Administració de seguretat

Josep Jorba Esteve

PID_00212467

UOC
Universitat Oberta de Catalunya

Índex

Introducció 5

Objectius 7

1. Tipus i mètodes dels atacs 9
1.1. Tècniques utilitzades en els atacs 12
1.2. Contramesures 20

2. Seguretat del sistema 25

3. Seguretat local 26
3.1. *Bootloaders* 26
3.2. Contrasenyes i *shadows* 28
3.3. Bits *sticky* i *setuid* 30
3.4. Habilitació de *hosts* 30
3.5. Mòduls PAM 31
3.6. Alteracions del sistema 33
3.7. Recursos limitats, *cgroups* i *chroot* 33
3.8. Protecció d'executables mitjançant Hardening-Wrapper 38

4. SELinux 41
4.1. Arquitectura 44
4.2. Crítica 47
4.3. Algunes alternatives 48

5. Seguretat en xarxa 50
5.1. Client de serveis 50
5.2. Servidor: inetd i xinetd 50

6. Eines de seguretat: detecció de vulnerabilitats i intrusions 53
6.1. OpenVAS 57
6.2. Denyhosts 60
6.3. Fail2ban 62

7. Protecció mitjançant filtratge (*TCP wrappers* i tallafocs) .. 65
7.1. Tallafocs 66
7.2. Netfilter: iptables 67
7.3. Paquets per a la gestió de tallafocs en les distribucions 72
7.4. Netfilter: nftables 74

7.5. Consideracions 76

8. Anàlisi de registres 78

9. Taller: anàlisi de la seguretat mitjançant eines 80

Resum 88

Activitats 89

Bibliografia 90

Introducció

El salt tecnològic que s'ha produït des dels sistemes d'escriptori aïllats fins als sistemes actuals integrats en xarxes locals i Internet, ha dut una nova dificultat a les tasques habituals de l'administrador: el control de la seguretat dels sistemes.

La seguretat és un camp complex, en el qual es barregen tècniques d'anàlisi amb altres de detecció o de prevenció dels possibles atacs. Les tècniques que cal utilitzar són tant computacionals com relacionades amb altres camps, com ara l'anàlisi de factors psicològics, pel que fa al comportament dels usuaris del sistema o a les possibles intencions dels atacants.

Els atacs poden provenir de moltes fonts i afectar des d'una aplicació o servei fins a algun usuari, a tots o el sistema informàtic sencer.

Els possibles atacs poden canviar el comportament dels sistemes, fins i tot "ferlos caure" (és a dir, inutilitzar-los), o donar una falsa impressió de seguretat, que pot ser difícilment perceptible. Podem trobar-nos amb atacs d'autenticació (obtenir accés per part de programes o usuaris prèviament no habilitats), escoltes (redirigir o punxar els canals de comunicació i les dades que hi circulen) o substitució (de programes, màquines, comunicacions o usuaris per d'altres, sense que es notin els canvis).

Seguretat absoluta

La seguretat absoluta no existeix. Una falsa impressió de seguretat pot ser tan perjudicial com no tenir-ne. L'àrea de la seguretat és molt dinàmica i cal mantenir actualitzats constantment els coneixements.

Una idea clara que cal tenir sempre present és que és impossible aconseguir una seguretat del 100%.

Les tècniques de seguretat són una arma de doble tall, que fàcilment poden donar-nos una falsa impressió de control del problema. La seguretat actual és un problema ampli, complex i, el que és més important, dinàmic. Mai no podem esperar o assegurar que la seguretat estigui garantida en un determinat sistema, sinó que amb força probabilitats serà una de les àrees a les quals l'administrador haurà de dedicar més temps i mantenir actualitzats els seus coneixements sobre el tema.

En aquest mòdul examinarem diferents problemàtiques amb què podem trobar-nos, com podem verificar i prevenir parts de la seguretat local i en entorns de xarxa. També examinarem tècniques de detecció d'intrusions i algunes eines bàsiques que ens poden ajudar en el control de la seguretat.

També cal esmentar que en aquest mòdul només podem fer una breu introducció a alguns dels aspectes que intervenen en la seguretat d'avui en dia. Per a qualsevol aprenentatge real amb més detall, es recomana consultar la bibliografia disponible, i els manuals associats als productes o eines comentats.

Objectius

En aquest mòdul mostrarem els continguts i les eines procedimentals per a aconseguir els objectius següents:

1. Conèixer els principals tipus i mètodes d'atacs a la seguretat dels sistemes, així com algunes contramesures bàsiques i eines útils per al seu tractament.
2. Saber fer un seguiment de la seguretat local i en xarxa en els sistemes GNU/Linux.
3. Conèixer la metodologia d'ús d'eines de detecció d'intrusions.
4. Saber elaborar mesures de prevenció mitjançant filtratge de serveis o paquets *wrappers* i tallafocs (*firewalls*).
5. Conèixer les eines de seguretat disponibles per a controlar accessos i permisos, a nivell de MAC (*Mandatory Access Control*).

1. Tipus i mètodes dels atacs

La seguretat computacional, en administració, pot ser entesa com el procés que ha de permetre a l'administrador del sistema prevenir i detectar usos no autoritzats d'aquest sistema. Les mesures de prevenció ajuden a aturar els intents d'usuaris no autoritzats (els coneguts com a **intrusos**) per a accedir a qualsevol part del sistema. La detecció ajuda a descobrir quan es van produir aquests intents o, en el cas de dur-se a terme, establir les barreres perquè no es repeteixin i poder recuperar el sistema si ha estat trencat o compromès.

Els intrusos (també coneguts col·loquialment com a *hackers*, *crackers*, *furoners, atacants* o *pirates*) normalment volen controlar el sistema, o bé per a causar funcionaments erronis, per a malmetre el sistema o les seves dades, per a guanyar recursos a la màquina o bé, simplement, per a llançar atacs contra altres sistemes i així protegir la seva veritable identitat i ocultar l'origen real dels atacs. També hi ha la possibilitat que l'atac es produeixi per a examinar o robar la informació del sistema, el pur espionatge de les accions del sistema o per a causar danys físics a la màquina, que poden consistir a formatar el disc, canviar dades, esborrar o modificar programari crític, etc.

Respecte als intrusos, cal establir algunes diferències que no solen estar gaire clares en els termes col·loquials. Normalment amb ***hacker*** [Him01] ens referim a una persona amb grans coneixements d'informàtica, més o menys apassionada pels temes de programació i seguretat informàtica i que, normalment sense finalitat malèvola, utilitza els coneixements per a protegir-se a ell mateix o protegir tercers, introduir-se en xarxes per a demostrar-ne les fallades de seguretat i, en alguns casos, per a obtenir el reconeixement públic de les seves habilitats. Un exemple seria la mateixa comunitat GNU/Linux, que deu molt als seus *hackers*, ja que cal entendre el terme *hacker* com a expert en uns temes computacionals, més que com a intrús que afecta la seguretat.

D'altra banda, trobaríem els pirates o ***crackers***. Aquí és quan s'utilitza el terme, de manera més o menys despectiva, per a referir-se a aquells que utilitzen les habilitats per a malmetre (o destrossar) sistemes, només per a obtenir fama, per motius econòmics, per ganes de causar dany o simplement per a molestar, per motius d'espionatge tecnològic, com a actes de ciberterrorisme, etc.

Així mateix, es parla de *hacking* o *cracking* o pirateig, respectivament, quan ens referim a tècniques d'estudi, detecció i protecció de la seguretat, o al contrari, a tècniques destinades a causar dany trencant la seguretat dels sistemes.

Altres termes, més comuns actualment per a diferenciar *hackers* (quan el terme s'usa de manera genèrica) en la comunitat de seguretat (a causa del mal

ús dels anteriors), són *black hat* per a referir-se a un *hacker* que viola la seguretat informàtica per raons més enllà de la malícia o per al benefici personal. Són la personificació d'allò que s'entén clàssicament per criminal informàtic. Els *hackers black hat* entren a xarxes segures per a destruir les dades o fer-les inutilitzables per a aquells que tinguin accés autoritzat. Normalment el procés que segueixen consta de l'elecció dels objectius, la recopilació d'informació i recerca i, finalment, l'execució. D'altra banda, es coneixen com *white hat* aquells *hackers* que trenquen la seguretat per raons no malicioses, potser per posar a prova els seus propis sistemes, desenvolupant programari de seguretat o analitzant la seguretat de sistemes o organitzacions sota un acord contractual. També s'acostumen a denominar *hackers* ètics. Uns altres de més curiosos són els denominats *gray hat*, que actuen com els *black hat* atacant sistemes, però s'ofereixen per a arreglar-los a un mòdic preu. O els *script kiddies*, que són inexperts i aprofiten eines automatitzades empaquetades i efectuades per altres, generalment sense cap (o poca) comprensió dels conceptes subjacents i moltes vegades sense ser amb prou feines conscients dels danys que poden provocar.

Malauradament, obtenir accés a un sistema (desprotegit o parcialment segur) és bastant més fàcil del que sembla a primera vista. Els intrusos descobreixen permanentment noves **vulnerabilitats** (anomenades de vegades *forats de seguretat*), que els permeten introduir-se en les diferents capes de programari (aplicacions, serveis o parts del sistema operatiu). Així, definirem una vulnerabilitat com l'error d'un programa (o capa de programari) que permet, mitjançant la seva explotació, violar la seguretat d'un sistema informàtic. D'altra banda, un *exploit* és un programari, conjunt de dades o seqüència d'ordres que, traient profit de vulnerabilitats, permet causar comportaments inesperats o no anticipats, a programari, maquinari o dispositius electrònics.

La complexitat cada vegada més gran del programari (i del maquinari) fa que augmenti la dificultat per a provar de manera raonable la seguretat dels sistemes informàtics. L'ús habitual dels sistemes GNU/Linux en xarxa, sigui a la mateixa Internet o en xarxes pròpies amb tecnologia TCP/IP com les intranets, ens duu a exposar els nostres sistemes, com a víctimes, a atacs de seguretat [Bur02, Fen02, Line].

El primer que cal fer és trencar el mite de la seguretat informàtica: simplement no existeix. El que sí que podem aconseguir és un cert grau de seguretat que ens faci sentir segurs dins de certs paràmetres. Però com a tal, això només és una percepció de seguretat, i com totes les percepcions, pot ser falsa i d'això podem adonar-nos-en en el darrer moment, quan ja tinguem els nostres sistemes afectats i compromesos. La conclusió lògica és que la seguretat informàtica exigeix un esforç important de constància, realisme i aprenentatge pràcticament diari.

Hem de ser capaços d'establir en els nostres sistemes unes polítiques de seguretat que ens permetin prevenir, identificar i reaccionar davant dels possibles

atacs. I tenir present que la sensació que puguem tenir de seguretat no és res més que això, una sensació. Per tant, no s'ha de descuidar cap de les polítiques implementades i cal mantenir-les al dia, de la mateixa manera que els nostres coneixements del tema.

Amenaces

Les amenaces afecten la confidencialitat, la integritat o l'accessibilitat dels nostres sistemes.

Els possibles atacs són una amenaça constant als nostres sistemes i poden comprometre'n el funcionament, així com les dades que gestionem. Davant de tot això, sempre hem de definir una certa política de requisits de seguretat sobre els nostres sistemes i dades. Les amenaces que podem patir podrien afectar els aspectes següents:

- **Confidencialitat**: la informació ha de ser accessible només a aquells que hi estiguin autoritzats; estem responent a la pregunta *qui podrà accedir a la informació?*

- **Integritat**: la informació només podrà ser modificada per aquells que hi estiguin autoritzats: què es podrà fer amb la informació?

- **Accessibilitat**: la informació ha d'estar disponible per als qui la necessitin i quan la necessitin, si hi estan autoritzats: de quina manera, i quan, es podrà accedir a la informació?

Passem a esmentar una determinada classificació (no exhaustiva) dels tipus d'atacs habituals que podem patir:

- **Autenticació**: atacs en els quals es falsifica la identitat del participant, de manera que obté accés a programes o serveis als quals en principi no tenia accés.

- **Intercepció** (o escolta): mecanisme pel qual tercers intercepten dades quan aquestes no anaven dirigits a ells.

- **Falsificació** (o reemplaçament): substitució d'alguns dels participants, tant màquines com programari o dades, per altres de falsos.

- **Robatori de recursos**: ús dels nostres recursos sense autorització.

- **Vandalisme**: al cap i a la fi, sol ser bastant comú l'existència de mecanismes que permeten interferir en el funcionament adequat del sistema o dels serveis i causar molèsties parcials i l'aturada o cancel·lació de recursos.

Els mètodes utilitzats i les tècniques necessàries poden variar molt (encara més, cada dia es creen novetats) i ens obliguen, com a administradors, a estar en contacte permanent amb el camp de la seguretat per a conèixer a què ens podem enfrontar diàriament.

Pel que fa a on es produeix l'atac, hem de tenir clar què pot fer-se o quin serà l'objectiu dels mètodes:

Finalitat dels atacs

Els atacs poden tenir finalitats destructives, inhabilitadores o d'espionatge dels nostres components (tant maquinari com programari o sistemes de comunicació).

- **Maquinari**. Quant a això, l'amenaça està directament sobre l'accessibilitat: què podrà fer algú que tingui accés al maquinari? En aquest cas normalment necessitarem mesures "físiques", com ara controls de seguretat per a accedir als locals on estiguin situades les màquines, per a evitar problemes de robatori o ruptura de l'equip a fi d'eliminar-ne el servei. També pot comprometre's la confidencialitat i la integritat si l'accés físic a les màquines permet utilitzar alguns dels dispositius, com les unitats de disc (extraïbles o no), l'arrencada de les màquines o l'accés a comptes d'usuari que podrien estar oberts.

- **Programari**. Si l'accessibilitat es veu compromesa en un atac, es poden esborrar o inutilitzar programes, denegant-ne l'accés. En cas de confidencialitat, pot provocar còpies no autoritzades de programari. En integritat podria alterar-se el funcionament per defecte del programa, perquè falli en algunes situacions o bé perquè faci tasques que puguin ser interessants per a l'atacant, o podria simplement comprometre la integritat de les dades dels programes: fer-los públics, alterar-los o simplement robar-los.

- **Dades**, ja siguin estructurades, com en els serveis de base de dades, gestió de versions (com cvs) o simples arxius. Mitjançant atacs que amenacin l'accessibilitat, aquestes dades poden ser destruïdes o eliminades, de manera que s'hi denegui l'accés. En el cas de la confidencialitat, estaríem permetent lectures no autoritzades i la integritat es veuria afectada quan es produïssin modificacions o creació de noves dades.

- **Canal de comunicació** (a la xarxa, per exemple). Per als mètodes que afecten l'accessibilitat, ens pot provocar la destrucció o l'eliminació de missatges i impedir l'accés a la xarxa. En confidencialitat, es pot donar la lectura i observació del trànsit de missatges, des de la màquina o cap a la màquina. I respecte a la integritat, podem veure'ns afectats per qualsevol modificació, retard, reordenació, duplicació o falsificació dels missatges entrants o sortints.

1.1. Tècniques utilitzades en els atacs

Els mètodes utilitzats són múltiples i poden dependre d'un element (maquinari o programari) o de la seva versió. Per tant, cal mantenir actualitzat el programari per les correccions de seguretat que vagin apareixent i seguir les indicacions del fabricant o distribuïdor per a protegir l'element, a més d'utilitzar les mesures de seguretat activa que puguem aplicar.

Malgrat això, normalment sempre hi ha tècniques o mètodes "de moda" del moment actual. Algunes indicacions breus d'aquestes tècniques d'atac (actuals) són:

Evolució de les tècniques d'atac

Les tècniques dels atacs són molt variades i evolucionen constantment pel que fa als detalls tecnològics usats.

- **Explotació de forats (*bug exploits*)**: es tracta de l'explotació de forats o errors [Cerb, Ins, San] d'un maquinari, programari, servei, protocol o el mateix sistema operatiu (per exemple, en el nucli o *kernel*) i, normalment, d'alguna de les seves versions en concret. En general, qualsevol element informàtic és més o menys propens a errors en la seva concepció o simplement a coses que no s'han tingut en compte o previst. Periòdicament, es descobreixen forats (de vegades s'anomenen *holes*, *exploits* o simplement *bugs*), que poden ser aprofitats per un atacant per a trencar la seguretat dels sistemes. Solen utilitzar-se tècniques d'atac genèriques, com les que s'expliquen a continuació, o bé tècniques particulars per a l'element afectat. Cada element tindrà un responsable, tant fabricant, desenvolupador, distribuïdor com la comunitat GNU/Linux, de produir noves versions o pedaços per a tractar aquests problemes. Nosaltres, com a administradors, tenim la responsabilitat d'estar informats i de mantenir una política d'actualització responsable per a evitar els riscos potencials d'aquests atacs. En cas que no hi hagi solucions disponibles, també podem estudiar la possibilitat d'utilitzar alternatives a l'element, o bé inhabilitar-lo fins que tinguem solucions.

- **Virus**: es tracta de programes normalment annexos a d'altres i que utilitzen mecanismes d'autocòpia i transmissió. Són habituals els virus adjunts a programes executables, a missatges de correu electrònic o incorporats en documents o programes que permeten algun llenguatge de macros (no verificat). Són potser la pesta més important de seguretat d'avui en dia en alguns sistemes.

  Els sistemes GNU/Linux estan protegits gairebé totalment (n'hi ha diversos casos, com també proves de concepte) contra aquests mecanismes vírics per diverses raons: en els programes executables, tenen un accés molt limitat al sistema, en particular al compte de l'usuari, amb excepció de l'usuari *root*, el compte del qual s'hauria de restringir als usos adequats, i també hauria de limitar-se o evitar-se completament l'ús com a usuari del sistema; el correu no sol utilitzar llenguatges de macros no verificats (com en el cas de l'Outlook i Visual Basic Script a Windows, que en el seu dia, van ser un aport important de forats d'entrada per a virus), mentre que en el cas dels documents, estem en condicions semblants, ja que no suporten llenguatges de macros no verificats (com el VBA en Microsoft Office). Tot i això, cal comentar que diversos aspectes d'aquest tipus comencen a tenir una certa importància, ja que alguns dels paquets d'ofimàtica per a GNU/Linux utilitzen ja llenguatges de macros, i els clients de correu cada vegada incorporen més suport d'HTML encastat amb suport de JavaScript, una de les fonts amb més problemes, com veurem en alguns apartats següents. Que aquests problemes no hagin estat explotats (o només mínimament) és no-

més una qüestió DE l'ús d'una plataforma, com GNU/Linux. A mesura que creix l'ús, també es fa més atractiva per als atacants.

En qualsevol cas, caldrà fer atenció al que pugui passar en un futur, ja que podrien sorgir alguns virus específics per a GNU/Linux aprofitant alguns errors o forats dels components del sistema. Un punt que sí que cal tenir en compte és el dels sistemes de correu, ja que si bé nosaltres no generarem virus, sí que podem arribar a transmetre'ls; per exemple, si el nostre sistema funciona com a *router*, *relay*, o simplement com a servidor de correu, podrien arribar missatges amb virus i aquests podrien ser enviats a d'altres. Aquí es pot aplicar alguna política de detecció i filtratge de virus, si en els nostres sistemes hi ha entorns Windows, en els quals es poden propagar virus externs que hàgim rebut. Un altre problema molt important que podria entrar dins de la categoria de virus són els correus brossa (*spam*), que si bé no solen ser utilitzats com a elements atacants (tot i que poden combinar-se amb altres tècniques, com intents de pesca), sí que podem considerar-los com un problema per la "virulència" de la seva aparició i pel cost econòmic que poden representar (pèrdues de temps i de recursos).

Cuc Morris

Un dels primers cucs, el conegut com a Cuc Morris, va ser molt important perquè va fer de la seguretat un tema important en uns moments en què hi havia certa innocència en aquests temes. Vegeu http://en.wikipedia.org/wiki/Morris_worm.

- **Cucs (*worms*)**: normalment es tracta d'un tipus de programes que aprofiten algun forat del sistema per a executar codi sense permís. Solen ser utilitzats per a aprofitar recursos de la màquina, com l'ús de CPU, quan es detecta que el sistema no funciona o no està en ús o, si són malintencionats, amb l'objectiu de robar recursos o utilitzar-los per a aturar o bloquejar el sistema. També solen utilitzar tècniques de transmissió i replicació.

- **Troians o cavalls de Troia (*trojans* o *Trojan horses*)**: programes útils que incorporen alguna funcionalitat però n'oculten d'altres, que són les utilitzades per a obtenir informació del sistema o comprometre'l. Un cas particular pot ser el dels codis de tipus mòbil en aplicacions web, com els Java, JavaScript o ActiveX; aquests normalment demanen consentiment per a executar-se (ActiveX a Windows) o tenen models limitats de què poden fer (Java, JavaScript). Però, com qualsevol programari, també poden tenir forats, i són un mètode ideal per a transmetre cavalls de Troia.

- **Portes secretes (*back doors*)**: és un mètode d'accés amagat en un programa que pot utilitzar-se per a donar accés al sistema o a les dades manejades sense que ho sapiguem. Altres efectes poden ser canviar la configuració del sistema o permetre la introducció de virus. El mecanisme que es fa servir pot ser des d'estar inclòs en algun programari comú fins a venir en un troià que produeixi portes secretes.

- **Bombes lògiques**: programa incrustat en un altre que comprova que es donin determinades condicions (temporals, accions de l'usuari, etc.) per a activar-se i emprendre accions no autoritzades.

- **Enregistradors de teclat (*keyloggers*)**: es tracta d'un programa especial que es dedica a segrestar les interaccions amb el teclat o el ratolí de l'usuari. Poden ser programes individuals o bé cavalls de Troia incorporats en altres programes. Normalment, necessitarien introduir-se en un sistema obert al qual es tingués accés (tot i que, cada vegada més sovint, poden anar incorporats en troians que s'instal·lin). La idea és captar qualsevol introducció de tecles, de manera que es capturin contrasenyes (per exemple, les bancàries), la interacció amb aplicacions, els llocs visitats per la xarxa, els formularis emplenats, etc.

- **Escaneig de ports (*port scanning*)**: més que un atac, seria un pas previ, que consistiria en la recol·lecció de possibles objectius. Bàsicament, consisteix a utilitzar eines que permetin examinar la xarxa a la recerca de màquines amb ports oberts, tant TCP, UDP com altres protocols que indiquen la presència d'alguns serveis. Per exemple, escanejar màquines buscant el port 80 TCP indica la presència de servidors web, dels quals podem obtenir informació sobre el servidor i la versió que utilitzen i així aprofitar-nos de vulnerabilitats conegudes.

- **Detectors (*sniffers*)**: permeten la captura de paquets que circulen per una xarxa. Amb les eines adequades podem analitzar comportaments de màquines: quines són servidors i quines són clients, quins protocols s'utilitzen i, en molts casos, obtenir contrasenyes de serveis no segurs. Al principi van ser molt utilitzats per a capturar contrasenyes de Telnet, rsh, rcp, FTP, etc., serveis no segurs que ja no s'haurien d'utilitzar (cal fer servir, en canvi, les versions segures: ssh, scp, sftp). Tant els detectors com els escàners de ports no són necessàriament eines d'atac, ja que també poden servir per a analitzar les nostres xarxes i detectar errors, o simplement per a analitzar el nostre trànsit. Normalment, un intrús sol utilitzar tant les tècniques d'escaneig com les dels detectors amb l'objectiu de trobar les vulnerabilitats del sistema, o bé per a conèixer dades d'un sistema desconegut (escàners) o bé per a analitzar-ne la interacció interna (detectors).

- **Segrest (*hijacking*)**: són tècniques que intenten col·locar una màquina de manera que intercepti o reprodueixi el funcionament d'algun servei en una altra màquina de la qual ha punxat la comunicació. Solen ser habituals els casos per a correu electrònic, transferència de fitxers o web. Per exemple, en el cas web, es pot capturar una sessió i reproduir el que l'usuari està fent, pàgines visitades, interacció amb formularis, etc. En alguns casos pot ser a causa de segrestos en l'àmbit de xarxa, perquè es capturen o escolten paquets de les comunicacions, o pot produir-se un segrest fora de línia (*offline*) mitjançant l'execució arbitrària de codi de *scripts*. Per exemple, és habitual, en sistemes de correu electrònic per web (*webmail*) o en aplicacions web que incloguin l'ús d'HTML en algunes de les seves finestres, que el sistema permeti injectar codi JavaScript o PHP (o altres llenguatges d'*script* similars) i això, si no es controla o limita de manera correcta, pot provocar el robatori de dades de l'usuari (o bé els identificadors de sessió o bé les ga-

letes –*cookies*– utilitzades), cosa que permet a l'atacant reproduir o segrestar *a posteriori* les sessions de correu o d'aplicació web sense necessitat de cap contrasenya, i així segrestar la sessió i la identitat de l'usuari.

- **Desbordament (*buffer overflows*)**: tècnica bastant complexa que aprofita errors de programació en les aplicacions. La idea bàsica és aprofitar desbordaments (*overflows*) de memòria intermèdia (*buffers*) de l'aplicació, ja siguin cues, matrius, vectors, etc. Si no se'n controlen els límits, un programa atacant pot generar un missatge o dada més gran de l'esperat i provocar fallades mitjançant l'escriptura per sobre dels límits permesos per l'estructura de dades i, així, sobreescriure memòria adjacent. Per exemple, en moltes aplicacions C amb *buffers* mal escrits, en matrius, si sobrepassem el límit podem provocar una sobrescriptura del codi del programa, la qual cosa provoca un funcionament incorrecte o la caiguda del servei o màquina. Encara més, una variant més complexa permet incorporar en l'atac trossos de programa (compilats en C o bé *scripts* de l'intèrpret d'ordres) que poden permetre l'execució de qualsevol codi que l'atacant vulgui introduir.

  Algunes variacions d'aquesta tècnica es poden aprofitar per a atacar de manera similar altres parts del codi executable d'un programa, com per exemple la pila del programa (parlem llavors de *stack overflows*) o la gestió de les peticions dinàmiques de memòria que faci l'executable (*heap overflows*). L'alteració tant de la pila com de la memòria dinàmica també pot permetre a l'atacant l'execució de codi arbitrari afegit o la caiguda de l'executable o servei.

- **Denegació de servei, *denial of service*, DoS**: aquest tipus d'atac provoca que la màquina caigui o que se sobrecarreguin un o més serveis, de manera que no siguin utilitzables. Una altra tècnica és la DDoS (*distributed DoS*, DoS distribuïda), que es basa a utilitzar un conjunt de màquines distribuïdes perquè produeixin l'atac o sobrecàrrega de servei. Aquest tipus d'atacs se solen solucionar amb actualitzacions del programari, i amb una correcta configuració dels paràmetres de control de càrrega del servei, ja que normalment es veuen afectats aquells serveis que no van ser pensats per a una càrrega de treball determinada i no se'n controla la saturació. Els atacs DoS i DDoS són bastant utilitzats en atacs a llocs web o servidors DNS, que es veuen afectats per vulnerabilitats dels servidors, per exemple, de versions concretes d'Apache o BIND. Un altre aspecte que es pot tenir en compte és que el nostre sistema també podria ser usat per a atacs de tipus DDoS, mitjançant control, ja sigui a través d'una porta secreta o d'un troià que formi part d'una *botnet*, per exemple, que disposi d'una sèrie de programes a múltiples màquines interconnectades, les quals podrien trobar-se en un estat latent i participar *a posteriori* en atacs DDoS a demanda.

