Universitatea Politehnica Bucuresti
Facultatea de Electronica, Telecomunicatii si Tehnologia Informatiei

Securitatea Windows – Linux

Bucuresti 2012

Prefata

In lucrarea de fata sunt prezentate cateva aspecte legate de securitatea Windows – Linux, si este intocmita de urmatorii studenti:

I. Implementarea securitatii la Linux si Windows: comparatie
(Gheorghe Corina Roxana)

II. SELinux
(Moraru Andrei)

III. Apelurile de sistem legate de securitate la Linux
(Teodorascu Paula Daniela)

IV. Kerberos
(Neagu Samuel)

V. Cele mai intalnite probleme in securitatea sistemelor Linux (probleme si solutii)
1. Porturi de retea deschise
2. Versiunile vechi de software
3. Programe nesigure sau configurare defectuos
4. Tipuri de virusi **(Langa Mihai)**
1. Boot-area securizata
2. Insuficienta resurselor si alocarea proasta a prioritatilor
(Clabescu Catalin)

VI. Concepte fundamentale: exemple si comparatii intre Linux si Windows, UID si SETUID la liunx, SID si jetonul de acces la Windows
(Stancescu Mihai Alexandru)

VII. Functiile principale din Win32 API legate de securitate
(Radulescu Vlad Alexandru)

Cuprins

VII. Protocolul
VIII. Principii
IX. Implementare Kerberos
X. Slabiciune Kerberos
XI. Kerberos pentru utilizator
XII. Concluzii

V. Cele mai intalnite probleme in securitatea sistemelor Linux (probleme si solutii)

1. Porturi de retea deschise
2. Versiunile vechi de software
3. Programe nesigure sau configurare defectuos
4. Insuficienta resurselor si alocarea proasta a prioritatilor
5. Boot-area securizata
6. Tipuri de virusi

VI. Concepte fundamentale: exemple si comparatii intre Linux si Windows, UID si SETUID la liunx, SID si jetonul de acces la Windows

a. Comparatii LINUX/WINDOWS
b. Super-User, UID, GID si Setuid in Linux
c. SID in Windows
d. IV. Jetonul de acces in Windows

VII. Functiile principale din Win32 API legate de securitate

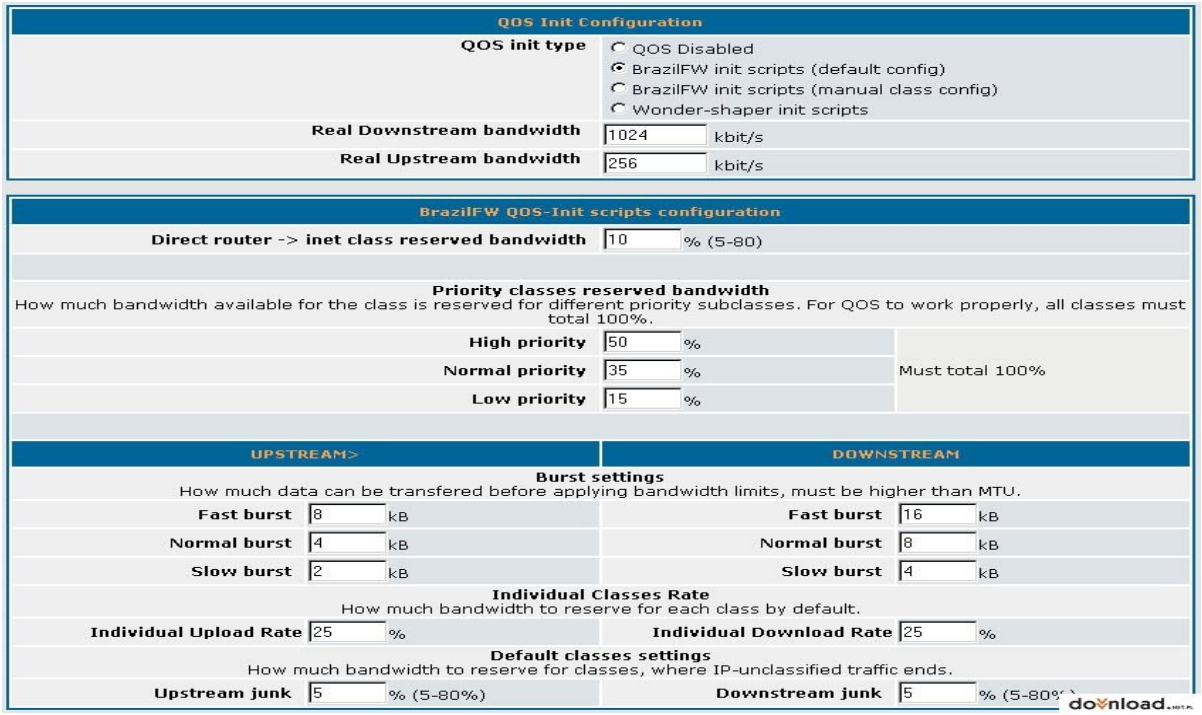
I. Implementarea securitatii la Linux si Windows: comparatie (Gheorghe Corina Roxana)

Arhitectura securitatii

Linux si Windows au o arhitectura a securitatii extreme de diferita. Firewall-ul fiecarui sistem de operare este un prim exemplu ce ilustreaza acest lucru.

Firewall-ul Windows-ului este unu minimalist, acesta filtrand doar pachetele din interior si find necesar sa fie construit simplu. Pentru un administrator de sistem cu experienta acesta este mult prea simplu sin u ofera suficiente facilitate.

Firewall-ul de Linux prezinta o functionalitate care poate face concurenta firewall-urilor comerciale. Acesta este extensibil, permitand folosirea de diferite filtrari in functie de nevoie. Linux este sistemul de operare preferat de majoritatea inginerilor si administratorilor de retea intrucat permite folosirea oricarei forme de NAT (Network Address Translation), translatiei de porturi si modificarea pachetelor inainte sau dupa rutare. Acest lucru permite proxiuri transparente, QoS de o calitate ridicata si politici de rutare diverse. Singura problema pe care o intalnim la firewall-ul de Linux este faptul ca e complex de configurat.



Configurarea Firewall-ului la Linux *

*Figura preluata de la http://download.net.pl/img/c038ee1cae516b4c800621faf89c615f.jpg

Structura fisierelor de sistem

Linux, ca si alte system de operare UNIX, structureaza fisierele de sistem in asa fel incat acestea se gaseasc in aceleasi directoare. Fisierele binare care ar trebui sa fie accesibile doar superutilizatorului sunt salvate in directoare diferite fata de cele accesibile tuturor utilizatorilor. Fisierele de date sunt statice si sunt salvate in directoare diferite fata de cele dinamice.

Permisiunea de acces si de proprietate asupra fisierelor de sistem este folosita pentru a impiedica utilizatorii sa ruleze sau sa acceseze anumite programe. Aceste mecanisme de securitate restrictioneaza si rularea programelor si daemon-urilor. Administratorii de sistem pot folosi unelte precum permisiuni de acces, proprietate, diferite atribute pentru a proteja cat mai bine aceste fisiere. De asemenea, anumite fisiere de sistem care contin fisiere statice sunt realizate astfel incat sa nu poata fi modificate, prin urmare nu pot fi atacate.

Windows-ul lasa impresia ca este structurat la fel dar, in realitate, nu este asa. Acest lucru se poate verifica usor in directorul Program Files (de pe partitia pe care este instalat sistemul de operare) unde fisierele binare sunt amestecate cu alte tipuri de fisiere. Datele de configuratie ale sistemului (insemnand registrii) se gasesc in acelasi loc ca si fisierele binare. Prin urmare, separarea diferitelor tipuri de fisiere este un dezavantaj al Windows-ului.
In continuare vom prezenta cate un exemplu de structura, pentru UNIX si pentru Windows.

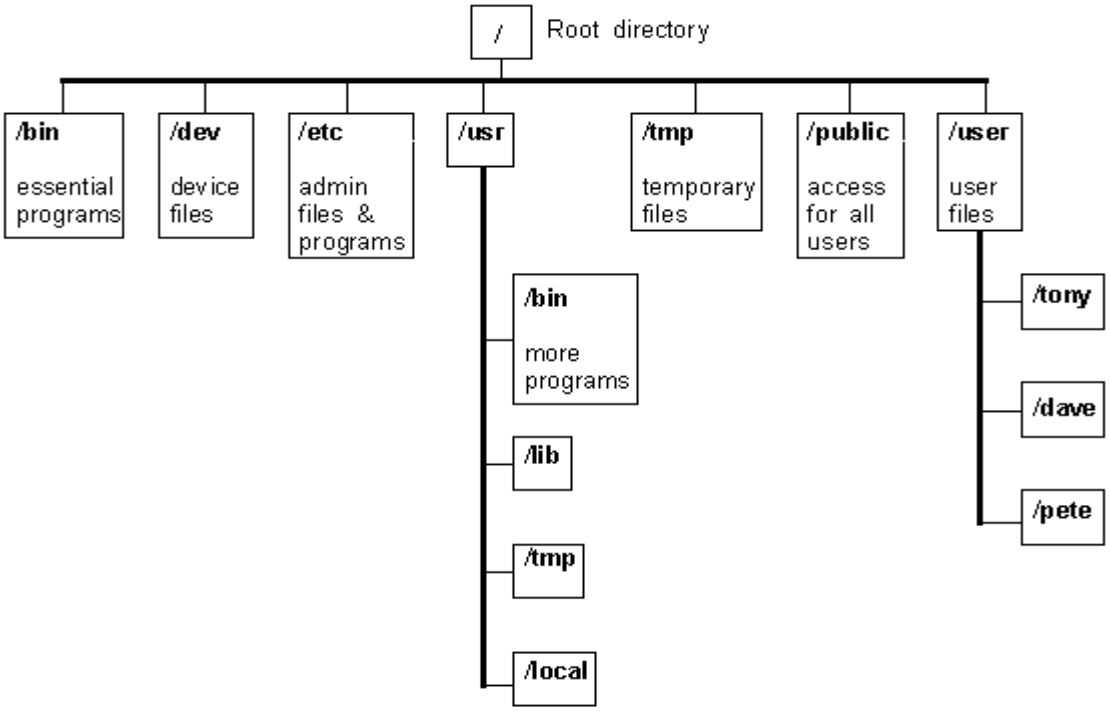
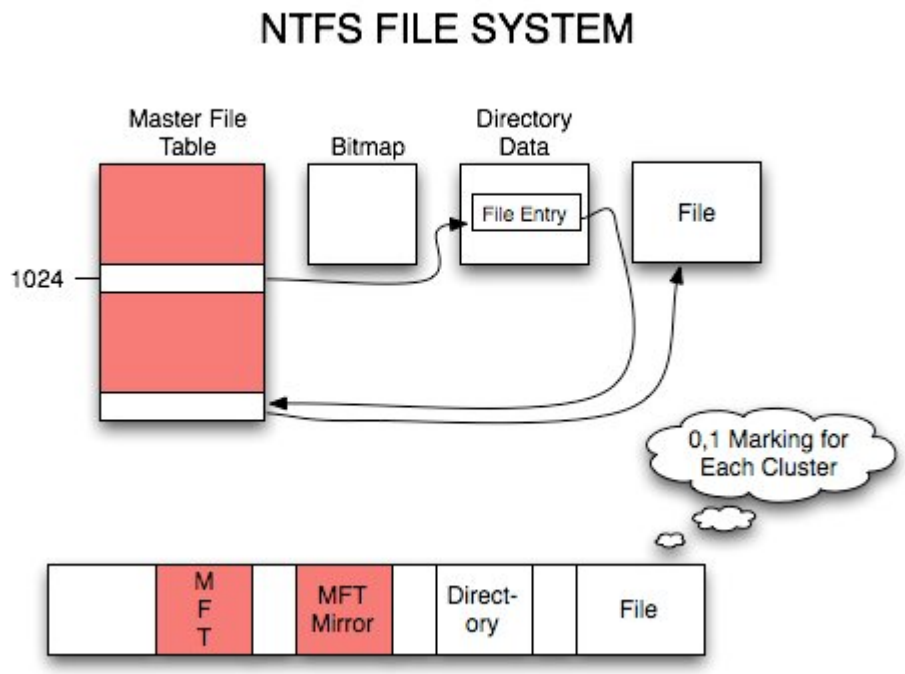


