí: https://secure.eicar.org/eicar.com.txt):

*X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H**

Un antivirus amb protecció en temps real hauria de detectar-lo immediatament i un escaneig en busca de virus també hauria de detectar-lo, fins i tot si està dins d'un arxiu comprimit (sense contrasenya). Lògicament, per a què la prova funcioni, els programadors d'antivirus han d'establir la cadena d'EICAR com un virus verificat com qualsevol altra signatura. La prova EICAR, però, no serveix per saber quins virus n'es capaç de detectar o el nivell d'eficàcia de l'antivirus.

1.-a) Descarrega't el fitxer EICAR (o crea'l directament) i guarda'l dins de la carpeta "/opt" del sistema on has instal·lat ClamAV. Fes servir la comanda *clamscan* adient per escanejar el contingut d'aquesta carpeta de forma recursiva. ¿L'escaneig detecta el fitxer EICAR?

b) Torna a anar a la base de dades https://bazaar.abuse.ch i descarrega't el malware que ara mateix sigui el més nou de la llista. Descomprimeix-lo (fent servir la contrasenya "infected") i tot seguit utilitza la comanda *clamscan* per comprovar si el fitxer descomprimit és detectat, efectivament, com a malware (o no). ¿Passa el mateix si l'escaneig es fa directament sobre el fitxer comprimit?

c) Recupera l'arxiu "diamorphine.ko" (generat a l'exercici 2 de la primera part d'aquest document) i escaneja'l amb *clamscan* ¿És reconegut com una amença?

> NOTA: ClamAV no és un detector de "rootkits", ja que no inspecciona la memòria dels altres processos que puguin estar executant-se a la màquina; ClamAV només pot alertar de rootkits si troba al disc signatures de fitxers associats a ells

2.-a) Descomenta la línia *LocalSocket* (i també les línies *LocalSocketGroup=* i *LocalSocketMode=* que l'acompanyen) del fitxer de configuració del servidor "clamd" per tal de què el servei atengui peticions locals d'escaneig i tot seguit posa en marxa el servei així: *sudo systemctl start clamd@scan* (o *sudo systemctl start clamav-daemon*, segons sigui la teva distribució Linux de treball)

b) Per a què no tinguis problemes de permisos a l'hora de connectar al socket local en executar *clamdscan*, el teu usuari (suposem que és "usuari") ha de pertànyer al grup "virusgroup". Recorda que això ho pots aconseguir executant s*udo usermod -a -G virusgroup usuari* i reiniciant sessió. Un cop fet tot això, dispara un escaneig a tot el contingut que hi hagi sota la carpeta "/opt" (per defecte els escanejos són recursius fins el nivell marcat per la directiva de configuració *MaxDirectoryRecursion=* de *clamd*) simplement executant la comanda *clamdscan --fdpass /opt* ¿Què passa?

> NOTA: Si tens problemes amb SELinux en Fedora, executa la comanda *sudo setsebool -P antivirus_can_scan_system 1*
>
> NOTA: La comanda *clamdtop* permet monitoritzar el funcionament en temps real del servidor *clamd* (indicat al fitxer de configuració especificat al seu paràmetre **-c** o bé indicat directament com a valor en la línia de comandes, bé en forma ruta del socket local o bé en forma d'adreça IP i port) ; és molt interessant aquesta eina, per tant, per veure en temps real els recursos emprats durant un escaneig. Una altra eina, en aquest cas per repassar la configuració de clamd, és *clamconf*

Creació de signatures pròpies

La manera més fàcil de crear una signatura per ClamAV és utilitzant hashes de fitxers per fer comparacions estàtiques (no obstant, cal tenir en compte que aquest mètode no funcionarà tan bon punt un simple bit canvïi en el fitxer inspeccionat). La manera concreta seria tal com es proposa al següent exercici:

3.-a) Calcula el hash MD5 del fitxer */bin/ls* i guarda'l en un arxiu de hashes anomenat "prova.hdb" (en aquest arxiu hi podrien haver molts més hashes, cadascun a una línia diferent). Això ho pots fer amb la comanda *sigtool --md5 /bin/ls > prova.hdb*

