Administració de seguretat

Josep Jorba Esteve

P07/M2003/02288

UOC
www.uoc.edu

Índex

Introducció 5

1. Tipus i mètodes dels atacs 7
1.1. Tècniques utilitzades en els atacs 10
1.2. Contramesures 16

2. Seguretat del sistema 21

3. Seguretat local 22
3.1. *Bootloaders* 22
3.2. Contrasenyes i *shadows* 23
3.3. Suid i sticky bits 24
3.4. Habilitació de *hosts* 25
3.5. Mòduls PAM 26
3.6. Alteracions del sistema 27

4. SELinux 29
4.1. Arquitectura 32
4.2. Crítica 35

5. Seguretat en xarxa 36
5.1. Client de serveis..... 36
5.2. Servidor: inetd i xinetd 36

6. Detecció d'intrusions 39

7. Protecció mitjançant filtratge (*wrappers* i tallafocs) 40
7.1. Tallafocs 41
7.2. Netfilter: IPtables 42
7.3. Paquets de tallafocs en les distribucions 45
7.4. Consideracions finals 46

8. Eines de seguretat 48

9. Anàlisi de *logs* 51

10. Taller: anàlisi de la seguretat mitjançant eines 53

Activitats 59

Altres fonts de referència i informació 59

Introducció

El salt tecnològic dels sistemes d'escriptori aïllats, fins als sistemes actuals integrats a xarxes locals i Internet, ha comportat 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. Com l'anàlisi de factors "psicològics", respecte del comportament dels usuaris del sistema o les possibles intencions dels atacants.

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

Els possibles atacs poden canviar el comportament dels sistemes, fins i tot ferlos caure (inutilitzar-los), o donar una falsa impressió de seguretat, que pot ser difícilment perceptible. Ens podem trobar 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 (substituir programes, màquines o usuaris per altres, sense que es notin els canvis).

Nota

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

Una idea clara que cal tenir en ment és que és impossible poder aconseguir una seguretat al 100%.

Les tècniques de seguretat són una arma de doble tall, que fàcilment ens poden donar una impressió falsa de control del problema. La seguretat actual és un problema ampli, complex, i el que és més important, dinàmic. Mai no podem esperar o dir que la seguretat està garantida, sinó que amb força probabilitat 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 aquesta unitat examinarem alguns tipus d'atacs amb els quals ens podem trobar, com podem verificar i prevenir parts de la seguretat local, i també en entorns de xarxa. Així mateix, examinarem tècniques de detecció d'intrusos i algunes eines bàsiques que ens poden ajudar en el control de la seguretat.

També cal esmentar que en aquesta unitat només podem fer una mera 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, així com els manuals associats als productes o eines comentats.

1. Tipus i mètodes dels atacs

La seguretat computacional en administració pot ser entesa com el procés que ha de permetre que l'administrador del sistema en previngui i en detecti usos no autoritzats. Les mesures de prevenció ajuden a parar 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 s'ha trencat.

Els intrusos (també coneguts col·loquialment com a *hackers, crackers,* "atacants" o "pirates"...) normalment volen obtenir control sobre el sistema, ja sigui per a causar un mal funcionament, corrompre el sistema o les seves dades, guanyar recursos a la màquina, o simplement utilitzar-lo per a llançar atacs contra altres sistemes, i així protegir la seva verdadera identitat i ocultar l'origen real dels atacs. També hi ha la possibilitat d'examinar la informació (o robar-la) del sistema, l'espionatge pur de les accions del sistema o causar danys físics a la màquina, ja sigui formatar el disc, canviar dades, esborrar o modificar programari crític, etc.

Respecte als intrusos, cal establir algunes diferències, que no solen ser gaire clares en els termes col·loquials. Normalment ens referim a *hacker* [Him01] com aquella persona amb grans coneixements en informàtica, més o menys apassionada pels temes de programació i seguretat informàtica, i que, normalment sense finalitat malèvola, utilitza els seus coneixements per a protegir-se a si mateix, o a tercers, introduir-se en xarxes per tal de demostrar els seus errors de seguretat, i, en alguns casos, com a reconeixement de les seves habilitats.

Un exemple seria la pròpia comunitat GNU/Linux, que deu molt als seus *hackers*, ja que cal entendre el terme *hacker* com a expert en uns temes (més que com a intrús en la seguretat).

D'altra banda, trobaríem els *crackers*. Aquí és quan s'utilitza el terme de manera més o menys despectiva, cap a aquelles persones que utilitzen les seves habilitats per a corrompre (o destrossar) sistemes, ja sigui només per fama pròpia, per motius econòmics, per ganes de causar danys o simplement per a molestar; per motius d'espionatge tecnològic, actes de ciberterrorisme, etc. Així mateix, es parla de *hacking* o *cracking*, quan ens referim a tècniques d'estudi, detecció i protecció de la seguretat, o al contrari, a tècniques destinades a causar danys trencant la seguretat dels sistemes.

Desafortunadament, obtenir accés a un sistema (ja sigui desprotegit o parcialment segur) és bastant més fàcil del que sembla. Els intrusos descobreixen per-

manentment noves vulnerabilitats (anomenades de vegades "forats", o *exploits*), que els permeten introduir-se a les diferents capes de programari. La complexitat cada vegada més gran del programari (i del maquinari), fa que augmenti la dificultat per a testejar de manera raonable la seguretat dels sistemes informàtics. L'ús habitual dels sistemes GNU/Linux en xarxa, ja sigui la pròpia Internet o en xarxes pròpies amb tecnologia TCP/IP com les intranets, ens porta 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 nivell de seguretat que ens faci sentir segurs dins de certs paràmetres. Però com a tal, només és una percepció de seguretat, i com totes les percepcions, pot ser falsa i d'això ens podem adonar en l'últim moment, quan ja tinguem els nostres sistemes afectats. 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 més que això, una sensació. Per tant, no s'ha de descuidar cap de les polítiques implementades i cal mantenir-les al dia, així com els nostres coneixements del tema.

Els possibles atacs són una amenaça constant als nostres sistemes i poden comprometre el seu funcionament, així com les dades que manegem; abans que res, sempre hem de definir una certa política de requisits de seguretat sobre els nostres sistemes i dades. Les amenaces que podem sofrir podrien afectar els aspectes següents:

Nota

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

- **Confidencialitat**: la informació ha de ser accessible només a aquelles persones que estiguin autoritzades; estem responent a la pregunta: Qui hi podrà accedir?

- **Integritat**: la informació només podrà ser modificada per aquelles persones que estiguin autoritzades: Què se'n podrà fer?

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

Passem a esmentar una certa 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 de què no disposava al principi.

- **Intercepció** (o escolta): mecanisme pel qual s'intercepten dades, per part de tercers, quan aquestes no estaven dirigides a ells.

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

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

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

Els mètodes utilitzats i les tècniques precises poden variar molt (és 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.

Per a cadascun d'aquests tipus d'atacs, normalment es poden utilitzar un o més mètodes d'atac, que al seu torn poden provocar també un o més dels tipus d'atacs.

Respecte a on es produeix l'atac, hem de tenir clar què es pot fe o quin serà l'objectiu dels mètodes:

Nota

Els atacs poden tenir finalitats destructives, inhabilitadores o d'espionatge, dels nostres components, ja sigui maquinari, programari o sistemes de comunicació.

- **Maquinari**: referent 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 controls de seguretat per a accedir als locals on siguin les màquines per a evitar problemes de robatori o ruptura de l'equip a fi d'eliminar el seu servei. També es pot comprometre la confidencialitat i la integritat si l'accés físic a les màquines permet utilitzar alguns dels seus dispositius com les disqueteres, o l'arrencada de les màquines, o l'accés a comptes d'usuari que podrien estar obertes.

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

- **Dades**: ja siguin estructurades, com en els serveis de base de dades, o gestió de versions (com cvs), o simples arxius. Mitjançant atacs que amenacin l'accessibilitat, poden ser destruïdes o eliminades, i denegar-hi així 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 afectin l'accessibilitat, ens pot provocar destrucció o eliminació de missatges, i impedir l'accés a la xarxa. En confidencialitat, lectura i observació del trànsit de missatges, des de o cap a la màquina. I respecte a la integritat, qualsevol modificació, retard, reordenació, duplicació o falsificació dels missatges entrants i/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 versió d'aquest. Per tant, cal mantenir actualitzat el programari per a les correccions de seguretat que vagin apareixent, i seguir les indicacions del fabricant o distribuïdor per a protegir l'element.

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

Nota

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

- ***Bug exploits:*** o explotació d'errors o forats [CERb] [Ins][San], ja sigui d'un maquinari, programari, servei, protocol o del propi sistema operatiu (per exemple, en el nucli), i normalment d'alguna de les seves versions en concret. Normalment, 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 es denominen *holes, exploits*, o simplement *bugs*), que poden ser aprofitats per un atacant per a trencar la seguretat dels sistemes. Se solen utilitzar o bé tècniques d'atac genèriques, com les que s'expliquen a continuació, o bé tècniques particulars per a l'element afectat. Cada element afectat tindrà un responsable –ja sigui fabricant, desenvolupador, distribuïdor o 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**: programa normalment annex a d'altres i que utilitza mecanismes d'autocòpia i transmissió. Són habituals els virus annexats 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 plaga de seguretat més gran d'avui en dia.

Els sistemes GNU/Linux estan protegits gairebé totalment contra aquests mecanismes per diverses raons: en els programes executables, tenen un accés molt limitat al sistema, en particular al compte de l'usuari. A excepció de l'usuari *root*, amb qui cal anar molt amb compte amb el que executa. El correu no sol utilitzar llenguatges de macros no verificats (com en el cas d'Outlook i Visual Basic Script a Windows, que és un forat d'entrada de virus), i en el cas dels documents, estem en condicions semblants, ja que no suporten llenguatges de macros no verificats (com el VBA a Microsoft Office).

En tot cas, caldrà fer atenció al que pugui passar en un futur, ja que podrien sorgir alguns virus específics per a GNU/Linux en aprofitar alguns *bugs* o *exploits*. 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 encaminador de correu, podrien arribar missatges amb virus que podria ser enviat a d'altres. Aquí es pot implementar alguna política de detecció i filtratge de virus. Una altra forma de plaga que podria entrar dins de la categoria de virus són els missatges d'*spam*, que si bé no solen ser utilitzats com a elements atacants, sí que els podem considerar com a problemes per la seva "virulència" d'aparició, i el cost econòmic que poden causar (pèrdues de temps i recursos).

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

- ***Trojan horse*** (o 'cavalls de Troia', o 'troians'): programes útils que incorporen alguna funcionalitat, però n'oculten 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 el seu consentiment per a executar-se (ActiveX a Windows), o tenen models limitats del que poden fer (Java, JavaScript). Però com qualsevol programari, també tenen forats i són un mètode ideal per a transmetre troians.

- ***Back door*** (o *trap door*, 'porta posterior'): mètode d'accés a un programa amagat que es pot utilitzar amb finalitats d'atorgar accés al sistema o a les dades manejades sense que el coneguem. Altres efectes poden ser canviar la configuració del sistema, o permetre introduir-hi virus. El mecanisme usat pot ser des de venir inclòs en algun programari comú, fins a ser en un troià.

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

- ***Keyloggers***: programa especial que es dedica a segrestar les interaccions amb el teclat, i/o ratolí, de l'usuari. Poden ser programes individuals o bé troians incorporats en altres programes.

  Normalment, necessitarien introduir-se en un sistema obert a què es disposés d'accés (encara que cada vegada més poden venir incorporats en troians que s'hi instal·lin). La idea és captar qualsevol introducció de tecles, de manera que es capturin contrasenyes (per exemple, les bancàries), interacció amb aplicacions, llocs visitats per la xarxa, formularis emplenats, etc.