Diagrama fisierelor de sistem in UNIX *

*Imagine preluata de la : http://unix4humans.files.wordpress.com/2010/04/unix1.gif?w=600



Structura fisierului de sistem NTFS *

*Imagine preluata de la : http://thinkdifferent.typepad.com/photos/uncategorized/04ntfsfilesystem.png

Servicii/Daemons

Serviciile din Windows corespund daemons-urilor din UNIX. In Linux, cand se ruleaza daemons, administratorul de sistem allege utilizatorul sub care acesta ruleaza. Administratorii care se axeaza pe securitate creeaza cate un daemon separate pentru fiecare utilizator. Fiecare dintre aceste conturi de utilizator are propriile permisiune de acces la fisierul de sistem, astfel incat daemonul are acces doar la fisierele si directoarele absolut necesare.

Un exemplu de daemon in Linux:

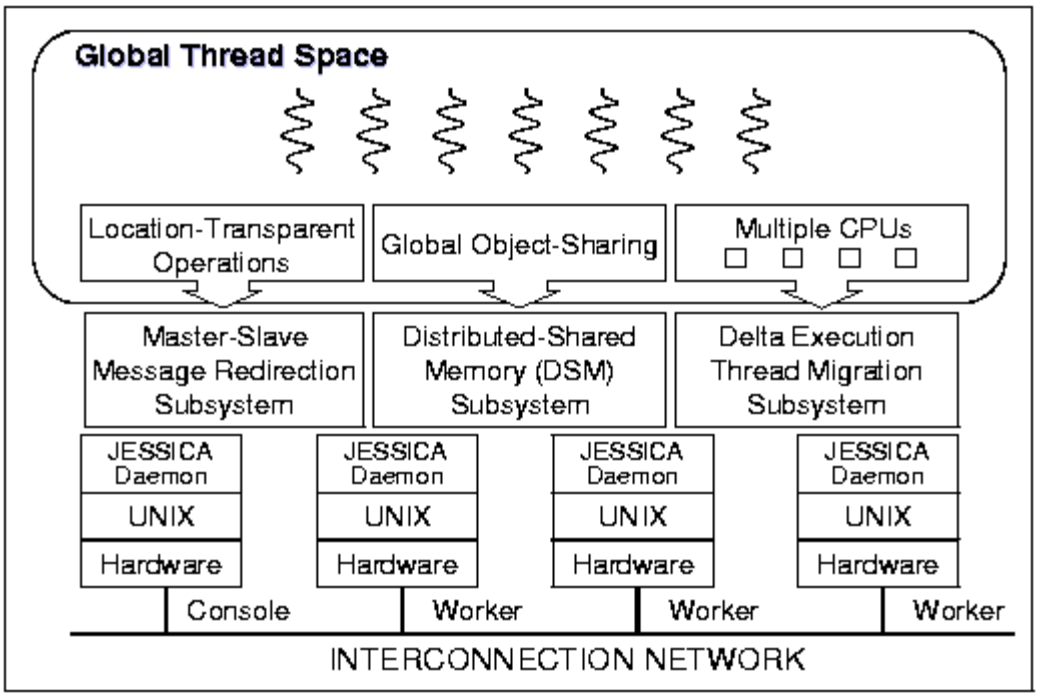


Figure 2: Global thread space creates an SSI illusion over a cluster of computers

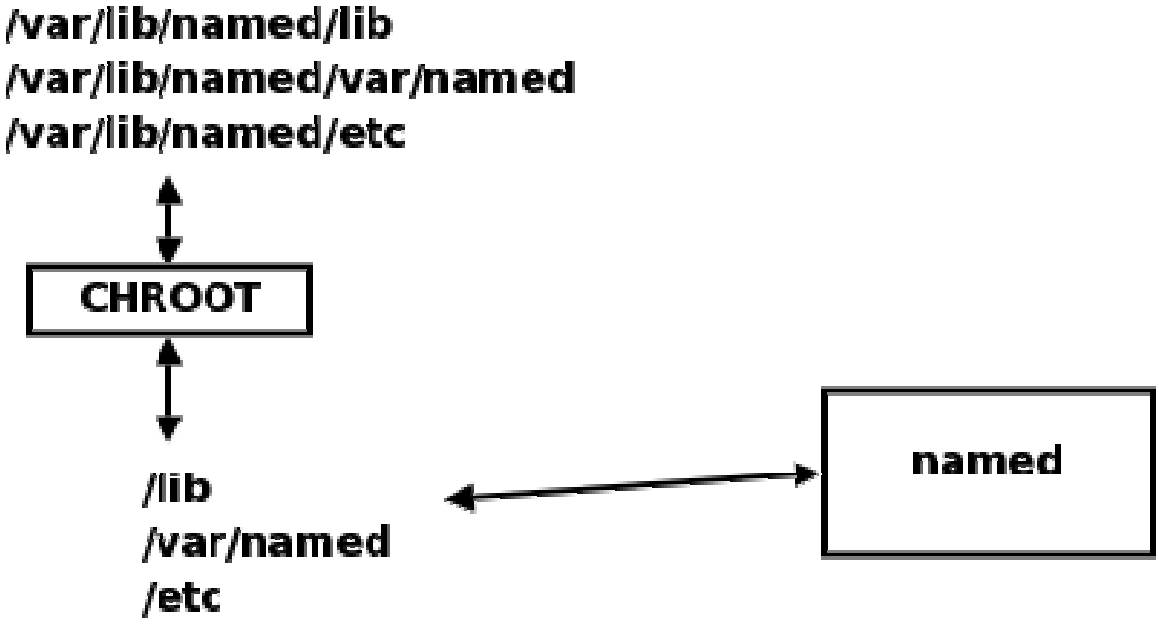
*

*Imagine preluata de la : http://www.srg.csis.hku.hk/homepage/images/jessica9.gif

Daca este atacat un daemon, efectele nu se vor simti decat in cadrul fisierului de sistem respectiv. Alte parti ale sistemului nu vor fi afectate in nici intr-un fel.

Majoritatea serviciilor din Windows ruleaza ca utlizatori privilegiati cu acces la aproape tot sistemul, ceea ce inseamna ca, in caz de atac asupra unuia dintre ele, intregul calculator este compromis.

Daca un atac asupra unui daemon are success, atunci se confera acces la acele portiuni ale fisierului de sistem la care utilizatorul daemon-ului avea permisiune de acces. Daca un utilizator ruleaza mai multe daemnouri, atunci si celelalte ar putea fi afectate. Solutia este de a oferi daemonurilor acces doar la portiune de sistem strict necesara. Acest lucru se realizeaza cu ajutorul “chroot”, dupa cum se ilustreaza in figura urmatoare:



*Imagine preluata de la : http://www.biznix.org/images/jail.png

II. SELinux

(Moraru Andrei)

Ce este SELinux ?

SELinux, acronim pentru Security Enhanced Linux, reprezintă o modalitate de îmbunătă□ire a securită□ii sistemelor de operare bazate pe Linux, fiind implementat dealtfel ca un modul de securitate (Linux Security Module). Dezvoltat de Agen□ia Na□ională pentru Securitate din America (National Security Agency – NSA), SELinux se bazează pe tehnica Mandatory Access Control (MAC) care permite implementarea unei mari varietă□i de poli□e de securitate. Pentru a putea folosi această tehnică, modulul SELinux adaugă o referin□ă către kernel-ul Linux-ului, cu ajutorul căreia implementează MAC-ul.

Dar cu ce ajută de fapt acest Mandatory Access Control ? Principiul pe care se bazează această tehnică este acela de a refuza execu□ia aplica□iilor ale căror permisiuni nu sunt explicit descrise, prin urmare nu se □tie exact care va fi efectul executării lor. Marele avantaj al SELinux-ului este însă acela că oferă posibilitatea de a seta anumite limite ale comportamentului unor aplica□ii ce se încadrează în categoria celor poten□ial dăunătoare sistemului de operare. Astfel, respectivele aplica□ii nu pot ie□i din acel model comportamental, indiferent care este menirea lor. De men□ionat ar mai fi ca SELinux con□ine toate informa□iile legate de pagubele care ar putea fi produse de o astfel de aplica□ie.