> NOTA: Si comproves el contingut del fitxer ".hdb" creat (amb *cat*, per exemple) veuràs que el format de la línia és *cadenahash:nommalware* Respecte el nom que ha de tenir el malware, no hi ha una regla estàndard oficial però la tradició històrica (explicada aquí https://docs.clamav.net/manual/Signatures/SignatureNames.html), anima a que sigui un nom estructurat de la següent manera: *plataforma.categoria.nom-id* (on "plataforma" podria ser "Windows", "Linux", "Java", etc; "categoria" podria ser "Virus", "Trojan", "Ransomware", etc; "nom" seria en aquest cas el nom propi del malware i "id" un identificador, normalment numèric, de la signatura en sí)

aII) Comprova ara si ClamAV "detecta" la presència d'aquest hash dins de la carpeta */bin* del teu sistema, així: *clamscan -d prova.hdb -i -r /bin* ¿Què passa?

b) Calcula el hash SHA1 (o SHA256, com tu vulguis) del fitxer */bin/ls* i guarda'l en un arxiu de hashes anomenat "prova.hsb" Això ho pots fer amb la comanda *sigtool --sha1 /bin/ls > prova.hsb (o sigtool --sha256 /bin/ls > prova.hsb*, respectivament)

NOTA: Si comproves el contingut del fitxer ".hsb" creat (amb *cat*, per exemple) veuràs que el format de la línia és *cadenahash:midafitxerenbytes:nommalware* Com a valor de la mida del fitxer es podria indicar "*" si aquesta dada no fos coneguda

bII) Comprova ara si ClamAV "detecta" la presència d'aquest hash dins de la carpeta */bin* del teu sistema, així: *clamscan -d prova.hsb -i -r /bin* ¿Què passa?

NOTA: També és possible crear hashes no de fitxers sencers sinó de seccions específiques dins d'executables PE (és a dir, els ".exe" i ".dll" de Windows). Això es pot fer mitjançant el paràmetre *--mdb* de *sigtool*, havent-se de guardar llavors aquests hashes en arxius amb extensió ".mdb" (si són MD5) o ".msb" (si són SHAx). Però aquest ja és un tema avançat que està millor explicat a https://docs.clamav.net/manual/Signatures/HashSignatures.html

c) Si no es vol haver d'especificar manualment l'arxiu ".hdb/.hsb" a utilitzar en l'escaneig, aquest arxiu es pot moure dins de la carpeta on està la base de dades oficial de signatures (és a dir, per defecte, "/var/lib/clamav") i serà utilitzat com un fitxer més d'aquesta base de dades per defecte. Comprova-ho movent l'arxiu "prova.hsb" dins de "/var/lib/clamav" i tot seguit executant *clamscan -i -r /bin*. ¿Què passa?

d) Els arxius ".cvd" no són més que contenidors comprimits (i signats) de múltiples fitxers ".hdb/.hsb/.mdb/.msb" (i de molts altres tipus que coneixerem als propers exercicis). Aquests fitxers (per exemple, els inclosos dins de "main.cvd") es poden extreure a la carpeta on estiguem situats en aquell moment (i així poder-los consultar/manipular individualment, per exemple) si executem la comanda *sigtool -u /var/lib/clamav/main.cvd* Fes-ho, comprova quin és el resultat obtingut i explora per curiositat (amb *cat*) el contingut d'alguns d'aquests fitxers extrets. Finalment, esborra'ls tots (no els farem servir més).

NOTA: Per fer el procés contrari (és a dir, per crear un arxiu ".cvd" a partir d'un conjunt de fitxers de signatures individuals, cal executar una comanda similar a la següent dins de la carpeta mateixa on estan ubicats aquests fitxers de signatures: **sigtool --unsigned -b nomarxiu.cud** , on l'arxiu generat té l'extensió ".cud" en lloc de ".cvd" perquè no estarà signat per ClamAV (és per això que cal afegir també el paràmetre *--unsigned*)

e) Executa la comanda *sigtool -f "Linux"*. ¿Què veus? (pots consultar la pàgina manual de la comanda per esbrinar què fa aquest paràmetre)

El format més habitual de signatures, però, és el basat en seqüències de bytes (en hexadecimal) corresponents a part del contingut de fitxers. Aquestes seqüències admeten, a més, determinats símbols que tenen un significat especial com són:

??	Equival a un byte qualsevol	*	Equival a qualsevol nombre de bytes
?a	Equival a un byte on els seus quatre bits més significatius poden ser qualssevol	a?	Equival a un byte on els seus quatre bits menys significatius poden ser qualssevol
{n}	Equival a *n* bytes qualssevol	{-n}	Equival a *n* o menys bytes qualssevol
{n-}	Equival a *n* o més bytes qualssevol	{n-m}	Equival a entre *n* i *m* bytes qualssevol
(aa|bb|cc|..)	Equival al byte "aa" o "bb" o "cc", etc. Els diferents valors-alternativa poden ser tenir una longitud de més d'un byte (i no cal que tots ells en tinguin la mateixa). També es poden indicar altres símbols (??, *, etc)	!(aa|bb|cc|..)	Equival a qualsevol byte excepte "aa" o "bb" o "cc", etc. Els diferents valors-alternativa poden ser tenir una longitud de més d'un byte, però tots ells hauran de tenir-ne en tot cas la mateixa. Tampoc es pot indicar cap altre símbol (??, *, etc)
(B)	Equival a un límit de paraula (inclou final de fitxer)	(L)	Equival a un salt de línia (CR, CRLF) o final de fitxer
(W)	Equival a un caracter no alfanumèric		

NOTA: Els símbols * i {} virtualment separen les signatures en dues parts; per exemple, la tira "aabbcc*bbaacc" és tractada com dues sub-tires, "aabbcc" i "bbaacc" (amb un nombre de bytes qualsevol entre elles)

Tota signatura basada en seqüència de bytes s'ha d'escriure dins d'un arxiu ".ndb" en un format tal com el següent: *nommalware:tipustarget:offset:sequenciabytes* , on:

* *tipustarget* ha de ser algun dels següents valors, que serveixen per indicar el tipus de fitxer on s'espera trobar la signatura (veure https://docs.clamav.net/appendix/FileTypes.html#Target-Types per veure la llista completa). Aquest valor serveix per evitar haver de fer la comprovació de signatures en cas que el fitxer a escanejar no sigui del tipus indicat en ella

0	Qualsevol fitxer (la signatura es comprova sempre, sigui quin sigui el tipus de fitxer a escanejar)	1	PE (Portable Executable):el format dels executables (*.exe, *.dll) de Windows
3	HTML normalitzat	5	Arxius gràfics
6	ELF (Executable Linux File): el format dels executables (*.so,...) de Linux	7	ASCII normalitzat
9	Mach-O: el format del executables de Mac	10	PDF

* *offset* ha de ser el nombre de bytes a comptar des del començament del fitxer on s'espera trobar el principi de la la seqüència indicada. També pot valer "*" per indicar "qualsevol" o el valor especial EOF-*nº* per especificar l'offset comptant des del final del fitxer en lloc de des del seu començament, entre altres valors més avançats (per indicar inici de seccions o de punts d'entrada en executables, etc). D'altra banda, també es pot especificar el valor *offset,maxshift* , on *maxshift* ha de ser un número sencer que serveix per indicar que la seqüència a buscar pot començar al byte ubicat a la posició *offset* , o bé a la posició *offset + 1 byte*, o la posició *offset + 2bytes*, etc, fins arribar com a màxim a la posició *offset + maxshift* bytes.

4.-a) Crea un fitxer de text a la teva carpeta personal anomenat "bitxo.txt" que contingui tan sols la següent frase (sense cometes): *"Hola, em dic Pepe"*. L'objectiu de l'exercici és crear una signatura basada en el contingut d'aquest fitxer de forma que si aquest es trobés en qualsevol altre fitxer, es consideraria malware.

aII) Abans de generar la signatura del fitxer anterior, l'hem de "normalitzar". Aquest és un procediment que cal fer amb els fitxers que són de text pla (i també els que són HTML) i que consisteix en vàries transformacions (com ara convertir tots els caràcters a minúscules, treure els caràcters no imprimibles i col·lapsar els espais en blanc i salts de línia, entre altres accions); ClamAV només trobarà signatures si aquestes han sigut normalitzades. Per normalitzar el fitxer "bitxo.txt" cal que executis la següent comanda: *sigtool --ascii-normalise bitxo.txt* ; el resultat serà un nou fitxer (ubicat a la carpeta de treball) anomenat "normalised_text"