- ***Scanner*** (escaneig de ports): més que un atac, seria un pas previ, que consistiria en la recollida de possibles objectius. Bàsicament, consisteix a utilitzar eines que permetin examinar la xarxa a la recerca de màquines amb ports oberts, siguin TCP, UDP o altres protocols, els quals indiquen la presència d'alguns serveis. Per exemple, escanejar màquines per buscar-hi el port 80 TCP, indica la presència de servidors web, dels quals podem obtenir informació sobre el servidor i la versió que utilitzen per a aprofitar-nos de vulnerabilitats conegudes.

- ***Sniffers*** ('detectors'): permeten la captura de paquets que circulen per una xarxa. Amb les eines adequades podem analitzar comportaments de màquines: quines són servidors, 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 telnet, rsh, rcp, ftpd... serveis no segurs que no s'haurien d'utilitzar (usar al seu lloc les versions segures: ssh, scp, sftp). Tant els detectors (com els *scanners*) no són necessàriament una eina d'atac, ja que també poden servir per a analitzar les nostres xarxes i detectar-ne errors, o simplement analitzar el nostre trànsit. Normalment, tant les tècniques d'*scanners* com les de detectors les sol utilitzar un intrús amb l'objectiu de trobar les vulnerabilitats del sistema, ja sigui per a conèixer dades d'un sistema desconegut (*scanners*), o bé per a analitzar la interacció interna (detector).

- ***Hijacking*** (o 'segrest'): són tècniques que intenten col·locar una màquina de manera que intercepti o reprodueixi el funcionament d'algun servei en una altra màquina que 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.

- ***Buffer overflows:*** tècnica bastant complexa que aprofita errors de programació en les aplicacions. La idea bàsica és aprofitar desbordaments (*overflows*)

en *memòries intermèdies* de l'aplicació, ja siguin cues, matrius, etc. Si no es controlen els límits, un programa atacant pot generar un missatge o dada més gran de l'esperat i provocar fallades. Per exemple, moltes aplicacions C amb *buffers* malament escrits, en matrius, si sobrepassem el límit podem provocar una sobreescriptura del codi del programa, cosa que causaria un mal funcionament o caiguda del servei o màquina. És més, una variant més complexa permet incorporar en l'atac trossos de programa (compilats C o bé *shell scripts*), que poden permetre l'execució de qualsevol codi que l'atacant vulgui introduir.

- ***Denial of Service*** ('atac 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), que es basa a utilitzar un conjunt de màquines distribuïdes perquè produeixin l'atac o sobrecàrrega de servei. Aquest tipus d'atacs se sol solucionar amb actualitzacions del programari, ja que normalment es veuen afectats aquells serveis que no van ser pensats per a una càrrega de treball determinada i no es 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 d'un *backdoor* o un troià.

  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 fan res més amb ella, simplement la deixen oberta; això gasta recursos del sistema en estructures de dades del nucli, i recursos de connexió per xarxa. 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 correu *bombing*, o simplement reenviament de correu (normalment amb emissor fals) fins que se saturen els comptes de correu, el sistema de correu cau, o es torna tan lent que és inutilitzable. Aquests atacs són en certa mesura senzills si es fan 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.

Nota

SYN flood, vegeu:
http://www.cert.org/advisories/CA-1996-21.html
E-Mail bombing, vegeu:
http://www.cert.org/tech tips/email_bombing_spamming.html

- ***Spoofing*** ('falsejament d'identitat'): les tècniques de falsejament d'identitat engloben diversos mètodes (normalment molt complexos) de falsificar tant la informació, com els participants en una transmissió (origen i/o destinació). Alguns falsejaments d'identitat són els de:

  – IP *spoofing*, falsificació d'una màquina, cosa que permet que generi trànsit fals o escolti trànsit que anava dirigit a una altra màquina. Combinat amb altres atacs es pot saltar fins i tot protecció de tallafocs.

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

– 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 força utilitzades les tècniques del que es denomina enginyeria social, que bàsicament intenta enganyar d'una manera raonable l'usuari, un exemple clàssic són correus falsos de l'administrador de sistema, o bé, per exemple, del banc en el qual tenim el nostre compte corrent, en què s'indica que hi ha hagut problemes en els comptes i que s'ha d'enviar informació confidencial, o la contrasenya anterior per a solucionar-los, o en els quals es demana que la contrasenya es canviï per una de concreta. Sorprenentment, aquesta tècnica (també coneguda com a *phising*) aconsegueix enganyar un nombre considerable d'usuaris. Fins i tot amb (enginyera social de) mètodes senzills: algun *cracker* 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 *crackers* van obtenir la firma privada de programari Microsoft amb només una trucada en què deien que trucaven de part de l'empresa, que se'ls havia presentat un problema i tornaven a necessitar la seva clau. Resumint, uns nivells alts de seguretat informàtica se'n poden anar en orris per una simple trucada, o un correu que un usuari malinterpreti.

Nota

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

- ***SQL injection***: és una tècnica orientada a bases de dades, i servidors web en particular, que s'aprofita generalment de programació incorrecta de formularis web, en els quals no s'ha controlat correctament la informació que es proporciona. No es determina que la informació d'entrada sigui del tipus correcte (fortament tipada respecte al que s'espera), o no es controlen quins tipus o caràcters literals s'hi introdueixen. La tècnica s'aprofita del fet que els literals obtinguts pels formularis (per exemple web, encara que els atacs es poden sofrir des de qualsevol API que permeti accés a base de dades, per exemple php o perl) són utilitzats directament per a construir les consultes (en SQL), que atacaran una base de dades determinada (a la qual en principi no es té accés directe). Normalment, si hi ha vulnerabilitats i poc control dels formularis, es pot injectar codi SQL al formulari, de manera que es construeixin consultes SQL que proporcionin la informació buscada. En casos dràstics es podria obtenir la informació de seguretat (usuaris i contrasenyes de la base de dades), o fins i tot taules o la base de dades sencera, quan no es podrien produir pèrdues d'informació, o esborraments intencionats de dades. En particular, aquesta tècnica en ambients web pot ser greu, a causa de les lleis que protegeixen la privacitat de dades personals que un atac d'aquest tipus pot provocar. En aquest cas, més que una qüestió de seguretat de sistema, suposa un problema de programació i control

amb tipatge fort de les dades esperades en l'aplicació, a més del control adequat de coneixement de vulnerabilitats presents del programari usat (base de dades, servidor web, API com php, perl...).

- ***Cross-side scripting*** (o XSS): una altra problemàtica relacionada amb ambients web, en particular amb alteracions del codi html, i/o *scripts* que pot obtenir un usuari en visualitzar un lloc web determinat, que pot ser alterat dinàmicament. S'aprofita generalment dels errors a l'hora de validar codi HTML (tots els navegadors tenen problemes amb això, per la pròpia definició d'HTML, 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 d'*script*, o bé mitjançant accés a les *cookies* de l'usuari des del navegador, o mitjançant l'alteració del procés pel qual es redirecciona una pàgina web a una altra. També hi ha tècniques mitjançant *frames*, que permeten redirigir l'html que s'estigui veient, o penjar directament el navegador. En particular, poden ser vulnerables els motors de cerca dels llocs web, que poden permetre execució de codi d'*script*. En general són atacs amb diverses tècniques complexes, però amb l'objectiu de capturar informació com *cookies*, que es pot utilitzar en sessions, i permetre així substituir una persona determinada mitjançant redireccions de llocs web, o obtenir informació seva. Una altra vegada des de la perspectiva de sistema, és més una qüestió de programari en ús. Cal controlar i conèixer vulnerabilitats detectades en navegadors (i aprofitar els recursos que ofereixen per a evitar aquestes tècniques), i controlar l'ús de programari (motors de cerca utilitzats, versions del servidor web, i API utilitzades en els desenvolupaments).

Algunes recomanacions generals (molt bàsiques) per a la seguretat, podrien ser:

- Controlar un factor problemàtic: els usuaris. Un dels factors que pot afectar més a 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 dins del propi sistema. La majoria dels atacs solen venir de dins del sistema, és a dir, una vegada l'atacant ha guanyat accés al sistema.

- Entre els usuaris, hi ha aquell que és una mica oblidós (o indiscret), que bé oblida la contrasenya cada dos per tres, l'esmenta en converses, l'escriu en un paper que oblida, o que és al costat (o enganxada) a l'ordinador o sobre la taula de treball, o que simplement la presta a altres usuaris o coneguts. Un altre tipus és el que col·loca contrasenyes molt predicibles, ja sigui el seu mateix id d'usuari, el seu nom, el seu DNI, el nom de la seva xicota, el de la seva mare, el del seu gos, etc., coses que amb un mínim d'informació es poden trobar fàcilment. Un altre cas són els usu-

aris normals amb un cert coneixement, que col·loquen contrasenyes vàlides, però sempre cal tenir en compte que hi ha mecanismes que les poden trobar (*cracking* de contrasenyes, *sniffing,* falsejament d'identitats...). Cal establir una certa "cultura" de seguretat entre els usuaris, i mitjançant tècniques obligar-los que canviïn les contrasenyes, no utilitzin paraules típiques, les contrasenyes han de ser llargues (tenir més de 2 o 3 caràcters), etc. Últimament, a moltes empreses i institucions s'està implantant la tècnica de fer firmar un contracte 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 (clar que això no impedeix que altres ho facin per ell).

- No utilitzar ni executar programes dels quals no en puguem garantir l'origen. Normalment, molts distribuïdors utilitzen mecanismes de comprovació de firmes per a verificar que els paquets de programari són tals, com per exemple les sumes md5 (ordres md5sum) o la utilització de firmes GPG [Hatd](ordre gpg). El venedor o distribuïdor proveeix una suma md5 del seu arxiu (o imatge de CD), i en podem comprovar l'autenticitat. Últimament, en les distribucions s'estan usant tant firmes per a paquets individuals, com les firmes per als dipòsits de paquets, com a mecanisme per a assegurar la fiabilitat del proveïdor.

- No utilitzar usuaris privilegiats (com *root*) per al treballs normals de la màquina; qualsevol programa (o aplicació) tindria els permisos per a accedir a qualsevol part.

- 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.

- No utilitzar elements que no sabem com actuen ni intentar descobrir-ho a base d'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 assegura que no es pugui colar algun programa maliciós que burli la seguretat si l'executem amb els permisos adequats. És a dir, que en general hem d'anar molt amb compte amb tots aquests tipus d'activitats que suposin accessos i execució de tasques de formes més o menys privilegiades.

1.2. Contramesures

Respecte a les mesures que es poden prendre sobre 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 les examinarem més endavant):

- ***Password cracking***: en atacs de força bruta per a trencar les contrasenyes, sol ser habitual intentar obtenir accés per *logins* de manera repetida; si s'hi aconsegueix entrar, la seguretat de l'usuari ha estat compromesa i es deixa la porta oberta a altres tipus d'atacs com, per exemple, els *backdoor*, o simplement la destrucció del compte. Per a prevenir aquest tipus d'atacs, cal reforçar la política de contrasenyes, demanar-ne una longitud mínima i canvis de contrasenya periòdics. Una cosa que cal evitar és l'ús de paraules comunes a 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, argot, etc.). Aquest tipus de contrasenyes seran les primeres a caure. També pot ser fàcil obtenir informació de l'atacat, com noms, DNI o la seva adreça, 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, etc. Una bona elecció sol ser elegir contrasenyes d'entre 6 a 8 caràcters com a mínim, amb contingut de caràcters alfabètics, numèrics i algun caràcter especial.