SELinux s-a dezvoltat foarte rapid □i a ajuns să fie utilizat pe foarte multe sisteme de operare bazate pe Linux, pe unele implicit(Fedora 3, Enterprise 4) iar pe unele doar daca utilizatorul dore□te acest lucru. Problema cea mare cu acest modul este legată de dificultatea utilizării lui, lucru de în□eles într-o anumită măsură dacă ne gândim că, de□i securitatea unui sistem de operare este de dorit, nu to□i utilizatorii care au implementat acest modul pe calculatorul lor personal doresc să-i folosească toate op□iunile.

Dificultatea men□ionată provine de la faptul că SELinux implementează for□at toate acele poli□e de securitate □i există posibilitatea ca ele să nu fie dorite de to□i utilizatorii. Desigur, există posibilitatea ca ele să fie dezactivate, însă în primele versiuni acest lucru era destul de greu de realizat, astfel că cei ce nu doreau acele func□ionalită□i ajungeau să dezactiveze sistemul în totalitate. Această problemă avea să se remedieze în timp gra□ie în primul rând dezvoltatorilor acestor poli□e de securitate, care au început să le implementeze astfel încât să supervizeze □i să gestioneze doar activitatea proceselor care reprezentau cu adevărat un pericol pentru sistem dacă ar fi func□ionat anormal (cele care accesau Internetul □i mai ales cele care se legau direct de func□ionarea sistemului). Acest gen de poli□e este util pentru că permite cre□terea nivelului de securitate fără a interevni în desfă□urarea activită□ilor zilnice ale utilizatorilor □i/sau administratorului calculatorului respectiv.

În al doilea rând, îmbunătă□irile se mai datorează □i unui salt tehnologic în dezvoltarea modulului, care a permis, cu ajutorul unor setări □i module suplimentare, ca utilizatorii să rezolve singuri anumite bre□e de securitate ce ar fi putut apărea.

După cum am men□ionat mai sus, SELinux utilizează foarte mult tehnica Mandatory Access Control, însă pe lângă aceasta, mai este utilizat conceptul de ”cel mai pu□in privilegiat” (least privileage). După cum îi spune □i numele, acest concept se bazează pe ideea că un program sau un utilizator trebuie să beneficieze doar de un minim de resurse care să-i permită func□ionarea în condi□ii optime. Dacă este utilizat acest concept, atunci sistemul de operare nu va permite accesul la alte resurse decât cele alocate cu ajutorul permisiunilor. Acest lucru ajută la limitarea daunelor produse ca urmare a unui atac, cauzat fie de un hacker, fie de programul în sine, dacă a fost scris cu inten□ia de a cauza bre□e de securitate. Securizarea poate fi îmbunăta□ită □i prin utilizarea unui firewall pe sistemele Windows □i Unix.

În SELinux există încorporat un ”monitor de referin□ă” (reference monitor) cu ajutorul căruia se verifică poli□ele de securitate de fiecare dată când are loc o comandă controlată. SELinux integrează arhitectura Flask pentru securitate cu scopul de a ajuta la controlul semnalul, al accesului la memorie, la sistemul de fi□iere etc cu scopul de a verifica dacă există sau nu bre□e de securitate.

La sistemele Unix, modelul de acces la controlul sistemului era de tip discret (Discretionary Access Control). Acest tip de model include un cont de super-utilizator care permite utilizatorilor să seteze gradul de acces la control pentru propriile date. Avantajele acestui tip de model sunt simplitatea □i faptul că este u□or de utilizat, însă nu poate fi utilizat pentru implementarea conceptului least privileage, ceea ce a dus la implementarea MAC, model care preia rolul atribuirii de permisiuni de la utilizator. MAC necesită ca procesele □i fi□ierele să fie etichetate iar administratorul de securitate va seta, cu ajutorul poli□elor, nivelul de acces pentru fiecare obiect, prin intermediul etichetelor. Aceste poli□e nu pot fi suprascrise de utilizator.

Modelul MAC este asociat de obicei cu securitatea multi-nivel (Multi-Level Security) unde un proces important nu î□i va împăr□i datele cu un alt proces care nu se află într-o categorie. Pe lângă aceasta, modelul MAC mai poate fi folosit, cum am spus □i mai sus, la implementarea conceptului least privileage □i, de asemenea, mai poate fi utilizat chiar □i atunci când se dore□te implementarea unei poli□e permisive. De men□ionat ar mai fi □i că spre deosebire de DAC, unde procesele mo□tenesc automat privilegiile utilizatorului, în cazul MAC această mo□tenire nu mai există, permi□ând astfel fiecărui proces să aibă propria poli□ă de securitate.

Un ultim lucru important care trebuie men□ionat este acela că SELinux permite implementarea oricărui tip de poli□ă de securitate, ceea ce a dus la dezvoltarea a numeroase aplica□ii ce au devenit apoi disponibile □i utilizatorilor, iar această flexibilitate este de fapt motivul pentru care acest modul a ob□inut sus□inerea comunită□ii open-source.

În concluzie, SELinux a reprezentat un mare pas înainte în ceea ce prive□te implementarea securită□ii în sistemele de operare, mai ales cele bazate pe Unix. Tehnicile de

implementare a acestei securită□i sunt cunoscute de decenii bune, iar încorporarea acestor tehnici a generat un entuziasm foarte mare în rândul dezvoltatorilor open-source.

Îmbunătă□irea securită□ii dispozitivelor ce utilizează sistemul de operare Android cu ajutorul SELinux

Android-ul este un sistem de operare tip open-source dezvoltat de cei de la Google. Scopul său principal este de a oferi, pentru telefoanele de tip smart-phone, servicii □i utilitare foarte asemănătoare cu cele oferite de un calculator □i anume: mesagerie, e-mail, posibilitatea căutării pe Internet cu ajutorul browser-elor etc. Însă aceste servicii aduc cu ele □i un poten□ial inamic, cauzat evident de atacuri menite să compromită confiden□ialitatea, integritatea □i valabilitatea utilizatorului □i a serviciilor de date.

Modalită□ile de atac cele mai des întâlnite sunt cele prin Bluetooth, Internet (Wi-Fi, 3G etc), USB □i alte periferice. Desigur, soft-uri □i tool-uri menite să contracareze atacuri tip spam □i malware există, însă utilizatorul obi□nuit de smart-phone-uri nu consideră telefonul său ca fiind un calculator în adevăratul sens al cuvântului, prin urmare tinde să nu utilizeze aceste programe.

Majoritatea programelor pentru Android se bazează pe kernel-ul Linux-ului, care asigură servicii de nivel jos pentru restul sistemului □i, în plus, asigură □i baza necesară cre□terii securită□ii în sistemele tip Android. Pe de altă parte, atunci când se utilizează un sistem de operare tip Linux, există posibilitatea apari□iei unor probleme de securitate a căror solu□ie este mai greu de găsit. Cea mai des întâlnită este problema cauzată de faptul că procesele □i serviciile de bază în Linux rulează de obicei la cel mai înalt nivel de privilegiu iar atacatorul poate ob□ine foarte u□or controlul asupra telefonului. De aceea este uneori necesar ca un smart-phone să con□ină anumite solu□ii menite să rezolve eventualele probleme apărute ca urmare a utilizării necorespunzătoare a telefonului de către cel care-l de□ine. Un exemplu în acest sens ar fi acela de a nu permite modificarea setărilor re□elei în care se află telefonul.

Sistemele obi□nuite tip Linux nu includ mecanismele necesare eliminării acestor riscuri, prin urmare este necesară utilizarea modulelor de securitate Linux (Linux Security Models – LSM) pentru îmbunătă□irea securită□ii. Utilizarea acestor module permite administratorului să limiteze accesul proceselor □i aplica□iilor la resursele sistemului, astfel încât eventualele aplica□ii dăunătoare sistemului să nu poată produce pagube. Această limitare a pagubelor se referă, în cazul telefoanelor, la protec□ia informa□iei confiden□iale □i la asigurarea func□ionării subsistemelor.

După cum am mai men□ionat, kernel-ul Linux-ului este cel pe care se bazează majoritatea programelor Android, programe a□ezate după rolul îndeplinit pe mai multe nivele. În prima categorie intră driverele pentru dispozitive, re□ele □i gestionarea sistemului de fi□iere, a memoriei, proceselor □i a consumului de putere. Google a implementat câteva îmbunătă□iri acestui kernel cu scopul de a-l face compatibil cu Android-ul.

Următorul nivel în structura acestei organizări a software-ului con□ine librăriile native Android-ului. Aceste librării sunt scrise în limbajul C/C++ □i sunt folosite de componentele sistemului din nivelele superioare. Urmează apoi ma□ina virtuală Dalvik □i librăriile nucleului. Ma□ina Delvik are rolul de a rula executabilele cu extensia .dex în timp ce librăriile, scrise în limbaj Java, au rolul de a asigura anumite librării □i func□ionalită□i specifice Android-ului.

Următorul nivel, scris tot in limbaj Java, con□ine utilitarele distribuite de Google □i extensii ale unor servicii.

Nivelul cel mai de sus este cel destinat aplica□iilor. Aplica□iile pentru Android se găsesc de obicei arhivate într-o arhivă cu extensia .apk, similară cu fi□ierele jar în sensul că în ea se găsesc toate resursele de cod □i comportament necesare aplica□iei respective.

Fiecare aplica□ie are nevoie de propriul proces Linux cu propria instan□ă a ma□inii Dalvik. De asemenea, la instalare i se atribuie un identificator. Prin urmare, cel pu□in teoretic, codul a 2 aplica□ii nu poate fi rulat simultan în acela□i proces □i cele două programe nu se vor strica unul pe celălalt. Procesele de sistem rulează la cel mai înalt nivel de privilegiu. În aceste procese este inclus □i **zygote**, proces de tip Dalvik destinat împăr□irii noilor procese de tip Dalvik ori de câte ori a aplica□ie de tip Dalvik este pornită. De exemplu, system_server, un astfel de proces, lansat de zygote, are rolul de a rula mai multe servicii native ale sistemului. Un alt exemplu este **init**, care reprezintă primul proces lansat de kernel □i care are rolul de a ini□ializa sistemul. Ar mai fi apoi **mountd**, care ini□ializeaza dispozitivele de stocare(carduri SD □i drive-uri USB) □i **rild**.

Linux furnizează numerose modalită□i de control al accesului, toate având la bază utilizatorul. Un utilizator, reprezentat în calculator printr-un număr de indentificare, de□ine obiectele create de el în acel calculator(procese, fi□iere, directoare).