  Un exemple d'aquest atac (DoS) bastant senzill és el conegut com a *SYN flood*, que tracta de generar paquets TCP que obren una connexió, però ja no hi fan res més, simplement la deixen oberta. Això consumeix recursos del sistema en estructures de dades del nucli i recursos de connexió per xar-

xa. Si es repeteix aquest atac centenars o milers de vegades, s'aconsegueix ocupar tots els recursos sense utilitzar-los, de manera que quan alguns usuaris vulguin utilitzar el servei, els sigui denegat perquè els recursos estan ocupats. Un altre cas conegut és el bombardeig de correu (*mail bombing*) o, simplement, el reenviament de correu (normalment amb emissor fals) fins que se saturen els comptes de correu i el sistema de correu cau o es torna tan lent que és inutilitzable. Aquests atacs són, en certa manera, de realització senzilla amb les eines adequades, i no tenen una solució fàcil, ja que s'aprofiten del funcionament intern dels protocols i serveis; en aquests casos hem de prendre mesures de detecció i control posterior, així com posar límits als paràmetres i la càrrega de serveis involucrats en aquests problemes.

Enllaços d'interès

Sobre *SYN flood*, vegeu: http://www.cert.org/advisories/CA-1996-21.html.
Sobre bombardeig de correu i correu brossa, vegeu: http://www.cert.org/tech_tips/email_bombing_spamming.html.

- **Falsejament d'identitat (*spoofing*)**: les tècniques de falsejament d'identitat engloben diversos mètodes (normalment complexos) per a falsificar tant la informació com els participants en una transmissió (origen i/o destinació). Alguns mètodes de falsejament d'identitat són els següents:

– Falsejament d'IP (*IP spoofing*), falsificació d'una màquina, de manera que generi trànsit fals o escolti trànsit que anava dirigit a una altra màquina. Combinat amb altres atacs, pot saltar-se fins i tot la protecció de tallafocs.

– Falsejament d'ARP (*ARP spoofing*), tècnica complexa (utilitza un DDoS) que intenta falsificar les adreces de fonts i destinataris d'una xarxa, mitjançant atacs de les memòries cau d'ARP que posseeixen les màquines, de manera que se substitueixin les adreces reals per d'altres en diversos punts d'una xarxa. Aquesta tècnica permet saltar-se tot tipus de proteccions, tallafocs inclosos, però no és una tècnica senzilla.

Enllaç d'interès

Vegeu el cas de Verisign-Microsoft a: http://www.computerworld.com/softwaretopics/os/windows/story/0,10801,59099,00.html.

– Correu electrònic. És potser el més senzill. Es tracta de generar continguts de correus falsos, tant pel contingut com per l'adreça d'origen. En aquest tipus són bastant utilitzades les tècniques del que s'anomena *enginyeria social*, que bàsicament intenta enganyar l'usuari d'una manera creïble. Un exemple clàssic són els correus falsos de l'administrador del sistema, o bé del banc on tenim el nostre compte corrent, en què s'esmenten problemes amb la gestió dels nostres comptes i ens demanen enviar informació confidencial o la contrasenya per a solucionar-los, o se'ns demana que es canviï la contrasenya per una altra de concreta. Sorprenentment, aquesta tècnica (també coneguda com a *pesca* o *phishing*) aconsegueix enganyar un nombre considerable d'usuaris. Fins i tot amb enginyeria social de mètodes senzills: algun pirata famós comentava que el seu mètode preferit era el telèfon. Com a exemple, exposem el cas d'una empresa de certificació (Verisign), de la qual els pirates van obtenir la signatura privada de programari de Microsoft simplement amb una trucada, esmentant que trucaven de part de l'empresa, que se'ls havia presentat un problema i que tornaven a necessitar la clau. Resumint, un grau molt alt de seguretat in-

formàtica se'n pot anar en orris per una simple trucada o un correu que un usuari malinterpreti.

- ***SQL injection***: és una tècnica orientada a bases de dades, i a servidors web en particular, que s'aprofita generalment de programació incorrecta de formularis web, en els quals no es controla correctament la informació que es proporciona: no es determina que la informació d'entrada sigui del tipus correcte (molt tipificada respecte al que s'espera) o no es controla quins tipus o caràcters literals s'introdueixen. La tècnica s'aprofita del fet que els caràcters literals obtinguts pels formularis (per exemple web, tot i que els atacs poden patir-se des de qualsevol API que permeti l'accés a una base de dades, per exemple PHP o PERL) són utilitzats directament per a construir les consultes (en SQL) que atacaran una determinada base de dades (a la qual en principi no es té accés directe). Normalment, si existeixen les vulnerabilitats i hi ha poc control dels formularis, es pot injectar codi SQL al formulari, de manera que es construeixin consultes SQL que proporcionin la informació cercada. En casos dràstics podria obtenir-se la informació de seguretat (usuaris i contrasenyes de la base de dades) i, fins i tot, taules o la base de dades sencera, i també podrien produir-se pèrdues d'informació o esborraments intencionats de dades. En particular, aquesta tècnica en ambients web pot dur a resultats greus (per a l'empresa proveïdora del servei, amb fortes conseqüències econòmiques), a causa de les lleis que protegeixen la privacitat de dades personals, que es poden veure compromeses, fer-se públiques o ser venudes a tercers si s'obtenen per un atac d'aquest tipus. En aquest cas d'atac, més que una qüestió de seguretat del sistema, es tracta d'un problema de programació i control amb tipatge fort de les dades esperades en l'aplicació, a més del control adequat i el coneixement de les vulnerabilitats presents en el programari que es fa servir (base de dades, servidor web, API com PHP, PERL, etc.).

- ***Cross-site scripting*, XSS**: és una altra problemàtica relacionada amb ambients web, en particular amb alteracions del codi HTML i *scripts* quan un usuari visualitza un determinat lloc web que pot ser alterat dinàmicament. S'aprofita generalment dels errors a l'hora de validar codi HTML (tots els navegadors tenen més o menys problemes, per la mateixa definició de l'HTML original, que permet llegir pràcticament qualsevol codi HTML per incorrecte que sigui). En alguns casos, la utilització de vulnerabilitats pot ser directa, mitjançant *scripts* a la pàgina web, però normalment els navegadors en tenen un bon control. D'altra banda, indirectament hi ha tècniques que permeten inserir codi de *script*, o bé mitjançant accés a les galetes de l'usuari des del navegador, o bé mitjançant l'alteració del procés pel qual es redirecciona una pàgina web a l'altra. També hi ha tècniques mitjançant marcs (*frames*) que permeten redirigir l'HTML que s'està veient o penjar directament el navegador. En particular, poden ser vulnerables els motors de cerca dels llocs web, que poden permetre l'execució de codi de *scripts*. En general, són atacs amb diverses tècniques complexes, però amb l'objectiu de capturar informació com ara galetes, que poden ser usades

en sessions i permetre així la substitució d'una determinada persona mitjançant readreçaments de llocs web o l'obtenció d'informació seva. Novament, des de la perspectiva de sistema, és més una qüestió de programari en ús. Cal controlar i conèixer les vulnerabilitats detectades en navegadors (i aprofitar els recursos que ofereixen per a evitar aquestes tècniques) i controlar l'ús de programari (motors de cerca emprats, versions del servidor web i API utilitzades en els desenvolupaments).

En general, algunes recomanacions (molt bàsiques) per a la seguretat podrien ser:

- Controlar un factor problemàtic: els usuaris. Un dels factors que poden afectar més la seguretat és la confidencialitat de les contrasenyes, i aquesta es veu afectada pel comportament dels usuaris; això facilita a possibles atacants les accions des de l'interior del mateix sistema. La majoria dels atacs solen venir de dins del sistema, és a dir, una vegada l'atacant ja hi ha guanyat l'accés.

- Entre els usuaris, sempre hi ha aquell que és una mica oblidadís (o indiscret), que oblida la contrasenya cada dos per tres, l'esmenta en converses, l'escriu en un paper que oblida, o que la deixa escrita al costat (o enganxat) de l'ordinador o sobre la taula de treball, o que simplement la deixa a altres usuaris o coneguts. Un altre tipus és el que col·loca contrasenyes molt predictibles, com el seu identificador d'usuari, el seu nom, el seu DNI, el nom de la seva parella, el de la seva mare, el de la seva mascota, etc., coses que amb un mínim d'informació poden trobar-se fàcilment (i més en aquests moments d'apogeu de les xarxes socials, en què és fàcil obtenir aquest tipus d'informació des de diferents xarxes en què participi l'usuari). Un altre cas són els usuaris normals amb un cert coneixement que col·loquen contrasenyes vàlides, però sempre cal tenir en compte que hi ha mecanismes que poden trobar-les (trencament de contrasenyes, detectors, suplantació, etc.). Cal establir una certa **cultura de la seguretat** entre els usuaris i, mitjançant diferents tècniques, obligar-los que canviïn les contrasenyes, que no utilitzin paraules típiques, que usin contrasenyes llargues (amb més de 6 o 8 caràcters, com a mínim), etc. Darrerament, en moltes empreses i institucions s'està implantant la tècnica de fer signar un contracte (o carta de compromisos) a l'usuari, de manera que se l'obliga a no divulgar la contrasenya o cometre actes de vandalisme o atacs des del seu compte (és clar que això no impedeix que d'altres ho facin per ell).

- No utilitzar ni executar programes dels quals no puguem garantir l'origen. Normalment, molts distribuïdors utilitzen mecanismes de comprovació de signatures per a verificar que els paquets de programari són tals, com per exemple les sumes md5 (ordre `md5sum`) o la utilització de signatures GPG [Hatd] (ordre `gpg`). El venedor o distribuïdor proporciona una suma md5 del seu arxiu (o imatge de CD/DVD), de manera que podem comprovarne l'autenticitat. Últimament, en les distribucions s'estan fent servir tant

signatures per a paquets individuals com signatures per als dipòsits de paquets, com a mecanisme per a garantir la fiabilitat del proveïdor.

- No utilitzar usuaris privilegiats (com el *root*) per al treball normal de la màquina, ja que qualsevol programa (o aplicació) tindria els permisos per a accedir a qualsevol zona del sistema d'arxius o a serveis en execució.

- No accedir remotament amb usuaris privilegiats ni executar programes que puguin tenir privilegis. I més si no coneixem, o no hem comprovat, els nivells de seguretat del sistema i les seves connexions. En particular, un altre dels problemes actuals són les xarxes sense fil (com Wi-Fi) insegures, ja que, per exemple, alguns xifratges com el que es fa servir en xarxes Wi-Fi WEP són molt febles. No s'han d'usar connexions crítiques sobre xarxes insegures.

- No utilitzar elements (programes o serveis) que no sabem com actuen ni intentar descubrir-ho amb execucions repetides.

Aquestes mesures poden ser poc productives, però si no hem assegurat el sistema, no podem tenir cap control sobre el que pot passar i, tot i així, ningú no garanteix que no es pugui introduir algun programa maliciós que burli la seguretat si l'executem amb els permisos adequats. En general, cal considerar que hem de tenir una vigilància activa amb tot tipus d'activitats que impliquin accessos i execució de tasques de manera més o menys privilegiada i un ús de mecanismes de comunicació insegurs.

1.2. Contramesures

Pel que fa a les mesures que es poden prendre contra els tipus d'atacs presentats, en podem trobar algunes de preventives i de detecció del que succeeix en els nostres sistemes.

Vegem alguns tipus de mesures que podríem prendre en els àmbits de prevenció i detecció d'intrusos (s'esmenten eines útils, algunes de les quals examinarem més endavant):

- **Trencament de contrasenyes (*password cracking*)**: en atacs de força bruta per a trencar les contrasenyes sol ser habitual intentar obtenir l'accés per inici de sessió de manera repetida; si s'aconsegueix entrar, la seguretat de l'usuari ha estat compromesa i es deixa la porta oberta a altres tipus d'atacs com, per exemple, les portes secretes o, simplement, a la destrucció del compte. Per a prevenir aquest tipus d'atacs, cal reforçar la política de contrasenyes, demanant una longitud mínima i canvis de contrasenya periòdics. Una cosa que cal evitar és l'ús de paraules comunes en les contrasenyes: molts d'aquests atacs es fan mitjançant la força bruta, amb un fitxer de diccionari (amb paraules en l'idioma de l'usuari, termes comuns,

noms propis, argot, etc.). Aquest tipus de contrasenyes seran les primeres a caure. També pot ser fàcil obtenir informació de l'atacat, com ara noms, DNI o la seva adreça i informació procedent de les seves xarxes socials, i usar aquestes dades per a provar les contrasenyes. Per tot això tampoc no es recomanen contrasenyes amb DNI, noms (propis o de familiars, etc.), adreces, noms de mascotes, etc. Una bona elecció sol ser triar contrasenyes d'entre 6 i 8 caràcters com a mínim i que continguin caràcters alfabètics, numèrics i algun caràcter especial. Malgrat que la contrasenya estigui ben triada, pot ser insegura si s'utilitza en serveis o en xarxes obertes, no segures. Per tant, es recomana reforçar els serveis mitjançant tècniques i serveis de xifratge que protegeixin les contrasenyes i les comunicacions. Al contrari, s'han d'evitar (o no usar) tots aquells serveis que no suportin xifratge i que, consegüentment, siguin susceptibles de ser atacats amb mètodes, per exemple, de detectors; entre aquests podríem incloure serveis com Telnet, FTP, rsh i rlogin, entre d'altres, dels quals ja hi ha alternatives més segures.

- **Explotació de forats**: cal evitar disposar de programes que no s'utilitzin, que siguin antics o que no s'actualitzin (perquè són obsolets). Cal aplicar els últims pedaços i actualitzacions disponibles, tant per a les aplicacions com per al sistema operatiu, provar eines que detectin vulnerabilitats i mantenir-se al dia de les vulnerabilitats que es vagin descobrint. Encara que cal assenyalar que, en determinades ocasions, les pròpies correccions d'alguna fallada de seguretat passada ens en poden portar d'altres de noves. Com a cas particular, cal estar especialment atent a les divulgacions de vulnerabilitats de tipus *0day*, que acostumen a explotar vulnerabilitats no conegudes i que els desenvolupadors no han tingut temps d'arreglar. En alguns casos, la finestra de vulnerabilitat (temps entre el coneixement de la vulnerabilitat i la seva solució), contra el que podria semblar, podria allargar-se setmanes, o fins i tot anys, i convertir-se en una fallada de seguretat desconeguda durant molt de temps per al públic general, però no així per a cert mercat, que pot involucrar des de *crackers* fins a agències governamentals, les quals aprofiten aquestes *0day* per a motius polítics (ciberguerra), invasió de la privadesa o espionatge industrial.

Enllaços d'interès

Sobre vulnerabilitats, una bona eina és Nessus. Per a descobrir vulnerabilitats noves, vegeu CERT a: http://www.cert.org/advisories/, lloc antic, i http://www.us-cert.gov/cas/techalerts/index.html.

Cas Snowden

Vegeu NSA *versus* Snowden: http://en.wikipedia.org/wiki/Edward_Snowden

- **Virus**: s'han d'utilitzar mecanismes o programes antivirus, sistemes de filtratge de missatges sospitosos i evitar l'execució de sistemes de macros (que no es puguin verificar). No convé minimitzar els possibles efectes dels virus, cada dia es perfeccionen més i tècnicament és possible crear virus simples que poden desactivar xarxes en qüestió de minuts (només cal observar alguns dels virus dels últims anys en ambients Windows). En especial, si en els nostres sistemes hi ha entorns Windows, seria interessant la possibilitat d'examinar virus que puguin afectar aquests sistemes. Hi ha antivirus per a UNIX i Linux, com ClamAV, que evitaran que es transmetin virus als nostres sistemes interns. En sistemes de correu basats en GNU/Linux, podem establir combinacions de productes antivirus i contra el correu brossa per a evitar aquestes transmissions, com per exemple ClamAV combinat amb altres productes, com SpamAssasin.

Enllaços d'interès

Vegeu pedaços i vulnerabilitats per al sistema operatiu a: http://www.debian.org/security, http://www.redhat.com/security, http://fedoraproject.org/wiki/Security.

- **Cucs**: cal controlar l'ús de les nostres màquines o usuaris en hores no previstes, amb patrons de comportament infreqüents, així com el control del trànsit de sortida i entrada.

- **Cavalls de Troia**: s'ha de verificar periòdicament la integritat dels programes mitjançant mecanismes de suma o de signatures, detectar el trànsit anòmal de sortida o entrada al sistema i utilitzar tallafocs per a bloquejar trànsit sospitós. Una versió bastant perillosa dels troians la formen les eines d'intrusió (*rootkits*, que comentarem més endavant), que fan més d'una funció gràcies a un conjunt variat d'eines. Per a la verificació de la integritat, podem utilitzar mecanismes de sumes, com md5 o gpg, o eines que automatitzen aquest procés, com Tripwire o AIDE.

- **Portes secretes**: cal obtenir dels proveïdors o venedors del programari la certificació que no conté cap tipus de porta secreta amagada no documentada i, sens dubte, acceptar el programari provinent només de llocs que ofereixin garanties. Quan el programari sigui de tercers o de fonts que podrien haver modificat el programari original, molts fabricants (o distribuïdors) integren algun tipus de verificació de programari basat en codis de suma o signatures digitals (tipus md5 o gpg) [Hatd]. Sempre que aquestes estiguin disponibles, fóra útil verificar-les abans d'instal·lar el programari. També pot provar-se el sistema de manera intensiva, abans de col·locar-lo com a sistema de producció. Un altre problema pot consistir en l'alteració del programari *a posteriori*. En aquest cas també poden ser útils els sistemes de signatures o sumes per a crear codis sobre programari ja instal·lat i controlar que no es produeixin canvis en programari vital. També resulten útils les còpies de seguretat, amb les quals podem fer comparacions per a detectar canvis.

- **Bombes lògiques**: en aquest cas, s'acostumen a ocultar amb activacions per temps o per accions de l'usuari. Podem verificar que no existeixin en el sistema treballs no interactius introduïts de tipus `crontab`, `at` i altres processos (per exemple, de tipus `nohup`), que disposin d'execució periòdica o que estiguin en execució en segon pla des de fa molt de temps (ordres `w`, `jobs`). En qualsevol cas, podrien utilitzar-se mesures preventives que impedissin treballs programats no interactius als usuaris (`crontab`) o que només els permetessin a aquells que realment els necessitin.

- **Registres de teclat i eines d'intrusió**: en aquest cas hi haurà algun procés intermedi que intentarà capturar les nostres pulsacions de tecles i les emmagatzemarà o comunicarà a algun lloc. Caldrà examinar situacions en què aparegui algun procés estrany pertanyent al nostre usuari, o bé detectar si tenim algun fitxer obert amb el qual no estiguem treballant directament (per exemple, podria ser d'ajuda `lsof`, vegeu man), o bé connexions de xarxa, si es tractés d'un registrador de tecleig amb tramesa externa. Per a provar un funcionament molt bàsic d'un registrador de tecleig molt senzill, pot veure's l'ordre de sistema `script` (vegeu man script). L'altre cas,

Enllaç d'interès

L'eina chkrootkit es pot trobar a: http://www.chkrootkit.org.

l'eina d'intrusió (que sol incloure també algun registre de teclat) sol ser un paquet d'uns quants programes amb diverses tècniques, i permet a l'atacant, una vegada ha entrat en un compte, utilitzar diversos elements, com un registre de teclat, portes secretes, cavalls de Troia (substituint ordres del sistema), etc., per a obtenir informació i portes d'entrada al sistema. Moltes vegades s'acompanya de programes que netegen els registres, per a eliminar les proves de la intrusió. Un cas particularment perillós el formen les eines d'intrusió (*kernel rootkits*), que s'usen o vénen en forma de mòduls de nucli, la qual cosa els permet actuar en l'àmbit del mateix nucli. Una eina útil per a verificar les eines d'intrusió és `chkrootkit`, o altres alternatives, com ara `rkhunter`.

- **Escaneig de ports**: els escàners solen llançar sobre un o més sistemes, bucles d'escaneig de ports coneguts, per detectar els que queden oberts i aquells en els quals els serveis estaran funcionant (i obtenir, per tant, informació de les versions dels serveis), i així conèixer si podrien ésser susceptibles d'atacs.

- **Detectors**: han d'evitar-se intercepcions i impedir així la possibilitat que s'introdueixin escoltes. Una tècnica és l'ús de maquinari específic per a la construcció de la xarxa, de manera que pugui dividir-se en segments perquè el trànsit només circuli per la zona que s'utilitzarà, posar tallafocs per a unir aquests segments i poder controlar el trànsit d'entrada i sortida. També cal usar tècniques de xifratge perquè els missatges no puguin ser llegits i interpretats per algú que escolti la xarxa. Per al cas tant d'escàners com de detectors, podem utilitzar eines com `Wireshark` (antic `Ethereal`) i `Snort` per a fer comprovacions sobre la nostra xarxa o, per a l'escaneig de ports, *Nmap*. En el cas dels detectors, es poden detectar a la xarxa mitjançant la cerca de màquines en mode Ethernet promiscu (estan a les escoltes de qualsevol paquet que circula). Normalment, la targeta de xarxa només hauria de capturar el trànsit que va cap a ella (o de tipus *broadcast* o *multicast*).

- **Segrest**: en aquest cas algunes contramesures són implementar mecanismes de xifratge en els serveis, demanar autenticació i, si és possible, que aquesta autenticació es renovi periòdicament; controlar el trànsit entrant o sortint mitjançant tallafocs, i monitorar la xarxa per a detectar fluxos de trànsit sospitosos. En casos d'aplicacions web o de correu electrònic, limitar al màxim la utilització de codis de *script* o, senzillament, eliminar la possibilitat d'execució de *scripts* interns de finestres HTML quan provinguin de fonts no fiables.

- **Desbordaments**: solen ser habituals en *bugs* o forats del sistema i solen solucionar-se (si han estat prèviament detectats) mitjançant una actualització del programari o paquet afectat. En qualsevol cas, poden observar-se, gràcies als registres del sistema, situacions estranyes de caiguda de serveis que haurien d'estar funcionant. També poden maximitzar-se els controls de processos i accessos a recursos per a aïllar el problema quan es produeixi

en entorns d'accés controlat, com el que ofereix SELinux. Existeixen altres alternatives, com ara Grsecurity o PaX, que són pedaços dels nuclis oficials de Linux per a obtenir proteccions addicionals en aquest sentit, tant per a problemes de desbordament de *buffers*, com de pila o *heap*.

Vegeu també

SELinux es tracta en l'apartat 4 d'aquest mòdul.

- **Denegació de servei i altres com *SYN flood* o bombardeig de correu**: s'han de prendre mesures de bloqueig de trànsit innecessari en la nostra xarxa (per exemple, per mitjà de tallafocs). En els serveis en què es pugui, caldrà controlar les mides de la memòria intermèdia, el nombre de clients per atendre, el temps d'espera (*timeouts*) de tancament de connexions, les capacitats del servei, etc.

- **Falsejament d'identitat**: a) falsejament d'IP, b) falsejament d'ARP, c) correu electrònic. Aquests casos necessiten un xifratge fort dels serveis, control per tallafocs i mecanismes d'autenticació basats en diversos aspectes (per exemple, no basar-se en la IP, si pogués veure's compromesa). Es poden aplicar mecanismes que controlin les sessions establertes i que es basin en diversos paràmetres de la màquina alhora (sistema operatiu, processador, IP, adreça Ethernet, etc.). També es poden monitorar sistemes DNS, caus d'ARP, cues de correu, etc., per a detectar canvis en la informació que n'invalidin d'anteriors.

- **Enginyeria social**: no és pròpiament una qüestió informàtica, però també és necessària perquè les persones no empitjorin la seguretat. Existeixen diverses mesures adequades, com augmentar la informació o educar els usuaris (perquè no divulguin dades privades ni informació que pugui oferir pistes sobre les contrasenyes, advertir-los sobre l'ús que facin de les xarxes socials, etc.) i divulgar tècniques bàsiques per a millorar la seva seguretat. També s'han de tenir en compte altres temes de seguretat: controlar quin personal disposarà d'informació crítica de seguretat i en quines condicions pot cedir-la a d'altres. Els serveis d'ajuda i manteniment d'una empresa poden ser un punt crític, i ha de controlar-se qui té informació de seguretat, i com la fa servir.

Respecte als usuaris finals, cal esforçar-se per a millorar la seva cultura de seguretat, tant en contrasenyes (i en la deixadesa del seu ús), com en la gestió del programari que utilitzen diàriament, per tal de mantenir-los actualitzats, així com proporcionar serveis confiables.

2. Seguretat del sistema

Davant dels possibles atacs, hem de disposar de mecanismes de prevenció, detecció i recuperació dels nostres sistemes.

Per a la prevenció local, cal examinar els diferents mecanismes d'autenticació i permisos d'accés als recursos per a poder definir-los correctament, de manera que es garanteixi la confidencialitat i la integritat de la nostra informació.

Fins i tot en aquest cas, no estarem protegits d'atacants que ja hagin obtingut accés al nostre sistema, ni d'usuaris hostils que vulguin saltar-se les restriccions imposades en el sistema.

Xifratge de sistemes de fitxers

En els sistemes GNU/Linux, s'implementen diverses opcions per al xifratge de dades, per exemple destacar *Encfs*, un sistema de fitxers en espai d'usuari que ens permet xifrar les dades d'usuari; vegeu: http://www.arg0.net/encfs

Respecte a la seguretat en xarxa, hem de garantir que els recursos que oferim (si proporcionem uns determinats serveis) tinguin els paràmetres de confidencialitat necessaris (i encara més si aquests estan garantits per lleis jurídiques nacionals, l'incompliment de les quals pot provocar sancions greus i publicitat negativa indirecta per la falta de compliment). Els serveis oferts no poden ser usats per tercers no desitjats, de manera que un primer pas serà controlar que els serveis oferts siguin els que realment volem i que no estem oferint, a més, altres serveis que no tenim controlats. En el cas de serveis dels quals nosaltres som clients, també caldrà garantir els mecanismes d'autenticació, en el sentit d'accedir als servidors correctes i que no hi hagi casos de suplantació de serveis o servidors (normalment força difícils de detectar a nivell d'usuari).

Quant a les aplicacions i als mateixos serveis, a més de garantir la configuració correcta de nivells d'accés mitjançant permisos i l'autenticació dels usuaris permesos, haurem de vigilar la possible explotació d'errors en el programari. Qualsevol aplicació, per molt ben dissenyada i implementada que estigui, pot tenir un nombre més o menys alt d'errors que poden ser aprofitats per a, amb certes tècniques, saltar-se les restriccions imposades. En aquest cas, practiquem una política de prevenció que passa per mantenir el sistema actualitzat en la mesura del que sigui possible, de manera que, o bé l'actualitzem davant de qualsevol nova correcció, o bé som conservadors i mantenim aquelles versions que siguin més estables en qüestió de seguretat. Normalment, això significa verificar periòdicament uns quants llocs de seguretat, de solvència coneguda, per a conèixer les últimes fallades detectades en el programari, tant d'aplicacions com de sistema, i les vulnerabilitats que se'n deriven i que podrien exposar els nostres sistemes a fallades de seguretat, localment o per xarxa.