  Encara que la contrasenya estigui ben elegida, aquesta pot ser insegura si s'utilitza en serveis no segurs. Per tant, es recomana reforçar els serveis mitjançant tècniques d'encriptació que protegeixin les contrasenyes i els missatges. I al contrari, evitar (o no utilitzar) tots aquells serveis que no suportin encriptació, i conseqüentment, susceptibles de ser atacats amb mètodes, per exemple d'*sniffers*; entre aquests podríem incloure serveis telnet, ftp, rsh, rlogin (entre d'altres).

- ***Bug exploits:*** evitar disposar de programes que no s'utilitzin, siguin antics, o no s'actualitzin (per ser obsolets). Aplicar els últims pedaços i actualitzacions que estiguin disponibles, tant per a les aplicacions com per al sistema operatiu. Provar eines que detectin vulnerabilitats. Mantenir-se al dia de les vulnerabilitats que es vagin descobrint.

Nota

Vegeu pedaços per al sistema operatiu en:

http://www.debian.org/security

http://www.redhat.com/security

http://fedoraproject.org/wiki/Security

- **Virus**: utilitzar mecanismes o programes antivirus sistemes de filtratge de missatges sospitosos, evitar l'execució de sistemes de macros (que no es puguin verificar). No s'han de minimitzar els possibles efectes dels virus, cada dia es perfeccionen més, i tècnicament és possible fer virus simples que puguin desactivar xarxes en qüestió de minuts (només cal veure alguns dels virus recents del món Windows).

Nota

Sobre vulnerabilitats, una bona eina és Nessus. Per a descobrir noves vulnerabilitats, vegeu CERT en:

http://www.cert.org/advisories/

- ***Worm*** (o 'cuc'): controlar l'ús de les nostres màquines o usuaris en hores no previstes, i el control del trànsit de sortida i/o entrada.

- ***Trojan horse*** (o 'cavalls de Troia', o 'troians'): verificar la integritat dels programes periòdicament, mitjançant mecanismes de suma o firmes. De-

tecció de trànsit anòmal de sortida o entrada al sistema. Utilitzar tallafocs per a bloquejar trànsit sospitós. Una versió bastant perillosa dels troians la formen els *rootkits* (comentats 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.

- ***Backdoor*** (o *trap door*, 'porta posterior'): cal obtenir dels proveïdors o venedors del programari la certificació que aquest no conté cap tipus de *backdoor* amagat no documentat, 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 firmes digitals (tipus md5 o gpg) [Hatd]. Sempre que aquestes estiguin disponibles, seria útil verificar-les abans de procedir a la instal·lació del programari. També pot provar-se el sistema intensivament, 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 poden ser també útils els sistemes de firmes o sumes per a crear codis sobre programari ja instal·lat i controlar que no es produeixin canvis en programari vital. O bé còpies de seguretat, amb què podem fer comparacions per a detectar-hi canvis.

- **Bombes lògiques**: en aquest cas se solen ocultar després d'activacions per temps o per accions de l'usuari. Podem verificar que no hi hagi 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 estiguin en execució en segon pla des de fa molt de temps (ordres w, jobs). En tot cas, es podrien utilitzar mesures preventives que impedissin treballs no interactius als usuaris, o que només els permetessin a aquells que ho necessitessin.

- ***Keyloggers*** i ***rootkits***: en aquest cas hi haurà algun procés intermediari que intentarà capturar les nostres pulsacions de tecles i les emmagatzemarà en 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 de *keylogger* amb tramesa externa. Per a provar un funcionament molt bàsic d'un *keylogger* molt senzill, es pot veure l'ordre de sistema *script* (vegeu man *script*). L'altre cas, el *rootkit* (que sol incloure també algun *keylogger*) sol ser un paquet d'uns quants programes amb diverses tècniques, i permet que l'atacant, una vegada entri en un compte, utilitzi diversos elements com un *keylogger*, *backdoors*, troians (substituint ordres del sistema), etc., per tal d'obtenir informació i portes d'entrada al sistema, moltes vegades s'acom-

panya de programes que fan neteja dels *logs,* per a eliminar les proves de la intrusió. Un cas particularment perillós el formen els *rootkits*, que s'usen o vénen en forma de mòduls de nucli, que els permet actuar en l'àmbit de nucli. Per a la seva detecció, seria necessari controlar que no hi hagi trànsit extern que surti fins a una certa adreça. Una eina útil per a verificar els *rootkits* és *chrootkit*.

- ***Scanner*** (escaneig de ports): els *scanner* solen llançar sobre un o més sistemes bucles d'escaneig de ports coneguts per a detectar els que queden oberts i aquells serveis que estan funcionant (i obtenir informació de les versions dels serveis) i que podrien ser susceptibles d'atacs.

Nota

L'eina chrootkit es pot trobar en:
http://www.chrootkit.org

- ***Sniffers*** ('detectors'): evitar intercepcions i impedir així la possibilitat que s'introdueixin escoltes. Una tècnica és la construcció maquinari de la xarxa, que es pot dividir en segments perquè el trànsit només circuli per la zona que s'utilitzarà, posar-hi tallafocs per a unir aquests segments i poder controlar el trànsit d'entrada i sortida. Usar tècniques d'encriptació que els missatges no puguin ser llegits i ser interpretats per algú que escolti la xarxa. En el cas tant d'*scanners* com d'*sniffers*, podem utilitzar eines com Whireshark [Wir] (antic Ethereal) i Snort, per a fer comprovacions sobre la nostra xarxa, o per al port scanning Nmap. En el cas dels *sniffers*, aquests poden ser detectats a la xarxa mitjançant la recerca de màquines en format Ethernet promiscu (estan a l'escolta de qualsevol paquet que circula), normalment, la targeta de xarxa només captura el trànsit que va cap a ella (o de tipus *broadcast* o *multicast*).

- ***Hijacking*** (o 'segrest'): implementar mecanismes d'encriptació en els serveis, requerir autenticació i, si és possible, que aquesta autenticació es renovi periòdicament. Controlar el trànsit entrant o sortint per tallafocs. Monitorar la xarxa per a detectar fluxos de trànsit sospitosos.

- ***Buffer overflows***: solen ser comuns com *bugs* o forats del sistema, se solen solucionar per a actualitzar el programari. En tot cas, es poden observar per *logs* situacions estranyes de caiguda de serveis que haurien d'estar funcionant. També es poden maximitzar els controls de processos i accessos a recursos per tal d'aïllar el problema quan es produeixi en entorns d'accés controlat, com el que ofereix SELinux (vegeu més endavant en el mòdul).

- ***Denial of Service*** ('atac DoS'), i altres com SYN flood, o correus *bombing*, prendre mesures de bloqueig de trànsit innecessari a la nostra xarxa (per exemple, per mitjà de tallafocs). En aquells serveis que es pugui, caldrà controlar mides de memòria intermèdia, nombre de clients per atendre, temporitzacions de tancament de connexions, capacitats del servei, etc.

- ***Spoofing***: com ara IP *spoofing*, ARP *spoofing*, correu electrònic. Aquests casos necessiten una forta encriptació dels serveis, control per tallafocs, mecanismes d'autenticació basats en diversos aspectes (per exemple, no basar-se en

la IP, si aquesta es pogués veure compromesa), es poden implementar mecanismes que controlin sessions establertes i es basin en diversos paràmetres de la màquina alhora (sistema operatiu, processador, IP, adreça Ethernet, etc.). També monitorar sistemes DNS, *cachés* d'ARP, *spools* 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. Mesures adequades com augmentar la informació o educar usuaris i tècnics en 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: controlar qui té informació de seguretat, i com la usa.

  Respecte a usuaris finals, millorar la seva cultura de contrasenyes, evitar que la deixin apuntada en qualsevol lloc, a la vista de tercers, o simplement la divulguin.

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'accessos als recursos per a definir-los correctament i poder garantir la confidencialitat i la integritat de la nostra informació. En aquest cas, ens estarem protegint d'atacants que hagin obtingut accés al nostre sistema, o bé d'usuaris hostils que es vulguin saltar les restriccions imposades en el sistema.

Respecte a la seguretat en xarxa, hem de garantir que els recursos que oferim (si proporcionem uns serveis determinats) tenen els paràmetres de confidencialitat necessaris i que els serveis 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 assegurar els mecanismes d'autenticació, en el sentit que accedim als servidors correctes i no hi ha casos de suplantació de serveis o servidors (normalment bastant difícils de detectar).

Respecte a les aplicacions i als mateixos serveis, a més de garantir la configuració correcta de nivells d'accés mitjançant permisos i d'autenticació dels usuaris permesos, haurem de vigilar la possible explotació de *bugs* 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, amb certes tècniques, per a saltar-se les restriccions imposades. En aquest cas, practiquem una política de prevenció que passa per mantenir el sistema actualitzat en la mesura que puguem, de manera que, o bé 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 per a conèixer els últims errors detectats en el programari i les vulnerabilitats que se'n deriven, i que podrien exposar els nostres sistemes a errors de seguretat, ja sigui local o per xarxa.

3. Seguretat local

La seguretat local [Peñ] [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.

Nota

Diversos atacs, encara que vinguin de l'exterior, tenen com a finalitat aconseguir l'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 ja es troba en l'arrencada del sistema. Si el sistema es pot arrencar des de disc o CD, un atacant podria accedir a les dades d'una partició GNU/Linux (o també Windows) només amb muntar el sistema de fitxers i es podria col·locar com a usuari *root* sense necessitat de conèixer cap contrasenya. En aquest cas, es necessita protegir l'arrencada del sistema des de la BIOS, per exemple, protegint l'accés per contrasenya, de manera que no es permeti l'arrencada des de CD (per exemple mitjançant un LiveCD) o disquet. També és raonable actualitzar la BIOS, ja que també pot tenir errors de seguretat. A més, cal anar amb compte, perquè molts dels fabricants de BIOS ofereixen contrasenyes extres conegudes (una espècie de *backdoor*), amb la qual cosa no podem dependre d'aquestes mesures en exclusiva.

El pas següent és protegir el *bootloader* (o 'carregador d'arrencada'), ja sigui lilo o grub, perquè l'atacant no pugui modificar les opcions d'arrencada del nucli o modificar directament l'arrencada (en el cas del grub). Qualsevol dels dos es pot protegir també per contrasenyes.

En grub, el fitxer /sbin/grub-md5-crypt demana la contrasenya i genera una suma md5 associada. Després, el valor obtingut s'introdueix en /boot/grub/grub.conf. Sota la línia *timeout*, s'introdueix:

```
contrasenya --md5 summa-md5-calculada
```

Per a lilo es col·loca, o 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 passats al nucli des de línia d'ordres. Cal anar amb compte de posar el fitxer /etc/lilo.conf protegit a només lectura/escriptura des del *root* (chmod 600).

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, es provoca una operació d'aturada a la màquina. Aquest comportament és definit en /etc./inittab, amb una línia com:

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

Si es comenta, aquesta possibilitat de reiniciar quedarà desactivada. O al contrari, es pot crear un fitxer /etc/shutdown.allow, que permet que alguns usuaris concrets puguin reiniciar.

3.2. Contrasenyes i *shadows*

Les contrasenyes típiques dels sistemes UNIX inicials (i de primeres versions GNU/Linux) estaven encriptades mitjançant uns algoritmes DES (però amb claus petites, i una trucada de sistema s'encarregava d'encriptar i desencriptar, en concret crypt, vegeu el man).

Normalment, es trobaven en el fitxer /etc/passwd, en el segon camp, per exemple:

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

Però el problema és que aquest fitxer és llegible per qualsevol usuari, amb la qual cosa un atacant podia obtenir el fitxer i utilitzar un atac de força bruta, fins que desencriptés les contrasenyes contingudes en el fitxer, o bé mitjançant atacs de força bruta per mitjà de diccionaris.