În sistemul de permisiuni al Linux-ului, fiecare fi□ier are asociat un utilizator, care face parte la rândul lui dintr-un grup □i este identificat cu ajutorul numărului de indentificare □i 3 tipuri de permisiuni: read, write □i execute (rwx). Primul tip este for□at de către kernel să fie al utilizatorului, al doilea este al celor care fac parte din acela□i grup cu utilizatorul iar al treilea este cel asignat utilizatorilor din alte grupuri.

Fi□ierele din Android, fie executabile sau de sistem, se supun □i ele actesor reguli ale permisiunilor. În general, fi□ierele de sistem apar□in utilizatorului telefonului în timp ce fi□ierele aplica□ie apar□in celui care le-a creat. De asemenea, fi□ierele create de o aplica□ie nu vor fi accesibile decât dacă acest lucru este specificat în mod explicit în aplica□ie.

Se poate spune fără a gre□i că inima securită□ii aplica□iilor în Android este reprezentată de mecanismul permisiunilor aplica□iei. Dezvoltatorii folosesc acest mecanism pentru a for□a anumite restric□ii opera□iilor pe care o aplica□ie le poate efectua. Android-ul oferă aproximativ 100 de permisiuni built-in menite să supervizeze opera□iile de apelare, fotografiere, utilizarea Internetului etc, însă orice aplica□ie î□i poate declara permisiuni suplimentare, cu condi□ia de a le cere în mod explicit.

După cum am mai men□ionat, aplica□iile în Android sunt executate ca utilizatori diferi□i, fapt care duce la o exploatare foarte bună a resurselor sistemului întrucât două

aplica□ii nu se pot încurca una pe cealaltă, de□i există posibilitatea rulării mai multor aplica□ii sub acela□i nume de utilizator.

Problema la sistemele Android este ca mecanismele de securitate, native sau mo□tenite de la Linux, nu sunt suficiente pentru asigurarea protec□iei totale. Un exemplu în acest sens este o aplica□ie care, având permisiunea căutării pe Internet, poate accesa orice port, poate comunica cu toate protocoalele etc. Altfel spus, mecanismul permisiunilor pentru fi□iere le protejează de toată lumea în afară de utilizatorul principal al telefonului (de tip **root**).

Pentru rezolvarea acestor probleme s-a recurs la implementarea unor module de securitate Linux, deoarece suportă numeroare modele de control al accesului □i datorită acestui lucru pot refuza executarea anumitor ac□iuni, dacă acestea ar putea avea un efect negativ asupra resurselor sistemului.

Security-Enhanced Linux în sistemele Android

Modulul SELinux a fost cel ales pentru rezolvarea problemelor de securitate întrucât vine cu multe solu□ii □i tipuri de securitate □i este cel care se pliază cel mai bine pe sistemele Android deoarece asigură toate func□ionalită□ile necesare.

Una din solu□iile propuse de SELinux este aceea de a eticheta procesele □i fi□ierele din sistem. Etichetele atribuite sunt diferite de cele obi□nuite atribuite utilizatorilor □i grupurilor, un exemplu în acest sens fiind eticheta **init_t**, care este atribuită procesului de ini□ializare a sistemului. Rolul acestor etichete este de a ajuta la luarea unei decizii legate de autoriza□ia procesului respectiv iar atribuirea lor este definită într-un fi□ier de poli□ă. Eticheta unui proces se nume□te domeniu iar cea a unui fi□ier se nume□te tip.

În general accesul nu este permis decât dacă acest lucru este definit explicit. Regulile de acces se leagă în principal de domeniu, tip □i permisiunile necesare, acestea din urmă fiind alese dintr-un set de aproximativ 200 de opera□iuni pre-definite.

Un exemplu pentru a clarifica aceste no□iuni ar putea fi un player audio, care trebuie sa aibă acces la placa de sunet. Player-ul va avea eticheta **audio_player_t** iar pseudo-fi□ierul menit să reprezinte placa va fi etichetat **sound_card_t**. Poli□a de securitate va include o regulă ce va permite audio_player_t să citească, să scrie □i să afle atributele oricărui fi□ier de tip sound_card_t, dar nu va putea să îl □teargă sau să modifice permisiunile sau pe cel care-l de□ine.

La crearea unui fi□ier, acesta va avea aceea□i etichetă ca □i directorul în care a fost creat. Similar se întâmplă □i când rulăm un fi□ier, procesul rezultat având aceea□i etichetă ca □i programul executat. Însă, prin intermediul op□iunii type-transition, putem atribui altă etichetă procesului rezultant, atribuirea făcându-se automat.

SELinux are două moduri de operare: prin constrângere (în care sistemul impune poli□ele de securitate □i deciziile de a permite sau nu accesul) □i permisiv (în care violările de protec□ie sunt doar re□inute în log-uri, fără a se impune vreo ac□iune). De obicei o poli□ă

este construită pe modelul permisiv, putând fi pusă în modul de constrângere când sistemul nu mai înregistrează violări ale poli□ei respective.

Crearea unei poli□e este un proces dificil. O poli□ă direc□ionată va restric□iona accesul doar pentru aplica□iile cunoscute deja □i va lăsa în grija altor mecanisme de control aplica□iile necunoscute. O poli□ă strictă va îngrădi accesul pentru toate aplica□iile, fiind deci necesară definirea unui set de reguli pentru fiecare proces. În general, poli□ele stricte sunt recomandate serverelor importante, în timp ce desktop-urile rulează de obicei pe poli□e permisive.

Pentru sistemele Android se va folosi poli□a direc□ionată întrucât nu putem cunoa□te în avans fiecare proces al sistemului □i mai ales ceea ce va face respectivul proces în sistem. Astfel vor fi restric□ionate doar procesele doar procesele cunoscute, în timp ce procesele necunoscute vor rămâne la nivelul de privilegiu de bază, cu care au fost create.

După cum am mai men□ionat, SELinux este un model mai restrictiv când vine vorba de securitate, utilizatorii □i sistemul având alocate volumul minim □i gradul minim de resurse □i permisiuni de care au nevoie pentru a func□iona în condi□ii bune. Acest lucru, după cum am zis, este folositor în cazul în care un program este atacat, deoarece el nu poate afecta resursele din afara domeniului său de lucru. În al doilea rând, un proces neprivilegiat (un browser de exemplu) va putea apela tot doar ceea ce are nevoie, prin urmare sistemul este protejat □i de apelurile de programe neutilizate de acel proces.

Un rezultat pozitiv al restric□ionării controlului asupra unei aplica□ii este acela că în timpul editării poli□ei pentru acea aplica□ie se pot detecta eventuale vulnerabilită□i cu ajutorul sistemului. Un exemplu ar fi acela că există acces la descriptorul fi□ierului, acces ce ar permite apoi proceselor să ajungă în zone ale resurselor în care teoretic nu ar trebui să aiba acces.

Cercetătorii au permis folosirea modulului SELinux pe dispozitive embedded □i telefoane mobile Android. Björn Vogel □i Bernd Steinke au testat SELinux pe o tabletă Nokia 770. Ei au modificat sistemul de fi□iere astfel încât să suporte atribute noi □i au redus □i au actualizat poli□a pentru aceste noi cerin□e. Conform unor sondaje, sistemele SELinux suferă p pierdere de 7% în ceea ce prive□te performan□ele, de□i cei doi nu au testat să vadă dacă acest lucru este adevărat.

Japonezii Yuichi Nakamura □i Yoshiki Sameshima au adresat problema utilizării mari de resurse a SELinux. Ei au abordat 3 chestiuni. În primul rând, au creat un editor de poli□e care permite sistemului să scrie poli□e simplificate □i care ocupă mai pu□in spa□iu. În al doilea rând au folosit micro-optimizarea structurilor de date □i a codurilor pentru a reduce overhead-ul memoriei □i al CPU-ului. În al treilea rând, au redus overhead-ul memoriei în spa□iul utilizatorului eliminând necesitatea utilizării codului cerut de creatorii poli□elor de securitate din librăria SELinux.

Utilizarea SELinux în sistemele Android

Există numeroase situa□ii în care este recomandată folosirea SELinux în sistemele de operare Android. Pentru scenariile de mai jos vom presupune că există deja vulnerabilitatea □i vom arăta cum utilizarea SELinux va avea un efect benefic pentru sistemul nostru.

1.Protec□ia proceselor

Sistemele Android con□ine numeroase componente pe care atacatorii le pot utiliza în scopul ob□inerii unor privilegii inaccesibile în mod obi□nuit. Cinci procese rulează ca utilizator root, două procese sunt de sistem □i încă două rulează ca un utilizator radio. De□i unele dintre componente de încredere sunt codificate în kernel, sporindu-le nivelul de privilegiu, altele sunt suficient de mare pentru a cauza dificultatea validării lor formale. Costul asigurării securită□ii acestor componente este destul de mare.

Daemonii android care rulează ca utilizator sunt init, mountd, debuggerd, zygote □i installd. Daemonul installd este cel care se ocupă de dezarhivarea □i manevrarea fi□ierelor tip apk. El trebuie să ruleze ca root pentru că necesită setarea anumitor privilegii □i permisiuni pentru fi□ierele dezarhivate. Se poate întâmpla însă ca aceste fi□iere apk să provină de la hackeri □i să fie destinate distrugerii sistemului □i odată intrat în sistem, poate executa cod arbitrar utilizând privilegiile corespunzătoare nivelului root. Acest lucru va duce inevitabil la pagube, de la stricarea dispozitivului la accesul la informa□iile confiden□iale ale utilizatorului.
De asemenea, cel care a creat fi□ierul va avea posibilitatea ob□inerii accesului la toate resursele ma□inii.

2.Protec□ia fi□ierelor

Pentru a asigura integritatea sistemului, trebuie prevenită posibilitatea utilizatorilor neautoriza□i de a scrie în executabilele critice sau posibilitatea de a modifica fi□ierele de configura□ie. În plus, uneori e necesară interzicerea accesului □i la fi□ierele read-only întrucât pot con□ine informa□ii confiden□iale precum parole.

Metoda de bază utilizată în restric□ionarea accesului la fi□iere pentru sistemele de operare Linux este mecanismul permisiunilor pentru fi□iere. Însă trebuie men□ionat că utilizatorul root are posibilitatea suprascrierii acestor permisiuni. SELinux poate limita accesul la fi□iere al unui proces, chiar dacă acel proces este de tip root. Un exemplu este acela de a nu permite procesului init să scrie informa□ie pe disc.