NOTA: En el cas de fitxers HTML, també és necessari normalitzar-los abans de generar la seva signatura, però en aquest cas el paràmetre de *sigtool* a utilitzar és *--html-normalise* i el resultat obtingut no és un sol fitxer sinó tres: "nocomment.html" (que seria el fitxer a normalitzar, en principi, ja que conté el mateix codi HTML però en minúscules, sense comentaris ni espais en blanc ni salts de línia superflus), "notags.html" (el mateix però sense cap etiqueta HTML) i "javascript" (contenint, també normalitzats, tots els scripts Javascripts trobats). Per més informació podeu consultar https://docs.clamav.net/manual/Signatures.html#writing-signatures-for-special-files

aIII) Obté la seqüència hexadecimal corresponent al contingut del fitxer "normalised_text" i guarda'l en un arxiu anomenat "prova.ndb" (en aquest arxiu hi podrien haver moltes més seqüències, cadascuna a una línia diferent). Això ho pots fer amb la comanda *sigtool --hex-dump < normalised_text > prova.ndb* Tot seguit pots esborrar l'arxiu "normalised_text", no el necessitarem més.

NOTA: Si s'usa el paràmetre *--hex-dump*, la comanda *sigtool* accepta les dades a convertir en tira hexadecimal només des de l'entrada estàndar. És per això que cal indicar l'operador "<" a l'exemple de l'apartat anterior i és per això mateix que, si no s'indica cap fitxer, pot funcionar interactivament des del teclat (d'aquesta manera, es pot obtenir en temps real la conversió inmediata de cadenes curtes).
NOTA: Depenent del fitxer a convertir, el resultat pot ser una tira hexadecimal mooooolt llarga (i per tant, molt lenta d'escanejar). Sovint no caldrà fer la conversió de tot el fitxer sinó d'una part concreta. Per exemple, la següent comanda només genera la signatura dels primers 2KB del fitxer passat per l'entrada estàndard: *cat testfile | sigtool --hex-dump | head -c 2048 > customsig.ndb* En teoria, només es necessita generar una signatura d'una part del fitxer que sigui única (i no necessàriament des del seu inici)

aIV) Edita el contingut de l'arxiu "prova.ndb" per afegir-ne el prefix "*Linux.Trojan.Pepe:7:0:"* (sense cometes) a l'inici de la tira hexadecimal allà present.

NOTA: En aquest cas era molt evident, però si en alguna altra ocasió tinguéssim un fitxer per escanejar del qual no sapiguéssim el seu tipus (o millor dit, el tipus amb què ClamAV el reconeix) per tal de saber indicar el valor numèric del segon camp de la signatura, el que podem fer és escanejar-lo amb el mode "debug" activat per observar una línia que

comenci per "Recongnized", la qual indicarà el tipus de dada reconegut. És a dir, exeucta una comanda similar a *clamscan **--debug** arxiudesconegut 2>&1 | grep -i "Recognized"*

aV) Comprova ara si ClamAV "detecta" la presència d'aquesta seqüència de bytes dins d'algun fitxer ubicat dins de la teva carpeta personal, així: *clamscan -d prova.ndb -i $HOME* ¿Què passa?

NOTA: Com passava amb els arxius ".hdb/.hsb", si no es vol haver d'especificar explícitament l'arxiu ".ndb" a utilitzar en un escaneig, aquest arxiu (o arxius) es poden moure a la carpeta "/var/lib/clamav" per tal de tractar-los de forma equivalent a la base de dades de signatures oficial (és a dir, per fer-los servir per defecte en qualsevol escaneig sense haver d'especificar-los manualment)

aVI) Canvia el zero que apareix quasi al final del prefix de la signatura present en "prova.ndb" per un asterisc i tornar a executar la mateixa comanda *clamscan* de l'apartat anterior. ¿Què passa i per què? Tot seguit, canvia el set que apareix també en el prefix de la mateixa signatura per un altre número qualsevol i torna a executar de nou la mateixa comanda *clamscan*. ¿Què passa ara i per què?

b) Ara afegirem a "prova.ndb" una segona signatura, en aquest cas la corresponent als primers 123 bytes (per exemple) d'un binari (concretament, */bin/ls*), així: *sigtool --hex-dump < /bin/ls | sed "s/^/Linux.Trojan.Pepa:6:0:/" | head -c 123 >> prova.ndb*