El primer pas és utilitzar els nous fitxers /etc/shadow, en què es guarda ara la contrasenya. Aquest fitxer és només llegible pel *root*, i per ningú més. En aquest cas, en /etc/passwd apareix un asterisc (*) en el lloc en què abans hi havia la contrasenya encriptada. 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 d'encriptació de les contrasenyes per un de més complex i difícil de trencar. Ara, tant Fedora com Debian ofereixen contrasenyes per md5; normalment ens deixen escollir el sistema en temps d'instal·lació. Cal anar amb compte amb les contrasenyes md5, ja que si usem NIS,

podríem tenir algun problema; si no, tots els clients i servidors usaran md5 per a les seves contrasenyes. Les contrasenyes es poden reconèixer en /etc/shadow perquè vénen amb un prefix "1".

Altres actuacions possibles 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.

Respecte a les eines, és interessant disposar d'un *cracker* de contrasenyes (és a dir, un programa per a obtenir contrasenyes), per a comprovar la situació real de seguretat dels comptes dels nostres usuaris, i forçar així el canvi en els que detectem insegurs. Dues de les més utilitzades per administradors són John the Ripper i "crack". Poden funcionar també per diccionari, amb la qual cosa, serà interessant disposar d'algun ASCII d'espanyol (es poden trobar per la xarxa). Una altra eina és "Slurpie", que pot provar diverses màquines alhora.

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 sistemes de detecció d'intrusos habilitats i podem ser objecte de denúncia per intents d'intrusió, ja sigui 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 al límit entre ser de seguretat o d'intrusió.

3.3. Suid i sticky bits

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

El bit sticky s'utilitza sobretot en directoris temporals, en què volem que en alguns grups (de vegades no relacionats), qualsevol usuari pugui escriure, però només pugui esborrar bé el propietari del directori, o bé el propietari del fitxer que sigui 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 hi hagi més que els purament necessaris com a temporals. El bit es col·loca mitjançant (chmod +t dir), i es pot treure amb -t. En un ls apareixerà com un directori amb permisos drwxrwxrwt (fixeu-vos en l'última t).

El bit setuid permet que un usuari executi (ja sigui un executable o un *shell script*) 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, encara que no té permisos de *root*, pot executar un programa amb setuid que pot disposar de permisos interns d'usuari *root*. Això és

molt perillós en cas d'*scripts*, ja que es podrien editar i modificar per a fer qualsevol cosa. 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, ja sigui aplicant-lo al propietari (s'anomena llavors suid) o al grup (s'anomena bit sgid); es pot treure amb –s. En el cas de visualitzar amb ls, el fitxer apareixerà amb -rwSrw-rw (fixeu-vos en la S), si és només suid, en sgid la S apareixeria després de la segona w.

En el cas d'utilitzar chmod amb notació octal, s'usen 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).

3.4. Habilitació de *hosts*

En el sistema hi ha uns quants fitxers de configuració especials que permeten habilitar l'accés a una sèrie de *hosts* 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...) sense necessitat d'introduir la contrasenya del compte. Això és potencialment perillós, ja que una mala configuració de l'usuari podria permetre l'entrada d'usuaris no desitjats, o que un atacant (amb accés al compte de l'usuari) canviés les adreces en .rhosts per a 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".

- /etc/hosts.equiv: és exactament el mateix que els fitxers .rhosts però en l'àmbit 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 ser-hi aquest fitxer, i sempre hi ha l'alternativa del servei ssh als "r".

- /etc/hosts.lpd: en el sistema d'impressió LPD s'utilitzava per a posar a les màquines que podien accedir al sistema d'impressió. Cal anar amb molt de compte, si no estem servint, de deshabilitar completament l'accés al sistema, i en cas que ho estiguem fent, restringir al màxim les màquines que realment en fan ús. O intentar canviar a un sistema CUPS o LPRng, que té molt més control sobre els serveis. El sistema LPD és un blanc habitual d'atacs de tipus cuc o *buffer overflows*, i hi ha documentats diversos *bugs* importants. Cal estar atents si utilitzem aquest sistema i el fitxer hosts.lpd.

3.5. Mòduls PAM

Els mòduls PAM [Peñ][Mor03] són un mètode que permet que l'administrador controli 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 es poden incorporar a les aplicacions com a mètode per a controlar l'autenticació dels seus usuaris. És més, es pot canviar el mètode d'autenticació (mitjançant la configuració dels mòduls PAM), sense que sigui necessari canviar l'aplicació.

Els mòduls PAM (les biblioteques) solen ser en el directori /lib/security (en forma de fitxers objecte carregables dinàmicament). I la configuració de PAM és present en el directori /etc/pam.d, en què apareix un fitxer de configuració de PAM per cada aplicació que està usant mòduls PAM. Ens trobem amb la configuració d'autenticació d'aplicacions i serveis com ssh, de connexió gràfica de X Window System, com xdm, gdm, kdm, xscreensaver... o, per exemple, de la connexió del sistema (l'entrada per identificador d'usuari i contrasenya). En les antigues versions de PAM, s'utilitzava un arxiu (típicament en /etc/pam.conf), que era en el qual es llegia la configuració PAM si el directori */etc/pam.d* no existia.

Nota

Per a saber més, es pot veure la "The Linux-PAM System Administrators' Guide":

http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html

La línia típica d'aquests fitxers (en /etc/pam.d) tindria aquest format (si s'utilitza /etc/pam.conf caldria afegir el servei a què pertany com a primer camp):

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

en què s'especifica:

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

b) *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 sintaxi. Hi ha una altra de més actual que treballa en parelles valor i acció.

c) la ruta (*path*) del mòdul.

d) arguments que es passen al mòdul (depenen de cada mòdul).

Atès que alguns serveis necessiten diverses línies de configuració comunes, hi ha possibilitat d'operacions d'inclusió de definicions comunes d'altres serveis, només cal afegir-hi 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 de *login* (s'han enumerat també les línies incloses provinents d'altres serveis):

```
auth                          requisite pam_securetty.so
auth                          requisitepam_nologin.so
auth                          requiredpam_env.so
auth                          requiredpam_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 el *login* (entrada al sistema). Un dels mòduls pam_unix.so és el que realment fa la verificació de la contrasenya de l'usuari (veient fitxers passwd, shadow...).

Altres controlen la seva sessió per a veure quan ha entrat per última vegada, o guarden 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 canviï la contrasenya (si està obligat a fer-ho en la primera connexió que faci) i que tingui de 4 a 8 lletres, i es pot utilitzar md5 per a l'encriptació.

En aquest exemple podríem millorar la seguretat dels usuaris: l'*auth* i la contrasenya 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 de *login*. El mateix es pot fer en el fitxer de configuració de passwd (en aquest cas, l'ordre de canvi de la 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 = 16.

3.6. Alteracions del sistema

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

Nota

CERT UNIX checklist:
http://www.cert.org/tech_tips/usc20_full.html

Un cas típic és la possibilitat de forçar el *root* perquè executi ordres falses de sistema; per exemple, si el *root* inclogués el "." en la seva variable de PATH, això permetria l'execució d'ordres des del seu directori actual, la qual cosa ha-

bilitaria per a col·locar arxius que substituïssin ordres del sistema i que serien executats en primer terme abans que els de sistema. El mateix procés es pot fer amb un usuari, encara que per ser més limitats els seus permisos, pot no afectar tant al sistema, com més a la pròpia seguretat de l'usuari. Un altre cas típic és el de les pantalles de *login* falses, que poden substituir el procés típic de connexió, passwd, per un programa fals que emmagatzemi les contrasenyes introduïdes.

En el cas d'aquestes alteracions, serà imprescindible usar polítiques d'auditoria dels canvis, ja sigui per mitjà de càlcul de firmes 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 bé utilitzar eines com *chkrootkit*, si aquests vinguessin de la instal·lació d'algun *rootkit* conegut.

Nota

Chkrootkit, vegeu:
http://www.chkrootkit.org

4. SELinux

La seguretat tradicional dins del sistema s'ha basat en les tècniques DAC *(discretionary access control*), en què normalment cada programa disposa de control complet sobre els accessos als seus recursos. Si un programa determinat (o l'usuari en permetre-ho) decideix fer un accés incorrecte (per exemple, deixa dades confidencials en obert, ja sigui per desconeixement o per mal funcionament). Així en DAC, un usuari té un 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 en especial afectarà els objectes (altres programes, fitxers, o recursos) a què l'usuari tingui accés. En el cas del *root*, això comprometria la seguretat global del sistema.

D'altra banda, les tècniques MAC (*mandatory access control*), desenvolupen polítiques de seguretat (definides per l'administrador) en què el sistema té un control complet sobre els drets d'accés que es concedeix sobre cada recurs. Per exemple, amb permisos (de tipus Unix) podem donar accés a fitxers, però mitjançant polítiques MAC tenim control extra per a determinar a quins fitxers explícitament es permet l'accés per part d'un procés, i quin nivell 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 de tipus MAC recent inclòs en la bifurcació 2.6.x del nucli, que les distribucions van incloent progressivament: Fedora/Xarxa Hat el porten habilitat per defecte (encara que és possible canviar-lo durant la instal·lació), i a Debian és un component opcional.

SELinux implementa polítiques de seguretat de tipus MAC, cosa que 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 *log,* però no el reescriuria o truncar-lo (tècniques utilitzades habitualment per atacants per a esborrar les pistes dels seus accessos). En un altre exemple, es podria permetre que programes de xarxa s'enllacessin a un port (o ports) que necessiten, i tanmateix denegar l'accés a altres ports (per exemple, podria ser una tècnica que permetés controlar alguns troians o *backdoors*).

Nota

Alguns recursos sobre SELinux:

http://www.redhat.com/docs/manuals/enterprise/RHEL-4-manual/selinux-guide/

http://www.nsa.gov/selinux/

http://fedora.redhat.com/docs/selinux-faq/

http://selinux.sourceforge.net/

SELinux va ser desenvolupat per l'agència US NSA, 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 a diferents distribucions GNU/Linux.

Disposem en SELinux d'un model de domini-tipus, en què cada procés corre en un denominat context de seguretat, i qualsevol recurs (fitxer, directori, *socket*, etc.) té un tipus associat a ell. Hi ha un conjunt de regles que indiquen qui-

nes accions es poden efectuar en cada context sobre cada tipus. Un avantatge d'aquest model context-tipus és que les polítiques que es defineixin poden ser analitzades (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 SELlinux cada procés del sistema té un context format per 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 hi 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 per aquesta terna de valors com: `root:sysadm_r:sysadm_t;` és el context per a l'administrador del sistema, defineix la seva identitat, el seu rol, i el seu 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 en els 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 en 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
```

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

Cal anar amb compte amb l'activació i desactivació de SELinux, en especial amb l'etiquetatge de contextos en els fitxers, ja que en períodes que s'activi/desactivi es poden perdre etiquetes, o no ser etiquetat simplement. Així mateix, la realització de còpies de seguretat del sistema de fitxers ha de tenir en compte que es conservin les etiquetes de SELinux.

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

Respecte a la política usada, SELinux suporta dos tipus diferents: *targered* i *strict*. En *targered* la majoria de processos operen sense restriccions, i només serveis (alguns *daemons*) específics són posats 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 confinen polítiques definides, de manera que qualsevol acció és controlada per les polítiques definides. En principi, aquests són els dos tipus de polítiques definides generalment, però l'especificació està oberta a incloure'n més.

Un cas especial de política és la MLS (*multilevel security*), que és una política multinivell de tipus *strict*. La idea és dins de la mateixa política definir diferents nivells de seguretat, els contextos de seguretat tenen associat un camp addicional de nivell d'accés. Aquest tipus de política de seguretat (com MLS)

sol ser utilitzada a organitzacions governamentals i militars, en què 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, es necessita disposar de polítiques de seguretat d'aquest tipus.