3. Seguretat local

La seguretat local [Pen, Hatb] és bàsica per a la protecció del sistema [Deb, Hatc], ja que, normalment, després d'un primer intent d'accés des de la xarxa, és la segona barrera de protecció abans que un atac aconsegueixi fer-se amb part del control de la màquina. A més, la majoria dels atacs acaben fent ús de recursos interns del sistema.

Accés local

Diversos atacs, tot i que procedeixin de l'exterior, tenen com a finalitat aconseguir un accés local.

3.1. *Bootloaders*

Respecte a la seguretat local, ja en l'arrencada se'ns poden presentar problemes a causa de l'accés físic que un intrús pogués tenir a la màquina.

Un dels problemes es troba en la mateixa arrencada del sistema. Si el sistema pot arrencar-se des de dispositius extraïbles o des de CD/DVD, un atacant podria accedir a les dades d'una partició GNU/Linux (o també en entorns Windows) solament muntant el sistema de fitxers i podria col·locar-se com a usuari *root* sense necessitat de conèixer cap contrasenya. En aquest cas, cal protegir l'arrencada del sistema des de la BIOS, per exemple, protegint-ne l'accés mitjançant una contrasenya, de manera que no es permeti l'arrencada des de CD/DVD (amb un CD autònom o *live CD*, per exemple), USB o altres connexions externes.

També és raonable actualitzar la BIOS, ja que també pot tenir errors de seguretat. A més, cal anar amb compte, perquè la majoria de fabricants de BIOS ofereixen contrasenyes addicionals conegudes (una espècie de porta secreta), de manera que no podem dependre d'aquestes mesures en exclusiva.

El pas següent és protegir el carregador de l'arrencada (*bootloader*), Lilo o Grub, de manera que l'atacant no pugui modificar les opcions d'arrencada del nucli o modificar directament l'arrencada (cas de Grub). Qualsevol dels dos pot protegir-se també per mitjà de contrasenyes addicionals.

En Grub-Legacy (Grub1), el fitxer `/sbin/grub-md5-crypt` demana la contrasenya i genera una suma md5 associada. En algunes distribucions en què no està disponible aquesta ordre, es pot fer de manera alternativa durant l'arrencada: en carregar Grub, accedim amb la tecla `c` a *shell* de Grub en el moment d'inici i introduïm `md5crypt` per obtenir el *hash* md5 associat a una contrasenya que proposem.

En Grub, el fitxer `/sbin/grub-md5-crypt` demana la contrasenya i genera una suma md5 associada. En algunes distribucions en què no està disponible aquesta ordre, pot fer-se de manera alternativa durant l'arrencada: en carregar Grub, accedim amb la tecla *C* a l'intèrpret d'ordres de Grub i hi introduïm `md5crypt` per a obtenir el *hash* md5 associat a una contrasenya que proposem.

Després, el valor obtingut s'introdueix a `/boot/grub/grub.conf`. Sota la línia `timeout` s'introdueix:

```
password --md5 suma-md5-calculada
```

Per a Lilo es col·loca bé una contrasenya global amb

```
password=contrasenya
```

o bé una en la partició que vulguem:

```
image = /boot/vmlinuz-version
password = contrasenya
restricted
```

En aquest cas `restricted` indica, a més, que no es podran canviar els paràmetres que es passen al nucli des de l'indicador d'ordres. Cal anar amb compte de posar el fitxer `/etc/lilo.conf` protegit en el mode de només lectura/escriptura des del *root* (`chmod 600`), si no, qualsevol podria llegir les contrasenyes.

En el cas de Grub 2, és possible fer-ho mitjançant *menuentry*, disponible en la configuració, definint *passwords* per a usuaris concrets, que després es poden afegir a cada *menuentry* definida, a més hi ha la possibilitat d'especificar un superusuari que disposa d'accés a totes les entrades. Un exemple simple de configuració seria el següent: un superusari que té accés a totes les entrades, mentre que un segon usuari només pot accedir a la segona entrada.

```
set superusers=rootgrub
password rootgrub password1
password otheruser password2

menuentry "GNU/Linux" {
set root=(hd0,1)
linux /vmlinuz
}
```

```
menuentry "Windows" --users otheruser {
set root=(hd0,2)
chainloader +1
}
```

En aquest cas, la configuració de Grub2 fa servir *passwords* en net, i cal assegurar-se que el fitxer no disposi d'accés de lectura per a altres usuaris, o en el seu lloc fer servir mètodes alternatius per a la creació de *passwords*, per exemple, mitjançant *grub-mkpasswd-pbkdf2*, que ens permetrà generar *hashes* del *password* que cal utilitzar. En cas de voler deixar *menuentries* disponibles per a tots, també pot usar-se *–unrestricted* en la definició de l'entrada.

En aquest darrer punt val la pena recordar una recomanació general: els fitxers amb informació sensible mai no haurien d'estar disponibles amb permisos de lectura a tercers, i sempre que sigui possible hauríem d'evitar-los o xifrar-los. En particular, per a dades sensibles, seria desitjable algun tipus de sistema de fitxers que permetés el xifratge dels fitxers presents (o fins i tot el disc o sistema sencer). En aquest sentit podem destacar Encryptfs i EncFS, que ja està disponible en diverses distribucions, per al xifratge dels arxius dels directoris privats dels usuaris o el sistema de fitxers sencer.

Sistemes de fitxers xifrats

Els sistemes de fitxers xifrats també són una bona opció per a protegir les dades sensibles d'usuaris, com per exemple, entre d'altres, Ecryptfs i EncFS. Vegeu, per exemple, Encfs: http://www.arg0.net/encfs Una altra opció són sistemes de xifratges de disc, com dm-crypt/LUKS: https://code.google.com/p/cryptsetup/wiki/DMCrypt

Un altre tema relacionat amb l'arrencada és la possibilitat que algú que tingui accés al teclat reiniciï el sistema, ja que si es prem `CTRL + ALT + DEL` (en diverses distribucions aquesta opció ja sol estar desactivada per defecte), es provoca una operació de tancament en la màquina. Aquest comportament és definit (en un sistema amb sysvinit) a `/etc/inittab`, amb una línia com ara:

```
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
```

Si es posa com a comentari, aquesta possibilitat de reiniciar quedarà desactivada. O, al contrari, pot crear-se un fitxer `/etc/shutdown.allow`, que permet que usuaris determinats puguin apagar i/o reiniciar.

3.2. Contrasenyes i *shadows*

Les contrasenyes típiques dels sistema UNIX inicials (i de les primeres versions de GNU/Linux) estaven xifrades mitjançant uns algorismes DES (però amb claus petites, i una crida de sistema s'encarregava de xifrar i desxifrar, en concret `crypt`, vegeu-ne el man).

Normalment, es trobaven al fitxer `/etc/passwd`, en el segon camp; així per exemple:

```
user:sndb565sadsd:...
```

Però el problema resideix en el fet que aquest fitxer era llegible per qualsevol usuari del sistema, de manera que un atacant podia obtenir el fitxer i utilitzar un atac de força bruta, fins a desxifrar les contrasenyes contingudes en el fitxer, o bé, amb atacs de força bruta una mica més intel·ligents, per mitjà de diccionaris.

El primer pas és utilitzar els fitxers `/etc/shadow`, on es desa ara la contrasenya. Aquest fitxer només és llegible per l'usuari *root*. En aquest cas, a `/etc/passwd` apareix un asterisc (*****) on abans era la contrasenya xifrada. Per defecte, les distribucions de GNU/Linux actuals usen contrasenyes de tipus *shadow*, tret que se'ls digui que no les usin.

Un segon pas és canviar el sistema de xifratge de les contrasenyes per un de més complex i difícil de trencar. Ara, tant Fedora com Debian ofereixen contrasenyes per a md5; normalment ens deixen escollir el sistema en temps d'instal·lació. Cal anar amb compte amb les contrasenyes md5, ja que si fem servir NIS podríem tenir algun problema, segons les versions; si no, tots els clients i servidors usaran md5 per a les contrasenyes. Les contrasenyes es poden reconèixer a `/etc/shadow` perquè vénen amb un prefix "`$id$`" amb `id=1`. No són els únics algorismes de xifratge que poden usar-se, `id=5` és sha-256 o `id=6` és sha-512. De fet, per a utilitzar contrasenyes xifrades més fiables, es pot fer servir (en algunes distribucions) l'ordre `mkpasswd`, a la qual pot aportar-se el xifratge utilitzat i la contrasenya que s'ha de xifrar (vegeu els man de `mkpasswd` i `crypt`).

Altres possibles actuacions consisteixen a obligar els usuaris a canviar la contrasenya amb freqüència (l'ordre `change` pot ser útil), imposar restriccions en la mida i el contingut de les contrasenyes i validar-les amb diccionaris de termes comuns (definint aquestes polítiques per una combinació de paràmetres en `/etc/passwd` o mitjançant control per mòduls PAM).

Pel que fa a les eines, és interessant disposar d'un trencador de contrasenyes (és a dir, un programa per a provar i trencar contrasenyes) per a comprovar la situació real de seguretat dels comptes dels nostres usuaris i forçar així el canvi en les que detectem insegures. Dues de les més utilitzades pels administradors són `John the Ripper (paquet john)` i `crack`. Poden funcionar també per diccionari, de manera que és interessant disposar d'algun ASCII d'espanyol o català (poden trobar-se a la Xarxa).

Una qüestió que s'ha de tenir en compte sempre és fer aquestes proves sobre els nostres sistemes. No s'ha d'oblidar que els administradors d'altres sistemes (o el proveïdor d'accés o ISP) tindran habilitats sistemes de detecció d'intrusos i podem ser objecte de denúncia per intents d'intrusió davant de les autoritats competents (unitats de delictes informàtics) o en el nostre ISP perquè se'ns tanqui l'accés. Cal anar amb molt de compte amb l'ús d'eines de seguretat, que són sempre a la frontera entre ser de seguretat o d'intrusió.

3.3. Bits *sticky* i *setuid*

Un altre problema important són alguns permisos especials que són utilitzats sobre fitxers o *scripts*.

El bit *sticky* s'utilitza sobretot en directoris temporals, on volem que, en alguns grups (de vegades no relacionats), qualsevol usuari pugui escriure, però només pugui esborrar el propietari del directori o bé el propietari del fitxer que estigui en el directori. Un exemple clàssic d'aquest bit és el directori temporal `/tmp`. Cal vigilar que no hi hagi directoris d'aquest tipus, ja que poden permetre que qualsevol hi escrigui, per la qual cosa caldrà comprovar que no n'hi hagi més que els estrictament necessaris com a temporals. El bit es col·loca mitjançant (`chmod +t dir`) i pot treure's amb `-t`. En un `ls` apareixerà com un directori amb permisos `drwxrwxrwt` (observeu l'última `t`).

El bit *setuid* permet a un usuari executar (o bé un programa binari executable o bé un *script* de l'intèrpret d'ordres) amb els permisos d'un altre usuari. Això en algun cas pot ser d'utilitat, però és potencialment perillós. És el cas, per exemple, de programes amb *setuid* de *root*: un usuari, malgrat que no tingui permisos de *root*, pot executar un programa amb *setuid* que pot disposar de permisos interns d'usuari *root*. Això és molt perillós en cas de *scripts*, ja que podrien editar-se i modificar-se per a fer qualsevol tasca. Per tant, cal tenir controlats aquests programes i, en cas que no es necessiti el *setuid*, eliminar-lo. El bit es col·loca mitjançant `chmod +s`, aplicant-lo o bé al propietari (llavors s'anomena *suid*) o bé al grup (s'anomena bit *sgid*); pot treure's amb `-s`. En el cas de visualitzar amb `ls`, el fitxer apareixerà amb `-rwSrw-rw` (observeu la `S`) si és només *suid*, en *sgid* la `S` apareixeria després de la segona `w`.

En cas d'utilitzar `chmod` amb notació octal, es fan servir quatre xifres, en què les tres últimes són els permisos clàssics `rwxrwxrwx` (recordeu que cal sumar en la xifra 4 per a `r`, 2 `w` i 1 per a `x`) i la primera té un valor per a cada permís especial que es vulgui (que se sumen): 4 (per a *suid*), 2 (*sgid*) i 1 (per a *sticky*).

Fóra desitjable fer una anàlisi periòdica del sistema de fitxers per a detectar els fitxers amb *suid* i *sgid* en el sistema i determinar si són realment necessaris o poden provocar problemes de seguretat.

3.4. Habilitació de *hosts*

En el sistema hi ha uns quants fitxers de configuració especials que permeten habilitar l'accés a una sèrie d'amfitrions (*hosts*) per a alguns serveis de xarxa, però els errors dels quals poden permetre atacar després la seguretat local. Ens podem trobar amb:

- `.rhosts` d'usuari: permet que un usuari pugui especificar una sèrie de màquines (i usuaris) que poden usar el seu compte mitjançant ordres "r" (rsh, rcp, etc.) sense necessitat d'introduir la contrasenya del compte. Això és

potencialment perillós, ja que una mala configuració de l'usuari podria permetre entrar a usuaris no desitjats, o que un atacant (amb accés al compte de l'usuari) canviés les adreces a `.rhosts` per poder entrar còmodament sense cap control. Normalment, no s'hauria de permetre crear aquests arxius i, fins i tot, caldria esborrar-los completament i deshabilitar les ordres "r". Una anàlisi periòdica del sistema serà útil per a detectar la creació d'aquests fitxers.

- `/etc/hosts.equiv`: és exactament el mateix que els fitxers `.rhosts` però a nivell de màquina, i especifica quins serveis, quins usuaris i quins grups poden accedir sense control de contrasenya als serveis "r". A més, un error com posar en una línia d'aquest fitxer un "+" permet l'accés a "qualsevol" màquina. Normalment, avui en dia tampoc no sol existir per defecte aquest fitxer creat, i sempre existeixen com a alternativa als "r" els serveis de tipus ssh.

- `/etc/hosts.lpd`: en el sistema d'impressió LPD s'utilitzava per a habilitar les màquines que podien accedir al sistema d'impressió. Cal tenir especial cura, si no estem servint impressió, de deshabilitar completament l'accés al sistema, i en cas que ho estiguem fent, restringir al màxim les màquines que realment en fan ús. També es pot intentar canviar a un sistema d'impressió CUPS, per exemple, que té molt més control sobre els serveis. El sistema d'impressió LPD havia estat un blanc habitual d'atacs de tipus cuc o de desbordament, i estan documentats diversos errors importants. Cal estar atents si encara utilitzem aquest sistema i el fitxer `hosts.lpd`.

3.5. Mòduls PAM

Els mòduls PAM [Pen, Mor03] són un mètode que permet a l'administrador controlar com es fa el procés d'autenticació dels usuaris per a determinades aplicacions. Les aplicacions han d'haver estat creades i enllaçades a les biblioteques PAM. Bàsicament, els mòduls PAM són un conjunt de biblioteques compartides que poden incorporar-se a les aplicacions com a mètode per a controlar l'autenticació dels usuaris. A més a més, es pot canviar el mètode d'autenticació (mitjançant la configuració dels mòduls PAM) sense que calgui canviar l'aplicació.

Enllaç d'interès

Si voleu saber més coses sobre els mòduls PAM, podeu consultar la *The Linux-PAM System Administrators Guide* a http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html.

Els mòduls PAM (les biblioteques) acostumen a estar a `/lib/security` o `/lib64/security` (en forma de fitxers objecte carregables dinàmicament). La configuració de PAM és en el directori `/etc/pam.d`, on apareix un fitxer de configuració de PAM per a cada aplicació que està usant mòduls PAM. Ens trobem amb la configuració d'autenticació d'aplicacions i serveis com ssh, d'inici de sessió de l'entorn gràfic d'X Window System, com *xdm, gdm, kdm, xscreensaver*, etc. o, per exemple, de l'inici de sessió del sistema (l'entrada per a identificador d'usuari i contrasenya). En les antigues versions de PAM, s'utilitzava un arxiu de configuració general (típicament a `/etc/pam.conf`), que era on es llegia la configuració PAM si el directori `/etc/pam.d` no existia.

La línia típica d'aquests fitxers (a `/etc/pam.d`) tindria aquest format (si s'utilitzava l'antic `/etc/pam.conf` caldria afegir el servei al qual pertany com a primer camp):

```
module-type control-flag module-path arguments
```

on s'especifica:

1) Tipus de mòdul: si és un mòdul que requereix que l'usuari s'autentifiqui (`auth`) o és d'accés restringit (`account`); coses que cal fer quan l'usuari entra o surt (`session`), o s'ha d'actualitzar la contrasenya (`password`).

2) *Flags* de control: especifiquen si és necessari (`required`), si és un requisit previ (`requisite`), si és suficient (`sufficient`) o si és opcional (`optional`). Aquesta és una de les possibles sintaxis, però per a aquest camp existeix una alternativa més actual que treballa en parelles valor i acció.

3) El camí (*path*) del mòdul.

4) Arguments que es passen al mòdul (depenen de cada mòdul).

A causa del fet que alguns serveis necessiten diverses línies de configuració comunes, hi ha la possibilitat d'operacions d'inclusió de definicions comunes d'altres serveis. Per a això només cal afegir una línia amb:

```
@include servei
```

Un petit exemple de l'ús de mòduls PAM (en una distribució Debian), pot ser el seu ús en el procés d'inici de sessió (*login*) (també hi ha la llista de les línies incloses provinents d'altres serveis):

```
auth      requisite pam_securetty.so
auth      requisite pam_nologin.so
auth      required  pam_env.so
auth      required  pam_unix.so nullok
account   required  pam_unix.so
session   required  pam_unix.so
session   optional  pam_lastlog.so
session   optional  pam_motd.so
session   optional  pam_mail.so standard noenv
password  required  pam_unix.so nullok obscure min = 4 max = 8 md5
```

Això especifica els mòduls PAM necessaris per a controlar l'autenticació d'usuaris en l'inici de sessió. Un dels mòduls `pam_unix.so` és el que realment fa la verificació de la contrasenya de l'usuari (examina com a dades fitxers `passwd`, `shadow`, etc.).

Altres mòduls controlen la sessió per veure quan ha entrat per darrera vegada, o desen quan entra i surt (per a l'ordre `lastlog`). També hi ha un mòdul que

s'encarrega de verificar si l'usuari té correu per llegir (també cal autentificar-se) i un altre que controla que es canviï la contrasenya (si està obligat a fer-ho en el primer inici de sessió que faci) i que tingui de 4 a 8 lletres. Pot utilitzar-se md5 per al xifratge de contrasenyes.

En aquest exemple podríem millorar la seguretat dels usuaris: el `auth` i el `password` permeten contrasenyes de longitud nul·la: és l'argument `nullok` del mòdul. Això permetria tenir usuaris amb contrasenyes buides (font de possibles atacs). Si traiem aquest argument, ja no permetem contrasenyes buides en el procés d'inici de sessió. El mateix pot fer-se en el fitxer de configuració de `passwd` (en aquest cas, l'ordre de canvi del contrasenya), que també presenta el `nullok`. Una altra possible acció és incrementar en ambdós fitxers la mida màxima de les contrasenyes, per exemple, amb `max = valor`.

3.6. Alteracions del sistema

Enllaç d'interès

Teniu més informació sobre AusCert UNIX checklist a http://www.auscert.org.au/5816.

Un altre problema pot ser l'alteració d'ordres o configuracions bàsiques del sistema, mitjançant la introducció de troians o portes secretes en el sistema, per la simple introducció de programari que substitueixi o modifiqui lleugerament el comportament del programari de sistema.

Un cas típic és la possibilitat de forçar l'usuari *root* perquè executi ordres falses de sistema; per exemple, si el *root* inclogués el "." en la variable de PATH, això permetria l'execució d'ordres des del seu directori actual, cosa que habilitaria la col·locació d'arxius que substituïssin ordres del sistema i que serien executades en primer terme, abans que les de sistema. El mateix procés pot fer-se amb un usuari, tot i que com que els permisos que té són més limitats, pot no afectar tant el sistema, sinó més aviat la seguretat de l'usuari. Un altre cas típic és el de les pantalles d'inici de sessió falses, que poden substituir el típic procés d'inici de `login`, `passwd`, per un programa fals que emmagatzemi les contrasenyes introduïdes.

Enllaç d'interès

Teniu més informació sobre chkrootkit a http://www.chkrootkit.org.

En cas d'aquestes alteracions, serà imprescindible usar polítiques d'auditoria de canvis del sistema, o bé per mitjà de càlcul de signatures o sumes (gpg o md5), o bé mitjançant algun programari de control com `Tripwire` o `AIDE`. Per als troians podem fer diferents tipus de deteccions, o utilitzar eines com `chkrootkit` o `rkhunter`, si aquestes vinguessin de la instal·lació d'alguna eina d'intrusió coneguda.

3.7. Recursos limitats, *cgroups* i *chroot*

La gestió comuna dels processos existents de la màquina, tant si són perfectament vàlids com nocius (intencionadament o per distracció), ens pot provocar situacions d'esgotament dels recursos, en forma de CPU disponible, memòria disponible, recursos de xarxa o simplement sessions d'usuari concurrents i/o processos executant-se.

Per a controlar aquest problema d'esgotament, podem introduir límits a alguns dels recursos usats mitjançant l'ordre `ulimit`, encara que la configuració global de límits es manté en el fitxer `/etc/security/limits.conf`. De fet, aquests límits, que poden ser imposats sobre temps de CPU, nombre màxim de processos, quantitat de memòria i altres de similars, són llegits per mòduls PAM durant el procés de *login* dels usuaris, i establerts per defecte a partir dels límits imposats a cada usuari.

L'elecció dels límits també es definirà pel perfil de funcionament del sistema: tenim un gran nombre d'usuaris al qual hem de limitar els recursos, algun servidor costós en memòria, o per contra només necessitem uns pocs processos per a mantenir un servidor simple. Si podem perfilar la tipologia del nostre servidor, podrem imposar límits raonables que ens permetin posar límit a situacions problemàtiques; pot succeir fins i tot que aquestes s'estiguin donant per atacs de seguretat orientats al consum de recursos per a deixar inoperants els servidors o el sistema en conjunt. Normalment ens concentrarem a controlar, i limitar adequadament, els recursos emprats pel *root* i pels usuaris comuns per a intentar evitar possibles atacs o comportaments defectuosos d'aplicacions corrompudes en el seu funcionament i l'ús dels recursos.

No establir límits i usar el comportament per defecte pot fer caure el nostre sistema, amb certa facilitat, en processos en mal funcionament (per mala programació o descuits de sintaxi), tant si són executables de sistema com si són *scripts*; per exemple, és bastant fàcil crear el que es denomina una *Fork Bomb*. En sistemes UNIX (Linux inclòs), la crida a sistema *fork()* és la que permet crear un procés derivat (fill) a partir de la imatge d'un procés inicial (procés pare). Posteriorment, el procés fill pot especialitzar el seu codi mitjançant altres crides a sistema i/o comunicar-se amb el procés pare. De fet, aquesta és la informació de parentiu que s'observa entre els processos en *ps* amb els seus PID i PPID (Parent PID). Si aquest tipus de crides entren en bucle infinit (més aviat finit, fins que s'esgoten els recursos), ja sigui per distracció, mala programació o intencionadament, la qual cosa aconseguirem en una màquina sense control dels límits, bàsicament farem caure el sistema directament (o el mantindrem en uns extrems de servei mínims).

En el cas de `/etc/security/limits.conf`, cada línía permet uns camps denominats:

```
domini tipus item valor
```

on el *domini* és el *login* d'un usuari o grup. El tipus serà *soft* o *hard* en funció de si és un límit amb certa laxitud o per contra no es pot sobrepassar. L'ítem és referit al tipus de límit (*fsize, nofile, cpu, nproc, maxlogins*), que es refereixen a: grandària màxima de fitxer, nombre de fitxers oberts, màxima CPU en minuts usada, nombre total de processos de l'usuari i nombre màxim de *logins* de l'usuari, i finalment el *valor* corresponent estable. Normalment el més ra-

onable és restringir als usuaris el nombre màxim de processos (per a limitar problemes com *fork bomb*) i, tret que hi hagi alguna raó explícita, podem deixar sense ús altres límits. En alguns casos hi ha alternatives. Per exemple, en el cas de l'ús de disc, els sistemes de fitxers podrien aportar control de quotes d'espai de disc per usuari.

Si, per exemple, volguéssim limitar a 64 els processos màxims per usuari, l'entrada de `limits.conf` seria:

```
* hard nproc 64
```

Podem comprovar el funcionament sortint i tornant a entrar en el compte d'usuari per a observar la sortida de l'ordre `ulimit -a`.