Folosind acest control al accesului la fi□iere pe Android este necesar mai ales în mediile corporatiste, unde dispozitivele de□inute de personalul de management pot con□ine date care dacă sunt făcute publice pot cauza daune imense companiei respective. Linux tratează multe din func□ionalită□ile sistemului ca pseudo-fi□iere iar o consecin□ă a acestui fapt este aceea că metodele care protejează fi□iere pot de asemenea să protejeze resursele importante ale sistemului cum ar fi driver-e, socket-uri, directoare etc.

3.Protec□ia sistemului

În majoritatea sistemelor de operare, odată descoperită o vulnerabilitate, distribuitorul acelui sistem lansează un update care rezolvă bre□a respectivă de securitate. Acest lucru este valabil □i la Android, însă acest mecanism este de obicei dezactivat dacă sistemul este exploatat, un exemplu fiind cel dat de update-urile neoficiale, care-l dezactivează pentru a nu fi suprascrise de cele oficiale. În plus, se poate preveni repararea manuală a problemei de către un utilizator legitim. Prin urmare este foarte important ca noi să prevenim dezactivarea mecanismului de actualizare, chiar □i în cazul apari□iei unei vulnerabilită□i.

Componenta cea mai importantă a codului unui update se află pe o parti□ie neutilizată în func□ionarea obi□nuită a sistemului. Prin interizcerea posibilită□ii înlocuirii acestui cod, un utilizator poate actualiza manual sistemul. Pe lângă aceasta trebuie prevenită □i suprascrierea mecanismului de actualizare, adică să men□inem integritatea kernel-ului. SELinux face acest lucru prin limitarea ac□iunilor permise, cum ar fi posibilitatea încărcării modulelor de kernel, înlocuirea imaginii kernel-ului de pe disc, încărcarea unei poli□e diferite de securitate sau accesul direct al memoriei de sistem sau al discului.

Integrarea SELinux în sistemele Android

Această integrare s-a făcut cu ajutorul unei poli□e creată special pentru Android, cu un layout propriu al fi□ierelor, un sistem init propriu □i daemoni personaliza□i.

În cazul în care Android-ul nu suportă SELinux, este necesar un proces cu nivel de privilegiu de tip root care reconfigurează kernel-ul pentru a suporta SELinux.

O problemă mai dificilp este dacă dispozitivul Android nu are implementată o modalitate de a încărca poli□a SELinux. O solu□ie pentru aceasta este de a utiliza o unealtă embedded destinată încărcării poli□elor, însă aceasta nu rezolvă problema încărcării poli□ei prea devreme □i etichetarea incorectă a procesului init. Solu□ia cea mai bună a fost adăugarea în fi□ierul init.rc a trei noi comenzi:

a)**loadpolicy** – încarcă o poli□ă în kernel

b)**chcon** – modifică eticheta unui fi□ier

c)**context** – setează eticheta daemonilor lansa□i de procesul init.

Crearea unei poli□e SELinux pentru Android

Pentru a crea o poli□ă, primul lucru pe care-l facem este să □tergem toate modulele care nu au nicio relevan□ă cu poli□a noastră. Apoi, folosind script-ul mdp (make dummy policy) vom genera o poli□ă goală. Apoi, pornind sistemul în modul permisiv, vom adăuga reguli în acea poli□ă până când toate avertismentele legate de utilizarea dispozitivului dispar. Această poli□ă se bazează pe layout-ul fi□ierelor Android □i va restric□iona doar procesele existente pe Android.

O decizie fundamentală în design-ul sistemelor Android a fost aceea ca doar un proces Dalvik (zygote) să fie executat la un moment dat, restul fiind derivate din acesta □i împăr□ind memoria. Acest lucru limitează aplicabilitatea SELinux pentru că acesta nu poate eticheta procesul Dalvik întrucât acesta nu necesită execu□ia unui fi□ier.

SELinux permite proceselor să-□i modifice etichetele în mod explicit printr-o comandă care specifică numele etichetei. Această capabilitate este suportată de directiva type_change_policy, o func□ionalitate aflată în contrast cu tranzi□ia de tip implicită.

Portarea SELinux pe Android

Pentru a porta SELinux pe Android, sunt parcurse următoarele etape:

1.compilare kernel cu suport pentru SELinux

2.creare □i compilare poli□ă de securitate specifică pentru Android

3.modificarea codului de proces init □i a scriptului init.rc pentru a putea utiliza comenzi adi□ionale pentru a încărca poli□a □i a seta etichetele ini□iale

4.crearea unei noi imagini de disc ce va con□ine init-ul actualizat □i fi□ierele poli□ei.

În figura de mai jos sunt exemplificate scriptul init.rc (a), un exemplu de reguli de autoriza□ie (1b) □i ie□irea ps –Z într-un sistem Android cu SELinux-ul activat.

În figura 1b. pe primul rând vedem că procesele cu eticheta kernel_t pot lista fi□ierele cu eticheta rootfs_t. Pe rândul 3 observăm că init are dreptul de a crea o copie (un copil) după sine însu□i. În rândul 4 se observă ca procesul init_t poate crea un socket Unix. În rândul 7, adbd poate citi □i scrie în fi□ierul /dev/null, iar în rândul 8 se observă că poate scrie în fi□ierul /dev/ashmem. În fine, în rândul 9 vedem regula tranzi□iei de tip care specifică faptul că atunci când un proces init_t execută un fi□ier adbd_exec_t, procesul adbd rezultat va fi etichetat adbd_t. Fără această tranzi□ie, procesul adbd ar fi re□inut eticheta init_t.

În figura 1c, argumentul –Z îi transmite lui ps să afi□eze conextul de securitate al proceselor care rulează în sistem. În această listă, thread-urile kernel sunt etichetate kernel_t iar procesele critice sunt corect identificate □i etichetate.

```
on init
  loglevel 3
  # SELinux must be initialized very early
  mkdir / selinux
  mount selinuxfs selinuxfs / selinux
  loadpolicy / se-policy
  write /proc/1/attr/current system_u:system_r:init_early_t
  chcon / init system_u:object_r:init_exec_t
  chcon /sbin/abdb system_u:object_r:adbd_exec_t
  chcon /dev/null system_u:object_r:devnull_t
  chcon /dev/ ashmem system_u:object_r:ashmem_t
(a)
```

```
1  allow kernel_t rootfs_t:dir { search };
2  allow init_early_t tmpfs_t:filesystem { mount };
3  allow init_t init_t:process { fork setpgid };
4  allow init_t init_t:unix_stream_socket { create bind };
5  allow init_t tmp_t:sock_file { create setattr unlink };
6  allow init_t adbd_t:process { transition };
7  allow adbd_t devnull_t:chr_file { read write };
8  allow adbd_t ashmem_t:chr_file { read write };
9  type_transition init_t adbd_exec_t:process adbd_t;
(b)
```

```
# ps -Z
      PID CONTEXT                      STAT    COMMAND
1     system_u:system_r:init_t         S       /init
2     system_u:system_r:kernel_t       SW<     [ kthreadd ]
3     system_u:system_r:kernel_t       SW<     [ ksoftirqd / 0 ]
4     system_u:system_r:kernel_t       SW<     [events0/]
...
16    system_u:system_r:kernel_t       SW<     [ rpciod / 0 ]
17    system_u:system_r:unconfined_t   S       /system/bin/sh
18    system_u:system_r:servicemanager S       /system/bin/servicemanager
19    system_u:system_r:mountd_t       S       /system/bin/ mountd
20    system_u:system_r:debuggerd_t    S       /system/bin/ debuggerd
23    system_u:system_r:unconfined_t   S       zygote /bin/ app_process -Xzygote /s
24    system_u:system_r:mediaserver_t  S       /system/bin/ mediaserver
26    system_u:system_r:dbusd_t        S       /system/bin/ dbus -daemon --system --
27    system_u:system_r:installd_t     S       /system/bin/ installd
30    system_u:system_r:unconfined_t   S       /system/bin/ qemud
33    system_u:system_r:adbd_t         S       / sbin / adbd
51    system_u:system_r:unconfined_t   S       system_server
86    system_u:system_r:unconfined_t   S       com.android.phone
92    system_u:system_r:unconfined_t   S       android.process.acore
107   system_u:system_r:unconfined_t   S       com.android.mms
123   system_u:system_r:unconfined_t   S       com.google.process.gapps
138   system_u:system_r:unconfined_t   S       android.process.media
152   system_u:system_r:unconfined_t   S       com.android.alarmclock
(c)
```

Figura 1

Benchmarking

Pentru testarea performan□elor sistemelor au fost rulate mai multe benchmark-uri pentru a vedea efectele utilizării SELinux asupra utilizatorului dispozitivului. Benchmarking-ul s-a concentrat pe lărgimea benzi I/O, laten□ă, gradul de utilizare a CPU □i memorie. Testele au fost realizate pe un dispozitiv G1, bazat pe un procesor Qualcomm MSM7201A cu 192 MB RAM, memorie flash internă de 256MB □i un slot pentru carduri microSD. În ceea ce prive□te software-ul, s-a folosit versiunea RC30 a T-Mobile G1 USA cu modificări minime, activarea suportului pentru abd □i adăugarea unui shell root □i a unui kernel personalizat. Au fost verificate atât kernele cu SELinux activat (configura□ia minimă necesară func□ionării) cât □i cu el dezactivat (identic cu RC30). Rezultatele sunt în poza de mai jos:

Table 1. Evaluation results.