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

4.1. Arquitectura

L'arquitectura de SELinux és formada pels components següents:

- Codi a nivell nucli
- La llibreria compartida de SELinux
- La política de seguretat (la base de dades)
- Eines

Examinem algunes consideracions respecte a cada component:

- El codi de nucli monitoritza l'activitat del sistema, i assegura que les operacions sol·licitades estan autoritzades sota la configuració de polítiques de seguretat de SELinux actual, i no permeten les operacions no autoritzades, i normalment genera entrades en *log* de les operacions denegades. El codi actualment es troba integrat en els nuclis 2.6.x, en els anteriors és ofert com a sèrie de *patchs*.

- La majoria d'utilitats i components SELinux no directament relacionats amb el nucli, fan ús de la llibreria compartida, anomenada libselinux1.so. que proporciona una API per a interaccionar amb SELinux.

- La política de seguretat és la que està integrada a 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 /etc/security/selinux (encara 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 Fedora) no instal·len les fonts per defecte), les quals habitualment podem trobar en /etc/security/selinux/src/policy o en /etc/selinux.

Normalment, aquestes fonts consisteixen en diversos grups d'informació:

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

– Fitxers de la configuració inicial, usuaris i rols associats.

– 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 desenes de milers de línies. Amb la qual cosa es pot veure el problema de trobar *bugs* o definir canvis en les polítiques.

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

- Eines: inclouen ordres usades per a administrar i usar SELinux. Versions modificades d'ordres estàndards Linux. I eines per a l'anàlisi de polítiques i el desenvolupament.

Vegem d'aquest últim apartat 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, inclosa la compilació de les polítiques a binari, típicament s'invoca des de les operacions de makefile.
getenforce	Genera missatge amb el mode actua de SELinux (*permissive* o *enforcing*). O bé desactivat si escau.
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 l'altre.
runn_init	Utilitzat per a activar un servei (*start, stop*), i s'assegura que es fa 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, és utilitzat típicament 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 per exemple cp, mv, install, ls, ps.... 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). L'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, es modifiquen alguns altres programes comuns per a suportar SELinux com:

- cron: modificat per a incloure els contextos per als treballs 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 aquest entra en el sistema.
- I diversos programes addicionals que modifiquin /etc/passwd o /etc/shadow.

També en algunes distribucions s'inclouen eines per a la gestió de SELinux, com les setools(-gui), que porten diverses eines de gestió i anàlisi de les polítiques. Així com alguna eina especifica per a controlar els contextos associats als diferents serveis suportats per SELinux en la distribució, per exemple l'eina system-config-securitylevel a Fedora disposa d'un apartat per a la configuració de SELinux com podem observar en la figura següent:

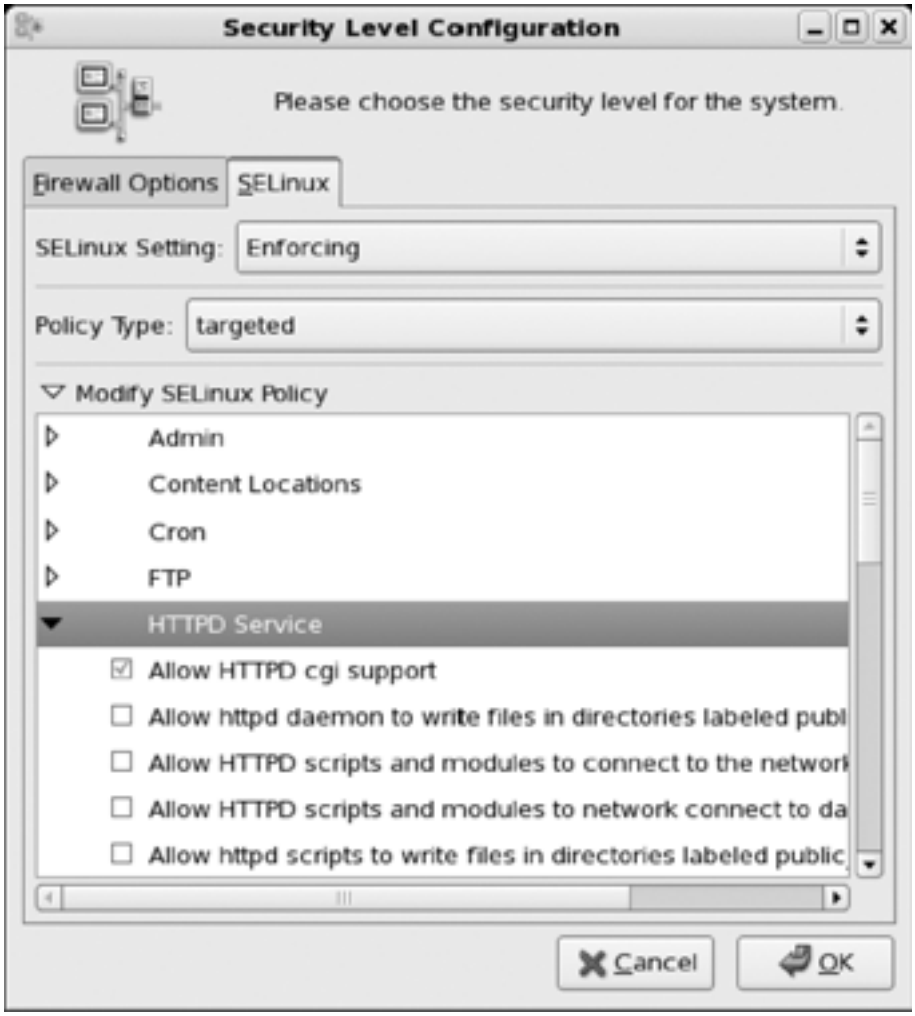

Figura 1. Interfície a Fedora per a configuració SELinux

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

A Fedora, per exemple, trobem suport de booleans per a (entre d'altres): 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 com a valors condicionals de les regles de la política usada, això permet modificacions de la política sense necessitat de carregar una nova política.

4.2. Crítica

Alguns administradors, i experts en seguretat, han criticat especialment SELinux per ser massa complex per a configurar i administrar. S'argumenta que, per la seva complexitat intrínseca, fins i tot usuaris experimentats poden cometre errors, i deixar la configuració de SELinux no segura o inservible, i el sistema vulnerable. Encara que això és discutible fins a cert punt, ja que encara que tinguéssim SELinux mal configurat, encara seguirien actius els permisos UNIX, SELinux no permetrà una operació que els permisos originals ja no permetien, de fet podem veure-ho com una etapa més de seguretat més estricta.

També hi ha factors de rendiment que poden afectar, a causa de l'enorme mida de les polítiques, poden disminuir les prestacions, a causa del gran ús de memòria i al temps inicial de càrrega, i en algun cas al processament de les regles. Cal pensar que estem parlant pràcticament d'un sistema d'unes 10.000 regles de la política. I encara pot ser un número més gran si escollim política de tipus *strict,* amb la necessitat d'especificar absolutament totes les opcions per 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 determinat, quin és l'origen o causa inicial. Atès que 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 servei determinat.

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

5. Seguretat en xarxa

5.1. Client de serveis

Com a clients de serveis, bàsicament hem d'assegurar que no posem en perill els nostres usuaris (o que ells es posin en perill sols) per usar serveis insegurs. Evitar l'ús de serveis que no utilitzin encriptació de dades i contrasenyes (ftp, telnet, correu no segur). Utilitzar en aquest cas tècniques de connexió encriptada, com SSH i SSL.

Nota

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

Un altre punt important 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 potents, que ens permetin verificar l'autenticitat dels servidors (per exemple, SSH i SSL tenen alguns d'aquests mecanismes). I també haurem de verificar la xarxa a la recerca d'intrusos que intentin substitucions de servidors, a més de polítiques correctes de filtratge de paquets mitjançant tallafocs (*firewalls*), que ens permetin treure els nostres paquets de petició i usar els servidors adequats, en controlar els paquets d'entrada que rebem com a respostes.

5.2. Servidors: inetd i xinetd

La configuració dels serveis de xarxa [Mou01], tal com hem vist, es fa des de diversos llocs [Ano99][Hat01][Peñ]:

- En /etc/inetd.conf o l'equivalent directori/etc/xinetd.d: aquests sistemes són una espècie de "superservidors", ja que controlen diversos serveis fills i les seves condicions d'arrencada. El servei inetd és utilitzat a Debian i xinetd, i a Fedora (a Debian pot ser instal·lat opcionalment en substitució d'inetd).

- Servidors iniciats en arrencada: segons el *runlevel* tindrem una sèrie de serveis engegats. L'arrencada s'originarà en el directori associat al *runlevel*, per exemple, a Debian el *runlevel* per defecte és el 2, els serveis arrencats seran arrencats des del directori /etc/rc2.d, segurament amb enllaços als *scripts* continguts en /etc/init.d, en els quals s'executarà amb el paràmetre start, stop, restart, segons correspongui.

- Altres serveis de tipus RPC: associats a trucades remotes entre màquines són utilitzats, per exemple, en NIS i NFS. Es pot examinar quins hi ha amb l'ordre `rpcinfo -p`.

Altres fitxers de suport (amb informació útil) són: /etc/services, que és format per una llista de serveis locals o de xarxa coneguts, juntament amb el nom del

protocol (tcp, udp o d'altres), que s'utilitza en el servei, i el port que utilitza; /etc/protocols és una llista de protocols coneguts, i /etc/rpc és una llista de servidors RPC, juntament amb els ports usats. Aquests fitxers vénen amb la distribució i són una espècie de base de dades que utilitzen les ordres i eines de xarxa per a determinar els noms de serveis, protocols, o rpc i els seus ports associats. Es pot destacar que són fitxers més o menys històrics, que no han de contenir totes les definicions de protocols i serveis, poden buscar-se així mateix diferents llistes de ports coneguts a Internet.

Una de les primeres accions que ha de dur a terme l'administrador és deshabilitar tots aquells serveis que no estigui utilitzant o no tingui previst utilitzar, caldrà que es documenti sobre la utilització dels serveis [Mou01], i quin programari els pot necessitar. [Neu]

En el cas de /etc/inetd.conf, només hem de comentar la línia del servei que cal deshabilitar, col·locant un "#" com a primer caràcter en la línia.

En l'altre model de serveis, utilitzat per defecte a Fedora (i opcional en Debian), el xinetd, la configuració rau en el fitxer /etc/xinetd.conf, en què es configuren alguns valors per defecte de control de *logs,* i després la configuració de cada servei fill es fa per mitjà d'un fitxer dins del directori /etc/xinetd.d. A cada fitxer es defineix la informació del servei, equivalent a la que apareix en el inetd.conf, en aquest cas, per a desactivar un servei només cal posar una línia disable = yes dins del fitxer del servei. Xinetd té una configuració més flexible que inetd, en separar la configuració dels diferents serveis en diferents arxius, i té força opcions per a limitar les connexions a un servei, en el nombre o la capacitat d'aquestes connexions; tot això permet un millor control del servei i, si es configura adequadament, podem evitar alguns dels atacs per denegació de servei (DoS o DDoS).

Respecte al tractament dels serveis dels *runlevel* des d'ordres de la distribució, ja esmentem diverses eines en la unitat d'administració local, que permetien habilitar o deshabilitar serveis. També hi ha eines gràfiques com ksysv de KDE, o el system-config-services i ntsysv a Fedora (a Debian són recomanables sysv-rc-conf, rcconf o bum). I en un nivell inferior, podem anar al nivell de *runlevel* que vulguem (/etc/rcx.d), i desactivar els serveis que volguem canviant les S o K inicials de l'*script* per un altre text: un mètode seria, per exemple, canviar S20ssh, per STOP_S20ssh, i ja no arrencarà; la propera vegada, quan tornem a necessitar-lo, traiem el prefix i el tornem a tenir actiu. O potser l'ús recomanat d'utilitats simples per a posar, treure o activar un servei determinat (com service i chkconfig a Fedora, o similars a Debian com update-rc.d i invoke.rc.d).