Els límits per defecte en la majoria de distribucions acostumen a ser bastant folgats. En una Debian, per exemple, el cas per defecte (i el valor anterior proposat, 64, contra el qual exhibeix per defecte la distribució, 16.006 processos màxims per usuari) seria:

```
# ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 16006
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 16006
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

Amb una correcta posada en escena de límits raonables segons el perfil del nostre sistema i dels nostres usuaris, podem mitigar de manera molt important els perills de caiguda dels nostres servidors.

En particular, per a l'establiment de límits, cal assenyalar que també disposem d'una funcionalitat en un nivell de kernel, introduïda en les últimes branques i feta servir majoritàriament pel sistema *Systemd*, denominada *cgroups*, que permet controlar i sol·licitar recursos de sistema de diferents tipus: CPU,

memòria o accés a la xarxa. Podem examinar la llista dels subsistemes suportats amb *cat /proc/cgroups*. S'aporta també una llibreria (*libcgroup*) i un conjunt d'utilitats administratives (*libcgroup-tools*) que poden ser usades per a controlar aquests grups, i monitorar l'ús dels recursos i els límits imposats. *Cgroups* s'utilitza a partir d'un pseudosistema de fitxers, *cgroupfs*, en el qual se situen els recursos en grups de control, als quals es poden associar processos que els estiguin usant i limitar el seu ús. Per exemple, podem esbrinar quins grups trobem i on estan muntats, amb tipus de *filesystem cgroupsfs*, on podem observar els directoris i la informació de control de cada grup de recursos mantinguda. També podem observar la configuració en `/etc/cgconfig.conf` (encara que algunes d'aquestes les durà a terme *Systemd* en temps d'arrencada):

```
# cat /proc/cgroups
#subsys_name hierarchy num_cgroups enabled
cpuset 2 1 1
cpu 3 1 1
cpuacct 3 1 1
memory 4 1 1
devices 5 1 1
freezer 6 1 1
net_cls 7 1 1
blkio 8 1 1
perf_event 9 1 1
net_prio 7 1 1
hugetlb 10 1 1

# lsusbsys -am
cpuset  /sys/fs/cgroup/cpuset
cpu,cpuacct /sys/fs/cgroup/cpu,cpuacct
memory  /sys/fs/cgroup/memory
devices  /sys/fs/cgroup/devices
freezer  /sys/fs/cgroup/freezer
net_cls,net_prio /sys/fs/cgroup/net_cls,net_prio
blkio  /sys/fs/cgroup/blkio
perf_event  /sys/fs/cgroup/perf_event
hugetlb  /sys/fs/cgroup/hugetlb
```

Posem un exemple de control per a una aplicació costosa de recursos que podríem controlar amb *cgroups* especificada en `/etc/cgconfig.conf`. Sigui APP l'executable de l'aplicació i $USER l'usuari que l'executarà:

```
# Prevenir que APP obtingui tota la memòria
#    i CPU disponibles
group APP {
    perm {
```

```
        admin {
            uid = $USER;
        }
        task {
            uid = $USER;
        }
    }
# 4 cores usados
#(suposem un sistema amb més disponibles)
    cpuset {
        cpuset.mems="0";
        cpuset.cpus="0-3";
    }
    memory {
# 4 GiB limit
        memory.limit_in_bytes = 4294967296;
    }
}
```

Aquest *cgroup* permetrà a APP usar els *cores* 0 a 3 (4 en total) i limitar el seu ús de memòria a 4 GB, i la posaríem en execució amb el límit de recursos amb (no s'ha d'oblidar substituir APP i $USER pel nom de l'aplicació i usuari):

cgroups info

Podeu consultar la documentació del kernel a https://www.kernel.org/doc/Documentation/cgroups/

```
$ cgexec -g memory,cpuset:APP /opt/APPdir/APP -opcions
```

Respecte als *cgroups*, recentment, en la versió de kernel 3.16 (agost de 2014), s'ha establert un nou mètode d'accés que permet disposar-los com una única jerarquia.

Enllaç d'interès

Novetats *cgroups* en kernel 3.16: http://lwn.net/Articles/601840/

Examinem una tercera possibilitat per al control de límits de recursos lliurats a una determinada aplicació, servei o usuari. És el que es coneix com a *creació d'ambients chroot*, el concepte dels quals és la creació d'una espècie d'ambient de presó (conegut com a *jail*), en què posem uns límits molt clars a allò que es pot fer sense afectar la resta del sistema.

Es basa en una crida al sistema *chroot()* que bàsicament redefineix el que el procés fill associat percep com a directori arrel (/), que, per a l'ambient (o gàbia) creat, pot associar-se en una determinada posició del sistema de fitxers, percebuda pel procés com a arrel (/), i buscarà a partir d'allà totes les configuracions, components del sistema, biblioteques, etc. des d'aquesta falsa arrel.

Ja que es canvia l'estructura del sistema de fitxers per una de nova, el procés només funcionarà si disposa de totes les seves dades a la zona nova. Per tant, haurem de replicar totes les dades, fitxers i ordres necessàries.

Convé recalcar que un ambient de *chroot* protegeix contra l'accés a fitxers fora de la zona (de gàbia), però no protegeix contra altres límits, com la utilització del sistema, accessos a memòria o d'altres de similars.

Una vegada escollim el procés que volem restringir, podem copiar els seus fitxers de dades en l'estructura adequada de directoris i podem examinar els executables del procés amb *ldd executable*, la qual cosa ens aporta informació de les biblioteques necessàries per a la seva execució, que poden ser copiades i situades en els punts de l'estructura de fitxers adequats.

Un cop disposem de l'estructura simplement amb:

```
chroot /nova/arrel executable
```

li passem l'execució a la seva nova gàbia (dins de l'arrel obtinguda resolent el camí */nova/arrel*).

3.8. Protecció d'executables mitjançant Hardening-Wrapper

Amb aquest nom es coneixen una sèrie de tècniques usades per a mitigar diversos problemes de seguretat relacionats amb atacs duts a terme per executables. En concret, es tracta dels problemes de desbordament de *stack* (pila), *heap* i proteccions contra l'accés a zones de memòria de dades i codi dels executables.

S'utilitzen aquestes tècniques en associació amb el compilador (per exemple, GNU *gcc*), mitjançant una sèrie de paràmetres i opcions (*flags*) passades en temps de compilació, ja sigui per a la compilació d'aplicacions d'usuari, clients de serveis o la part *server* de serveis (*daemons*), que a partir de les seves fonts de codi i el procés de compilació són protegides contra diverses tècniques d'atac als executables.

En Debian podem instal·lar les utilitats i fitxers complementaris per al procés de compilació, a més de disposar de la utilitat *hardening-check*, que ens permet comprovar les proteccions de què disposa un determinada ordre, aplicació o *daemon*. Posem l'exemple de la instal·lació dels paquets en Debian i la comprovació d'un *daemon* (*sshd* en aquest cas):

```
# apt-get install hardening-wrapper hardening-includes

# hardening-check /usr/sbin/sshd
/usr/sbin/sshd:
 Position Independent Executable: yes
 Stack protected: yes
```

```
Fortify Source functions: yes (some protected functions found)
Read-only relocations: yes
Immediate binding: yes
```

Entre les comprovacions que s'efectuen en l'executable (i les relacions amb paràmetres [*flags*] de compilació en GNU gcc, els que s'esmenten poden ser passats per línia d'ordres a gcc, o bé en el corresponent fitxer de Makefile que efectuï el procés de compilació):

- *Position Independent Executable*, que permet al programa que la seva posició en memòria del sistema sigui movible, sense disposar necessàriament de posició fixa i fins i tot de manera aleatòria entre diferents execucions. En gcc aconseguim aquesta funcionalitat amb els *flags* de compilació *-peu -fPIE* aplicats a tots els components de l'executable. El kernel que maneja els processos ha de tenir suport d'allò que es coneix com ASLR (*Address Space Layout Randomization*), que permet protecció contra atacs de desbordament de *buffer* (*buffer overflows*). Per a prevenir que un atacant pugui atacar una determinada funció en memòria, ASLR reordena de manera aleatòria àrees clau del procés (la imatge de l'executable en memòria) per a canviar posicions de la base de la part executable, així com les posicions de l'*stack*, el *heap* o les biblioteques dinàmiques dins de l'espai d'adreces del procés. Així s'aconsegueix que sigui difícil predir les posicions de memòria quan s'intenten atacs amb aquesta fi. Pot esbrinar-se si hi ha suport en el kernel per a ASLR examinant */proc/sys/kernel/randomize_va_space*, que hauria de donar un valor d'1 o 2 per a opcions amb ASLR activades i 0 quan estan desactivades.

- *Stack Protected*. En aquest cas s'han ofert mètodes addicionals per a protegir els desbordaments de *stack*, per exemple, mitjançant la compilació gcc amb l'opció (*-fstack-protector*).

- *Fortify Source functions*. En aquest cas, quan durant el procés de compilació (en la part de *link*) els executables són enllaçats amb les funcions de la biblioteca estàndard C (funcions de *glibc* en un sistema GNU/Linux), hi ha una certa substitució d'algunes funcions per unes altres. Algunes funcions de *glibc* són conegudes per no ser segures en diversos aspectes, encara que tenen alternatives segures en la mateixa biblioteca, ja que tenen tests de seguretat incorporats, per exemple, enfront de *buffer overflows*. En el cas de gcc, en el procés de compilació es fa amb *-D_FORTIFY_SOURCE=2 -O1* (o superior nivell d'optimització).

- *Read-only relocations*. Això permet al *linker*, quan detecta zones de memòria que pot deduir que seran de sol accés de lectura durant el procés de compilació, marcar-les com a tals abans de començar l'execució. D'aquesta manera, es redueix la possibilitat que un atacant pugui usar aquestes àrees per a injectar codi o que aprofiti *exploits* de corrupció de memòria. En compilació, poden passar-se a gcc els *flags -W1,-z,relro* per a aquesta opció.

- *Immediate binding*. És una possible combinació amb l'anterior, que permet al *linker*, prèviament a l'execució, resoldre moviments de dades i certs enllaços a memòria que evitin que aquests moviments (*relocates*) es portin a terme *a posteriori*. Així, en combinació amb l'opció anterior, es redueixen de manera sensible les zones de memòria disponibles contra atacs de corrupció de memòria.

En el cas de gcc, la majoria d'aquestes tècniques poden activar-se per defecte abans de la compilació activant la variable següent (amb valor igual a 1 per a activar o 0 per a desactivar):

```
$ export DEB_BUILD_HARDENING=1
$ gcc ....
```

Aquestes tècniques aporten una bona base per a protegir els executables del sistema contra atacs a la seva forma de procés mentre estan en execució, així com protegir les seves dades i el seu codi de canvis dinàmics que un atacant podria introduir.

Hardening-Wrapper

Es recomana per a més informació consultar les pàgines man de hardening-wrapper i hardening-check. En el cas de Debian, és recomanable: https://wiki.debian.org/hardening.

No obstant això, cal assenyalar alguns defectes, com que aquestes tècniques varien constantment, que certs problemes en la implementació de les tècniques de protecció poden ser explotats, o que en determinats casos aquestes proteccions poden donar falsos positius.

4. SELinux

La seguretat tradicional dins del sistema s'ha basat en les tècniques DAC (*discretionary access control*, control d'accés discrecional), en què normalment cada programa disposa de control complet sobre els accessos als recursos. Si un determinat programa (o l'usuari) decideix fer un accés incorrecte (per exemple, deixant dades confidencials en obert, per desconeixement o per mal funcionament), no hi ha res que li ho impedeixi. Així, en DAC, un usuari té control complet sobre els objectes que li pertanyen i els programes que executa. El programa executat disposarà dels mateixos permisos que l'usuari que l'està executant. Així, la seguretat del sistema dependrà de les aplicacions que s'estiguin executant i de les vulnerabilitats que poguessin tenir, o del programari maliciós que poguessin incloure i afectarà, en especial, els objectes (altres programes, fitxers o recursos) a què l'usuari tingui accés. En el cas de l'usuari *root*, això comprometria la seguretat global del sistema.

D'altra banda, les tècniques MAC (*mandatory access control*, control d'accés obligatori) desenvolupen polítiques de seguretat (definides per l'administrador) en les quals el sistema controla completament els drets d'accés que es concedeixen a cada recurs. Per exemple, amb permisos (de tipus UNIX) podem donar accés a fitxers, però mitjançant polítiques MAC tenim control addicional per a determinar a quins fitxers es permet accedir explícitament a un procés, i quin grau d'accés volem concedir. Es fixen contextos, en els quals s'indiquen en quines situacions un objecte pot accedir a un altre objecte.

SELinux [Nsab] és un component recent de tipus MAC inclòs en la branca 2.6.x del nucli que les distribucions van incloent progressivament: Fedora / Red Hat el duen habilitat per defecte (tot i que és possible canviar-lo durant la instal·lació) i en Debian és un component opcional. SELinux aplica polítiques de seguretat de tipus MAC i permet disposar d'un accés de permisos més fi que els tradicionals permisos d'arxiu UNIX. Per exemple, l'administrador podria permetre que s'afegissin dades a un arxiu de registre, però no reescriure'l o truncar-lo (tècniques utilitzades habitualment per atacants per a esborrar les pistes dels accessos). En un altre exemple, es podria permetre que programes de xarxa s'enllacessin a un port (o ports) que els calgués i, tot i així, denegar l'accés a altres ports (aquesta tècnica podria permetre controlar i limitar l'accés d'alguns troians o portes secretes). SELinux va ser desenvolupat per l'agència NSA dels Estats Units, amb aportacions de diverses companyies per a sistemes UNIX i lliures, com Linux i BSD. Es va alliberar l'any 2000 i des de llavors s'ha anat integrant en diferents distribucions GNU/Linux.

Enllaços d'interès

En els enllaços següents teniu alguns recursos sobre SELinux: http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/selinux-guide, http://www.nsa.gov/research/selinux/index.shtml, http://fedoraproject.org/wiki/SELinux, http://selinux.sourceforge.net/.

Amb SELinux disposem d'un model de domini-tipus, en què cada procés corre en un context de seguretat i qualsevol recurs (fitxer, directori, sòcol, etc.) té un

tipus associat amb ell. Hi ha un conjunt de regles que indiquen quines accions poden efectuar-se en cada context sobre cada tipus. Un avantatge d'aquest model context-tipus és que les polítiques que es defineixin es poden analitzar (hi ha eines) per a determinar quins fluxos d'informació estan permesos; per exemple, per a detectar possibles vies d'atac o si la política és prou completa per a cobrir tots els accessos possibles.

Es disposa del que es coneix com la *base de dades de polítiques de SELinux* (*SELinux policy database*), que controla tots els aspectes de SELinux: determina quins contextos pot utilitzar cada programa per a executar-se i especifica a quins tipus de cada context pot accedir.

En SELinux cada procés del sistema té un context compost de tres parts: una identitat, un rol i un domini. La identitat és el nom del compte de l'usuari, o bé `system_u` per a processos de sistema o `user_u` si l'usuari no disposa de polítiques definides. El rol determina quins contextos estan associats. Per exemple, `user_r` no té permès tenir el context `sysadm_t` (domini principal per a l'administrador de sistema). Així, un `user_r` amb identitat `user_u` no pot obtenir de cap manera un context `sysadm_t`. Un context de seguretat s'especifica sempre amb aquests tres valors, com per exemple:

```
root:sysadm_r:sysadm_t
```

que és el context per a l'administrador del sistema, en defineix la identitat, el rol i el context de seguretat.

Per exemple, en una màquina amb SELinux activat (una Fedora en aquest cas) podem veure amb l'opció `-Z` del `ps` els contextos associats als processos :

```
# ps -ax -Z

LABEL                           PID   TTY   STAT   TIME   COMMAND
system_u:system_r:init_t        1     ?     Ss     0:00   init
system_u:system_r:kernel_t      2     ?     S      0:00   [migration/0]
system_u:system_r:kernel_t      3     ?     S      0:00   [ksoftirqd/0]
system_u:system_r:kernel_t      4     ?     S      0:00   [watchdog/0]
system_u:system_r:kernel_t      5     ?     S      0:00   [migration/1]
system_u:system_r:kernel_t      6     ?     SN     0:00   [ksoftirqd/1]
system_u:system_r:kernel_t      7     ?     S      0:00   [watchdog/1]
system_u:system_r:syslogd_t     2564  ?     Ss     0:00   syslogd -m 0
system_u:system_r:klogd_t       2567  ?     Ss     0:00   klogd -x
system_u:system_r:irqbalance_t  2579  ?     Ss     0:00   irqbalance
system_u:system_r:portmap_t     2608  ?     Ss     0:00   portmap
system_u:system_r:rpcd_t        2629  ?     Ss     0:00   rpc.statd
user_u:system_r:unconfined_t    4812  ?     Ss     0:00   /usr/libexec/gconfd-2 5
user_u:system_r:unconfined_t    4858  ?     Sl     0:00   gnome-terminal
```

```
user_u:system_r:unconfined_t   4861   ?      S    0:00   gnome-pty-helper
user_u:system_r:unconfined_t   4862   pts/0  Ss   0:00   bash
user_u:system_r:unconfined_t   4920   pts/0  S    0:01   gedit
system_u:system_r:rpcd_t       4984   ?      Ss   0:00   rpc.idmapd
system_u:system_r:gpm_t        5029   ?      Ss   0:00   gpm -m /dev/input/mice -t exps2
user_u:system_r:unconfined_t   5184   pts/0  R+   0:00   ps ax -Z
user_u:system_r:unconfined_t   5185   pts/0  D+   0:00   Bash
```

I amb `ls` amb l'opció `-Z` podem veure els contextos associats a fitxers i directoris:

```
# ls -Z
drwxr-xr-x josep josep user_u:object_r:user_home_t Desktop
drwxrwxr-x josep josep user_u:object_r:user_home_t proves
-rw-r--r-- josep josep user_u:object_r:user_home_t yum.conf
```

I des de la consola podem conèixer el nostre context actual amb:

```
$ id -Z
user_u:system_r:unconfined_t
```

Pel que fa al mode de funcionament, SELinux en presenta dos, anomenats *permissive* i *enforcing*. En *permissive* es permeten els accessos no autoritzats, però són auditats en els registres corresponents (normalment, directament sobre `/var/log/messages` o bé, depenent de la distribució, amb l'ús de `audit` en `/var/log/audit/audit.log`). En *enforcing* no es permet cap tipus d'accés que no permetin les polítiques definides. També pot desactivar-se SELinux per mitjà del fitxer de configuració (habitualment a `/etc/selinux/config`), col·locant `SELINUX=disabled`.

Cal anar amb compte amb l'activació i desactivació de SELinux, especialment amb l'etiquetatge de contextos en els fitxers, ja que en períodes en què s'activi o desactivi es poden perdre etiquetes (perquè el sistema no estarà actiu) o simplement poden no ser etiquetats. Així mateix, la realització de còpies de seguretat del sistema de fitxers ha de tenir en compte que cal conservar les etiquetes de SELinux.

Un altre possible problema que cal tenir en compte és el gran nombre de regles de política de seguretat que poden arribar a existir i que poden provocar limitacions en el control dels serveis. Davant d'un tipus determinat de mal funcionament, en primer lloc cal determinar que no sigui precisament SELinux allò que està impedint el funcionament, per una limitació massa estricta de seguretat (vegeu el subapartat 4.2. de crítica a SELinux) o a causa d'opcions que no esperàvem tenir activades (poden requerir canviar la configuració dels booleans, com veurem).

Pel que fa a la política usada, SELinux suporta dos tipus diferents: *targeted* i *strict*. En *targeted* la majoria de processos operen sense restriccions, i només serveis específics (alguns dimonis –*daemons*) es posen en diferents contextos de seguretat que són confinats a la política de seguretat. En *strict* tots els processos són assignats a contextos de seguretat i confinats a polítiques definides, de manera que qualsevol acció és controlada per les polítiques definides. En principi aquests són els dos tipus de polítiques definits generalment, però l'especificació resta oberta a incloure'n més.

Un cas especial de política és la MLS (*multilevel security*, seguretat multinivell), que és una política multinivell de tipus *strict*. La idea és definir, dins de la mateixa política, diferents nivells de seguretat, i els contextos de seguretat tenen associats un camp addicional de nivell d'accés. Aquest tipus de política de seguretat (com MLS) sol ser utilitzat en organitzacions governamentals i militars, on hi ha organitzacions jeràrquiques amb diferents nivells d'informació privilegiada, nivells d'accés general i capacitats d'acció diferents en cada nivell. Per a obtenir algunes certificacions de seguretat (concedides o avalades per certes organitzacions) per als sistemes operatius, cal disposar de polítiques de seguretat d'aquest tipus.

Es pot definir quin tipus de política s'usarà a `/etc/selinux/config`, variable `SELINUXTYPE`. La política corresponent, i la seva configuració, normalment estarà instal·lada en els directoris `/etc/selinux/SELINUXTYPE/`. Per exemple, sol trobar-se en el subdirectori `policy` el fitxer binari de la política compilada (que és el que es carrega en el nucli en inicialitzar SELinux).

4.1. Arquitectura

L'arquitectura de SELinux està formada pels components següents:

- Codi en l'àmbit del nucli
- La biblioteca compartida de SELinux
- La política de seguretat (la base de dades)
- Eines

Examinem algunes consideracions sobre cada component:

- El codi de nucli monitora l'activitat del sistema, i assegura que les operacions sol·licitades estiguin autoritzades sota la configuració de polítiques de seguretat del SELinux actual; així, no permet les operacions no autoritzades i normalment genera entrades en el registre de les operacions denegades. El codi actualment es troba integrat en els nuclis 2.6.x, mentre que en els anteriors s'ofereix com a sèrie de pedaços.

- Gairebé la majoria d'utilitats i components SELinux no directament relacionats amb el nucli fan ús de la biblioteca compartida (que s'anomena `libselinux1.so`), que facilita una API per a interaccionar amb SELinux.

- La política de seguretat és la que està integrada en la base de dades de regles de SELinux. Quan el sistema arrenca (amb SELinux activat), carrega el fitxer binari de política, que habitualment resideix en la ruta següent: `/etc/security/selinux` (tot i que pot variar segons la distribució).

  El fitxer binari de polítiques es crea a partir d'una compilació (via `make`) dels fitxers font de polítiques i alguns fitxers de configuració.

  Algunes distribucions (com per exemple Fedora) no instal·len les fonts per defecte, que solen trobar-se a `/etc/security/selinux/src/policy` o a `/etc/selinux`. Normalment, aquestes fonts consisteixen en diversos grups d'informació:

– Els fitxers relacionats amb la compilació, `makefile` i *scripts* associats.

– Els fitxers de la configuració inicial, usuaris i rols associats.

– Els fitxers de `Type-enforcement`, que contenen la majoria de les sentències del llenguatge de polítiques associat a un context particular. Cal tenir en compte que aquests fitxers són enormes, típicament de desenes de milers de línies, amb la qual cosa pot aparèixer el problema de trobar fallades o definir canvis en les polítiques.

– Els fitxers que serveixen per a etiquetar els contextos dels fitxers i directoris durant la càrrega o en determinats moments.

- Les eines: inclouen ordres usades per a administrar i utilitzar SELinux, versions modificades d'ordres estàndard Linux i eines per a l'anàlisi de polítiques i el desenvolupament.

Vegem, d'aquest últim punt (les eines típiques de què solem disposar), algunes de les ordres principals:

Nom	Utilització
`chcon`	Etiqueta un fitxer específic o un conjunt de fitxers amb un context específic.
`checkpolicy`	Fa diverses accions relacionades amb les polítiques, incloent-hi la compilació de les polítiques a binari; típicament es crida des de les operacions de `makefile`.
`getenforce`	Genera missatges amb el mode actual de SELinux (*permissive* o *enforcing*). O bé desactivat, si és el cas.
`getsebool`	Obté la llista de booleans, és a dir, la llista d'opcions *on/off* per a cada context associat a un servei o opció general del sistema.
`newrole`	Permet a un usuari la transició d'un rol a un altre.
`runn_init`	S'utilitza per a activar un servei (*start, stop*), garantint que es faci en el mateix context que quan s'arrenca automàticament (amb `init`).
`setenforce`	Canvia de mode a SELinux: `0` *permissive*, `1` *enforcing*.
`setfiles`	Etiqueta directoris i subdirectoris amb els contextos adequats; s'utilitza habitualment en la configuració inicial de SELinux.
`setstatus`	Obté l'estat del sistema amb SELinux.

D'altra banda, es modifiquen algunes ordres GNU/Linux amb funcionalitats o opcions noves, com ara `cp`, `mv`, `install`, `ls`, `ps`, etc., per exemple modificant en els fitxers l'etiqueta per a associar el context de seguretat (com vam poder comprovar amb l'opció `-Z` de `ls`). `Id` es modifica per a incloure l'opció de mostrar el context actual de l'usuari. I en `ps` vam veure que incloïa una opció per a visualitzar els contextos de seguretat actuals dels processos.

A més a més, es modifiquen alguns altres programes comuns per a donar suport a SELinux com:

- cron: modificat per a incloure els contextos per a les tasques en execució per cron.
- login: modificat perquè col·loqui el context de seguretat inicial per a l'usuari quan entra en el sistema.
- logrotate: modificat per a preservar el context dels *logs* quan estiguin recopilats.
- pam: modificat per a col·locar el context inicial de l'usuari i per a usar l'API SELinux i obtenir accés privilegiat a la informació de contrasenyes.
- ssh: modificat per a col·locar el context inicial de l'usuari quan entra en el sistema.
- Diversos programes addicionals que modifiquen `/etc/passwd` o també `/etc/shadow`.

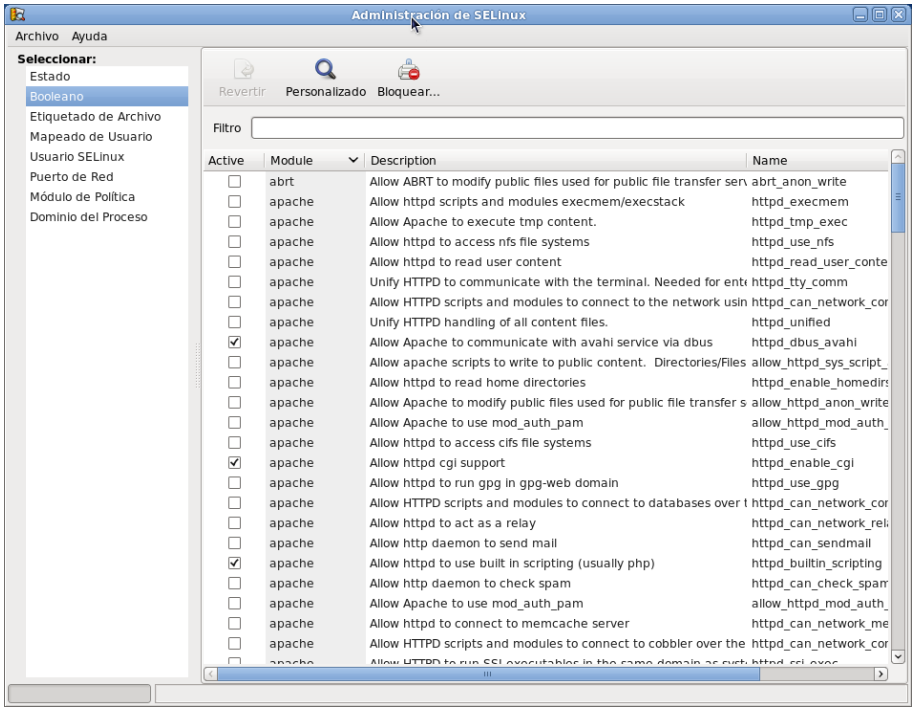
En algunes distribucions també s'inclouen eines per a la gestió de SELinux, com les `setools(-gui)`, que duen diverses eines de gestió i anàlisi de les polítiques. Així, com a eina específica per a controlar els contextos associats als diferents serveis suportats per SELinux en la distribució, en Fedora l'eina `system-config-securitylevel` disposa d'un apartat per a la configuració de SELinux, com podem observar en la figura 1.

En la figura s'observa la configuració de booleans per a diversos serveis i opcions genèriques, entre les quals el servidor web. També podem obtenir aquesta llista amb `getsebool -a`, i amb les ordres `setsebool`/`togglesebool` podem activar/desactivar les opcions.

En Fedora, per exemple, trobem suport de booleans, entre d'altres, per a cron, ftp, httpd (Apache), dns, grub, lilo, nfs, nis, cups, pam, ppd, samba, proteccions contra accessos incorrectes a la memòria dels processos, etc.

La configuració de booleans permet la personalització de la política SELinux en temps d'execució. Els booleans són utilitzats en SELinux com a valors condicionals de les regles de la política usada, fet que permet modificacions de la política sense necessitat de carregar una nova política.

Figura 1. Interfície en Fedora per a configuració SELinux (examinant booleans)



4.2. Crítica

Alguns administradors i experts en seguretat han criticat SELinux específicament, perquè és massa complex de configurar i administrar. S'argumenta que, per la complexitat intrínseca que té, fins i tot usuaris experimentats poden cometre errors, cosa que deixaria la configuració de SELinux no segura o inservible i el sistema, vulnerable. Això és discutible fins a cert punt, ja que tot i que tinguéssim SELinux mal configurat, encara seguirien actius els permisos UNIX i altres eines, i SELinux no permetria una operació que els permisos originals ja no permetessin. De fet, podem veure'l com una etapa addicional de seguretat més estricta.

També hi ha factors de rendiment que es poden veure afectats, a causa de l'enorme grandària de les polítiques; poden disminuir les prestacions com a conseqüència del gran ús de memòria i del temps inicial de càrrega i, en algun cas, del processament de les regles. Cal pensar que estem parlant pràcticament d'un sistema de més de 10.000 regles en la política. I encara pot ser un nombre més gran si escollim una política de tipus *strict*, amb la necessitat d'especificar absolutament totes les opcions que s'han de controlar. Normalment, el processament per política en format binari i l'ús de booleans per a inhabilitar regles permet un ús més eficient del sistema.

Un altre aspecte que sol molestar els administradors és el problema addicional de determinar, davant d'un mal funcionament específic, quin n'és l'origen o causa inicial. És habitual que finalment ens trobem que el problema provenia d'una configuració excessivament restrictiva (potser per desconeixement de l'administrador) de SELinux per a un determinat servei. En darrera instàn-

cia, cal assenyalar l'ampli suport per a la seguretat que proporciona SELinux i que, com a administradors, hem de ser conscients de les capacitats i perills de qualsevol nova tècnica que utilitzem.

4.3. Algunes alternatives

Davant la complexitat de SELinux i algunes deficiències del model (complexitat de polítiques o mala configuració), s'han produït algunes alternatives (o en alguns casos complements) a la utilització de SELinux. Hem de destacar en particular Grsecurity, una sèrie de pedaços per al nucli Linux que intenten millorar algunes deficiències. També cal destacar AppArmor, que algunes distribucions estan incloent com a alternativa a SELinux.

Grsecurity

http://grsecurity.net

Grsecurity, per contra, s'ofereix com una sèrie de pedaços de nucli publicats posteriorment a l'establiment d'una versió estable del nucli, i es poden obtenir de la seva pàgina web per a certes versions de nucli, a més de paquets ja preparats per a algunes distribucions.

Grsecurity inclou diferents prestacions, com un model d'accés a recursos mitjançant gestió de rols (tant locals com remots), gestionats mitjançant una eina denominada *gradm*; auditació de les operacions internes al nucli, i prevenció de codi arbitrari en aquest (de fet fa ús d'un pedaç de nucli denominat PaX, que ofereix algunes d'aquestes proteccions). Això evita possibles errors de tipus desbordament de *buffer*, *heap* o *stack* (tècniques aprofitades per *exploits* del kernel). També inclou notificacions d'alertes de seguretat basades en IP de l'usuari. Els pedaços oferts per al nucli s'apliquen com a pedaços al codi font del nucli i permeten activar, durant la configuració del nucli prèvia a la seva compilació, diferents opcions de protecció del codi del nucli relacionades amb les prestacions esmentades. A més, una vegada el nucli està configurat per al seu suport, l'eina *gradm* permet gestionar els rols i les tasques o proteccions associades a cada rol i als usuaris que incloguem en cadascun.

Enllaç d'interès

Sobre l'administració de Grsecurity pot consultar-se http://en.wikibooks.org/wiki/Grsecurity.

Grsecurity és un bon complement o substitut d'alguns punts febles de SELinux, ja que ofereix una interfície més simple d'usar i inclou algunes prestacions de seguretat addicionals.

AppArmor en Ubuntu

Una alternativa a SELinux utilitzada en Ubuntu és AppArmor: https://help.ubuntu.com/14.04/serverguide/apparmor.html.

Una altra alternativa, que està sorgint en algunes distribucions com Ubuntu i SuSe, és AppArmor, que permet a l'administrador associar a cada programa un perfil de seguretat que restringeixi les capacitats del programa. Es pot establir un perfil individual de permisos o bé es pot fer servir un mode d'autoaprenentatge basat en violacions de perfil registrades i habilitades pels usuaris o no.

Bàsicament, es compon d'un mòdul de nucli per a la gestió de perfils d'aplicacions, basat, igual que SELinux, en LSM (*Linux Security Modulis*), juntament amb diferents utilitats. Mitjançant aquestes utilitats podem establir permisos

dels programes per a l'accés al sistema de fitxers i als dispositius. La major diferència amb SELinux és que les polítiques de seguretat no s'apliquen amb etiquetes als fitxers, sinó respecte a polítiques aplicables als *pathnames*. AppArmor s'integra en el nucli com a mòdul LSM disponible a partir de les versions 2.6.36 del nucli.

Sobre mòduls LSM en el kernel

Podeu llegir comentaris dels diferents models d'LSM en Overview of LSM.

De fet, en les branques actuals del kernel es troba el *framework* denominat LSM (*Linux Security Modules*), que suporta una varietat de mòduls de seguretat per a evitar el favoritisme envers algun dels esmentats. Ara mateix, en LSM, per a les implementacions de seguretat de tipus MAC (*mandatory access control*), estan acceptats com a part del kernel, des de les branques 2.6, els models d'AppArmor, SELinux, Smack i TOMOYO (a més de Yama, que podria incorporar-s'hi en el futur).

5. Seguretat en xarxa

5.1. Client de serveis

Com a clients de serveis, bàsicament ens hem d'assegurar que no posem en perill els nostres usuaris (o que es posin en perill ells tots sols) pel fet d'usar serveis insegurs. Cal evitar l'ús de serveis que no utilitzin xifratge de dades i contrasenyes (FTP, Telnet, correu no segur) i utilitzar en aquest cas tècniques de connexió xifrada, com SSH i SSL/TSL. Així mateix, hem d'assegurar-nos que trobem en el sistema els programes client, i les biblioteques que aquests fan servir (com les esmentades prèviament) en les versions més actualitzades. En alguns moments crítics, davant de possibles vulnerabilitats de tipus *0day*, pot ser fins i tot necessari actualitzar serveis, clients o biblioteques, independentment de la distribució, per a assegurar-nos que el nostre sistema no és vulnerable, mentre no es disposi de solució a les vulnerabilitats detectades.

Clients i serveis insegurs

Com a clients de serveis, haurem d'evitar l'ús de serveis insegurs.

Cas HeartBleed

Un cas de vulnerabilitat crítica que va afectar les biblioteques openSSL durant el 2014: http://heartbleed.com/

Un altre punt important que s'ha de tenir en compte es refereix a la possible substitució de servidors per altres de falsos, o bé a tècniques de segrest de sessions. En aquests casos hem de disposar de mecanismes d'autenticació robustos que ens permetin verificar l'autenticitat dels servidors (per exemple, SSH i SSL/TSL tenen alguns d'aquests mecanismes). També hauríem de verificar la xarxa a la recerca d'intrusos, perills potencials, intents d'intrusió fallits o intents de substitució de servidors, a més de fer servir polítiques correctes de filtratge de paquets mitjançant tallafocs (*firewalls*) que ens permetin donar sortida als nostres paquets de dades vàlides i usar els servidors adequats, alhora que controlem els paquets d'entrada que rebem com a resposta.

5.2. Servidor: inetd i xinetd

La configuració dels serveis de xarxa [Mou02], tal com hem vist, es pot fer des de diversos llocs [Ray01, Hat08, Pen]:

- A `/etc/inetd.conf` o el directori equivalent `/etc/xinetd.d`: aquests fitxers de configuració de serveis estan associats al que es coneix com a *superservidors*, ja que controlen diversos serveis fills i les seves condicions d'arrencada. El servei inetd és utilitzat en Debian (tot i que cada vegada menys), i xinetd, en Fedora (en Debian, xinetd es pot instal·lar opcionalment en substitució d'inetd).

- Servidors iniciats en arrencada: segons el nivell d'execució (*runlevel*) tindrem una sèrie de serveis arrencats. L'arrencada s'originarà en el directori

Nota Systemd

En noves distribucions, el control de serveis s'està migrant, majoritàriament, des d'inetd i xinetd al concepte de *targets* basats en Systemd. Observeu: http://0pointer.de/blog/projects/security.html

associat al nivell d'execució; per exemple, en Debian el nivell per defecte és el 2, els serveis arrencats ho seran des del directori `/etc/rc2.d`, segurament amb enllaços als *scripts* continguts a `/etc/init.d`, en els quals s'executarà amb els paràmetres `start`, `stop`, `restart`, segons correspongui.

- Altres serveis de tipus RPC: s'usen, per exemple, en NIS i NFS associats a crides remotes entre màquines. Amb l'ordre `rpcinfo -p` es pot examinar quins hi ha.

Altres fitxers de suport (amb informació útil) són els següents: `/etc/services`, que està format per una llista de serveis locals o de xarxa coneguts, juntament amb el nom del protocol (tcp, udp o altres) que es fa servir en el servei i el port que utilitza; `/etc/protocols`, que és una llista de protocols coneguts, i `/etc/rpc`, que és una llista de possibles servidors RPC, juntament amb els ports (rpc) usats. Aquests fitxers vénen amb la distribució i són una mena de base de dades que utilitzen les ordres i les eines de xarxa per a determinar els noms de serveis, protocols o rpc i els seus ports associats. Cal destacar que són fitxers més o menys històrics que no han de contenir obligatòriament totes les definicions de protocols i serveis. També es poden buscar diferents llistes de ports coneguts a Internet.

Una de les primeres accions que haurà de fer l'administrador serà deshabilitar tots els serveis que no estigui utilitzant o no tingui previst utilitzar; en aquest sentit caldrà documentar-se sobre l'ús dels serveis [Mou02] i quin programari pot necessitar-los [Neu].

En el cas de `/etc/inetd.conf`, n'hi ha prou de comentar la línia del servei que cal deshabilitar, col·locant-hi un "`#`" com a primer caràcter de la línia.

En l'altre model de serveis, utilitzat per defecte en Fedora (i opcional en Debian), el xinetd, la configuració es troba en el fitxer `/etc/xinetd.conf`, on es configuren alguns valors per defecte de control de registres i, després, la configuració de cada servei fill es fa mitjançant un fitxer dins del directori `/etc/xinetd.d`. En cada fitxer es defineix la informació del servei, equivalent a la que apareix a `inetd.conf`; en aquest cas, per a desactivar un servei solament cal posar una línia `disable = yes` dins del fitxer del servei. xinetd té una configuració més flexible que inetd, ja que separa la configuració dels diferents serveis en diferents arxius, i té força opcions per a limitar les connexions a un servei, en nombre o en capacitat. Tot plegat permet un control més eficient del servei i, si el vam configurar adequadament, podrem evitar alguns dels atacs per denegació de servei (DoS o DDoS).

Quant al tractament dels serveis dels nivells d'execució des d'ordres de la distribució, ja hem esmentat diverses eines en la unitat d'administració local que permeten habilitar o deshabilitar serveis. També hi ha eines gràfiques com `ksysv` de KDE o el `system-config-services` i `ntsysv` en Fedora (en Debian són recomanables `sysv-rc-conf`, `rcconf` o `bum`). I a una escala inferior, podem anar al nivell d'execució que vulguem (`/etc/rcx.d`) i desactivar els

serveis que ens interessi canviant les `S` o `K` inicials de l'*script* per un altre text: un mètode seria, per exemple, canviar `S20ssh` per `STOPS20ssh` i ja no arrencarà; la propera vegada, quan tornem a necessitar-lo, esborrarem el prefix i el tornarem a tenir actiu. Malgrat que aquest mètode és possible, és més recomanable utilitzar una sèrie d'eines simples que proporcionen les distribucions per a col·locar, treure o activar un servei determinat. Per exemple, ordres com `service` i `chkconfig` en Fedora, o similars en Debian, com `update-rc.d` i `invoke-rc.d`.

Un altre aspecte és el tancament de serveis no segurs. Tradicionalment, en el món UNIX s'havien utilitzat serveis de transferència d'arxius com FTP, de connexió remota com Telnet i ordres d'execució (d'inici de sessió o de còpia) remotes, moltes de les quals començaven amb la lletra *r* (per exemple, rsh, rcp, rexec, etc.). Altres perills potencials són els serveis finger i rwhod, que permetien obtenir informació des de la xarxa dels usuaris de les màquines; aquí el perill rau en la informació que podia obtenir un atacant i que li podia facilitar molt el treball. Tots aquests serveis no s'haurien de fer servir actualment, a causa dels perills que impliquen. Pel que fa al primer grup:

- FTP, Telnet: en les transmissions per xarxa no xifren les contrasenyes, de manera que qualsevol pot obtenir les contrasenyes de serveis o els comptes associats (per exemple, mitjançant un detector).

- rsh, rexec, rcp: tenen, a més, el problema que, en algunes condicions, ni tan sols necessiten contrasenyes (per exemple, si es fa des de llocs validats en el fitxer `.rhosts`), amb la qual cosa són novament insegurs i deixen grans portes obertes.

L'alternativa és fer servir clients i servidors segurs que suportin el xifratge dels missatges i l'autenticació dels participants. Hi ha alternatives segures en els servidors clàssics, però actualment la solució més freqüent és l'ús del paquet OpenSSH (que pot combinar-se també amb OpenSSL per a entorns web). OpenSSH ofereix solucions basades en les ordres ssh, scp i sftp, que permeten substituir els antics clients i servidors (s'utilitza un dimoni anomenat *sshd*). L'ordre ssh permet les antigues funcionalitats de Telnet, rlogin i rsh, entre d'altres, i scp seria l'equivalent segur de rcp, i sftp, de l'FTP.

Pel que fa a SSH, també cal tenir la precaució d'usar ssh en la versió 2. La primera versió té algunes vulnerabilitats conegudes; cal anar amb compte a instal·lar OpenSSH i, si no necessitem la primera versió, instal·lar només suport per a ssh2 (en el fitxer de configuració `/etc/ssh/sshd_config` vegeu l'opció `Protocol`).

A més a més, la majoria de serveis que deixem actius en les nostres màquines s'haurien de filtrar posteriorment per mitjà de tallafocs, per a garantir que no fossin utilitzats o atacats per persones a les quals no anaven destinats.

6. Eines de seguretat: detecció de vulnerabilitats i intrusions

Detecció d'intrusos

Els sistemes IDS ens permeten la detecció a temps d'intrusos fent servir els nostres recursos o explorant els nostres sistemes en cerca de fallades de seguretat.

Amb els sistemes de detecció d'intrusos [Hat08] (IDS) es vol fer un pas més. Una vegada haguem pogut configurar més o menys correctament la nostra seguretat, el pas següent consistirà en una detecció i prevenció activa de les intrusions.

Els sistemes IDS creen procediments d'escolta i generació d'alertes en detectar situacions sospitoses, és a dir, busquen símptomes de possibles accidents de seguretat.

Hi ha sistemes basats en la informació local que, per exemple, recopilen informació dels registres del sistema, vigilen canvis en els fitxers de sistema o bé en les configuracions dels serveis típics. Altres sistemes estan basats en xarxa i intenten verificar que no es produeixen comportaments estranys com, per exemple, casos de suplantació, en què hi ha falsificacions d'adreces conegudes; controlen el trànsit sospitós i possibles atacs de denegació de servei, detectant trànsit excessiu cap a determinats serveis i controlant que no hi ha interfícies de xarxa en mode promiscu (símptoma de *sniffers* o capturadores de paquets).

Alguns exemples d'eines IDS serien:

- Logcheck: verificació de *logs*.
- TripWire: estat del sistema mitjançant sumes md5 aplicades als arxius.
- Portsentry: protegeixen contra escanejos de ports i detecten indicis d'activitat sospitosa.
- AIDE: una versió lliure de TripWire.
- Snort: IDS de verificació d'estat d'una xarxa completa.

Algunes de les eines de seguretat es poden considerar també eines d'atac. Per tant, es recomana provar aquestes eines sobre màquines de la nostra xarxa local o privada. Mai no s'han de portar a terme atacs de prova amb adreces IP a tercers, ja que aquests podrien prendre-ho com a intrusions i demanar-nos responsabilitats, demanar-les al nostre ISP o avisar les autoritats competents perquè ens investiguin o tanquin el nostre accés.

A continuació, comentem breument algunes de les eines i alguns dels usos que se'ls pot donar:

1) Logcheck: una de les activitats d'un administrador de xarxa és verificar diàriament (més d'una vegada per dia) els arxius *log* per a detectar possibles atacs/intrusions o esdeveniments que puguin donar indicis sobre aquestes qüestions. Aquesta eina selecciona (dels arxius *log*) informació condensada de problemes i riscos potencials i després envia aquesta informació al responsable, per exemple, a través d'un correu. El paquet inclou utilitats per a executar-se de manera autònoma i recordar l'última entrada verificada per a les subsegüents execucions. La llista d'arxius que cal monitorar s'emmagatzema a `/etc/logcheck/logcheck.logfiles` i la configuració per defecte és adequada (si no es va modificar gran part de l'arxiu `/etc/syslog.conf`). Logcheck pot funcionar en tres modalitats: paranoid, server i workstation. La primera és molt detallada i hauria de limitar-se a casos específics com, per exemple, *firewalls*. La segona (per defecte) és la recomanada per a la majoria de servidors, i l'última és l'adequada per a estacions d'escriptori. Aquesta eina permet una configuració total dels filtres i de les sortides, si bé pot ser complicat reescriure'ls. Les regles es poden classificar en les següents: intent d'intrusió (*cracking*), emmagatzemades a `/etc/logcheck/cracking.d/`; les d'alerta de seguretat, emmagatzemades a `/etc/logcheck/violations.d/`; i les que són aplicades a la resta de missatges. [Her13]

2) Tripwire: aquesta eina manté una base de dades de sumes de comprovació dels fitxers importants del sistema. Pot servir com a sistema IDS preventiu. Ens serveix per a portar a terme una foto del sistema, poder comparar després qualsevol modificació introduïda i comprovar que no hagi estat malmès per un atacant. L'objectiu aquí és protegir els arxius de la pròpia màquina, evitar que es produeixin canvis com, per exemple, els que podria haver provocat un *rootkit*. Així, quan tornem a executar l'eina, podem comprovar els canvis respecte a l'execució anterior. Cal triar un subconjunt d'arxius que siguin importants en el sistema o font de possibles atacs. Hi ha altres eines lliures de funcionament equivalent, entre aquestes una possibilitat és AIDE. Tripwire (http://sourceforge.net/projects/tripwire/) és una eina que ajudarà l'administrador notificant possibles modificacions i canvis en arxius per a evitar possibles danys (majors). Aquesta eina compara les diferències entre els arxius actuals i una base de dades generada prèviament per a detectar canvis (insercions i esborrat), la qual cosa és molt útil per a detectar possibles modificacions d'arxius vitals, com per exemple, d'arxius de configuració.

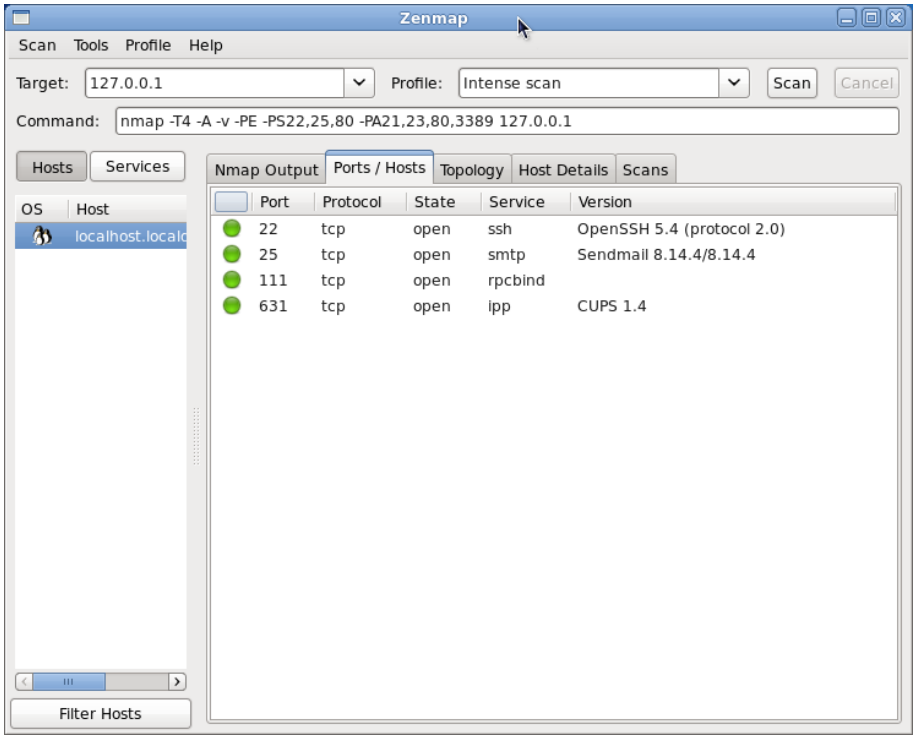
3) Portsentry: forma part d'un conjunt d'eines* que proporcionen serveis de seguretat de nivell de *host* per a GNU/Linux. PortSentry, LogSentry i Hostsentry protegeixen contra escanejos de ports i detecten indicis d'activitat sospitosa.

***http://sourceforge.net/projects/sentrytools/**

4) Nmap [Insb]: és una eina d'escaneig de ports per a xarxes (figura 2). Pot escanejar des de màquines individuals a segments de xarxa. Permet diversos tipus d'escaneig de ports, depenent de les proteccions que tingui el sistema.

També té tècniques que permeten determinar el sistema operatiu que fan servir les màquines remotes. Pot emprar diferents paquets de TCP i UDP per a provar les connexions. Hi ha algunes interfícies gràfiques per a KDE i Gnome, i alguna amb bastants possibilitats, com Zenmap.

Figura 2. Interfície gràfica per a Nmap durant una anàlisi de serveis locals



5) tcpdump: es tracta d'una eina molt potent que està en totes les distribucions i que ens servirà per a analitzar els paquets de xarxa. Aquest programa permet l'abocament del trànsit d'una xarxa i pot analitzar la majoria de protocols utilitzats avui dia (IPv4, ICMPv4, IPv6, ICMPv6, UDP, TCP, SNMP, AFS BGP, RIP, PIM, DVMRP, IGMP, SMB, OSPF, NFS, entre d'altres).

Donada la importància d'analitzar cap a on van els paquets i d'on vénen, mostrarem algunes ordres habituals de tcpdump:

```
# tcpdump
paràmetres per defecte -v o  -vv per al nivell d'informació mostrada, -q sortida ràpida
# tcpdump -D
interfícies disponibles per a la captura
# tcpdump -n
mostra IP en lloc d'adreces
# tcpdump -i eth0
captura el trànsit d'eth0
# tcpdump udp
només els paquets UDP
# tcpdump port http
només els paquets del port 80 (web)
# tcpdump -c 20
solament els 20 primers paquets
```

```
# tcpdump -w capture.log
envia les dades a un arxiu
# tcpdump -r capture.log
llegeix les dades d'un arxiu
```

Les dades capturades no són text, per la qual cosa es pot utilitzar aquesta mateixa eina per a la seva lectura o bé fer-ne servir una altra, com Wireshark, per a portar a terme la seva lectura i descodificació. Amb tcpdump *host* www.site.com només examinem els paquets que continguin aquesta adreça.

```
tcpdump src 192.168.1.100 and dst 192.168.1.2 and port ftp
```

Per a mostrar els paquets FTP que vagin de 192.168.1.100 a 192.168.1.2.

```
tcpdump -A       mostra el contingut dels paquets.
```

6) Wireshark (abans denominat Ethereal): és un analitzador de protocols i captura el trànsit de la xarxa (actua com *sniffer*). Permet visualitzar el trànsit capturat, veure estadístiques i dades dels paquets individuals i agrupar-los, ja sigui per origen, destinació, ports o protocol. Pot fins i tot reconstruir el trànsit d'una sessió sencera d'un protocol TCP.

7) Snort [Sno]: és un dels millors sistemes IDS en GNU/Linux, que permet fer anàlisis de trànsit en temps real i guardar registres dels missatges. Permet portar a terme anàlisis dels protocols i cerques per patrons (protocol, origen, destinació, etc.). Pot usar-se per a detectar diversos tipus d'atacs. Bàsicament, analitza el trànsit de la xarxa per a detectar patrons que puguin correspondre a un atac. El sistema utilitza una sèrie de regles per a anotar la situació (`log`), produir un avís (`alert`) o descartar la informació (`drop`).

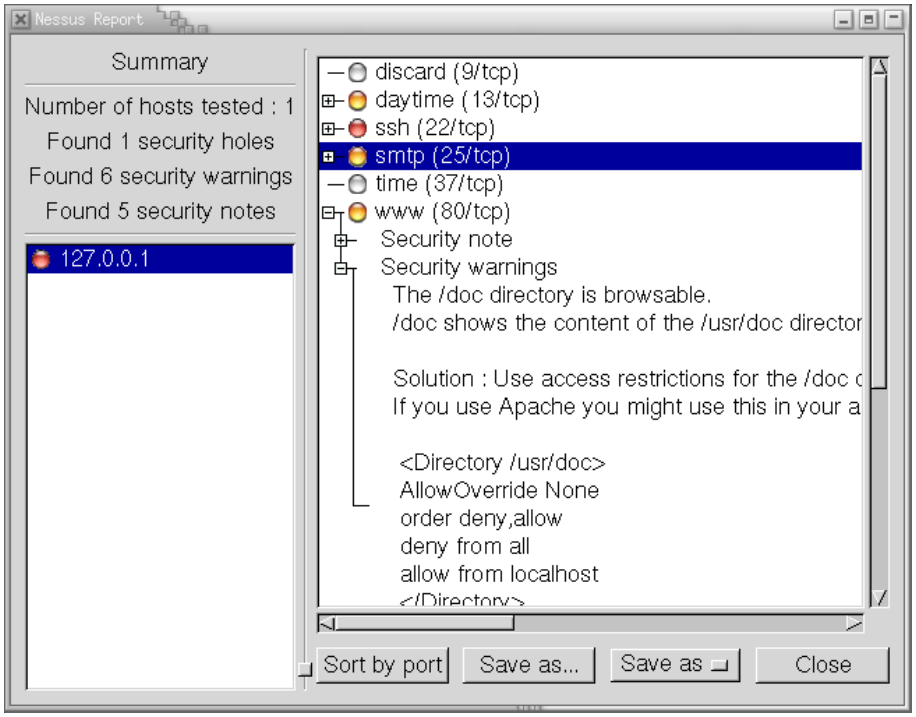
Instal·lació Snort

Snort és un sistema IDS altament recomanable. Per a la instal·lació detallada, podeu consultar les *Setup Guides* disponibles en https://www.snort.org/documents.

8) Nessus [Nes]: és un detector de vulnerabilitats conegudes, que prova diferents tècniques d'intrusió i assessora sobre com millorar la seguretat per a les detectades. És un programa modular que inclou més d'11.000 connectors (*plugins*) per a les diferents anàlisis. Utilitza una arquitectura client/servidor, amb un client gràfic que mostra els resultats i el servidor que porta a terme les diferents comprovacions sobre les màquines. Té capacitat per a examinar xarxes senceres i genera els resultats en forma d'informes que poden exportar-se a diferents formats (per exemple, HTML). Fins a l'any 2005, Nessus 2 era una eina lliure, però la companyia va decidir convertir-se en propietària en la versió Nessus 3. Encara en GNU/Linux, es pot trobar i seguir utilitzant Nessus 2 (figura 3), que es manté amb llicència GPL i una sèrie de connectors que es van actualitzant. Nessus 3+ (i versions posteriors) com a eina propietària per GNU/Linux és més potent, àmpliament utilitzada i una de les eines més populars de seguretat. Normalment es manté lliure de cost una versió amb els connectors menys actualitzats que la versió de pagament. I encara es manté Nessus 2 en llicència GPL, actualitzant-la sovint. També ha sorgit

una *fork* lliure a partir de Nessus 2, denominada OpenVAS, que està disponible pràcticament per a la majoria de distribucions GNU/Linux. Més endavant, analitzarem alguna de les seves possibilitats.

Figura 3. Client de Nessus 2 que mostra l'informe de vulnerabilitats i les possibles solucions.



Podem trobar moltes més eines de seguretat disponibles. Un bon lloc per a examinar és http://sectools.org, on els creadors del Nmap van mantenir una llista d'eines populars votades pels usuaris al llarg de diversos anys. Ara aquesta llista és una mica antiga, però es mantenen actualitzades les referències a les eines, les quals, a dia d'avui, són perfectament útils i de les més utilitzades.

Passarem ara, en les seccions següents, a analitzar diferents eines amb més detall i alguns casos d'ús. Algunes d'aquestes són bastant útils en aspectes generals de detecció de vulnerabilitats i intrusos.

6.1. OpenVAS

L'eina OpenVAS (*Open Vulnerability Assessment System*) és un conjunt d'eines que treballen a l'unísón per a córrer tests sobre màquines clients usant una base de dades d'*exploits* coneguts. La idea és saber en quina situació es troben els clients provats contra una sèrie d'atacs coneguts.

En aquest cas, utilitzarem una distribució Debian com a sistema base per a provar els clients corresponents. Comencem instal·lant el repositori necessari per a una Debian 7.x (wheezy):

(Nota: vigileu els salts de línia indicats en aquestes línies d'ordres amb \; haurien de ser una única línia sense aquest caràcter, ni espais afegits.)