Measure	Without SELinux	With SELinux	Slowdown (%)
bonnie++			
Seq output, Chr (KBps)	52.3 ± 1.26	62.76 ± 1.89	−16.66*
Seq output, Chr (% CPU)	96.82 ± 0.92	96.33 ± 0.89	−0.50
Seq output, block (KBps)	3,674.64 ± 15.76	3,682.06 ± 25.41	−0.20
Seq output, block (% CPU)	13.42 ± 0.5	13.79 ± 0.48	+2.71
Seq output, rewrite (KBps)	2,623.7 ± 5.7	2,624.27 ± 7.64	−0.02
Seq output, rewrite (% CPU)	11.7 ± 0.47	11.94 ± 0.24	+2.07
Seq input, Chr (KBps)	205.45 ± 16.44	157.85 ± 12.97	+30.16*
Seq input, Chr (% CPU)	99 ± 0	98.97 ± 0.17	−0.03
Seq input, block (KBps)	8,914.91 ± 10.22	8,895.91 ± 10.54	+0.21*
Seq input, block (% CPU)	20.3 ± 0.47	20.61 ± 0.5	+1.49
Random, seeks (KBps)	104.75 ± 2.93	103.48 ± 2.29	+1.22
Random, seeks (% CPU)	41.82 ± 2.32	43.88 ± 2.52	+4.93
Seq output, Chr, latency (ms)	340.48 ± 110.83	326.27 ± 109.23	−4.17
Seq output, block, latency (ms)	1,368.21 ± 70.93	1,332.24 ± 108.47	−2.63
Seq output, rewrite, latency (ms)	2,175.7 ± 107.89	2,132.18 ± 234.56	−2.00
Seq input, Chr, latency (ms)	47.66 ± 4.16	60.29 ± 4.64	+26.51*
Seq input, block, latency (ms)	34.72 ± 14.59	33.99 ± 14.17	−2.11
Random, seeks, latency (ms)	225.45 ± 37.23	234.88 ± 36.81	+4.18
Lmbench			
Simple syscall (μs)	4.83 ± 0.05	4.78 ± 0.05	−0.86
Simple read (μs)	7.84 ± 0.12	9.64 ± 0.14	+22.96*
Simple write (μs)	7.25 ± 0.83	10.02 ± 1.72	+38.13*
Simple stat (μs)	97.39 ± 2.03	184.95 ± 5.21	+89.91*
Simple fstat (μs)	12.81 ± 0.18	27.9 ± 1.81	+117.75*
Simple open/close (μs)	139.19 ± 5.36	261.5 ± 6.75	+87.87*
Signal handler installation (μs)	6.64 ± 0.05	6.49 ± 0.02	−2.22
Signal handler overhead (μs)	39.29 ± 0.51	44.25 ± 1.82	+12.63*
Protection fault (μs)	6.77 ± 1.12	1.81 ± 1.31	−73.19*
Pagefault (μs)	146.09 ± 1.13	149.27 ± 3.05	+2.17*
Pipe bandwidth (MBps)	39.85 ± 1.01	38.61 ± 0.3	+3.23*
Pipe latency (μs)	312.91 ± 8.38	454.06 ± 13.66	+45.11*
AF_UNIX sock stream bandwidth (MBps)	43.95 ± 1.09	42.7 ± 0.5	+2.93*
AF_UNIX sock stream latency (μs)	408.68 ± 14.03	677.62 ± 14.25	+65.81*
Context switches, size = 16k (2)	724.01 ± 8.77	826.33 ± 10.83	+14.13*
Context switches, size = 16k (4)	1,057.49 ± 15.33	1,147.86 ± 28.63	+8.55*
Context switches, size = 16k (8)	1,107.91 ± 13.48	1,190.78 ± 23.16	+7.48*
Context switches, size = 16k (16)	1,120.88 ± 19.38	1,200.32 ± 20.3	+7.09*
Context switches, size = 16k (24)	1,106.83 ± 17.98	1,188.16 ± 22.52	+7.35*
Context switches, size = 16k (32)	1,099.9 ± 25.15	1,178.78 ± 21.03	+7.17*
Context switches, size = 16k (64)	1,077.3 ± 31.04	1,157.98 ± 35.53	+7.49*
Context switches, size = 16k (96)	1,054.5 ± 33.86	1,126.61 ± 49.04	+6.84*
Disk and memory usage			
Free RAM	53,177 ± 828 Kbytes	51,945 ± 929 Kbytes	−2.32
Init file size	98,260 bytes	98,296 bytes	+0.04
Init.rc	8,630 bytes	9,469 bytes	+9.72
Kernel size	1,379,032 bytes	1,477,188 bytes	+7.11
Binary policy	N/A	17,400 bytes	N/A

*Indicates significant changes ($\alpha = 0.0013$)

În concluzie, implementarea SELinux în sistemele Android întăre□te sistemul □i for□ează controlul accesului de nivel jos pentru procesele privilegiate critice ale Linux-ului. Prin alegerea acestei căi, putem proteja sistemul de acele scenarii în care atacatorul exploatează o vulnerabilitate a unui proces cu nivel mare de privilegiu. În plus, această cale este una cu cost redus □i este una din solu□iile care a câ□tigat foarte mult teren în ultima vreme.

III. Apelurile de sistem legate de securitate la Linux

(Teodorascu Paula Daniela)

I. Introducere

Pe masura ce din ce in ce mai multe afaceri se desfasoara pe internet , securitatea sistemelor devine din ce in ce mai importanta. Exista mai multe cai pe care un utilizator le poate aborda pentru a abuza de vulnerabilitati , atat local cat si de la distanta . Pentru a imbunatatii securitatea sistemului , se incearca stratificarea de diferite mecanisme de securitate una peste alata , pentru ca una din ele sa poata respinge un atac malitios. Aceste nivele de securitate includ firewall-uri pentru a restriciona accesul in retea , primitive de sistem de operare precum stive non-executabile sau protective la nivel de aplicatie precum separarea privilegiilor. Atat in teorie cat si in practica , securitatea creste odata cu numarul de nivele care trebuiesc ocolite pentru ca un atac sa aiba success. Firewall-urile pot preveni conectarea de la distanta si restrictiona accesul de exemplu in mod exclusive la un web server. Daca cineva insa exploateaza cu succes un bug in serverul web si dobandeste privilegiile, se poate folosi de ele in atacuri ulterioare pentru a dobandi si mai multe privilegii. Cu acces local la un sistem , un utilizator poate obtine privilegii de root , prin exploatarea programelor setuid , prin acces localhost la retea sau apeluri de sistem speciale. Din acest motiv , se incearca ingradirea drepturilor utilizatorilor si limitarea potentialelor pagube. Pentru sistemele de fisiere , ACL-urile (listele de control al accesului) permit limitarea drepturilor de scriere si citire asupra fisierelor. Desi ACL-urlie sunt mai versatile decat modelul de acces UNIX traditional , nu permit delimitare stricta , si sunt dificil de configurat.

Singurul mod de a face schimbari persistenete la sistem este prin apelurile de sistem . Acestea sunt calea catre operatiile privilegiate pe kernel. Prin monitorizarea si restrictionarea apelurilor de sistem , o palicatie poate fi impiedicata sa cauzeze pagube. Solutii bazate pe interpunerea apelurilor de sistem au fost dezvoltate si in trecut. Interpunerea apelurilor de sistem permite detectia intruziunilor ca violari ale politicilor si oprirea acestora .Ramane insa problema specificarii unei politici precise de acces.

II. Securitatea

Calculatoarele stocheaza mari cantitati de informatie care, adesea, trebuie sa ramana confidentiale, la dorinta utilizatorilor. Aceste informatii pot include posta electronica, planuri de afaceri, recuperari de taxe si multe altele. Este la indemana sistemului de operare sa gestioneze securitatea sistemului astfel incat fisierele cu pricina, de exemplu, sa fie accesibile numai utilizatorilor autorizati.

Ca un exemplu siplu, care sa ilustreze cum lucreaza securitatea, se considera sistemul de operare UNIX. Fisierele din UNIX sunt protejate prin coduri de protectie de cate 9 biti, care

se atibuie fiecarui fisier. Codul de protectie este compus din 3 campuri de cate 3 biti, unul pentru proprietar, unul pentru ceilalti membrii ai grupui proprietarului si unul pentru oricine altcineva. Fiecare camp are un bit pentru accesul la citire, un bit pentru accesul la sccriere si inca un bit pentru accesul la executie. Acesti 3 biti sunt cunoscuti ca bitii “rwx”.
Pe langa protectia fisierelor, mai sunt si alte multe probleme legate de securietate. Protectia sistemului impotriva intrusilor, umani sau ne-umani(virusi, de exemplu) este una dintre ele.
Securitatea in Linux este foarte importanta. Deoarece Linux este un sistem de operare destinat in primul rand retelelor, el este gandit sa functioneze prin acordarea de permisii fisierelor si directoarelor, si utilizatori cu drepturi restrictionate. Cel mai puternic utilizator este root-ul, avand drepturi depline asupra sistemului.
Cele mai importante reguli de securitate:
· Se recomanda sa se utilizeze o parola de minim 8 caractere, compusa din litere, cifre, si care sa nu reprezinte un cuvant, o data de nastere sau un nume. Pentru protejarea parolei de root se mai folosesc sistemele de securizare shadow si MD5. MD5 este un algoritm complex de incriptare a parolei. Shadow este un sistem prin care parolele retinute in fisierul /etc/passwd sunt transferate intr-un fisier numit /etc/shadow;
· Sa nu se instaleze programe care nu sunt necesare;
· Creearea unui utilizator obisnuit pentru folosirea sistemului. Acesta se logheaza pe root doar atunci cand este absolut necesar;
· Folosirea ultimei versiuni a programelor;
· Pentru un sistem stabil, ar trebui sa se foloseasca versiunea stabila de kernel (cea care contine a 2-a cifra para, ex 2.2.x, 2.4.x). Este bine ca ultimul kernel sa fie stabil si sa se foloseasca toate patch-urile;
· Creearea partitilor: ar trebui sa se creeze partitii pentru fiecare din directoarele /home, /var/cache, /usr/local, /boot si /tmp;
· Se vor folosi programe de control la distanta doar in caz de necesitate.

III. Apeluri de sistem

Interfata dintre sistemul de operare si programele utilizatorilor este definita de multimea apelurilor de sistem pe care le ofera sistemul de operare. Pentru a intelege exact ceea ce fac sistemele de operare, trebuie sa examinam indeaproape aceasta interfata. Apelurile de sistem puse la dispozitie de interfata variaza de la un sistem de operare la altul.

In sectiunile urmatoare, vom examina unele dintre cele mai utilizate apeluri de sistem POSIX, sau mai exact, procedurile de biblioteca ce fac aceste apeluri de sistem. POSIX are aproximativ 100 de apeluri de sistem. Unele dintre ele, cele mai importante vor fi listate mai jos, grupa pentru claritate in 4 categorii.