Un altre aspecte és el tancament de serveis no segurs. Tradicionalment, al món UNIX s'havien utilitzat serveis de transferència d'arxius com ftp, de connexió remota com telnet, i ordres d'execució (*login* o còpia) remotes, moltes de les quals començaven amb la lletra r, (per exemple, rsh, rcp, rexec...). 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 era en la informació que podia obtenir un atacant i que li podia facilitar molt el treball. Tots aquests serveis no haurien de ser usats actualment a causa dels perills potencials que impliquen. Respecte al primer grup:

a) ftp, telnet, en les transmissions per xarxa no encripten les contrasenyes, i qualsevol pot fer-se amb les contrasenyes pde serveis o els comptes associats (per exemple, mitjançant un *sniffer*).

b) rsh, rexec, rcp a més presenten el problema que sota algunes condicions ni tan sols no necessiten contrasenyes (per exemple, si es fa des de llocs validats en fitxer .rhosts), amb la qual cosa tornen a ser insegurs, i deixen grans portes obertes.

L'alternativa és usar clients i servidors segurs que suportin l'encriptació dels missatges i autenticació dels participants. Hi ha alternatives segures als servidors clàssics, però actualment la solució més usada és mitjançant la utilització del paquet OpenSSH (que es pot combinar també amb OpenSSL per a entorns web). OpenSSH ofereix solucions basades en les ordres ssh, scp i sftp, cosa que permet substituir els antics clients i servidors (s'utilitza un *daemon* denominat 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 del ftp.

Respecte a SSH, també cal tenir la precaució d'usar ssh version 2. La primera versió té alguns *exploits* coneguts; cal anar amb compte en instal·lar OpenSSH, i si no necessitem la primera versió, instal·lar només suport per a ssh2 (vegeu opció Protocol en el fitxer de configuració /etc/ssh/ssh_config).

A més, la majoria de serveis que deixem actius a les nostres màquines tindrien després que ser filtrats mitjançant tallafocs, per tal d'assegurar que no fossin utilitzats o atacats per persones a les quals no estaven destinats.

6. Detecció d'intrusions

Amb els sistemes de detecció d'intrusos [Hat01] (IDS) es vol fer un pas més endavant. 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.

Nota

Els sistemes IDS ens permeten la detecció a temps d'intrusos, usant els nostres recursos o explorant els nostres sistemes a la recerca d'errors de seguretat.

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

Hi ha des de sistemes basats en la informació local, per exemple, recopilen informació dels *logs* 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 produeixin comportaments estranys, com per exemple els casos de falsejament d'identitats, en què hi ha falsificacions d'adreces conegudes; controlar trànsit sospitós, possibles atacs de denegació de servei, detectant trànsit excessiu cap a serveis determinats, controlant que no hi ha interfícies de xarxa en mode promiscu (símptoma d'*sniffers* o capturadors de paquets).

Exemples

Alguns exemples d'eines IDS serien: Logcheck (verificació de *logs*), TripWire (estat del sistema mitjançant sumes md5 aplicades als arxius), AIDE (una versió lliure de TripWire), Snort (IDS de verificació d'estat d'una xarxa completa).

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

Els **TCP wrappers** [Mou01] són un programari que actua de mitjancer entre les peticions de l'usuari de servei i els *daemons* 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 al *daemon* (dimoni) del servei per un altre que actua de mitjancer (anomenat tcpd, normalment en /usr/sbin/tcpd). Quan aquest rep una petició, verifica l'usuari i el seu origen, per a determinar si la configuració del *wrapper* del servei permet o no permet utilitzar-lo. A més, incorpora facilitats de generar *logs,* o bé informar per correu dels possibles intents d'accés i després executa el *daemon* adequat assignat al servei.

Per exemple, suposem l'entrada següent en inetd:

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

Aquesta es canvia per:

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

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

També hi ha un mètode alternatiu de *wrapper* TCP, que consisteix que l'aplicació original es compili amb la biblioteca de *wrappers*. Llavors l'aplicació no ha de ser en inetd, i la podrem controlar igual que el primer cas amb la configuració que comentem a continuació.

Nota

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

El sistema de *wrappers* es controla des dels fitxers /etc/hosts.deny, en què especifiquem quins serveis deneguem i a qui, per mitjà d'opcions, com un petit intèrpret de dades per a desar la informació de l'intent, i el fitxer /etc/hosts.allow, en què solem col·locar quin servei utilitzarem, seguit de la llista de a qui deixem entrar (més endavant, al taller, en veurem un petit exemple). També hi ha les ordres tcpdchk, que testegen la configuració dels fitxers *hosts* (vegeu man hosts_access i hosts_options), per a comprovar que són correctes, és a dir, en testeja la configuració. L'altra ordre útil és tcpdmatch, a la qual donem un nom de servei i un possible client (usuari, i/o *host*), i ens diu què faria el sistema davant d'aquesta situació.

7.1. Tallafocs

Un tallafocs (o *firewall*) és un sistema o grup de sistemes que reforça les polítiques de control d'accés entre xarxes. El tallafocs pot estar implementat en programari, com una aplicació especialitzada corrent en un computador individual, o bé es pot tractar d'un dispositiu especial dedicat a protegir un computador o més d'un.

En general disposarem, o bé de l'aplicació de tallafocs per a protegir una màquina concreta connectada directament a Internet (ja sigui directa o per proveïdor), o bé podem col·locar a 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 computador amb dos o més targetes de xarxa que aïllin les diferents xarxes (o segments de xarxa) connectades, de manera que el programari de tallafocs a la màquina (o si fos un maquinari especial) s'encarregui de connectar els paquets de les xarxes i determinar quins poden passar o no, i a quina xarxa.

Aquest tipus de tallafocs se sol combinar amb un *encaminador* 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/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 fugues d'informació per alguns atacs. Una tercera possibilitat és la màquina individual connectada a una única targeta de xarxa cap a Internet, directa o bé mitjançant un proveïdor. En aquest cas, només volem protegir la nostra màquina d'atacs d'intrusos, de trànsit no desitjat o que es vegi compromesa al robatori de dades.

És a dir, en aquests casos podem comprovar que el tallafocs –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 tallafocs en general permet definir a l'usuari una sèrie de polítiques d'accés (quines són les màquines a què es pot connectar o les que poden rebre informació i el tipus d'informació) per mitjà del control dels ports TCP/UDP permesos d'entrada (*incomming*) o de sortida (*outcomming*). Alguns tallafocs tenen polítiques preconfigurades; en algun cas només diuen si es vol un nivell de seguretat alt, mitjà o baix; d'altres permeten personalitzar les opcions totalment (màquines, protocols, ports, etc.).

Nota

Els tallafocs permeten establir seguretat en l'àmbit de paquets i connexions de comunicació.

Una altra tècnica de vegades relacionada és NAT (*network address translation*). Aquesta tècnica proporciona una via per a ocultar les adreces IP usades a la xarxa privada i les oculta d'Internet, però manté l'accés des de les màquines. Un

dels mètodes típics és el denominat *masquerading*. En utilitzar NAT masquerading, un o diversos dispositius a la xarxa poden aparèixer com una única adreça IP vistos des de fora. Això permet connectar diversos computadors a un únic dispositiu de connexió externa; per exemple, un cas d'encaminador ADSL a casa permet connectar diverses màquines sense necessitat que el proveïdor ens proporcioni diferents adreces IP. Els encaminadors ADSL solen oferir algun tipus de NAT *masquerading*, i també possibilitats de tallafocs. 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 tallafocs (els casos vistos abans), la configuració de la xarxa privada interna que vulguem protegir.

7.2. Netfilter: IPtables

El nucli Linux (a partir de versions 2.4.x) proporciona un subsistema de filtratge denominat Netfilter [Net], que proporciona característiques de filtratge de paquets i també NAT. Aquest sistema permet usar diferents interfícies de filtratge, entre elles, la més usada es denomina IPtables. L'ordre principal de control és iptables. Anteriorment [Hata], es proporcionava un altre filtratge denominat ipchains en els nuclis 2.2 [Gre], el sistema tenia una sintaxi diferent (encara que semblant). En els nuclis 2.0 s'utilitzava un altre sistema denominat ipfwadm. Aquí (i en els exemples posteriors) tractarem només amb Netfilter/IPTables (és a dir, amb nuclis versions 2.4/2.6).

Nota

Netfilter, vegeu:
http://www.netfilter.org
Ipchains, vegeu:
http://www.netfilter.org/ipchains/

La interfície de l'ordre IPtables permet efectuar les diferents tasques de configuració de les regles que afecten el sistema de filtratge: ja sigui generació de *logs*, accions de pre i *postrouting* de paquets, NAT, i *forwarding* (reenviament) de ports.

Nota

IPTables aporta diferents elements com les taules, *chains*, i les pròpies regles.

Arrencada del servei amb: `/etc/init.d/iptables start`, si no estava configurat ja en el *runlevel*.

L'ordre `iptables -L` enumera les regles actives en aquell moment a cadascuna de les cadenes (*chains*). Si no s'han configurat prèviament, solen ser per defecte acceptar tots els paquets de les cadenes (o *chains*) d'*input* (entrada), *output* (sortida) i *forward* (reenviament).

El sistema de IPTables té com a nivell superior les taules. Cadascuna conté cadenes diferents, que al seu torn contenen regles diferents. Les tres taules que hi ha són: Filter, NAT i Mangled. La primera serveix per a les pròpies 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 gestionar-los. Concretament, si estem amb un sistema directament connectat a Internet, utilitzarem, en general, tan sols la taula Filter. Si el sistema és en una xarxa privada que ha de passar per un encaminador, *gateway* o *proxy* (o 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 la taula NAT i la Filter. Si la màquina és en un sistema de xarxa privada, però és una de les màquines internes, n'hi haurà prou amb la taula Filter, tret que sigui un servidor el que faci NAT a un altre segment de xarxa.

Si un paquet arriba al sistema, en el tallafocs es mirarà primer si hi ha regles en la taula NAT, per si cal fer traduccions d'adreces cap a la xarxa interna (les adreces normalment no són visibles cap a fora); 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 tenim regles *forward* per a saber cap a on els reenviem. Al contrari, 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 efectuarien la traducció d'adreces de manera que s'emmascaressin. En la taula NAT hi sol haver dues cadenes: *prerouting* i *postrouting*. En la primera, les regles han de decidir si cal fer algun *routing* del paquet i quina serà l'adreça de destinació. En la segona, es decideix finalment si el paquet es passa o no cap a l'interior (la xarxa privada, per exemple). I també hi ha 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, podeu veure man iptables).

A continuació, comentarem alguns aspectes i exemples de configuració de la taula Filter (per a les altres taules, es pot consultar la bibliografia associada).

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

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

en què *chain* és *input, output o forward,* i *target* la destinació que es donarà al paquet que es correspongui amb la regla. L'opció -A afegeix la regla a les existents. Amb això cal anar amb compte, ja que l'ordre importa. Cal col·locar les menys restrictives al principi, ja que, si primer posem una regla que elimini els paquets, encara que hi hagi una altra regla, aquesta no es tindrà en compte. L'opció -j permet decidir què farem amb els paquets, típicament *accept* ('acceptar''), *reject* ('rebutjar'), o *drop* (simplement 'perdre'l'). 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 *log*. Normalment, en aquest cas hi ha dues regles, una amb el *log* i una altra d'igual amb *accept*, *drop* i *reject*, per a permetre enviar al *log* la informació de paquets acceptats, rebutjats o perduts.