NOTA: La comanda *sed* realitza una substitució (gràcies a la seva ordre interna "s"); concretament, substitueix l'inici de la línia ("^") per la cadena indicada ("Linux.Trojan.Pepa:6:0:"), de forma que la línia final serà la mateixa però amb el prefix afegit al seu inici
NOTA: La comanda *head* fa que només els primers 123 caràcters (bytes) rebuts de *sed* arribin a escriure's al fitxer final

bII) Comprova ara si ClamAV "detecta" la presència d'aquesta seqüència de bytes dins d'algun fitxer ubicat dins de */bin*, així: *clamscan -d prova.ndb -i -r /bin* ¿Què passa?

bIII) Modifica la signatura de */bin/ls* present en "prova.ndb" per a què alguna tira "*00000000*" (sense cometes) que hi aparegui sigui substituïda per la cadena "*(00|01|02){3-}*" (sense cometes). Torna a executar la mateixa comanda *clamscan* de l'apartat anterior. ¿Què passa i per què?

Més enllà de les signatures basades en seqüències de bytes (que anomenarem "body-based"), tenim la possibilitat d'utilitzar les anomenades signatures "lògiques", les quals permeten combinar múltiples signatures "body-based" mitjançant operadors lògics (bàsicament, AND i OR). Aquestes signatures lògiques s'han d'escriure dins d'un arxiu ".ldb" en un format tal com el següent: *nomsignatura;parellesdades;expressiologica;seqbytes0::mods;seqbytes1::mods;seqbytes2::mods;...* , on:

* *parellesdades* són parelles de tipus *nom:valor* (separades per comes) que aporten informació diversa sobre la signatura lògica en qüestió. D'entre tots els noms possibles, el que més sovint farem servir és *Target*, el qual pot tenir com a valor associat qualsevol dels números ja coneguts pels "targets" definits a ClamAV (0=qualsevol tipus de fitxer, 1=fitxers PE, 3=fitxers HTML, 5=fitxers gràfics, 6=fitxers ELF, etc), essent el significat el qual exactament el mateix que ja coneixíem: funcionar com a filtre per descartar la signatura lògica en el cas que el fitxers inspeccionat no concordi amb el valor "target" especificat.

* *expressiologica* descriu la relació lògica entre les seqüències de bytes indicades a continuació (és a dir, entre *seqbytes0*, *seqbytes1*, etc. Aquí cada seqüència s'identifica amb el número de posició en que apareixen dins de la signatura lògica en qüestió ("0", "1",...sense cometes) i les relacions lògiques poden ser les següents:

& : És un "I" lògic. Per exemple, 0&1 indica que el fitxer inspeccionat en aquell moment, per a que sigui considerat malware, ha de contenir la seqüència de bytes nº0 i la seqüència nº1, no importa on.
| : És un "O" lògic. Per exemple, 0|1 indica que el fitxer inspeccionat en aquell moment, per a que sigui considerat malware, ha de contenir la seqüència de bytes nº0 o la seqüència nº1, no importa on.

A=X : Indica que la seqüència especificada (aquí escrita com a "A" però que en realitat hauria de ser 0, 1, 2...) ha d'aparèixer dins del fitxer inspeccionat exactament el nombre de vegades *X* per a què sigui considerat malware, no importa on. En el cas concret que X valgués 0, indicaria que la signatura indicada no ha d'aparèixer per ser considerat el fitxer inspeccionat malware
A>X : Indica que la seqüència especificada (aquí escrita com a "A" però que en realitat hauria de ser 0, 1, 2...) ha d'aparèixer dins del fitxer inspeccionat més vegades que el nombre *X* per a què sigui considerat malware, no importa on.
A<X : Indica que la seqüència especificada (aquí escrita com a "A" però que en realitat hauria de ser 0, 1, 2...) ha d'aparèixer dins del fitxer inspeccionat menys vegades que el nombre *X* per a què sigui considerat malware, no importa on.

* *seqbytesX* és la seqüència de bytes nºX tal qual, on es poden indicar igualment els mateixos símbols especials ja coneguts de les signatures "body-based" (*??, *, {n}, (aa|bb|..)*, etc). També es pot indicar un "offset" específic per una determinada sequência indicant el seu valor abans de la pròpia seqüència (separant-lo d'aquesta per ":"), així: *offset:seqbytesX*