Administrarea processelor

Apel	Descriere
PID=Fork()	Creare proces fiu identic cu parintele
pid=waitpid(pid, & statloc, Optiuni)	Asteptare terminare proces fiu
s=Execve(nume, argv,environp)	Imagine continut zona memorie a procesului
Exit(stare)	Terminare ececutie proces si intoarcere stare

Administrarea fisierelor

Apel	Descriere
fd=open(fisier, mod, …)	Deschidere fisier citire, scriere sau ambele
s=close(fd)	Inchidere fisier deschis
n=read(fd, zonamem, nocteti)	Citire date din fisier in zona mem. temapon
n=write(fd, zonamem, nocteti)	Scriere date din fisier in zona mem. temapon
pozitie=lseek(fd,deplasament,baza)	Mutarea cursorului asociat fisierului
s=Stat(nume, &buf)	Obtinere informatii de stare pentru fisier

Administrarea cataloagelor si a sistemelor de fisiere

Apel	Descriere
s=Mkdir(nume, Mod)	Creerea unui nou catalog
s=Rmdir(nume)	Stergerea unui catalog gol
s=Link(nume1, nume2)	Creerea unui noi intrari, nume2, ref la nume1
s=Unlink(nume)	Stergerea unei intrari de catalog
s=mount(special, Nume, flag)	Montarea unui sistem de fisiere
s=unmont(special)	Demontarea unui sistem de fisiere

Diverse

Apel	Descriere
s=Chdir(numedir)	Schimbarea catalogului de lucru curent
s=Chmod(nume, mod)	Modificarea bitilor de protectie pt. un fisier
s=kill(pid, semnal)	Trimiterea unui semnal catre un proces
Secunde=Time(&secunde)	Obtinerea timpului scurs de la 1 ian. 1970

Apelul de sistem este interfata fundamentala intre o aplicatie si kernelul linux.
Apelurile de sistem nu sunt in general apelate in mod direct , in schimb , majoritatea au functii wrapper in librarii C corespunzatoare care indeplinesc pasii necesari (precum captarea in modul de kernel) pentru a invoca apelul de sistem. Astfel, efectuarea unui apel de sistem apare ca si invocarea unei functii de librarie.
Valori returnate.
La erori , majoritatea apelurilor de sistem returneaza un numar negativ(de exemplu valoarea negata a uneia din constatntele descrise in errno) . Wrapper-ul de librarie C ascunde detaliile acestea fata de initiatorul apelului. Cand un apel de sistem returneaza un o valoare negativa ,

wrapper-ul copiaza valoarea absoluta in variabila erno , si returneaza -1 ca valoarea returnata de wrapper.
Valoarea returnata cand apelul de sistem se efectueaza cu succes depinde de apel. Multe apeluri de sistem returneaza 0 la un succes , dar unele pot returna valori nenule . Aceste detalii sunt descrise in paginile individuale de manual ale fiecarui apel.
In unele cazuri , programatorul trebuie sa defineasca un macro de trasaturi pentru a obtine declaratia apelului de sistem din fisierul de header specificat in sectiunea SYNOPSIS a paginilor de manual .
Lista completa a apelurilor de sistem se poate consulta la adresa http://www.linuxmanpages.com/man2/ Sectiunea 2: System Calls.
In continuare voi prezenta apelurile de sistem: syscall, strace si fcntl.

1. **syscall**

Pentru a invoca manual un apel de sistem , se poate folosi syscall.
Nume: syscall – indirect system call
Synopsis:
#define _GNU_SOURCE /* sau _BSD_SOURCE sau _SDIV_SOURCE */
#include <unistd.h>
#include <sys/syscall.h> /* perntru definirea SYS_xxx */
int syscall(int number, ...);

Descriere:
Syscall() lanseaza apelul de sistem al carei interfete de limbaj de asamblare are numarul specificat cu argumentele specificate. Constante simbolice pentru apelurile de sistem pot fi gasite in fisierul de header <sys/syscall.h>.
Valoarea returnata:
Valoarea returnata este definita de catre apelul de sistem invocat. In genera valoarea returnata in caz de succes este 0 , iar valoare -1 indica o eroare , si un cod de eroare este stocat in errno.

2. **strace**

Nume : strace – urmareste apelurile de sistem si semnalele
Synopsis:
strace [**-dffhiqrtttTvxx**] [**-a***column*] [**-e***expr*] ... [**-o***file*] [**-p***pid*] ... [**-s***strsize*] [**-u***username*] [**-E***var=val*] ... [**-E***var*] ... [*command* [*arg* ...]]
strace -c [**-e***expr*] ... [**-O***overhead*] [**-S***sortby*] [*command* [*arg* ...]]

Comanda strace urmareste executia altui program , listand orice apel de sistem facut de program si orice semnale pe care acesta le primeste.
Pentru a urmari apelurile de sistem si semnalele dintr-un sistem, trebuie doar invocat strace , urmat de program si argumentele din linia de comanda . Spre exemplu, pentru a urmari apelurile de sistem initiate de catre comanda hostname , se foloseste comanda :

% strace hostname

In output-ul rezultat , fiecare linie corespunde unui singur apel de sistem, pentru fiecare apel , este listat numele sau , urmat de argumentele sale (sau abrevierile acestora , daca sunt foarte lungi) si valoarea returnata.Acolo unde este posibil strace afiseasza in mod convenabil nume simbolice in locul valorilor numerice pentru argumente si valori returnate , si afiseaza campurile structurilor pasate de catre un pointer in apelul de sistem. Output-ul de la strace hostname arata in prima linie apelul de sistem execve care invoca programul hostname :

execve(“/bin/hostname”, [“hostname”], [/* 49 vars */]) = 0

Primul argument este numele programului de rulat ; al doilea este lista de argumente , ce consta in doar un element; iar al treilea este lista de variabile. Urmatoarele aproximativ 30 de linii fac parte din mecanismul de incarcare a librariei standard C dintr-un fisier de librarie comun.
Spre final sunt alte apeluri de sistem care contribuie la functionalitatea programului in cauza . Spre exemplu , apelul de sistem uname este folosit pentru a obtine hostname-ul sistemuli din kernel ,

uname({sys=”Linux”, node=”myhostname”, ...}) = 0

De observat este ca strace eticheteaza campurile sys si node , ale argumentului structurii. Aceasta structura este completata de catre apelul de sistem- Linux seteaza campul sys cu numele sistemului de operare si campul node este completat ci hostname-ul sistemului. In final , apelul de sistem write prodeuce output

write(1, “myhostname\n”, 11) = 11

Avand in vedere ca strace poate genera mult output , este adesea convenabil sa fie utilizat cu optiunea –o filename , care redirectioneaza output-ul comenzii catre un fisier.

acces : testarea permisiunilor fisierelor

Apelul de sistem acces determina daca procesul apelant are sau nu permisiuni de acces la un anumit fisier. Poate verifica orice combinatie de permisiuni de citire, scriere si executie; poate de asemenea verifica existenta unui fisier.
Apelul de sistem acces primeste doua argumente. Primul este calea catre fisierul de verificat, al doilea este un bitwise sau R_OK, W_OK, si X_OK, corespunzand permisiunilor de scriere, citire si executie. Daca fisierul exista dar procesul apelant nu are permisiune specificata , acces returneaza -1 si seteaza erno la EACCES (sau EROFS , daca a fost ceruta permisiunea de scriere pe un fisier pe un sistem de fisiere read-only)
Daca al doile argument est F_OK , acces doar verifica existenta fisierului . Daca acesta exista , valoarea returnata este 0; daca nu , valoarea returnata este -1 , iar errno este setat la ENOENT. A se observa ca errno poate fi setat la EACCES daca un director in calea de fisiere este inaccesibil.

3. fcntl :Lock-uri si alte operatiuni pe fisiere

Apelul de sistem fcntl este punct de acces pentru mai multe operatii avansate pe descriptori de fisiere. Primul argument al fcntl este un descriptor de fisier deschis , iar al doilea este o valoare deschisa care indica ce operatiune urmeaza a fi realizata. Pentru unele operatiuni , fcntl primeste un argument aditional. Cea mai importanta operatiune a fcntl este blocarea fisierelor.
Apelul de sistem fcntl permite unui program sa plaseze o blocare de citire sau de scriere asupra unui fisier , asemanator cu blocarile mutex. O blocare de citire este pusa pe descriptorul de citire al unui fisier , iar o blocare de scriere este pusa pe descriptorul de scriere al fisierului. Mai multe procese pot sa plaseze o blocare de scriere asupra aceluiasi fisier in acelasi timp, dar un singur proces poate plasa o blocare de scriere asupra unui fisier la un moment dat, si acelasi fiser nu poate sa aiba in acelasi timp si blocare de citire si blocare de scriere. Plasarea unei blocari asupra unui fiser nu impiedica alte procese sa deschida acel fisier , sa il citeasca sau sa scrie in e , decat daca primesc si ele blocari prin fcntl.
Pentru a plasa o blocare asupra unui fiser , mai intai trebuie creata si izolata o variabila struct flock. Campul l_type al structurii se seteaza la F_RDLCK pentru o blocare de citire sau F_WRLCK pentru o blocare de scriere. Apoi se apeleaza fcntl , pasandu-i un descriptor , codul operatiei F_SETLCW , si un pointer catre variabila struct flock. Daca un alt proces a plasat o blocare care impiedica blocarea noua sa fie obtinuta , fcntl se blocheaza pana cand acea blocare este eliberata.

IV. Monitorizarea apelurilor de sistem in mod fiabil si sigur.

Un mod de monitorizare a apelurilor de sistem si a semnalelor in Linux este strace, dar exista doua probleme in aceasta abordare:

1. „Agatarea” apelurilor de sistem este o metoda favorita utilizata de rootkit-uri . Este posibil ca ptrace sa fie inlocuit , sa fie furnizat output-ul unui alt proces sau alta metoda

2. Procesele nu sunt intotdeauna scrise pentru a fi usor de realizat debugging.

Exista doua metode clare anti-ptrace:

“if (ptrace(PTRACE_TRACEME, 0, 0, 0) < 0) {
    printf("being ptraced\n");
    exit(1);
}”

Aceasta metoda se bazeaza pe faptul ca un proces nu paote fi urmarit de doua ori. Daca nu se poate initializa un ptrace asupra procesului propriu , acesta este deja urmarit.