```
# echo "deb http://download.opensuse.org/repositories/security:/OpenVAS\
:/UNSTABLE:/v6/Debian_7.0/ ./" >> /etc/apt/sources.list
# wget http://download.opensuse.org/repositories/security:/OpenVAS\
:/UNSTABLE:/v6/Debian_7.0/Release.key
# apt-key add ./Release.key
# apt-get update
```

Així instal·lem el repositori d'OpenVAS per Debian, afegint-lo com a font de paquets, important el seu *hash* (del repositori), i afegint-lo al sistema perquè puguem autenticar els paquets del repositori. Seguidament, procedim a un *update* del sistema de paquets APT. I procedirem a instal·lar els paquets necessaris per als diferents components d'OpenVAS, paquets necessaris per a la generació d'informes i alguns de necessaris per a la creació de certificats posteriors:

```
# apt-get -y install greenbone-security-assistant openvas-cli openvas-manager \
openvas-scanner openvas-administrator sqlite3 xsltproc rsync
# apt-get -y install texlive-latex-base texlive-latex-extra \
texlive-latex-recommended htmldoc
# apt-get -y install alien rpm nsis fakeroot
```

El següent bloc d'ordres, totes com a usuari *root*, s'encarrega de generar els certificats necessaris per al lloc OpenVAS, preparar les bases de dades que s'utilitzaran (sincronitzant-la per Internet amb els servidors d'OpenVAS i els seus *plugins* disponibles), parar els *daemons* d'OpenVAS, reiniciar el *daemon* principal d'OpenVAS (es fa la sincronització de dades necessàries) i afegir usuari com a admin, per la qual cosa ens preguntaran el *password* que desitgem. Posteriorment, s'aturarà el *daemon* general d'OpenVAS i es reiniciaran els serveis. Aquest bloc següent pot executar-se directament com a *root* sobre una Debian 7.x (amb els passos anteriors prèviament portats a terme). Cal tenir en compte que la sincronització de dades amb els servidors OpenVAS pot portar un temps considerable:

```
test -e /var/lib/openvas/CA/cacert.pem  || openvas-mkcert -q
openvas-nvt-sync
test -e /var/lib/openvas/users/om || openvas-mkcert-client -n om -i
/etc/init.d/openvas-manager stop
/etc/init.d/openvas-scanner stop
openvassd
openvasmd --rebuild
openvas-scapdata-sync
openvas-certdata-sync
test -e /var/lib/openvas/users/admin || openvasad -c add_user -n admin -r Admin
killall openvassd
sleep 15
/etc/init.d/openvas-scanner start
/etc/init.d/openvas-manager start
/etc/init.d/openvas-administrator restart
/etc/init.d/greenbone-security-assistant restart
```

Una vegada efectuats tots aquests passos, podem accedir a l'eina en el nostre sistema arrencant el navegador web i obrint l'URL:

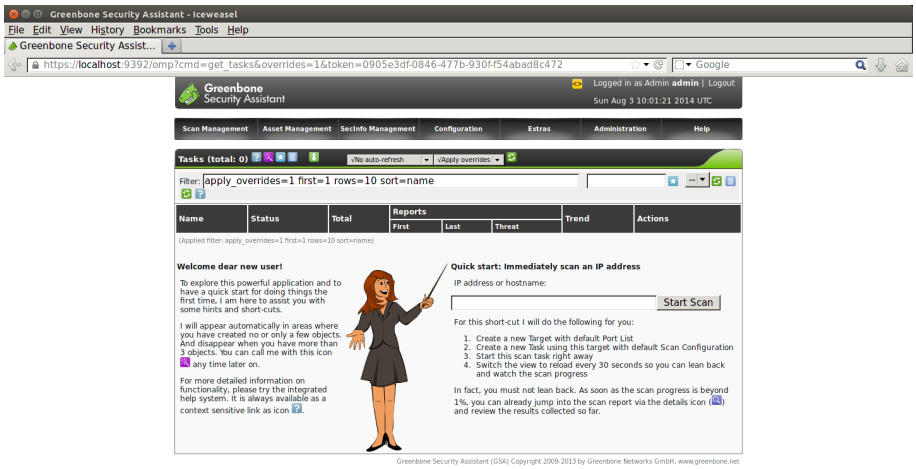
```
https://localhost:9392/
```

Quan obrim l'adreça esmentada, tindrem un procés de *login*, que superarem mitjançant l'usuari login creat.

En accedir, tindrem la interfície web disponible, on mitjançant un *wizard* podrem directament indicar la IP o DNS del sistema on farem l'escàner de vulnerabilitats.

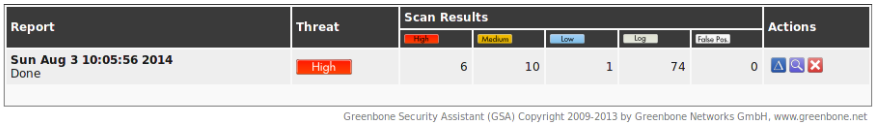
Com sempre, cal assenyalar de nou que utilitzarem l'eina per a escanejar un *host* propi i no contra tercers, que podrien prendre's l'escaneig com un intent d'atac.

Figura 4. Interfície web inicial OpenVAS per a *quick start*.



Una vegada que l'escàner hagi progressat, podem accedir al resultat mitjançant les icones de l'apartat *Actions* de la interfície associada.

Figura 5. VVulnerabilitats detectades amb el seu nivell de risc.



Al final, disposarem de l'informe en forma d'una llista agregada de potencials vulnerabilitats detectades, amb un nivell de *thread* i les vulnerabilitats associades a cada nivell de perill. Podem de nou examinar els detalls de l'informe (icona de lupa) per a disposar de la possibilitat de visualitzar-lo o descarregar-lo (en diversos formats).

En la interfície de l'informe (*Report Summary*), podem filtrar els resultats per diversos criteris, i a sota podem examinar cada incidència segons el seu nivell de perill, verificant l'explicació de la vulnerabilitat i indicacions per a la seva possible solució.

Figura 6. Una vulnerabilitat detectada.

Així doncs, ja disposem d'una instal·lació bàsica d'OpenVAS funcional per a controlar vulnerabilitats dels nostres servidors, i podem usar-la com a mitjà per a perfeccionar problemes de seguretat del nostre entorn.

Hi ha diverses opcions extra disponibles per a perfeccionar la instal·lació, entre les quals es troben tasques de programes d'escaneig, generar informes automàticament i enviar alertes per correu. Podeu explorar les opcions d'interfície per a obtenir una configuració més detallada i així optimitzar l'ús d'OpenVAS.

6.2. Denyhosts

En aquest cas, tractem amb un altre problema habitual. Es tracta de l'administració dels accessos a través d'SSH i de l'administració de comptes remots dels usuaris. El servei d'SSH és propens a rebre atacs de diverses tipologies, com ara:

1) Atacs controlats cap a ports, típicament el TCP 22, on s'allotja per defecte.

2) Atacs contra usuaris Linux predeterminats o de sistema, buscant comptes de tipus convidat que estiguessin amb *passwords* per defecte, o algunes contrasenyes d'usuaris associats a serveis (per exemple, bases de dades, CMS, control de versions o altres) que hagin estat instal·lades amb els seus comptes per defecte, o contrasenyes provinents de defecte.

3) Atacs contra usuari *root*, mitjançant diccionaris, guiats o simplement per la força bruta.

El servei SSH en si mateix ofereix certes possibilitats (mitjançant el fitxer de configuració del seu *daemon* `/etc/ssh/sshd_config`) per a mitigar aquests atacs, com canviar el port per defecte del servei, o en el plànol dels usuaris,

intentar mitigar els problemes mitjançant una selecció correcta de les seves contrasenyes i promocionant polítiques de canvi i ús de contrasenyes no trivials. A l'usuari *root* se li pot denegar, a més del seu accés en SSH (opció *PermitRootLogin no*), l'accés remot, la qual cosa no impedirà que l'atacant intenti aconseguir accés local mitjançant algun altre usuari. També podrem complementar serveis com SSH amb altres mesures: a més d'aquelles que veurem en l'apartat següent 7, amb *TCP wrappers*, amb l'ús de tallafocs (*firewalls*), o també inhabilitant l'ús de contrasenyes per a SSH i forçant així l'ús de claus pública/privada per als usuaris d'SSH.

Davant d'aquests problemes, Denyhosts ofereix la possibilitat de monitorar els *logs* d'SSH i impedir (*ban*) l'accés a usuaris que detecti que possiblement estan portant a terme un atac remot. Pot configurar-se per a permetre una sèrie d'intents de connexió i els temps entre aquests. Perquè Denyhosts funcioni correctament, el *daemon* d'SSHD ha d'estar compilat amb l'opció de *TCP wrappers* (comentarem aquesta tècnica en el següent apartat 7), encara que en el cas dels *daemons* instal·lats de paquets de la distribució ja ve així per defecte.

En el cas d'una distribució Debian (de manera semblant a una Fedora), el podem instal·lar amb:

```
# apt-get install denyhosts
```

I podem configurar els paràmetres en el fitxer `/etc/denyhosts.conf`, un exemple de valors de certs paràmetres (hi ha moltes més opcions, en particular les relacionades amb correu de notificació mitjançant opcions *SMTP_*):

```
SECURE_LOG = /var/log/auth.log
BLOCK_SERVICE = sshd
DENY_THRESHOLD_INVALID = 5
DENY_THRESHOLD_VALID = 10
DENY_THRESHOLD_ROOT = 1
DENY_THRESHOLD_RESTRICTED = 1
HOSTNAME_LOOKUP=YES
```

Primer establim el lloc per a examinar el *log* corresponent als intents d'SSH (en aquest cas, en Debian és el fitxer esmentat), que serà el fitxer que monitorarà Denyhosts. Controlarem el bloqueig del servei *sshd* (*daemon ssh*) si es donen les circumstàncies (poden controlar-se altres serveis). En aquest exemple, permetem fins a 5 intents fallits de comptes no vàlids (no existents en el sistema), 10 intents fallits de comptes registrats en el sistema (en `/etc/passwd`), i 1 intent sobre *root* (això té sentit, per exemple, si sabem que no està permès l'accés des de *root*); aquests seran els intents abans de bloquejar el *host* que els ha produït. En l'últim, es permet que es faci una cerca DNS per a l'IP que ha portat a terme la fallada per intents excessius.

Cal anar amb compte amb els valors de *Threshold* per a assegurar que complim uns mínims de possibilitats, ja que intents vàlids des de *hosts*, per oblits o fallades repetides, poden provocar que un *host* vàlid es bloquegi.

El control anterior dels hosts que es permeten i no es fa amb els fitxers `/etc/hosts.deny` i `/hosts.allow`, per bloquejats i permesos; per exemple, si ens volem assegurar que el nostre *host* habitual no es bloquegi (sempre que disposem d'IP estàtica), podríem col·locar (en `/etc/hosts.allow`):

```
ALL: la nostra_ip
```

Posteriorment, podem reiniciar el servei amb:

```
# /etc/init.d/denyhosts restart
```

Els *hosts* que es bloquegin al llarg del temps acabaran incloent-se en l'arxiu `/etc/hosts.deny`. També poden purgar-se al cap d'un temps (hi ha opcions denominades *PURGE_* en la configuració de *denyhosts* que ho controlen de manera automàtica); podran purgar-se explícitament mitjançant:

```
# /etc/init.d/denyhosts stop
# denyhosts --purge
# /etc/init.d/denyhosts start
```

6.3. Fail2ban

Fail2Ban, disponible en Debian i Fedora, és similar en concepte a l'eina prèvia *Denyhosts*, però en lloc d'enfocar-se a SSH, en aquest cas, pot ser configurada per a monitorar qualsevol servei que escrigui intents de *login* en un fitxer de *log* i, en lloc de bloquejar, posar l'*host* culpable d'atacs en `/etc/hosts.deny` (encara que també pot configurar-se perquè actuï així). Fa els bloquejos per mitjà d'*iptables* (sistema de tallafocs per defecte del kernel Linux, com veurem en l'apartat 7.

Instal·lem amb:

```
# apt-get install fail2ban
```

I disposem posteriorment de la seva configuració per al nostre sistema en el directori `/etc/fail2ban`, en el qual trobarem el comportament per defecte en el fitxer `/etc/fail2ban/jail.conf`. En algunes versions recents, es recomana no editar aquest fitxer i disposar la configuració local en el directori

`/etc/fail2ban/jail.d` o en `/etc/fail2ban/jail.local`. Consulteu la pàgina man depenent de la versió disponible en la nostra distribució.

Algunes de les configuracions presents (com a exemples) es refereixen a:

- ignoreip = 127.0.0.1/8 : si són IP de màquines conegudes que no volem que fail2ban bloquegi. En aquest cas, el propi *localhost* i adreces associades.

- bantime = 600 : temps en segons en què una màquina és bloquejada si es detecta com a problemàtica.

- findtime = 600 i maxretry=3 : un *host* és bloquejat si abans que arribi a *findtime* segons ha generat *maxretry* intents fallits.

Després, per a cada servei trobem diversos camps (en la secció del fitxer de configuració `[servei]`); normalment, per defecte només es troba activat el servei d'SSH:

- enabled: activat el control del servei.

- port: port de servei que s'està utilitzant, normalment fent servir els noms de `/etc/services`.

- filter: filtres aplicats dels disponibles en `/etc/fail2ban/filter.d`. Aquests filtres serveixen per a enganxar els missatges d'esdeveniments dels *logs* examinats, per a detectar si corresponen o no al servei.

- action: la mesura que cal prendre per al bloqueig, per exemple *iptables-allports*, que bloquejarà els ports que faci servir el servei per al *host* problemàtic. Les *actions* que tenim disponibles les podem trobar en el directori `/etc/fail2ban/action.d/`.

- logpath: fitxer de *log* que es comprova per a detectar *logins* fallits.

- maxretry: si canviem en el servei el nombre màxim d'intents fallits.

Per exemple, en una Debian podríem copiar `/etc/fail2ban/jail.conf` en `/etc/fail2ban/jail.local` i ajustar els paràmetres locals generals com creguem convenient, i dins del fitxer cadascuna de les seccions corresponents al servei.

Posteriorment, reiniciem el servei:

```
# /etc/init.d/fail2ban restart
```

I si per exemple tenim *fail2ban* emetent *logs* a `/var/log/fail2ban.log`, podríem observar sortides com:

```
2014-08-24 18:49:09,466 fail2ban.actions: WARNING [apache] Ban 1.2.3.4
2014-08-24 19:08:33,213 fail2ban.actions: WARNING [ssh] Ban 1.2.3.4
```

I observar amb *iptables* si s'han produït bloquejos (en aquest cas, es visualitza allò produït en el servei SSH):

```
#iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
fail2ban-ssh  tcp  --  anywhere          anywhere             multiport dports ssh
........
Chain fail2ban-ssh (1 references)
target     prot opt source               destination
DROP       all  --  1.2.3.4             anywhere
RETURN     all  --  anywhere            anywhere
......
```

si disposem dels *hosts* problemàtics bloquejats.

Algunes ordres útils per a examinar els *logs*, comprovar l'estatus general d'un servei particular o eliminar el bloqueig *iptables* d'una IP determinada:

```
# tail /var/log/fail2ban.log
# fail2ban-client status
# fail2ban-client status ssh
# iptables -D fail2ban-<CHAIN_NAME> -s <IP> -j DROP
```

En què en l'últim cas es desbloqueja una IP si el bloqueig es va produir per a *iptables*, i on normalment (tret que haguem canviat el tipus d'*action*), la *CHAIN_NAME* serà la corresponent al servei, en l'exemple anterior SSH (*fail2ban-ssh*).

7. Protecció mitjançant filtratge (*TCP wrappers* i tallafocs)

Els *TCP wrappers* [Mou02] són un programari que actua d'intermediari entre les peticions de l'usuari del servei i els dimonis dels servidors que ofereixen el servei. Moltes de les distribucions vénen ja amb els *wrappers* activats i podem configurar els nivells d'accés. Els *wrappers* se solen utilitzar en combinació amb inetd o xinetd, de manera que protegeixin els serveis que ofereixen.

El *wrapper* bàsicament substitueix el dimoni del servei per un altre que fa d'intermediari (anomenat *tcpd*, normalment a `/usr/sbin/tcpd`). Quan aquest rep una petició, en verifica l'usuari i l'origen, per determinar si la configuració del *wrapper* del servei permet utilitzar-lo o no. A més, incorpora facilitats per a generar registres o bé informar per correu dels possibles intents d'accés i després executa el dimoni adequat assignat al servei.

Per exemple, suposem l'entrada següent a inetd:

```
finger stream tcp nowait nobody /usr/etc/in.fingerd in.fingerd
```

Es canvia per:

```
finger stream tcp nowait nobody /usr/sbin/tcpd in.fingerd
```

De manera que, quan arribi una petició, serà tractada pel dimoni tcpd, que s'encarregarà de verificar-ne l'accés (per a més detalls, vegeu la pàgina *man tcpd*).

Wrappers

Els *wrappers* permeten controlar la seguretat mitjançant llistes d'accés en l'àmbit dels serveis.

També existeix un mètode alternatiu de *TCP wrapper*, en què l'aplicació original es compila amb la biblioteca de *wrappers*. Llavors l'aplicació no cal que sigui a inetd i podrem controlar-la de la mateixa manera que en el primer cas, amb la configuració que comentem a continuació.

El sistema de *wrappers* es controla des del fitxer `/etc/hosts.deny`, on especifiquem quins serveis deneguem i a qui, per mitjà d'opcions, com un petit intèrpret d'ordres per a guardar la informació de l'intent, i des del fitxer `/etc/hosts.allow`, on solem col·locar el servei que utilitzarem, seguit de la llista de qui deixem entrar (més endavant, en el taller, en veurem un petit exemple). També existeixen les ordres `tcpdchk`, que comproven la configuració dels fitxers amfitrions (vegeu *man hosts_access* i *hosts_options*), per verificar que són correctes; és a dir, comproven la configuració. L'altra ordre útil és

`tcpdmatch`, a la qual assignem un nom de servei i un possible client (usuari i/o amfitrió) i ens diu què faria el sistema davant d'aquesta situació.

7.1. Tallafocs

Un tallafoc (*firewall*) és un sistema o grup de sistemes que reforça les polítiques de control d'accés entre xarxes. El tallafoc pot estar implementat en programari, com una aplicació especialitzada que s'executa en un ordinador individual, o bé pot tractar-se d'un dispositiu especial dedicat a protegir un o més ordinadors.

En general, disposarem o bé de l'aplicació de tallafocs per a protegir una màquina concreta connectada directament a Internet (directament o per proveïdor), o bé podrem col·locar en la nostra xarxa una o diverses màquines dedicades a aquesta funció, de manera que protegeixin la nostra xarxa interna.

Tècnicament, la millor solució és disposar d'un ordinador amb dues o més targetes de xarxa que aïllin les diferents xarxes (o segments de xarxa) connectades, de manera que el programari de tallafocs de la màquina (o en el seu cas, un maquinari especial) s'encarregui de connectar els paquets de les xarxes i determinar quins poden passar i quins no, i a quina xarxa.

Aquest tipus de tallafocs sol combinar-se amb un encaminador (*router*) per a enllaçar els paquets de les diferents xarxes. Una altra configuració típica és la de tallafocs cap a Internet, per exemple amb dues targetes de xarxa: en una obtenim o proporcionem trànsit a Internet i en l'altra enviem o proporcionem el trànsit a la nostra xarxa interna, de manera que podem eliminar el trànsit que no va destinat a nosaltres, i també controlar el trànsit que es mou cap a Internet, per si no volem que es tingui accés a alguns protocols o bé sospitem que hi ha possibilitats de fuites d'informació a causa d'alguns atacs. Una tercera possibilitat és la màquina individual connectada amb una única targeta de xarxa cap a Internet, directament o bé a través d'un proveïdor. En aquest cas, només volem protegir la nostra màquina d'atacs d'intrusos, de trànsit no desitjat o de la possibilitat de robatori de dades.

És a dir, en aquests casos podem veure que el tallafoc, depenent de si és programari o no, de si la màquina té una o diverses targetes de xarxa o de si protegeix una màquina individual o una xarxa, pot tenir configuracions i usos diferents.

El tallafoc, en general, permet definir a l'usuari una sèrie de polítiques d'accés (quines són les màquines a les quals es pot connectar o quines poden rebre informació i el tipus d'informació que pot rebre) per mitjà del control dels ports TCP/UDP permesos d'entrada (*incoming*) o de sortida (*outcoming*). Algunes eines de gestió de tallafocs vénen amb una sèrie de polítiques preconfigurades, o només diuen si es vol un nivell de seguretat alt, mitjà o baix, o, finalment, permeten personalitzar les opcions totalment (màquines, protocols, ports, etc.).

Una altra tècnica de vegades relacionada és la NAT (*network address translation*, traducció d'adreces de xarxa). Aquesta tècnica proporciona una via per a ocultar les adreces IP usades en la xarxa privada i les oculta d'Internet, però hi manté l'accés des de les màquines. Un dels mètodes típics és el denominat *masquerading*. Si s'usa *NAT masquerading*, un o diversos dispositius en la xarxa poden aparèixer com una única adreça IP vistos des de fora. Això permet connectar diversos ordinadors a un únic dispositiu de connexió externa; per exemple, un cas d'encaminador ADSL a la llar permet connectar diverses màquines sense necessitat que el proveïdor ens proporcioni diferents adreces IP. Els encaminadors ADSL acostumen a oferir algun tipus de *NAT masquerading* i també possibilitats de tallafoc. Sol ser bastant comú utilitzar una combinació d'ambdues tècniques. En aquest cas, entra en joc, a més de la configuració de la màquina del tallafoc (els casos vistos abans), la configuració de la xarxa privada interna que volem protegir.

Seguretat a nivell de paquets

Els tallafocs permeten establir seguretat a nivell de paquets i en les connexions de comunicació.

7.2. Netfilter: iptables

El nucli Linux (a partir de versions 2.4.x) proporciona un subsistema de filtratge anomenat Netfilter [Net], que ofereix característiques de filtratge de paquets i també NAT. Aquest sistema permet fer servir diferents interfícies de filtratge, entre les quals la més utilitzada s'anomena *iptables*. L'ordre principal de control és `iptables`. Abans s'oferia un altre sistema de filtratge, anomenat *ipchains*, en els nuclis 2.2 [Gre] i el sistema tenia una sintaxi diferent (tot i que amb semblances conceptuals). En els nuclis 2.0 s'utilitzava un altre sistema anomenat *ipfwadm*. Aquí (i en els exemples posteriors) tractarem només amb Netfilter/iptables (és a dir, amb les possibilitats de tallafoc en nuclis de les versions 2.4/2.6/3.x). També comentarem noves propostes, com *nftables*, que serà el proper substitut de Netfilter/iptables, a partir d'algunes de les últimes branques kernel 3.x, i segurament el *firewall* per defecte en la branca 4.x i superiors. nftables té capes de compatibilitat amb iptables, amb la qual cosa segurament durant un temps veurem encara de manera majoritària regles de *firewall* compatibles amb iptables.

Enllaç d'interès

Sobre Netfilter vegeu http://www.netfilter.org.

Respecte a Netfilter, com comentàvem, el seu principal component en aquests moments és *iptables*, que funciona com una eina de tallafoc (*firewall*), amb el suport en un nivell de kernel, fet que permet les accions de filtratge dels paquets i gestió de translació d'adreces de xarxa (NAT).

La interfície de l'ordre `iptables` permet fer les diverses tasques de configuració de les regles que afecten el sistema de filtratge: generació de registres, accions de preencaminament i postencaminament de paquets, NAT i reenviament (*forwarding*) de ports.

Elements d'iptables

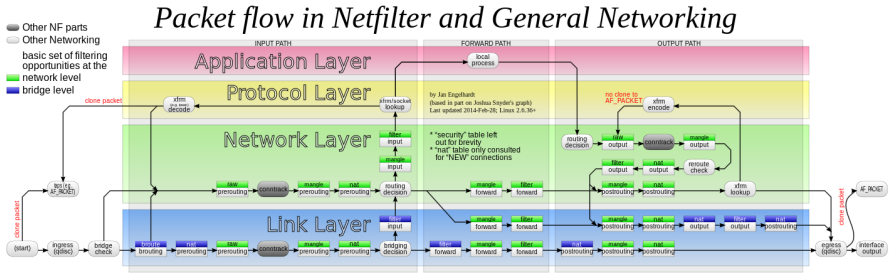
iptables aporta diversos elements, com les taules, cadenes (*chains*) i les regles mateixes.

L'arrencada del servei es fa amb `/etc/init.d/iptables start`, si no estava ja configurada en el nivell d'execució.

L'ordre `iptables -L` genera una llista de les regles actives en aquell moment en cada una de les cadenes. Si no s'han configurat prèviament, per defecte acostumen a ser l'acceptació de tots els paquets de les cadenes d'`INPUT` (entrada), `OUTPUT` (sortida) i `FORWARD` (reenviament).

El sistema d'iptables té com a nivell superior les taules. Cada una conté diferents cadenes que, al seu torn, contenen diferents regles. Les tres taules que hi ha són Filter, NAT i Mangled. La primera serveix per a les normes de filtratge, la segona, per a fer translació d'adreces dins d'un sistema que utilitzi NAT, i la tercera, menys usada, serveix per a especificar algunes opcions de control dels paquets i com es poden gestionar. Concretament, si estem amb un sistema directament connectat a Internet, utilitzarem, en general, només la taula Filter. Si el sistema és en una xarxa privada que ha de passar per un encaminador, passarel·la (*gateway*) o servidor intermediari (*proxy*), o una combinació d'aquests, segurament disposarem d'un sistema de NAT o IP *masquerading*; si estem configurant precisament la màquina que permet accés extern, haurem de tocar les taules NAT i la Filter. Si la màquina és en un sistema de xarxa privada, però és una de les màquines internes, serà suficient amb la taula Filter, tret que sigui un servidor el que faci NAT a un altre segment de xarxa.

Figura 7. Fluxos dels paquets de xarxa involucrats en la gestió de NetFilter/iptables



Més detall de la figura 7

La figura procedent de Wikipedia pot observar-se amb més detall en l'article: http://en.wikipedia.org/wiki/iptables

Si un paquet arriba al sistema, en el tallafoc implementat per iptables es mirarà primer si hi ha regles a la taula NAT, per si cal fer traduccions d'adreces cap a la xarxa interna (les adreces normalment no són visibles a l'exterior); després es miraran les regles de la taula Filter per a decidir si es deixaran passar els paquets o si no són per a nosaltres, i, si tenim regles `forward`, per a saber cap on els reenviem. En canvi, quan els nostres processos generen paquets, les regles `output` de la taula Filter controlen si els deixem sortir o no, i si hi hagués sistema NAT, les regles traduirien les adreces de manera que quedessin emmascarades. En la taula NAT sol haver-hi dues cadenes: `prerouting` i `postrouting`. En la primera, les regles han de decidir si cal fer algun encaminament del paquet i quina serà l'adreça de destinació. En la segona, es decideix finalment si el paquet es fa passar o no cap a l'interior (la xarxa privada, per exemple). També existeix una cadena `output` per al trànsit que es generi localment de sortida a la xarxa privada, ja que `prerouting` no ho controla (per a més detalls, es pot examinar *man iptables*).

Tot seguit comentarem alguns aspectes i exemples de configuració de la taula Filter*. La configuració típica de la taula Filter és d'una sèrie de regles que especifiquen què es fa dins d'una determinada cadena, com les tres anteriors (`input`, `output` o `forward`). Normalment, s'especifica:

***Per a les altres taules, es pot consultar la bibliografia associada.**