En posar la regla, també es pot usar l'opció -I (inserir) per a indicar una posició, per exemple:

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

que ens diu que es col·loqui la regla a la cadena *input* en tercera posició; i s'acceptaran paquets (-j) que provinguin (amb font, o *source* -s) de la subxarxa 10.0.0.0 amb netmask 255.0.0.0. Amb –D, de manera semblant, podem esborrar o un número de regla, o la regla exacta, 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 dir que es perdin tots els paquets per defecte, i s'habiliten després els que interessen, i moltes vegades també s'evita que hi hagi *forwarding* de paquets si no és necessari (si no actuem d'encaminador), això es podria indicar:

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

Tot això 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 concerneixen els paquets que vulguem utilitzar, i dir 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 utilitzin el nostre programari o serveis. Una altra tàctica seria deixar només actius aquells serveis que siguin imprescindibles i habilitar amb el tallafocs l'accés dels serveis a les màquines desitjades.

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

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

En què:

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

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

3) Els mateixos paquets que en el punt 2) però que provinguin de 10.x.x.x seran acceptats.

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

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

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

I es desen en:

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

A Debian es pot fer:

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

Cal anar amb compte que prèviament hi hagi el directori /var/log/iptables, que és en el qual es desen els fitxers; nom-regles serà un fitxer en el directori.

Amb (/etc/init.d/iptables load) podem carregar les regles (a Debian cal donar el nom del fitxer de regles), encara que Debian suporta uns noms per defecte de fitxers, que són *active* per a les regles normals (les que s'usaran en un *start* del servei) i *inactive* per a les que quedaran quan es desactivi el servei (es faci un *stop*). Una altra aproximació usada comunament és la de col·locar les regles en un fitxer *script* amb les trucades iptables que es necessitin i trucar-les per exemple col·locant-les en el *runlevel* necessari, o amb un enllaç cap a l'*script* en /etc/init.d.

7.3. Paquets de tallafocs en les distribucions

Respecte a eines de configuració més o menys automàtiques del 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:

- **lokkit**: en Fedora/Xarxa Hat, molt bàsic, només permet elegir l'usuari un nivell de seguretat que desitja (alt, mitjà o baix). Després ensenya els serveis

que són afectats i podem deixar, o no, passar el servei canviant la configuració per defecte. El mecanisme utilitzat per sota és iptables. Es pot veure la configuració final de regles que fa /etc/sysconfig/iptables que, al seu torn, és llegit pel servei iptables, que es carrega en arrencada, o bé per parada o arrencada mitjançant /etc/init.d/iptables amb les opcions start o stop. A Debian també és possible instal·lar-lo, però deixa la configuració de les regles en /etc/defaults/lokkit-l, i un *script* en /etc/init.d/lokkit-l. També hi ha una versió gràfica anomenada gnome-lokkit.

- **Bastille** [Proa]: programa de seguretat bastant complet i didàctic, ja que ens explica pas a pas diferents recomanacions de seguretat i si les volem aplicar, així com la configuració del tallafocs (el programa és interactiveBastille). Funciona en diverses distribucions, tant a Fedora com a Debian.

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

Nota

Vegeu:
http://www.fwbuilder.org/

- **firestarter**: una eina gràfica (Gnome) per a la creació del tallafocs. És molt completa, pràcticament maneja totes les possibilitats d'iptables, però així mateix, disposa d'assistents que faciliten la construcció intuïtiva del tallafocs. Disposa a més d'un monitor en temps real, per a detectar les intrusions.

Normalment, cadascun d'aquests paquets utilitza un sistema de regles que guarda en algun fitxer propi de configuració, i que sol arrencar com a servei o com a execució d'*script* en el *runlevel* per defecte.

7.4. Consideracions finals

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 bé falsejar dades que creïn confusió. A més, les necessitats de connectivitat modernes obliguen de vegades a crear programari que permeti el *bypass* (encreuament) dels tallafocs:

Nota

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

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

- Sovint s'utilitza (per exemple, els anteriors protocols i d'altres) una tècnica denominada *tunneling*, que bàsicament encapsula protocols no permesos, sobre la base d'altres que sí que ho estan; per exemple, si un tallafocs permet només el pas de trànsit http (port 80 per defecte), és possible escriure un client i un servidor (cadascun en un costat diferent del tallafocs) que

parlin qualsevol protocol conegut per ambdós, però que a la xarxa és transformat en un protocol http estàndard, amb el qual, el trànsit pot creuar el tallafocs.

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

Així, encara que els tallafocs són una molt bona solució a la major part de 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 (i confiar-hi) una única solució, i esperar que ens protegeixi absolutament; cal examinar els diversos problemes, plantejar solucions que els detectin a temps i polítiques de prevenció que ens curin en salut, pel que pugui passar.

8. Eines de seguretat

Algunes de les eines 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 fer-ho amb IP de tercers, ja que aquests podrien entendre-ho com a intrusions i demanar-nos responsabilitats, o al nostre ISP, o avisar les autoritats competents que ens investiguin o que tanquin el nostre accés.

A continuació, comentem breument algunes eines, així com alguns usos que poden utilitzar:

a) 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 "fer" una foto del sistema, poder comparar després qualsevol modificació introduïda i comprovar que no hagi estat corromput 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 elegir un subconjunt d'arxius que siguin importants en el sistema, o fonts de possibles atacs. TripWire és propietari, però hi ha una eina lliure equivalent anomenada AIDE.

b) Nmap [Insb]: és una eina d'escaneig de ports per a xarxes grans. Pot escanejar des de màquines individuals a segments de xarxa. Permet diversos models d'escaneig, depenent de les proteccions que tingui el sistema. També té tècniques que permeten determinar el sistema operatiu que usen les màquines remotes. Pot fer servir diferents paquets de TCP i UDP per a provar les connexions. Hi ha una interfície gràfica denominada xnmap.

c) Wireshark [Wir] (abans anomenat Ethereal): és un analitzador de protocols i captura el trànsit de la xarxa (actua d'*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 reconstruir fins i tot el trànsit d'una sessió sencera d'un protocol TCP.

d) Snort [Sno]: és un sistema IDS que permet fer una anàlisi de trànsit en temps real i desar *logs* dels missatges. Permet fer una anàlisi dels protocols, recerques per patrons (protocol, origen, destinació, etc.). Pot ser usat 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 o bé anotar la situació (*log*) o bé produir un avís (*alert*) o descartar la informació (*drop*).

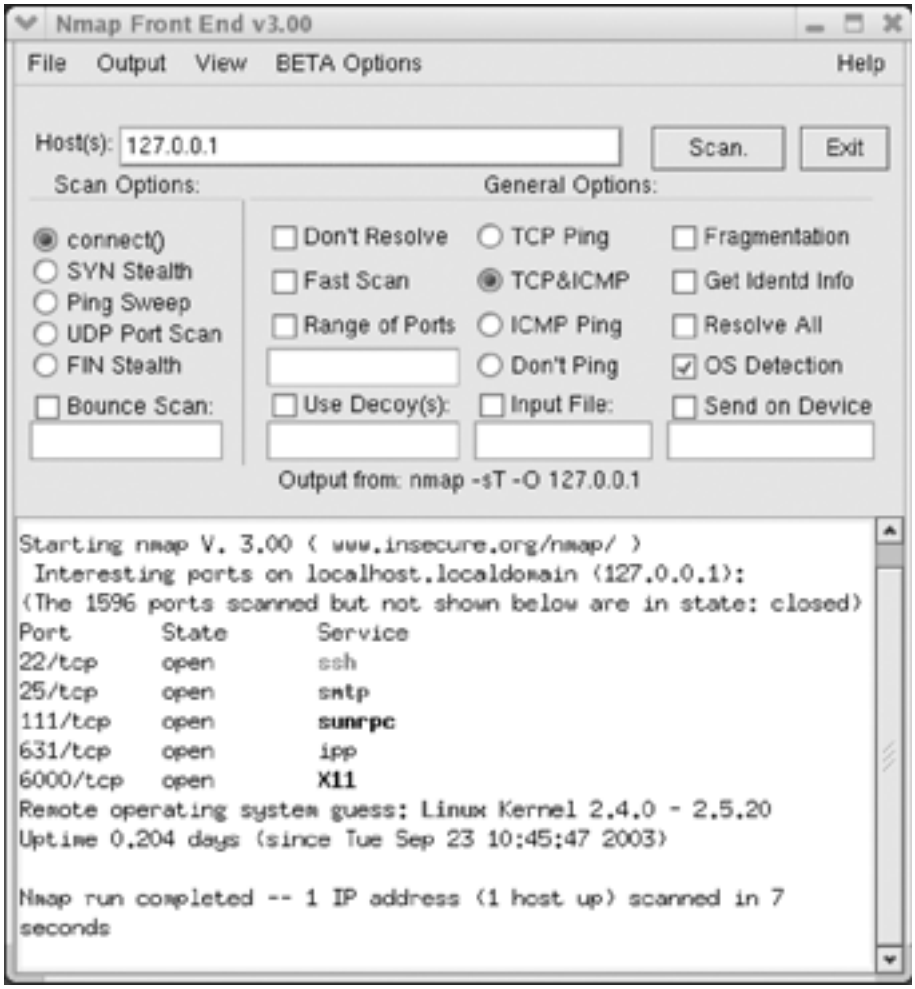

Figura 2. xnmap analitzant serveis locals

e) Nessus [Nes]: és un detector de vulnerabilitats conegudes (prova diferents tècniques d'intrusió) i assessora sobre com millorar la seguretat per a les detectades. És un programa modular que inclou una sèrie de *plugins* (més d'11.000) per a les diferents anàlisis. Utilitza una arquitectura client-servidor, amb un client gràfic que mostra els resultats i el servidor que fa les diferents comprovacions sobre les màquines. Té capacitat per a examinar xarxes senceres. Genera informes de resultats en forma de *reports* que es poden exportar a diferents formats (per exemple, HTML). Fins al 2005, Nessus 2 era una eina lliure, però la companyia va decidir convertir-la en propietària en la versió Nessus 3. Encara en GNU/Linux, es continua utilitzant Nessus 2, que segueix amb llicència GPL, i una sèrie de *plugins* que es va actualitzant. Nessus 3 com a eina propietària per a GNU/Linux, és més potent i és utilitzada àmpliament, i és una de les eines més populars de seguretat, normalment es manté lliure de cost una versió amb els *plugins* menys actualitzats que la versió de pagament.

Podem trobar moltes més eines de seguretat disponibles. Un bon lloc per examinar és http://sectools.org, en què els creadors del Nmap mantenen una llista d'eines populars votades pels usuaris.

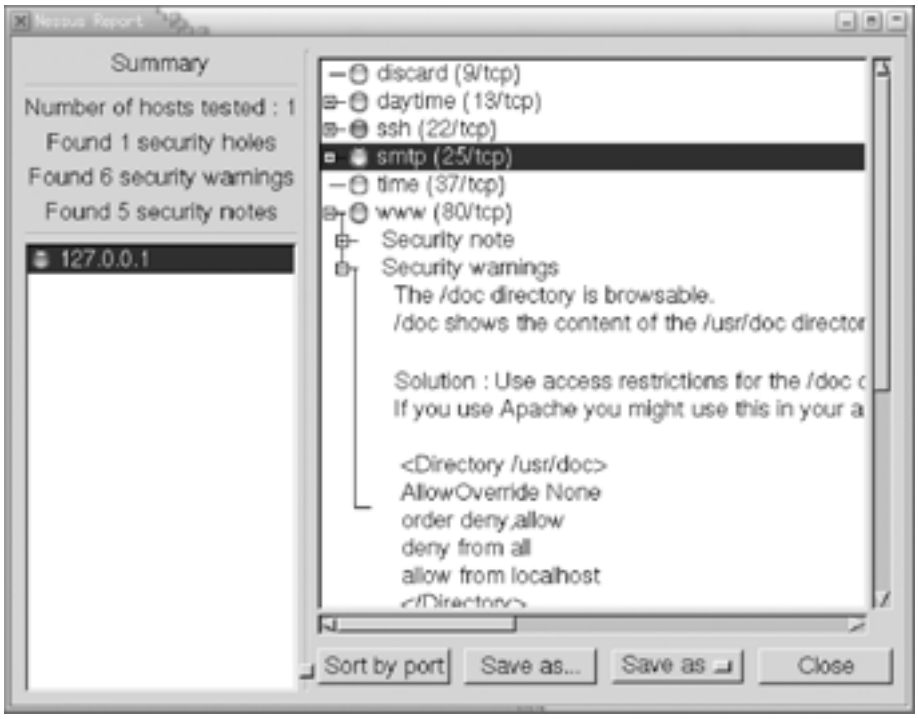

Figura 3. Client de Nessus mostrant l'informe de vulnerabilitats i possibles solucions

9. Anàlisi de *logs*

En observar els fitxers de *log* [Ano99][Fri02], ens podem fer una idea ràpida de l'estat global del sistema, així com de les últimes ocurrències, i detectar accessos (o intents) indeguts. Però també es pot tenir present que els *logs*, si ha succeït una intrusió real, poden haver estat netejats o falsejats. La majoria d'arxius de *log* rauen en el directori /var/log.