* *mods* representa un conjunt (opcional!) de modificadors associats a la seqüència de bytes precedent (i de la qual estan separats per "::"). Cadascun d'aquests modificadors té forma de lletra, i en podem destacar els següents:

i : Indica que la confrontació entre el fitxer inspeccionat i la seqüència de bytes en qüestió, si aquesta es correspon amb símbols alfabètics, es faci de forma "case-insensitive" (és a dir, indistintament utilitzant majúscules i minúscules)
f : Indica que la signatura en qüestió només serà considerada trobada si en el fitxer inspeccionat està delimitada entre dos caracters no alfanumèrics (com, per exemple, espais, tabuladors, salts de línia o altres símbols. La idea és que la signatura representi una paraula.
a : Indica que la signatura en qüestió representa caràcters ASCII
w : Indica que la signatura en qüestió representa caràcters codificats cadascun amb dos bytes

Per exemple, si es busquen fitxers de qualsevol tipus que continguin tant "AAAA" com "BBB" (sense importar si són majúscules o minúscules, ni tampoc l'ordre en què apareguin al fitxer inspeccionat!), la signatura lògica a indicar podria ser: *Pepi1;Target:0;**0&1**;41414141::i;424242::i* (noteu com les signatures 0 i 1 indicades estan formades pels valors hexadecimals corresponents als caràcters buscats -"A" i "B", dins de la taula ASCII).

Altres signatures lògiques més complexes són, per exemple, les següents:

*Pepi2;Target:0;**(0|1)&2&3**;4141::i;4242::i;4343::i;4444::i*
*Pepi3;Target:0;**0&(1|2)&(3|4)**;4141::i;4242::i;4343::i;4444::i;4545::i*
*Pepi4;Target:0;**(0&1)|(1&2)**;4141::i;4242::i;4343::i*
*Pepi5;Target:0;**(0|1|2)>1**;4141::i;4242::i;4343::i*
*Pepi6;Target:0;((**0&1&3)>2)&(2<3|2>5)**;4141::i;4242::i;4343::i;4444::i*

NOTA: A https://docs.clamav.net/manual/Signatures/LogicalSignatures.html hi ha més informació sobre com construir signatures lògiques
NOTA: Una altra sintaxis (potser més accessible) per escriure tant signatures "body-based" com, sobre tot, lògiques (que no deixen de ser una combinació d'aquelles) és l'anomenada sintaxis **Yara** (https://virustotal.github.io/yara). No obstant, ClamAV només n'és compatible fins a cert punt.

5.-a) Executa, a la teva màquina virtual de treball, la següent comanda: *sudo dd if=/dev/urandom of=file.bin bs=1 count=100* ¿Què fa aquesta comanda?

b) Instal·la, a la teva màquina virtual de treball, el paquet "hexedit" i tot seguit executa *hexedit file.bin* Consulta la "nota" següent per saber com editar el contingut de quatre bytes qualssevol d'aquest fitxer per a què tinguin en conjunt el valor hexadecimal 41414141 i el d'altres tres bytes qualssevol per a què tinguin el valor hexadecimal **62626**2. Fes-ho

NOTA: La comanda *hexedit* (https://github.com/pixel/hexedit) és un editor hexadecimal interactiu en mode text. Amb F1 es pot accedir a la seva pàgina del manual (*man hexedit*), on es mostra la referència de combinacions de tecles que es poden fer servir per treballar-hi, d'entre les quals en podem destacar les següents:

Tab: Passar de la columna hexadecimal a la ASCII i viceversa	F2 : Guardar
CTRL+X : Guardar i sortir	CTRL+C : Sortir sense guardar
/ : Buscar endavant	CTRL+R : Buscar endarrera

bII) Crea un arxiu anomenat "prova.ldb" que contingui la signatura "Pepi1" indicada als paràgrafs de teoria anteriors i tot seguit executa *clamscan -d prova.ldb -i file.bin* ¿Què passa i per què?

bIII) Executa la següent comanda i digues quina informació et mostra: *cat prova.ldb | sigtool --decode-sigs*

c) Modifica la signatura "Pepi1" dins del fitxer "prova.ldb" per a què ara sigui així: *Pepi1;Target:0;0&1;**25:**41414141::i;**5:**626262::i* i torna a executar la comanda *clamscan* anterior. ¿Què passa ara i per què? Modifica adientment el fitxer "file.bin" per a què en tornar a executar-hi la mateixa comanda *clamscan* ara obtinguis un resultat diferent (tingues en compte que el 1r byte és el 0, no pas el 1!!).