```
iptables -A chain -j target
```

En què `chain` és `input`, `output` o `forward`, i `target` és la destinació que es donarà al paquet que es correspongui amb la regla. L'opció `-A` afegeix la regla a les existents.

Amb aquesta fase d'afegir regles cal prendre precaucions, ja que l'ordre importa. Cal col·locar les menys restrictives al principi, ja que, si primer posem una regla que elimini els paquets, malgrat que hi hagi una altra regla, no es tindrà en compte. L'opció `-j` permet decidir què farem amb els paquets, habitualment `accept` (acceptar-los), `reject` (rebutjar-los) o `drop` (simplement perdre'ls). És important la diferència entre `reject` i `drop`. Amb el primer, rebutgem el paquet i normalment informem l'emissari que hem rebutjat l'intent de connexió (normalment per un paquet de tipus ICMP). Amb el segon (`drop`), simplement perdem el paquet com si mai no hagués existit i no enviem cap tipus de resposta. Un altre `target` utilitzat és `log`, per a enviar el paquet al sistema de registres. Normalment, en aquest cas hi ha dues regles, una amb el `log` i una altra d'igual amb `accept`, `drop` o `reject`, per a permetre enviar al registre la informació de quins paquets han estat acceptats, rebutjats o perduts. Amb les opcions de generar-los en el tallafoc cal controlar-ne l'ús amb precisió, ja que són capaços, depenent de l'ambient de xarxa, de generar una enorme quantitat d'informació en els fitxers de registre.

Quan col·loquem la regla, també es pot utilitzar l'opció `-I` (insertar) per a indicar una posició, per exemple:

```
iptables -I INPUT 3 -s 10.0.0.0/8 -j ACCEPT
```

Que ens diu que es col·loqui la regla en la cadena `input` en tercera posició, i que s'acceptaran paquets (`-j`) que provinguin (amb font, o *source*, `-s`) de la subxarxa 10.0.0.0 amb màscara de xarxa 255.0.0.0. De manera semblant, amb `-D` podem esborrar un número de regla o la regla exacta, tal com s'especifica a continuació, esborrant la primera regla de la cadena o la regla que esmentem:

```
iptables -D INPUT 1
iptables -D INPUT -s 10.0.0.0/8 -j ACCEPT
```

També hi ha regles que permeten definir una política per defecte dels paquets (opció `-P`): es farà el mateix amb tots els paquets. Per exemple, se sol establir

a l'inici que es perdin tots els paquets per defecte, i s'habiliten després els que interessen. Moltes vegades també s'evita que es reeenviïn paquets si no cal (si no actuem com a encaminador). Això podria posar-se així:

```
iptables -P INPUT DENY
iptables -P OUTPUT REJECT
iptables -P FORWARD REJECT
```

Tot plegat estableix unes polítiques per defecte que consisteixen a denegar l'entrada de paquets, no permetre sortir i no reenviar paquets. Ara es podran afegir les regles que fan referència als paquets que volem utilitzar, dient quins protocols, ports i orígens o destinacions volem permetre o evitar. Això pot ser difícil, ja que hem de conèixer tots els ports i protocols que utilitzen el nostre programari o els nostres serveis. Una altra tàctica seria deixar actius només aquells serveis que siguin imprescindibles i habilitar amb el tallafoc l'accés dels serveis a les màquines que ens interessen.

Alguns exemples d'aquestes regles de la taula Filter podrien ser:

```
iptables -A INPUT -s 10.0.0.0/8 -d 192.168.1.2 -j DROP
iptables -A INPUT -p tcp --dport 113 -j REJECT --reject-with tcp-reset
iptables -I INPUT -p tcp --dport 113 -s 10.0.0.0/8 -j ACCEPT
```

On:

1) Perdem els paquets que vinguin de 10.x.x.x amb destinació 192.168.1.2.

2) Rebutgem els paquets tcp amb destinació al port 113, i s'emet una resposta de tipus tcp-reset.

3) Acceptem els mateixos paquets que a 2), però que provinguin de 10.x.x.x.

Pel que fa als noms de protocols i ports, el sistema iptables fa servir la informació facilitada pels fitxers `/etc/services` i `/etc/protocols`, i es pot especificar la informació (de port i de protocol) de manera numèrica o per nom (cal anar amb compte, en aquest cas, que la informació dels fitxers sigui correcta i que no hagin estat modificats, per exemple, per un atacant).

La configuració d'iptables sol establir-se mitjançant crides consecutives a l'ordre `iptables` amb les regles. Això crea un estat de regles actives que poden consultar-se amb `iptables -L`. Si volem desar-les perquè siguin permanents, podem fer-ho, en Fedora, amb:

```
/etc/init.d/iptables save
```

I es desen a:

```
/etc/sysconfig/iptables
```

En Debian es pot fer, si existeix l'*script* anterior:

```
/etc/init.d/iptables save nom-regles
```

Si no s'ofereix suport directe per a desar o recuperar regles, sempre es pot amb les ordres `iptables-save` i `iptables-restore` rederigir-les a fitxers, per a desar o recuperar la configuració del tallafoc.

En el primer cas, cal estar ben segurs que existeixi prèviament el directori `/var/lib/iptables`, que és on es desen els fitxers; `nom-regles` serà un fitxer en el directori.

Amb `/etc/init.d/iptables load` podem carregar les regles (en Debian cal donar el nom del fitxer de regles o bé usar `iptables-restore`), tot i que Debian suporta uns noms per defecte de fitxers, que són `active` per a les regles normals (que s'utilitzaran en un `start` del servei), i `inactive` per a les que quedaran quan es desactivi el servei (quan es faci un `stop`). Una altra aproximació que es fa servir sovint és la de col·locar les regles en un fitxer *script* amb les crides iptables que calgui i cridar-les, per exemple, col·locant-les en el nivell d'execució necessari o amb un enllaç cap a l'*script* a `/etc/init.d`.

Per acabar, esmentarem un petit exemple d'allò que podria ser un fitxer complet d'iptables (el # indica comentaris):

```
*filter
# Permetre tot el trànsit de loopback (lo0) i denegar la resta de 127/8
-A INPUT -i lo -j ACCEPT
-A INPUT ! -i lo -d 127.0.0.0/8 -j REJECT
# Acceptar totes les connexions entrants prèviament establertes
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
# Acceptar tot el trànsit sortint
-A OUTPUT -j ACCEPT
# Permetre  HTTP i HTTPS des de qualsevol lloc
-A INPUT -p tcp --dport 80 -j ACCEPT
-A INPUT -p tcp --dport 443 -j ACCEPT
# Permetre les connexions d'SSH
# Normalment utilitza el port 22, verificar-lo en el arxiu /etc/ssh/sshd_config.
-A INPUT -p tcp -m state --state NEW --dport 22 -j ACCEPT
# Respondre  al ping icmp
-A INPUT -p icmp -m icmp --icmp-type 8 -j ACCEPT
# Rebutjar tot el trànsit restant d'entrada
-A INPUT -j REJECT
-A FORWARD -j REJECT
COMMIT
```

En aquest exemple, només acceptem trànsit entrant de ping, http, htpps, ssh i bloquegem tot el trànsit restant d'entrada. Quant al trànsit de sortida, es deixa que surti tot sense restriccions. Es poden salvar les regles en un arxiu, carregar-les i, per exemple, des d'una altra màquina provar la connectivitat o executar

algun programa com nmap, que ens mostrarà els ports oberts en la màquina configurada amb iptables.

7.3. Paquets per a la gestió de tallafocs en les distribucions

Quant a eines de configuració més o menys automatitzades per tallafocs, hi ha diverses possibilitats, però cal tenir en compte que no solen oferir les mateixes prestacions que la configuració manual d'iptables (que en la majoria de casos seria el procés recomanat). Algunes eines són:

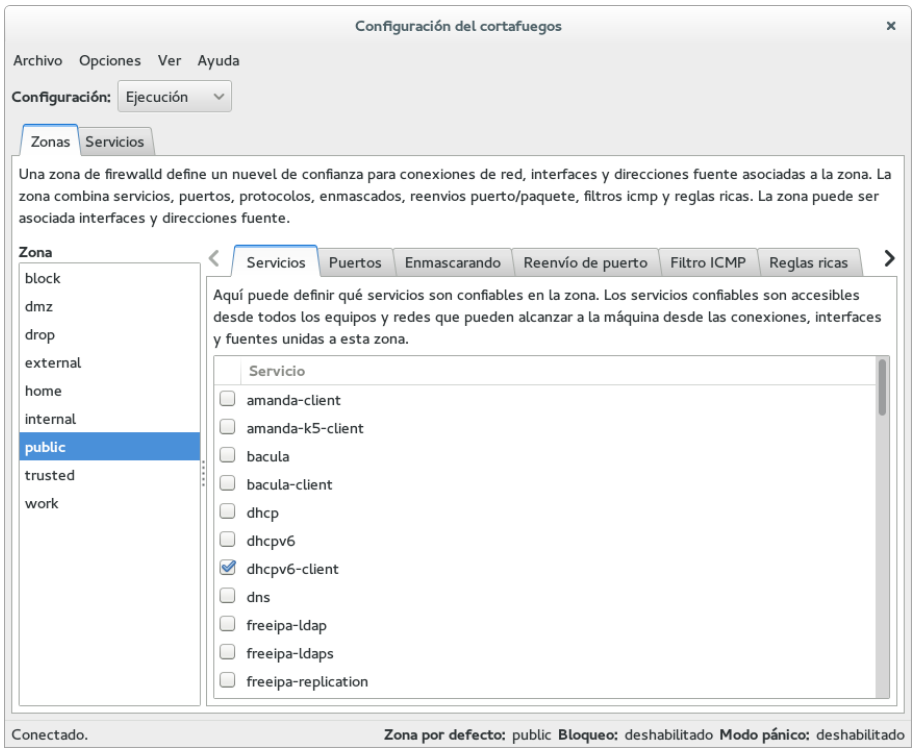
- *system-configure-firewall / Lokkit*: en la distribució Fedora / Red Hat, fins fa poc era inicialment un gestor de *firewall* molt bàsic i només permetia triar a l'usuari el nivell de seguretat que desitjava (alt, mitjà o baix). Després, ensenyava els serveis que es veien afectats i podíem deixar passar o no el servei canviant la configuració per defecte. En les últimes versions, ha evolucionat bastant i ja inclou possibilitats avançades de ports que cal controlar per a cada servei. El mecanisme utilitzat per sota és iptables. És possible veure la configuració final de regles que porta a terme `/etc/sysconfig/iptables` que, al seu torn, és llegit pel servei iptables, que es carrega en arrencada o per parada o arrencada mitjançant `/etc/init.d/iptables`, amb les opcions `start` o `stop` (o les equivalents en Systemd). S'ofereix en versió gràfica o textual anomenada *system-config-firewall-tui*.

- En les versions recents de Fedora / Red Hat, s'ha inclòs un nou gestor per defecte de *firewall*, que encapsula el funcionament d'iptables i que substitueix el previ (*system-config-firewall*), denominat *FirewallD*. Aquest proporciona control dinàmic del *firewall* i permet definir múltiples zones de *firewall* per a gestionar les diverses xarxes a les quals estigui connectat el sistema. Permet mantenir, de manera simultània, configuracions permanents i en temps d'execució. L'antic control de *firewall* de Fedora era estàtic i requeria que, cada vegada que les regles canviaven, s'hagués de reiniciar el *firewall*. FirewallD permet aplicar les noves regles sense necessitat de reiniciar el *firewall* complet. Algunes ordres útils d'aquest cas que ens permeten obtenir l'estatus, els serveis suportats, habilitar un servei, obtenir les zones i les zones amb serveis habilitats són:

  ```
  # firewall-cmd --state
  # firewall-cmd --get-services
  # firewall-cmd --add-service=http
  # firewall-cmd --get-zones
  # firewall-cmd --list-all-zones
  ```

  Es disposa també de l'eina gràfica *firewall-config* per a portar a terme una configuració més còmoda. En entorn de terminal, es recomana veure el man corresponent a *firewall-cmd*, a causa del gran nombre d'opcions disponibles.

Figura 8. Interfície gràfica firewall-config per a FirewallD



FirewallD

Es recomana examinar. https://fedoraproject.org/wiki/firewalld

- fwbuilder: una eina que permet construir les regles del tallafoc de manera gràfica. Es pot fer servir en diversos sistemes operatius (GNU/Linux tant Fedora com Debian, OpenBSD, MacOS), amb gestió de diferents tipus de tallafocs (incloent iptables).

- firestarter: una altra eina, ara obsoleta, però en distribucions antigues es feia servir com a eina gràfica (Gnome) per a la creació del tallafoc. Pràcticament maneja totes les possibilitats d'iptables, però així mateix disposa d'assistents que faciliten la construcció intuïtiva del tallafoc. Disposa, a més, d'un monitor en temps real per a detectar les intrusions.

- ufw (*Uncomplicated Firewall*): com el seu nom indica, pretén ser un *firewall* d'ús simple, que pot trobar-se en les distribucions Debian i Ubuntu, basat en ús de línia d'ordres, que permet un ús bastant senzill (però suportant gran part de les possibilitats d'iptables). Una petita sessió d'exemple:

```
$ sudo ufw allow ssh/tcp
$ sudo ufw logging on
$ sudo ufw enable
$ sudo ufw status
Firewall loaded

To                         Action  From
--                         ------  ----
22:tcp                     ALLOW   Anywhere
```

També es disposa d'un paquet gràfic per al seu ús anomenat *Gufw*.

ufw

Lloc i eina gràfica: https://launchpad.net/ufw http://gufw.org/

Normalment, cada un d'aquests paquets utilitza un sistema de regles que desa en algun fitxer propi de configuració, i que sol arrencar com a servei o com a execució de *script* en el nivell d'execució per defecte. Cal tenir una cura especial amb la consistència de les regles iptables del tallafoc, ja que moltes d'aquestes utilitats no suporten sincronització bidireccional de les regles: o bé s'actualitza el tallafoc sempre mitjançant l'eina de línia d'ordres o gràfica, o bé es fa amb l'ordre `iptables`; si es fan canvis d'ambdues maneres, pot ser que aquests canvis no siguin consistents o es pot perdre la configuració del tallafoc. Es recomana fer còpies de seguretat periòdiques de la configuració del tallafoc (en especial si aquest té un cert grau de complexitat).

7.4. Netfilter: nftables

NFtables es presenta com un subprojecte de Netfilter per a la següent generació de tallafocs per a Linux, amb la intenció de substituir la infraestructura actual basada en iptables i proporcionar un nou *framework* per al filtratge de paquets (inclòs en l'espai de kernel, a partir de la versió 3.13), una nova utilitat *nft* d'usuari (en espai d'usuari) i una capa de compatibilitat amb iptables.

Algunes diferències destacables amb iptables:

- Sintaxi diferent, més clara.
- Les taules i cadenes són totalment configurables per l'usuari, a diferència d'iptables, que es basa en una estructura de taules predefinida.
- Major control de la correspondència de protocols. En iptables, algunes vegades eren necessàries extensions.
- Diverses accions en una sola regla. En iptables se'n podien haver de generar unes quantes.
- Els protocols de xarxa suportats poden actualitzar-se portant a terme modificacions en les aplicacions clients, sense necessitat d'actualitzar el kernel, evitant així problemàtiques amb versions de kernel congelades en certes distribucions.

Quant als camins que segueixen els paquets, continuen tenint la mateixa estructura (de *hooks*, en terminologia Netfilter): quan un paquet arriba al sistema, passa per un *prerouting*, i accedeix després a una decisió de *routing* (és un paquet d'entrada per al sistema o de reexpedició cap a uns altres?); en cas que fem *forwarding* en el nostre sistema (perquè funciona d'encaminador, per exemple amb `echo 1 > /proc/sys/net/ipv4/ip_forward`), el paquet arribarà al *postrouting* i sortirà del sistema, depenent de les opcions de *routing*. Si estem en el cas contrari, i el paquet era per al sistema, passarà a `INPUT`, serà processat pel sistema, que pot generar paquets de sortida, `OUTPUT`, i després

d'una decisió de *routing* (és per al propi sistema o per a enviament extern?) arribarà al *postrouting* per a sortir del sistema.

Per a l'ús d'nftables, el kernel ha de disposar del mòdul *nf_tables*, que pot estar disponible en el kernel (3.13+) o pot arribar-se a construir a partir dels fonts per kernels anteriors. Així, també es necessita el mòdul de la família, per exemple *nf_tables_ipv4*. Si els tenim disponibles, podem procedir a la seva càrrega amb *modprobe*.

Una vegada disposem dels mòduls idonis, podem procedir a fer algunes funcions bàsiques (com hem comentat, no hi ha taules o cadenes [*chains*] predefinides, per tant les haurem de crear):

```
Crear taula:
# nft add table ip filter

Llistar taules:
# nft list tables ip

Llistar cadenes associades a una taula (en principi no estaran creades):
# nft list table ip filter

Esborrar una taula:
# nft delete table filter

Flush d'una taula (alliberar regles de la taula):
# nft flush table ip filter

Afegir una cadena (chain) base
(relacionada amb un Netfilter Hook comentat abans),
aquesta cadena veu els paquets de la pila TCP/IP de Linux,
exemples d'input i output (si fos necessari)
i tercer cas una cadena no base.
Amb delete o flush en lloc d'add, esborrem el flush de les regles.
# nft add chain ip filter input { type filter hook input priority 0 \; }
# nft add chain ip filter output { type filter hook output priority 0 \; }
# nft add chain ip filter test

Regles: llistar-les
(opcions -n desactiva resolució noms, -nn resolució servei),
una regla per a un port concret,
una regla a una posició prèvia concreta dins de la cadena
guardar regles d'una cadena en un fitxer
recuperar-les a la cadena
# nft list table filter
# nft add rule filter output tcp dport ssh counter
# nft add rule filter output position 8 ip daddr 127.0.0.8 drop
# nft list table filter > filter-table
# nft -f filter-table
```

Finalment estan les accions disponibles, ja sigui acceptant paquets, perdent-los, rebutjant-los, fent NAT, etc. Vegem un breu exemple d'aquestes possibilitats. Per a la protecció simple d'una màquina d'escriptori, per a ipv4 i ipv6, podria ser el següent (aquest contingut és un fitxer amb la definició de les taules i regles bàsiques, que pot recuperar-se amb `nft -f fitxer` per a carregar el *firewall*.)

Accions nftables

Per a més detalls, es recomana consultar *Possible actions on packets* en: http://wiki.nftables.org/wiki-nftables/index.php/main_page

```
# IPv4 filtering
table filter {
        chain input {
                table filter hook input priority 0;
                ct state established accept
                ct state related accept
                meta iif lo accept
                tcp dport ssh counter packets 0 bytes 0 accept
                counter packets 5 bytes 5 log drop
        }

        chain output {
                table filter hook output priority 0;
                ct state established accept
                ct state related accept
                meta oif lo accept
                ct state new counter packets 0 bytes 0 accept
        }
}
#IPv6 filtering
table ip6 filter {
        chain input {
                table filter hook input priority 0;
                ct state established accept
                ct state related accept
                meta iif lo accept
                tcp dport ssh counter packets 0 bytes 0 accept
                icmpv6 type { nd-neighbor-solicit, echo-request, \
   nd-router-advert, nd-neighbor-advert } accept
                counter packets 5 bytes 5 log drop
        }

        chain output {
                table filter hook output priority 0;
                ct state established accept
                ct state related accept
                meta oif lo accept
                ct state new counter packets 0 bytes 0 accept
        }

}
```

7.5. Consideracions

Nivells de seguretat

Mai no s'ha de confiar en un únic mecanisme o sistema de seguretat. Cal establir la seguretat del sistema amb diferents nivells.

Encara que disposem de tallafocs ben configurats, cal tenir present que no són una mesura de seguretat absoluta, ja que hi ha atacs complexos que poden saltar-se el control o falsejar dades que creïn confusió. A més, les necessitats de connectivitat modernes obliguen de vegades a crear programari que permeti la derivació o el pas (*bypass*) dels tallafocs:

- Tecnologies com IPP, protocol d'impressió utilitzat per CUPS, o el WebDAV, protocol d'autoria i actualització de llocs web, permeten passar per sobre (o cal que ho facin) de les configuracions dels tallafocs.

- Sovint s'utilitza (per exemple, els protocols anteriors i d'altres) una tècnica anomenada *tunelització* (*tunneling*), que bàsicament encapsula protocols no permesos sobre la base d'altres que sí que ho estan; per exemple, si un tallafoc permet només pas de trànsit HTTP (port 80 per defecte), és possible escriure un client i un servidor (cada un en un costat diferent del tallafoc) que parlin amb qualsevol protocol conegut per tots dos, però que a la xarxa

es transformi en un protocol HTTP estàndard, de manera que el trànsit pugui creuar el tallafoc. El *tunneling* per ssh també s'utilitza per a establir diferents tipus de túnels, per a connexions, amb els protocols i ports d'ssh.

- Els codis mòbils per a web (ActiveX, Java i JavaScript) creuen els tallafocs (via web) i, per tant, és difícil protegir els sistemes si són vulnerables als atacs contra forats descoberts.

Així, tot i que els tallafocs són una solució molt bona a la majoria d'amenaces a la seguretat, sempre poden tenir vulnerabilitats i deixar passar trànsit que es consideri vàlid, però que inclogui altres fonts possibles d'atacs o vulnerabilitats.

En seguretat mai no s'ha de considerar (ni confiar en) una única solució i esperar que ens protegeixi de manera absoluta; cal examinar els diversos problemes, plantejar solucions que detectin els perills a temps i establir polítiques de prevenció que ens curin en salut, pel que pugui passar.

8. Anàlisi de registres

Si observem els fitxers de registre (*logs*) del sistema [Ray01, Fri02], podem fernos una idea ràpida de l'estat global del sistema i dels últims esdeveniments produïts, i detectar accessos (o intents d'accés) indeguts. Però també cal tenir present que els registres, si s'ha produït una intrusió real, poden haver estat netejats o falsejats. La majoria d'arxius de registre són al directori `/var/log`.

Molts dels serveis poden tenir registres propis, que normalment s'estableixen en la configuració (mitjançant el corresponent fitxer de configuració). La majoria solen utilitzar les facilitats de registre incorporades en el sistema Syslog, mitjançant el dimoni Syslogd. La seva configuració és en el fitxer `/etc/syslog.conf`. Aquesta configuració sol establir-se per nivells de missatges: existeixen diferents tipus de missatges segons la importància que tenen. Normalment hi ha els nivells `debug`, `info`, `err`, `notice`, `warning`, `crit`, `alert` i `emerg`, ordenats més o menys en funció de la importància dels missatges (de més baixa a més alta). Normalment, la majoria dels missatges es dirigeixen cap al registre `/var/log/messages`, però pot definir-se que cada tipus es distribueixi cap a fitxers diferents i també es pot identificar qui els ha originat: habitualment el nucli, el correu, missatges (*news*), el sistema d'autenticació, etc.

En conseqüència, cal examinar (i en tot cas adaptar) la configuració de Syslog per a saber en quins registres podem trobar o generar la informació. Un altre punt important és controlar-ne el creixement, ja que segons els que estiguin actius i les operacions (i serveis) que s'executin en el sistema, els registres poden créixer bastant. En Debian i Fedora es controla per mitjà de `logrotated`, un dimoni que s'encarrega periòdicament de fer còpies dels registres més antics i comprimir-les; se'n pot trobar la configuració general a `/etc/logrotate.conf`, però algunes aplicacions fan configuracions específiques que es poden trobar en el directori `/etc/logrotate.d`.

En els punts següents comentem alguns fitxers de registre que caldria tenir en compte (potser els més utilitzats):

Fitxers de registre

Cal tenir en compte que en la majoria d'atacs sofisticats reeixits es falsegen o s'esborren les dades dels fitxers de registre. Tot i així, depenent de la tècnica usada, es pot detectar informació útil sobre l'atac.

1) `/var/log/messages`: és el fitxer de registre per defecte del dimoni Syslogd, però caldria revisar-ne la configuració, no fos cas que s'hagués canviat en una altra banda o que n'hi hagués més d'un. Aquest fitxer conté una gran varietat de missatges de diversos orígens (arrencada, diferents dimonis, serveis o el nucli mateix); tot allò que resulti anòmal s'hauria de verificar. En cas que s'hagi produït una intrusió, s'haurà de mirar al voltant de les dates de la intrusió en un determinat interval de temps, per a detectar possibles intents previs, la metodologia usada, primers avisos del sistema, etc.

2) `/var/log/utmp`: aquest fitxer conté informació binària per a cada usuari que és actiu actualment. És interessant per a determinar qui és dins del sistema. L'ordre `who` utilitza aquest fitxer per a proporcionar aquesta informació.

3) `/var/log/wtmp`: cada vegada que un usuari entra en el sistema, en surt o es reinicia la màquina, es desa una entrada en aquest fitxer. És un fitxer binari del qual l'ordre `last` obté informació en què s'esmenta quins usuaris han entrat en el sistema o n'han sortit, quan s'ha originat la connexió i on. Pot ser útil per a buscar on (en quins comptes) s'ha originat una intrusió o per a detectar usos de comptes sospitosos. També hi ha una variant de l'ordre, anomenada `lastb`, que genera una llista dels intents d'inici de sessió que no s'han pogut validar correctament i es fa servir el fitxer `/var/log/btmp` (pot ser necessari crear-lo si no existeix). Aquests mateixos errors d'autenticació també solen enviar-se al registre `auth.log`. De manera semblant, l'ordre `lastlog` utilitza un altre fitxer `/var/log/lastlog` per a verificar quina va ser la darrera connexió de cada un dels usuaris.

4) `/var/log/secure`: sol utilitzar-se en Fedora per a enviar els missatges dels *tcp wrappers* (o dels tallafocs). Cada vegada que s'estableix una connexió a un servei d'inetd, o bé per a xinetd (amb la seva pròpia seguretat), s'afegeix un missatge de registre a aquest fitxer. Poden buscar-se intents d'intrusió en serveis que no s'utilitzen habitualment, o bé màquines no familiars que s'intenten connectar.

En el sistema de registres, una altra cosa que caldria garantir és que només l'usuari *root* (o dimonis associats a serveis) pugui escriure en el directori de registres `/var/log`. En cas contrari, qualsevol atacant podria falsificar la informació dels registres. Tot i així, si l'atacant aconsegueix accés a *root*, pot esborrar les pistes dels seus accessos (i acostuma a fer-ho).

9. Taller: anàlisi de la seguretat mitjançant eines

A continuació seguirem alguns dels processos descrits en els apartats previs sobre una distribució Debian (encara que es poden seguir de manera equivalent en una Fedora / Red Hat, la majoria de processos i mètodes són equivalents) per a configurar i auditar la seva seguretat (observarem diferents sortides de les ordres, depenent de la versió i sistema físic que utilitzem).

Podem començar amb algunes comprovacions locals. No entrem en els detalls, ja que moltes d'aquestes s'han comentat prèviament, i en alguns casos són procediments típics d'administració local (segons examinem en el material d'*Introducció a l'administració*):

- Comprovar que no hi hagi usuaris o grups problemàtics. Podem analitzar `/etc/passwd` i `/etc/group` cercant usuaris o grups que no corresponguin a usuaris de sistema, o siguin usuaris de sistema o associats a un determinat servei. En aquest últim cas, també és una bona manera de detectar serveis que no teníem identificats com en funcionament, i que possiblement procedeixin d'una instal·lació prèvia sense configuració o que han deixat de ser útils. Seria un bon moment per a desinstal·lar aquests serveis, si se sap que no s'utilitzaran, ja que probablement estiguin deficientment configurats en temes de seguretat.

- Comprovar que no existeixin usuaris amb *passwords* buits. Recordeu que si està funcionant `/etc/shadow`, el segon paràmetre en `/etc/passwd` hauria de ser `x`. Si simplement està buit: `::` o en `/etc/shadow`, si està buit sense disposar de *hash* (el camp amb ! o * tampoc permet *login*). En aquestes dues últimes comprovacions també seria útil aprofitar l'ocasió per a inhabilitar comptes d'usuari que hàgim detectat com a no utilitzats, a més de verificar els últims comptes d'usuari afegits, per si detectem alguna incoherència amb la nostra activitat prèvia de gestió d'usuaris.

- Verificar també l'existència d'usuaris i grups amb Id 0 (associats a usuaris amb permisos de *root*).

- Verificar configuració de PAM, per a comprovar que no estigui alterada, especialment en aquells mòduls relacionats amb el procés de *login* i/o restriccions de *passwords* que puguem haver imposat.

- Verificar, amb ordres com `lastlog`, les últimes connexions dels usuaris per a detectar els últims usos interactius i des de quins llocs s'han fet.

- Comprovació de missatges de Syslog, Rsyslog o Systemd (journalctl), per a la detecció d'errors de sistema i autenticació.

- Verificació de l'existència d'arxius `Sticky Bit` i `suid/guid`. Podem comprovar que no s'hagin generat en el sistema arxius o directoris amb aquests permisos.

- Comprovació de signatures de binaris (i altres fitxers). Per exemple, si disposem de sumes prèvies, amb eines tipus tripware o AIDE.

- Verificació dels processos en execució. ordres com `ps`, `top`.

- Configuració de sudoers `/etc/sudoers`: quins usuaris disposen de permisos especials.

- Comprovació de treballs de cron: `/etc/cron.daily`, i altres períodes per a comprovar que no hagin estat introduïdes tasques no esperades.

- Comprovar amb el sistema de paquets si es disposa d'actualitzacions noves en el sistema, especialment correccions de seguretat.

Quant a comprovacions per a la seguretat en xarxa, detallarem alguns passos amb major detall, però una sèrie de tasques típiques en l'auditació i configuració de seguretat serien:

- Verificar configuració de xarxa (*ifconfig*), rutes disponibles *route*.

- Verificació de *runlevel* actiu (*runlevel*), estat `/etc/inittab` o *target* equivalent en arrencada Systemd.

- Verificació de l'estat dels serveis de xarxa. Quins? Estat?

- Verificació de les connexions de xarxa. ordres `netstat`.

- Comprovar la configuració d'SSH (`/etc/ssh/sshd_config`) per a les opcions habituals de no *login* per *root*, SSH 2 activat, entre d'altres.

- Verificació de l'estat del *firewall* (iptables), comprovar regles, segueix actiu?

- Verificació d'alguns serveis problemàtics: Samba (està actiu i el necessitem?). Serveis d'NTP (ens permeten disposar del sistema sincronitzat).

- Verificació dels *logs* específics d'alguns servidors, normalment disponibles en `/var/log/servei`, tant si és un fitxer com un directori amb els *logs* d'accés i error pertinents, per exemple, de servidor web o bases de dades. Les eines comentades, com *logcheck* o *logwatch*, ens permeten també una millor detecció a través dels seus resums obtinguts dels *logs* dels serveis i del general del sistema.

Examinem ara amb més detall algunes d'aquestes fases per controlar i auditar la seguretat del nostre sistema. En aquest cas, ho fem a partir d'un sistema exemple amb una distribució Debian (en particular, una versió antiga que tenia alguns problemes de seguretat en la configuració per defecte).

Primer examinarem què ofereix, quant a serveis, la nostra màquina a la xarxa. Per a això, utilitzarem l'eina *nmap* com a escanejador de ports. Amb l'ordre (des de *root*):