Molts dels serveis poden tenir *logs* propis, que normalment s'estableixen en la configuració (mitjançant el fitxer de configuració corresponent). La majoria solen utilitzar les facilitats de *log* incorporades en el sistema Sys *log,* mitjançant el dimoni Syslogd. La seva configuració rau en /etc/syslog.conf. Aquesta configuració se sol establir per nivells de missatges: hi ha diferents tipus de missatges segons la seva importància. Normalment solen aparèixer nivells debug, info, err, notice, warning, err, crit, alert, emerg, que més o menys aquesta seria l'ordenació d'importància dels missatges (de menor a major). Normalment, la majoria dels missatges es dirigeixen al *log* /var/log/messages, però es pot definir que cada tipus vagi a fitxers diferents i també es pot identificar qui els ha originat; típicament el nucli, correu, *news*, el sistema d'autenticació, etc.

Per tant, es pot examinar (i en tot cas adaptar) la configuració de Syslog per a conèixer en quins *logs* podem trobar/generar la informació. Un altre punt important és controlar el seu creixement, ja que segons els que hi hagi actius i les operacions (i serveis) que es facin en el sistema, els *logs* poden créixer bastant. A Debian i Fedora es controla mitjançant logrotated, un dimoni que s'encarrega periòdicament de fer còpies i comprimir-les dels *logs* més *anti*cs; es pot trobar la seva configuració general en /etc/logrotate.conf, algunes aplicacions fan configuracions específiques que es poden trobar en el directori /etc/logrotate.d.

Comentem en els punts següents alguns fitxers de *log* que caldria tenir en compte (potser els més utilitzats):

a) /var/log/messages: és el fitxer de *log* per defecte del dimoni Syslogd, però caldria revisar la seva configuració, no sigui que s'hagi canviat a un altre lloc, o n'hi hagi diversos. Aquest fitxer conté una àmplia varietat de missatges d'orígens diversos (diferents dimonis, serveis o el mateix nucli); tot el que s'observi com a anòmal hauria de ser verificat. En cas que hi hagi hagut una intrusió, mireu entorn de la data de la intrusió.

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

c) /var/log/wtmp: cada vegada que un usuari entra en el sistema, surt, o la màquina reinicia, es desa una entrada en aquest fitxer. És un fitxer binari del qual obté informació l'ordre, en la qual s'esmenta quins usuaris van entrar o van sortir, quan i on es va originar la connexió. Pot ser útil per a buscar on (en quins comptes) es va originar la intrusió o detectar usos de comptes sospitosos. També hi ha una variació de l'ordre anomenada lastb, que enumera els intents de *login* que no es van poder validar correctament i s'usa el fitxer /var/log/btmp (pot ser necessari crear-lo si no hi és). Aquests mateixos errors d'autenticació també se solen enviar al *log* auth.log. De manera semblant, l'ordre lastlog utilitza un altre fitxer /var/loglastlog per a verificar quina va ser l'última connexió de cadascun dels usuaris.

d) /var/log/secure: se solen utilitzar a Fedora per a enviar els missatges dels tcp *wrappers* (o els tallafocs). Cada vegada que s'estableix una connexió a un servei d'inetd, o bé en el cas de Xarxa Hat 9, per a xinetd (amb la seva pròpia seguretat) s'afegeix un missatge de *log* a aquest fitxer. Es poden buscar intents d'intrusos de serveis que no se solen usar, o bé màquines no familiars que s'intenten connectar.

En el sistema de *logs*, una altra cosa que caldria assegurar és que el directori de *logs* /var/ *log,* només es pugui escriure pel *root* (o *deamons* associats a serveis). D'una altra manera, qualsevol atacant podria falsificar la informació dels *logs*. Tot i així, si l'atacant aconsegueix accés a *root*, pot, i acostuma a fer-ho, esborrar les pistes dels seus accessos.

10. Taller: anàlisi de la seguretat mitjançant eines

Farem alguns dels processos descrits sobre un sistema Debian, per a configurar millor la vostra seguretat.

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

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

obtenim:

```
root@maquina:~# nmap -sUT -O localhost

starting nmap 3.27 ( www.insecure.org/nmap/ ) at 2003-09-17
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 2.4.0-2.5.20
Uptime 2.011 days (since Mon Sep 15 11:14:57 2003)
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 hi podria haver més: telnet, ftp, finger...), tant en protocols tcp com udp. Alguns serveis com discard, daytime, time poden ser útils alguna vegada, però normalment no haurien d'estar oberts a xarxa, ja que es conside-

ren insegurs. SMTP és el servei de reenviament, i *routing*, el de correu; 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 recerca de ports actius a l'escolta, això es pot fer amb netstat -lut.

El servei nmap també es pot aplicar 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 pròpia 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 fora.

Ara anirem a /etc/inetd.conf per a desactivar aquests serveis. Busquem línies com:

```
discard stream tcp nowait root internal
```

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

i els col·loquem un # al principi de la línia (només a aquells serveis que vulguem desactivar i sapiguem què fan realment (consulteu pàgines *man*), o se'n recomani la desactivació. Un altre cas de desactivació recomanat seria el dels serveis de ftp, telnet, finger... i usar 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
```

Del que ens queda, tenim el servei ssh, que volem deixar, i el servidor de web, que de moment el pararem:

```
/etc/init.d/apache 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 sobre a què es dedica un port determinat, podem mirar-ho 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-hi 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 el sistema no ha pogut determinar quin nmap està associat al port. Intentarem veure-ho nosaltres. Per 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 al que ens interessa, que són les connexions obertes i qui les usa. Intentarem 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 trucades RPC. El sistema de trucades RPC (*remote procedure call*) permet utilitzar el mecanisme de trucades remotes entre dos processos que són en diferents màquines. portmap és un *daemon* que s'encarrega de traduir les trucades que hi arriben pel port als nombres de serveis RPC interns que es tinguin, i utilitzen diferents servidors com NFS, NIS, NIS+.

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

```
root@maquina:~# rpcinfo -p
programa vers      proto        puerto
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
```

en què 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...). Per tant, portmap podria ser víctima d'intents de DoS que podrien provocar errors en els serveis o fer-los caure. Normalment, protegirem portmap per mitjà d'algun tipus de *wrapper* i/o tallafocs. Si no utilitzem,

i no tenim previst utilitzar-los, serveis NFS i NIS, el millor és desactivar completament portmap, i treure'l del *runlevel* en què s'activi. També els podem aturar momentàniament amb els *scripts* (a Debian):

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

i donar-los el paràmetre *stop* per a aturar els serveis RPC (en aquest cas NFS).

A continuació, controlarem la seguretat a partir d'un *wrapper* senzill. Suposarem que volem deixar pas mitjançant ssh d'una màquina determinada, l'anomenem 1.2.3.4 (adreça IP). Tancarem 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). Anem a fer un *wrapper* (suposem que els TCP *wrappers* estan ja instal·lats), modificant els fitxers hosts.deny -j 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 (atenció, aquells relacionats amb inetd) (primer all) a tots (all), i l'acció per prendre serà esbrinar qui ha demanat el servei i a quina màquina, i enviarem un correu al *root* per informar-lo de l'intent. Podríem escriure també un fitxer de *log*... Ara, en /etc/hosts.allow:

```
sshd: 1.2.3.4
```

habilitem l'accés per la màquina IP 1.2.3.4 al servidor sshd (del ssh). També podríem col·locar l'accés a portmap, només faria falta una línia portmap: la_ip. 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 intent determinat, per exemple:

```
root@maquina:~# tcpdmatch sshd 1.2.3.4

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

Ens diu que seria concedit l'accés. Un detall és que ens diu que sshd no hi és en el inetd.conf, i si ho verifiquem, veiem que així és: no s'activa pel servidor inetd, sinó per *daemon* en el *runlevel* que som. A més (a Debian), aquest és un cas de dimoni que està compilat amb les biblioteques de *wrappers* incloses (i, per tant, no necessita tcpd per a funcionar). A Debian hi ha diversos *daemons*, com per exemple: ssh, portmap, in.talk, rpc.statd, rpc.mountd, entre d'altres. Això permet assegurar aquests *daemons* mitjançant *wrappers*.

Una altra qüestió per verificar són les connexions actuals existents. Amb l'ordre netstat -utp, podem fer una llista de les connexions tcp, udp establertes amb l'exterior, ja sigui 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 establertes mitjançant lsof -i, de manera que s'hi poden detectar accessos indeguts a fitxers.

També podríem utilitzar un tallafocs per a processos similars (o bé com a mecanisme afegit). Començarem veient com estan les regles del tallafocs 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 tallafocs no està col·locant cap restricció en aquest moment, i permet l'entrada, sortida i reenviament de tots els paquets.

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

En el cas de posar els tallafocs, podem considerar si usar aquest mecanisme com a únic i treure els *wrappers*: es podria fer, 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 tallafocs podria ser més o menys suficient, però, per si de cas, més mesures de seguretat no vénen malament. I en cas que el tallafocs no estigués ben dissenyat i deixés escapar alguns paquets, el *wrapper* seria la mesura, en l'àmbit 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, la fossa i les primeres muralles serien el tallafocs, i la segona muralla de contenció, el *wrapper*.

Activitats

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

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

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

4. Suposem que volem deshabilitar un servei del qual sabem que té l'*script* /etc/init.d/servicio que el controla: volem desactivar-lo en tots els *runlevels* en els quals es presenta. Com trobem els *runlevels* en els quals hi és? (per exemple, buscant enllaços a l'*script*).

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

6. Practiqueu l'ús d'algunes de les eines de seguretat descrites (nmap, nessus, etc.).

7. Quines regles IPtables serien necessàries per a una màquina en la qual només volem accés per SSH des d'una adreça concreta?

8. I si volem únicament un accés al servidor web?

Altres fonts de referència i informació

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

[Peñ] Imprescindible per a Debian, molt bona descripció per a seguir pas a pas la configuració de seguretat, [Hatb] seria equivalent per a Fedora/Xarxa Hat.

[Mou01] Referència de seguretat excel·lent per a Xarxa Hat (aplicable també per a Debian).

[Hat01] Llibres seguretat GNU/Linux, que cobreixen amplis aspectes i tècniques.

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

[Sei] Guia pas a pas que identifica els punts clau que cal verificar i els problemes que puguin sorgir.

[Net] Projecte Netfilter i IPTables.

[Ian] Una llista de ports TCP/IP.

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

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

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

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

[NSAa][FBI][USA] Algunes “policies” de cibercrim als Estats Units.