cII) Modifica la signatura "Pepi1" dins del fitxer "prova.ldb" per a què ara sigui així: *Pepi1;Target:0;0&1;**25,3:**41414141::i;**5,2:**626262::i* i torna a executar la comanda *clamscan* anterior. ¿Què passa, i per què?

d) Sustitueix dins del fitxer "prova.ldb" la signatura "Pepi1" per "Pepi2" (també indicada a la teoria) i edita adientment el fitxer "file.bin" per a què en executar-hi la mateixa comanda *clamscan* es generi un positiu.

e) Sustitueix dins del fitxer "prova.ldb" la signatura "Pepi2" per "Pepi3" (també indicada a la teoria) i edita adientment el fitxer "file.bin" per a què en executar-hi la mateixa comanda *clamscan* es generi un positiu.

f) Sustitueix dins del fitxer "prova.ldb" la signatura "Pepi3" per "Pepi4" (també indicada a la teoria) i edita adientment el fitxer "file.bin" per a què en executar-hi la mateixa comanda *clamscan* es generi un positiu.

g) Sustitueix dins del fitxer "prova.ldb" la signatura "Pepi4" per "Pepi5" (també indicada a la teoria) i edita adientment el fitxer "file.bin" per a què en executar-hi la mateixa comanda *clamscan* es generi un positiu.

h) Sustitueix dins del fitxer "prova.ldb" la signatura "Pepi5" per "Pepi6" (també indicada a la teoria) i edita adientment el fitxer "file.bin" per a què en executar-hi la mateixa comanda *clamscan* es generi un positiu.

6.-a) Afegeix una signatura nova al fitxer "prova.ldb" que comprovi si al fitxer inspeccionat apareix exactament 2 vegades la cadena "hola" a qualsevol ubicació (recorda que pots trobar la seva conversió a hexadecimal molt fàcilment executant *echo -n "hola" | sigtool --hex-dump*) i més de 3 vegades però només en els darrers 30 bytes del fitxer (recorda la notació "EOF-nº,nº" per indicar l'offset -variable- a comptar des del final del fitxer) la cadena "adeu". Edita adientment el fitxer "file.bin" per a què en executar-hi la mateixa comanda *clamscan* es generi un positiu.

aII) Afegeix una nova cadena "hola" on vulguis en l'interior del fitxer "file.bin" i torna a passar la comanda *clamscan*. ¿Què passa i per què?

aIII) Desfés el que has fet a l'apartat anterior. Tot seguit, elimina una de les quatre cadenes "adeu" que hi havia al fitxer "file.bin" però escriu-la de nou al principi d'aquest fitxer. Torna a passar la comanda *clamscan*. ¿Què passa i per què?

b) Afegeix una signatura nova al fitxer "prova.ldb" que comprovi si el fitxer "file.bin" conté les cadenes "aa" i "bb" amb un mínim de tres bytes entre una i l'altra (recorda els diferents símbols especials que es poden indicar a les signatures).

c) Afegeix una signatura nova al fitxer "prova.ldb" que comprovi si un fitxer PDF qualsevol (recordem que aquests fitxers tenen un "TargetType" igual a 10) conté dins el seu interior algun enllaç HTTP (és a dir, les cadenes "http://" o "https://")

d) Escriu el següent codi C dins un fitxer anomenat "malware.c" i tot seguit compila'l amb la comanda *gcc -o malware.exe malware.c* per tal d'obtenir l'executable ELF "malware.exe".

```
#include <stdio.h>
int main () {
	char *user = "adm.user";
	printf("%s\n",user);
	return 0;
}
```

Si escriguissis a l'arxiu "prova.ldb" una regla tal com la següent: *Manoli1;Target:6;(0|1);61646d;636f7270* , ¿creus que aquest executable seria detectat com malware (o no) i per què? (pots observar el contingut hexadecimal de "malware.exe" amb *hexedit* si t'és d'ajuda)

7.-a) ¿Què explica aquest article: https://docs.clamav.net/manual/Signatures/FileTypeMagic.html ?

b) ¿Què expliquen l'article https://docs.clamav.net/manual/Signatures/DatabaseInfo.html i els apartats "Settings databases" i "Body-based/Hash-based/ signatures", "Alternative signature support" i "Other database files" de l'article https://docs.clamav.net/manual/Signatures.html ?

c) ¿Què explica aquest article: https://docs.clamav.net/appendix/CvdPrivateMirror.html?