```
nmap -sTU -O localhost
```

obtenim:

```
root@maquina:# nmap -sUT -O localhost
starting nmap 5.21 ( nmap.org ) 11:31 CEST
Interesting ports on localhost (127.0.0.1):

(The 3079 ports scanned but not shown below are in state: closed)
Port  State Service
9/tcp   open discard
9/udp   open discard
13/tcp  open daytime
22/tcp  open ssh
25/tcp  open smtp
37/tcp  open time
37/udp  open time
80/tcp  open http
111/tcp open sunrpc
111/udp open sunrpc
113/tcp open auth
631/tcp open ipp
728/udp open unknown
731/udp open netviewdm3
734/tcp open unknown

Remote operating system guess: Linux kernel 2.6.X
Uptime 2.011 days
Nmap run completed 1 IP address (1 host up) scanned in 9.404 seconds
```

Podem observar que ens ha detectat un gran nombre de serveis oberts (depenent de la màquina, podria haver-n'hi més: Telnet, FTP, finger, etc.), tant en protocols tcp com udp. Alguns serveis, com *discard*, *daytime* o *time*, poden ser útils en alguna ocasió, però normalment no haurien d'estar oberts a xarxa, ja que es consideren insegurs. SMTP és el servei de reexpedició i encaminament del correu electrònic; si actuem com a *host* o servidor de correu, hauria d'estar actiu, però si només llegim i escrivim correu mitjançant comptes POP3 o IMAP, no té per què estar-ho.

Una altra manera de detectar serveis actius seria mitjançant la cerca de ports actius a les escoltes, detectant les connexions de xarxa actives; això pot fer-se amb l'ordre `netstat -lut`.

L'ordre `nmap` també pot aplicar-se amb el nom DNS o IP de la màquina; així veiem el que es veuria des de l'exterior (amb *localhost* veiem el que pot

veure la mateixa màquina) o, millor fins i tot, podríem utilitzar una màquina d'una xarxa externa (per exemple, un PC qualsevol connectat a Internet) per a examinar el que veurien de la nostra màquina des de l'exterior. En els primers casos, no estarem tallant les connexions pel tallafoc si aquest existís. Només des de l'exterior podrem comprovar si aquest efectivament tanca els ports esperats.

Anem ara a `/etc/inetd.conf` per desactivar aquests serveis (si han aparegut o no en l'escaneig previ dependrà de la distribució GNU/Linux i de la configuració prèvia d'aquests serveis, o pot ser que `inetd` ja no estigui actiu en les últimes distribucions). Busquem línies com:

```
discard stream tcp nowait root internal
smtp stream tcp nowait mail /usr/sbin/exim exim -bs
```

I els col·loquem una `#` al principi (només per a aquells serveis que vulguem desactivar i sapiguem què fan realment -consulteu les pàgines man- o es recomani la seva desactivació). Un altre cas de desactivació especialment recomanada seria la dels serveis d'FTP, Telnet, finger, etc. i fer servir `ssh` per a substituir-los.

Ara hem de reiniciar `inetd` perquè torni a llegir la configuració que hem canviat: `/etc/init.d/inetd restart`.

Tornem a `nmap`:

```
22/tcp  open ssh
80/tcp  open http
111/tcp open sunrpc
111/udp open sunrpc
113/tcp open auth
631/tcp open ipp
728/udp open unknown
734/tcp open unknown
```

D'allò que ens queda, tenim el servei ssh, que volem deixar actiu, i el servidor de web, que l'aturarem de moment:

```
/etc/init.d/apache2 stop
```

ipp és el servei d'impressió associat a CUPS. En administració local, vam veure que hi havia una interfície web de CUPS que es connectava al port 631. Si volem tenir una idea de a què es dedica un port determinat, podem mirar en `/etc/services`:

```
root@maquina:# grep 631 /etc/services
ipp 631/tcp # Internet Printing Protocol
ipp 631/udp # Internet Printing Protocol
```

Si no estem actuant com a servidor d'impressió cap a l'exterior, hem d'anar a la configuració de CUPS i eliminar aquesta prestació (per exemple, col·locant un *listen* 127.0.0.1:631, perquè només escolti la màquina local) o limitar l'accés a les màquines permeses.

Ens apareixen també alguns ports com a desconeguts, en aquest cas el 728 i 734; això indica que nmap no ha pogut determinar quin servei està associat al port. Intentarem comprovar-ho directament. Per a això, executem sobre el sistema l'ordre `netstat`, que ofereix diferents estadístiques del sistema de xarxa, des de paquets enviats i rebuts, i errors, fins a allò que ens interessa, que són les connexions obertes i qui les fa servir. Intentem buscar qui està usant els ports desconeguts:

```
root@maquina: # netstat -anp | grep 728
udp 0 0 0.0.0.0:728 0.0.0.0:* 552/rpc.statd
```

I si fem el mateix amb el 734, observem també que qui ha obert el port ha estat rpc.statd, que és un *daemon* associat a NFS (en aquest cas el sistema té un servidor NFS). Si fem aquest mateix procés amb els ports 111 que apareixien com a `sunrpc`, observarem que el *daemon* que hi ha darrere és portmap, que s'usa en el sistema de crides RPC. El sistema de crides RPC (*remote procedure call*) permet utilitzar el mecanisme de crides remotes entre dos processos que estan en diferents màquines. portmap és un *daemon* que s'encarrega de traduir les crides que li arriben pel port als números de serveis RPC interns que es tinguin, i és utilitzat per diferents servidors, com ara NFS, NIS i NIS+.

Els serveis RPC que s'ofereixen es poden veure amb l'ordre `rpcinfo`:

```
root@maquina:# rpcinfo -p
programa  vers proto  port
100000  2 tcp 111  portmapper
100000  2 udp 111  portmapper
100024  1 udp 731  status
100024  1 tcp 734  status
391002  1 tcp 39797 sgi_fam
391002  2 tcp 39797 sgi_fam
```

on observem els serveis RPC amb alguns dels ports que ja s'havien detectat. Una altra ordre que pot resultar útil és `lsof`, que, entre altres funcions, permet relacionar ports amb els serveis que els han obert (per exemple: `lsof -i | grep 731`).

El *daemon* portmap és una mica crític amb la seguretat, ja que, en principi, no ofereix mecanismes d'autenticació del client, ja que se suposa que es deleguen en el servei (NFS, NIS, etc.). Per tant, portmap podria ser víctima d'intents de DOS/DDoS que podrien provocar errors en els serveis o fer-los caure. Normalment, protegirem portmap per mitjà d'algun tipus de *wrapper* i/o tallafoc. Si no utilitzem, i no tenim previst utilitzar, serveis NFS i NIS, el millor és desactivar completament portmap, traient-lo del *runlevel* en què s'activi. També podem parar-los momentàniament amb els *scripts* (en Debian):

```
/etc/init.d/nfs-common
/etc/init.d/nfs-kernel-server
/etc/init.d/portmap
```

I donar-los el paràmetre `stop` per a parar els serveis RPC (en aquest cas NFS). En tots aquests casos, només estem parant els serveis per a l'execució actual; si volem desactivar realment aquests serveis, hem d'anar al nivell d'execució concret (*runlevel*) i desactivar els serveis (com s'ha comentat en el mòdul), perquè si no, es reiniciaran en la següent arrencada del sistema.

A continuació, controlarem la seguretat sobre la base d'un *wrapper* senzill. Suposem que volem deixar pas a través d'SSH d'una màquina determinada, que denominem 1.2.3.4 (adreça IP). Tanquem portmap a l'exterior, ja que no tenim NIS, i de NFS tenim servidor però no estem servint res (podríem tancar-lo, però el deixarem per a futurs usos). Fem un *wrapper* (suposem que els *TCP wrappers* estan ja instal·lats, poden ser necessaris *tcpd i libwrap*), modificant els fitxers `hosts.deny` i `hosts.allow`. En `/etc/hosts.deny`:

```
ALL : ALL : spawn (/usr/sbin/safe_finger -l @%h \
| /usr/bin/mail -s "%c FAILED ACCESS TO %d!!" root) &
```

Estem denegant tots els serveis (compte, aquells relacionats amb inetd) (primer `all`) a tots (`all`), i l'acció a prendre serà esbrinar qui ha demanat el servei (només funcionarà si la màquina suporta *finger*) i a quina màquina, i enviarem un correu al *root* que informi de l'intent. Podríem també escriure un fitxer de registre. Ara, en `/etc/hosts.allow`:

```
sshd: 1.2.3.4
```

habilitem l'accés per a la màquina IP 1.2.3.4 en el servidor sshd (de l'ssh). Podem col·locar una llista de màquines o bé subxarxes que puguin utilitzar el servei (vegeu `man hosts.allow`). Recordeu que també tenim les ordres `tcpdchk`, per a comprovar que la configuració del *wrapper* sigui correcta, i

`tcpdmatch`, per a simular què passaria amb un determinat intent. Per exemple:

```
root@maquina:# tcpdmatch sshd 1.2.3.4

warning: sshd: no such process name in
   /etc/inetd.conf client: hostname maquina.domini.es
client: address 1.2.3.4
server: process sshd
matched: /etc/hosts.allow line 13
access: granted
```

Ens comenta que seria concedit l'accés. Un detall és que ens explicita que sshd no està en el `inetd.conf` i, si ho verifiquem, veiem que així és: no s'activa pel servidor inetd, sinó per *daemon* propi (sshd) en el *runlevel* en què estiguem. A més (en Debian), aquest és un cas d'un dimoni que està compilat amb les biblioteques de *wrappers* incloses. En Debian hi ha diversos *daemons* que tenen aquest suport de *wrappers* en compilar-se amb les biblioteques pertinents com rpc.statd, rpc.mountd, rpcbind i sshd, entre d'altres. Això permet assegurar aquests *daemons* mitjançant *wrappers* pels fitxers *hosts* esmentats.

Una altra qüestió a verificar són les connexions actuals existents. Amb l'ordre `netstat -utp`, podem llistar les connexions tcp o udp establertes amb l'exterior, ja siguin entrants o sortints; així, en qualsevol moment podem detectar els clients connectats i a qui estem connectats. Una altra ordre important (de múltiples funcions) és `lsof`, que pot relacionar fitxers oberts amb processos o connexions per xarxa establerts mitjançant `lsof -i`, i així es poden detectar accessos indeguts a fitxers.

També podríem utilitzar un tallafoc per a processos similars (o bé com a mecanisme afegit). Començarem veient com estan les regles del tallafoc en aquest moment (ordre `iptables -L`):

```
root@aopcjj:# iptables -L
Chain INPUT (policy ACCEPT)
target prot opt source destination
Chain FORWARD (policy ACCEPT)
target prot opt source destination
Chain OUTPUT (policy ACCEPT)
target prot opt source destination
```

És a dir, que el tallafoc no està col·locant cap restricció en aquest moment, i permet l'entrada, sortida i reexpedició de tots els paquets.

En aquest punt, podríem afegir un tallafoc que ens permetés una gestió més adequada de paquets que rebem i enviem, i que seria un control previ per a millorar la seguretat. Depenent de les nostres necessitats, establiríem les regles necessàries de manera semblant a aquelles que comentem en els exemples de tallafocs de la unitat.

En cas de col·locar actiu algun tallafoc, podem considerar si fer servir aquest mecanisme com a única garantia i treure els *wrappers*: podria fer-se, ja que els tallafocs (en aquest cas mitjançant iptables) ofereixen un mecanisme molt potent que ens permet seguir un paquet per tipus, per protocol i pel que està fent en el sistema. Un bon tallafoc podria ser més o menys suficient, però, per si de cas, més mesures de seguretat no vénen malament. I en cas que el tallafoc no estigués ben dissenyat i deixés escapar alguns paquets o *hosts*, el *wrapper* seria la mesura, en un nivell de servei, per a aturar els accessos no desitjats. Per a posar una metàfora que se sol utilitzar, si ens plantegéssim el nostre sistema com la defensa d'un castell medieval, el fossat i primeres muralles serien el tallafoc, i la segona muralla de contenció, els *wrappers*.

Les següents mesures ja podrien venir directament de cada servei (web, correu, impressió, etc.), amb les opcions de seguretat que ofereixi de manera addicional, mitjançant autenticació dels seus clients, mitjançant limitació dels accessos per perfils o ACL o simplement en oferir un subconjunt de funcionalitats requerides. Veurem en diferents mòduls orientats a serveis concrets algunes d'aquestes mesures en el nivell servei.

Com a opcions addicionals per a protegir el nostre sistema, podríem fer intervenir a algunes de les eines vistes en apartats anteriors, per exemple, per a mitjançant *Denyhosts* i *Fail2ban*, amb els procediments vistos en els subapartats 6.2. i 6.3., protegir les connexions SSH, i en el segon cas altres serveis com servidor web, correu, Samba, o altres que tinguem actius. Com a següent pas, podríem col·locar una eina de detecció de vulnerabilitats, com OpenVAS (vegeu el subapartat 6.1.), per a obtenir un informe de l'estat final del nostre servidor.

Resum

En aquest mòdul hem examinat els conceptes bàsics de seguretat aplicables als sistemes GNU/Linux, i hem identificat els possibles tipus d'atacs, tant locals com en sistemes en xarxa.

El coneixement del procés dels atacs ens permet dur a terme accions de seguretat activa mitjançant eines de detecció d'intrusions, i també de prevenció de possibles situacions problemàtiques.

Sistemes com SELinux ens permeten polítiques de seguretat, altament especificades, i ens ofereixen una àmplia sintaxi de permisos, controls i prevenció activa de la seguretat del sistema.

Cal examinar la seguretat tant des del punt de vista local del sistema, la qual cosa inclou la seguretat pel que fa a l'accés físic, com des del punt de vista dels sistemes en xarxa.

L'ús d'eines de seguretat en les diferents àrees de prevenció, detecció i actuació permet un control actiu de seguretat que ens pot evitar problemes més importants en els nostres sistemes.

També observem les limitacions de la seguretat dels sistemes informàtics i, en especial, les possibles sensacions d'una falsa seguretat total (difícil o impossible d'obtenir) que ens pot dur a una confiança cega, amb resultats pitjors que en el cas de no tenir-ne. La seguretat és un procés actiu que necessita un seguiment constant i participatiu per part de l'administrador de sistemes.

Activitats

1. Suposem que col·loquem un lloc web a la nostra màquina, per exemple amb Apache. El nostre lloc està pensat per a deu usuaris interns, però no controlem aquest nombre. Més endavant ens plantegem posar aquest sistema a Internet, ja que creiem que pot ser útil per als clients, i l'únic que fem és posar el sistema amb una IP pública a Internet. Quin tipus d'atacs podria patir aquest sistema?

2. Com podem detectar els fitxers amb SUID en el nostre sistema? Quines ordres caldran? I els directoris amb SUID o SGID? Per què cal, per exemple, que `/usr/bin/passwd` tingui bit de SUID?

3. Els fitxers `.rhosts`, tal com hem vist, són un perill important per a la seguretat. Podríem utilitzar algun mètode automàtic que en comprovés periòdicament l'existència? Com?

4. Com crearíem un ambient chroot per a `/bin/bash`? Descriviu els passos necessaris.

5. Suposem que volem deshabilitar un servei del qual sabem segur que el controla l'*script* `/etc/init.d/servei`: volem desactivar-lo en tots els nivells d'execució en què es presenta. Com trobem aquests nivells d'execució?, (per exemple, buscant enllaços a l'*script*). Com ho faríem en distribucions que suporten Systemd?

6. Examineu els serveis en actiu de la vostra màquina. Són tots necessaris? Com caldria protegir-los o desactivar-los?

7. Practiqueu l'ús d'algunes de les eines de seguretat que hem descrit (nmap, chkrootkit, wireshark, OpenVas, etc.).

8. Quines regles iptables serien necessàries per a una màquina amb la qual només vulguem accés per SSH des d'unes màquines habituals concretes?

9. I si volem només un accés al servidor web?

Bibliografia

[Aus] **CERT Australia**. *Australian CERT*.
<http://www.auscert.org.au>

[Bur02] **Burgiss, H.** (2002). *Security QuickStart HOWTO for Linux*. The Linux Documentation Project.

[Cera] **CERT**. *CERT site*.
<http://www.cert.org>

[Cerb] **CERT**. *CERT Vulnerability Database*.
<http://www.kb.cert.org/vuls>

[Deb] **Debian**. *Lloc de seguretat de Debian*.
<http://www.debian.org/security>

[Fbi] **Federal Bureau of Intelligence**. *Brigada del FBI para ciberdelitos*.
<http://www.fbi.gov/about-us/investigate/cyber/cyber>

[Fen02] **Kevin Fenzi**. *Linux security HOWTO*. The Linux Documentation Project.

[Fri02] **Frisch, A.** (2002). *Essential System Administration* (3a. ed.). O'Reilly.

[Gre] **Grennan, M.** *Firewall and Proxy Server HOWTO*. The Linux Documentation Project.

[Hat08] **Hatch, B.** (2008). *Hacking Linux Exposed* (3a. ed.) McGraw-Hill.

[Hatb] **Red Hat**. *Red Hat RHEL 7 Security Guide. 2014*.
<https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux
/7/html/Security_Guide/index.html>

[Hatc] **Red Hat**. *Lloc de seguretat de Red Hat, bases de dades de vulnerabilitats*.
<https://access.redhat.com/security/cve/>

[Hatd] **Red Hat**. *Utilització de signatures GPG en Red Hat*.
<https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux
/7/html-single/System_Administrators_Guide/#s1-check-rpm-sig>

[Her13] **Hertzog, R; and Mas, R.** (2013). *El libro del administrador de Debian*.
<http://debian-handbook.info/browse/es-ES/stable/index.html>

[Him01] **Himanen, P.** (2001). *La ética del hacker y el espíritu de la era de la información*. Destino.

[Incb] **Internet Storm Center** (2014). *Analisis de Vulnerabilidades, y algunos incidentes*.
<https://isc.sans.edu//>

[Ins] **Insecure.org** (1998). *Vulnerabilitats i exploits*.
<http://www.insecure.org/sploits.html>

[Insa] **Insecure.org** (2014). *Insecure.org site*.
<http://www.insecure.org>

[Insb] **Insecure.org** (2003). *Nmap Home site*.
<http://nmap.org/index.html>

[Line] **Linuxsecurity.com** (2002). *Linux Security Reference Card*.
<http://www.linuxsecurity.com/docs/QuickRefCard.pdf>

[Mor03] **Morill, D.** (2003). *Configuración de sistemas Linux*. Anaya Multimedia.

[Mou02] **Mourani, G.** (2002). *Securing and Optimizing Linux: The Ultimate Solution. v2.0*. The Linux Documentation Project.

[Nes] **Nessus.org**. *Nessus*.
<http://www.nessus.org>

[Net] **Netfilter.org.** *Projecte Netfilter/IPtables*.
<http://www.netfilter.org>

[Neu] **Neufeld, C.** *Setting Up Your New Domain MiniHOWTO*. The Linux Documentation Project.

[Nsaa] **NSA**. *NIST site*.
<http://csrc.nist.gov>

[Nsab] **NSA** (2009). *Security Enhanced Linux*.
<http://www.nsa.gov/research/selinux/index.shtml>

[Opv] **OpenVas**. *OpenVas vulnerability Scanner.*
<http://openvas.org>

[Pen] **Fernández-Sanguino Peña, J.** (2013). *Securing Debian Manual*.
<http://www.debian.org/doc/manuals/securing-debian-howto/>

[Ray01] **Ray, John** (2011). *Maximum Linux Security: A Hacker's Guide to Protecting. 2nd Edition, Sams*.

[San] **Sans** (2014). *Top20 de controls crítics de seguretat*.
<http://www.sans.org/critical-security-controls/>

[Sei] **Seifried, K.** (2002). *Securing Linux, Step by Step*.
<http://seifried.org/security/os/linux/20020324-securing-linux-step-by-step.html>

[Sno] **Snort.org**. *Snort*.
<http://www.snort.org>

[Usa] **The United States Department of Justice**. *Divisió del Departament de Justícia dels Estats Units per al ciberdelicte*.
<http://www.usdoj.gov/criminal/cybercrime/>

Sobre aquestes fonts de referència i informació

[Deb] [Hatc] Alguns llocs de seguretat de les distribucions.

[Pen] Imprescindible per a Debian, molt bona opció per a seguir pas a pas la configuració de seguretat; [Hatb] seria equivalent per a Fedora / Red Hat.

[Mou02] Excel·lent referència de seguretat per a Red Hat (aplicable també a Debian).

[Hat08] Llibres sobre seguretat en GNU/Linux, que cobreixen un gran nombre d'aspectes i tècniques.

[Line] Petita guia (2 pàgines) de seguretat.

[Sei] Guia pas a pas d'identificació dels punts clau que cal verificar i els problemes que puguin sorgir.

[Net] Projecte Netfilter/iptables i nftables.

[Ian] Una llista de ports TCP/IP.

[Proa] [Sno] [Insb] [Nes] Algunes de les eines de seguretat més utilitzades.

[NSAb] Versió de Linux per a la seguretat, produïda per l'NSA. Referència per a SELinux.

[CERa][Aus][Insa][Incb] [NSAa] Llocs d'organismes de seguretat.

[CERb][Ins][San] Vulnerabilitats i *exploits* dels diversos sistemes operatius.

[NSAa][FBI][USA] Alguns organismes governamentals de ciberdelictes als Estats Units.