“struct timespec spec;

signal(SIGALRM, SIG_IGN);
alarm(1);
spec.tv_sec = 2;
spec.tv_nsec = 0;
if (nanosleep(&spec, NULL) < 0) {
    /* EINTR */
    printf("being ptraced\n");
    exit(1);
}”

Aceasta metoda se bazeaza pe nanosleep() - intrerupe executia firului care initiaza apelul fie pana ce timpul specificat in *req a trecut, fie pana ce un semnal declanseaza invoarea unui handlre in firul ce a initiat apelul , sau care termina procesul.
O alta metoda de monitorizare a apelurilor de sistem sau a accesarii fisierelor este folosiind susitemul audit. Trebuie ca daemonul auditd sa fie pornit, si sa se configureze apelurile care se vor a fi monitorizate cu auditctl. Fiecare operatie este logata in /var/log/audit/audit.log

IV. Kerberos

(Neagu Samuel)

Securitate

O definiție largă a noțiunii de ``securitate'' pentru calculatoare ar fi următoarea: interzicerea accesului unor persoane neautorizate la anumite date. Putem distinge două tipuri de restricții: restricții asupra citirii unor date secrete şi restricții asupra modificării de către persoane ne-autorizate. Există mecanisme speciale pentru fiecare din aceste scopuri, şi există mecanisme care pot fi adaptate amîndurora.

Sisteme sigure

Înainte de a plonja în detalii să observăm că orice schemă de securitate trebuie să înceapă cu securitatea fizică a unor dispozitive de calcul (în sensul că accesul la aparatul însuşi este îngrădit). Dacă nu poți avea încredere în nici un calculator, atunci nu poți face nici un fel de comunicație sau stocare de date sigură. Dacă cineva are control asupra sistemul de operare al unui calculator atunci el poate intercepta toate tastele apăsate şi toate caracterele scrise pe ecran.

Atunci cînd lucrați pe un calculator trebuie să aveți deplină încredere cel puțin în consola la care lucrați şi în celălalt capăt al liniei. Dacă nu puteți garanta aceste lucruri nu trebuie să vă aşteptați la nici un fel de garanții de securitate din partea sistemului. Marile bănci au calculatoarele îngropate în nişte buncăre subterane extrem de bine păzite; unul dintre motive este, desigur, acela de a nu pierde informații extrem de importante în cazul unei calamități, dar una dintre rațiunile primordiale este garantarea securității fizice a sistemului.

Oricare ar fi scopul pentru care vrem să protejăm date, politica pe care o aplicăm în acest sens va fi o funcție de entitatea care vrea să acceseze datele. De pildă anumite documente militare vor fi secrete pentru gradele mici, dar vor putea fi accesate de un general.

Pentru a putea face astfel de distincții este deci necesar să avem la dispoziție un mecanism de autentificare.

Autentificare

Autentificarea (authentication) este procesul prin care se verifică identitatea unei instanțe (de exemplu un utilizator) care a produs nişte date. Cele mai sigure metode de autentificare sunt cele biometrice, care măsoară caracteristici fizice unice ale unui individ (cum ar fi amprentele digitale sau forma irisului). Acestea sunt foarte greu de păcălit. Din (ne)fericire sunt încă foarte costisitoare şi puțin răspîndite.

Metoda prin care în mod uzual calculatoarele identifică identitatea unei entități cu care comunică este verificarea că acea entitate ştie o informație care foarte probabil nu mai este cunoscută de nimeni altcineva. În sistemele tipice Unix acel secret este parola pe care utilizatorul trebuie să o tasteze înainte de a începe lucrul.

Să mai observăm că pentru a putea vorbi despre autentificare trebuie să avem o noțiune de identitate a utilizatorilor. Calculatorul trebuie să poată distinge cumva între diferiții indivizi care îl folosesc. Trebuie să existe un spațiu de nume pentru utilizatori; autentificarea atunci va pune în corespondență o entitate activă cu un astfel de nume. În Unix numele utilizatorilor autorizați sunt trecute într-o mică bază de date într-un fişier numit /etc/passwd (sistemele mai moderne au scheme mai complicate, dar înrudite); pentru un sistem Unix a autentifica un utilizator constă în a asigna unul dintre aceste nume cunoscute unui set de procese executate de acel utilizator. Dacă un individ nu are un nume în acel fişier atunci el nu există pentru calculator.

Cu alte cuvinte, pentru a putea vorbi despre autentificare trebuie ca părțile implicate în comunicare să aibă un spațiu de nume comun pentru utilizatori.

Criptografie

Dacă întreg sistemul cu care lucrăm este sigur fizic atunci nu avem mare nevoie de autentificare; noțiunea de securitate fizică implică faptul că persoane neautorizate nu pot accesa sistemul. De îndată însă ce datele trebuie să traverseze porțiuni nesigure trebuie să luăm măsuri suplimentare de precauție. Transformarea datelor în scopul de a împiedica accesul persoanelor neautorizate se numeşte criptare. Criptarea transformă un text inteligibil (plaintext) într-un cifru (ciphertext). Procesul de codificare se numeşte cifrare sau criptare (encryption) iar procesul invers se numeşte descifrare sau decriptare (decryption).

Funcții ``ne-inversabile''

Putem vedea criptarea unui mesaj ca o transformare a mesajului. În mod normal această transformare este una funcțională, în sensul că unui mesaj îi corespunde un singur cod. Pentru a putea folosi la ceva mesajele codificate, trebuie să existe şi o transformare inversă, prin care dintr-un cifru obținem textul inițial.

Dacă notăm funcția de criptare cu E (de la ``encryption'') şi cea de decriptare cu D, pentru un mesaj x trebuie să avem relația D(E(x)) = x. De aici rezultă în primul rînd că funcția E trebuie să fie injectivă (altfel inversa nu este bine definită), cu alte cuvinte oricare două codificări ale unor cuvinte diferite trebuie să fie diferite.

În realitate funcțiile de criptare şi decriptare mai au un parametru relativ mic (comparat cu lungimea mesajului x, numit cheia de criptare. Cheia este un obiect relativ uşor de descris şi manipulat, spre deosebire de funcțiile D şi E. E va fi atunci de forma E(x, k), unde k este cheia. $D(x, k_1)$ este funcția de decriptare, k_1 fiind cheia pentru decriptare.

Se deosebesc două feluri de sisteme de criptare: simetrice, în care $k = k_1$ şi asimetrice în care cheile sunt diferite.

Ceea ce este important este că funcțiile D şi E sunt cunoscute de toată lumea; atunci cînd vreau să stabilesc un canal sigur cu un alt individ noi trebuie doar să cădem de acord asupra cheilor pe care le folosim (k şi k_1).

Nu oricare două funcții care satisfac cerințele anterioare sunt bune pentru criptare. Dacă fixăm cheia k şi notăm funcția E(k,.) cu E_k, trebuie ca din cunoaşterea valorii lui $y = E_k(x)$ (cifrul) să fie practic imposibil de calculat x. Din cauza asta funcțiile de criptare se numesc (nu foarte precis) ``neinversabile'' sau ``greu inversabile'' (one-way functions).

Ce înseamnă de fapt ``imposibil de calculat''? Pentru a răspunde la această întrebare ne trebuie ceva cunoştințe de teoria complexității, pe care o să le omitem din prezentarea de față. În principiu asta înseamnă că orice algoritm care ar calcula cheia k ştiind y, D şi E trebuie să aibă o complexitate mai mare decît polinomială (de exemplu exponențială).

Să observăm că există întotdeauna algoritmi care pot calcula inversa, pur şi simplu iterînd prin toate cheile posibile k şi văzînd dacă D(y,k) este un cuvînt din limbajul de intrare. Dar dacă cheia are n biți, trebuie încercate 2^n combinații diferite. Pentru un n suficient de mare sunt mult prea multe încercări, chiar şi pentru cel mai perfecționat supercalculator.

De altfel unul dintre cele mai populare programe pentru ``spart'' parole pentru Unix, Crack, se bazează pe faptul că cheile de fapt nu sunt egal probabile; utilizatorii vor alege cu mare probabilitate anumite cuvinte (pe care le vor mutila în mici feluri), pentru că sunt mai uşor de memorat. ``Crack'' are la dispoziție un dicționar enorm şi o suită de reguli pentru a modifica cuvintele, şi pur şi simplu încearcă toate variantele; deşi asta poate părea mult, o limbă umană are în jur de 200 000 de cuvinte, o bagatelă pentru un calculator (comparați de pildă cu 128^8=72 057 594 037 927 936, cît ar fi toate combinațiile de 8 caractere ASCII).

P = NP?

Într-un articol mai vechi din PC Report despre algoritmi am prezentat felurite clase de complexitate, printre care şi clasele P şi NP, ale problemelor a căror soluție se poate calcula în timp polinomial, cu un calculator determinist, respectiv nedeterminist.

Problema P=NP este probabil cea mai importantă problemă deschisă din informatica; de peste 25 de ani s-au depus eforturi susținute pentru a vedea dacă există probleme care se pot rezolva cu maşini nedeterministe (care ``ghicesc'' o parte din soluție) repede dar nu şi cu maşini deterministe (cum sunt toate calculatoarele reale).

Ei bine, pentru a fi rezonabilă, problema aflării cheii cu care este criptat un mesaj nu poate fi mai complicată decît o problemă din clasa NP. Lucrurile se pot justifica perfect riguros; ideea de bază este că cel care cunoaşte cheia trebuie să decodifice relativ rapid mesajul primit (dacă decodificarea însăşi ar dura foarte mult metoda n-ar mai fi practică). Un străin care posedă însă o maşină (deocamdată fictivă) nedeterministă ar putea folosi maşina pentru a ``ghici'' cheia, după care ar decodifica cu această cheie şi ar verifica corectitudinea ghicirii.

De aici rezultă o consecință foarte interesantă: dacă se va demonstra că P=NP nu există criptografie! Atunci orice cifru cunoscut ar putea fi spart în timp polinomial! Dacă nu ați

crezut pînă acum că problema P=NP merită atenție, poate acum v-ați schimbat opinia: rezolvarea ei (în sensul confirmării egalității) ar supăra o grămadă de agenții de spionaj şi militare.

Desigur, cele spuse anterior consideră că orice problemă care are o complexitate polinomială este uşoară (chiar dacă polinomul este ceva de genul x^{10000}) şi că toate problemele exponențiale sunt grele. Este adevărat că 1.0001^x depăşeşte cu mult pe x^{10000}, dar asta numai de la o valoare foarte mare a lui x încolo.

Dar să lăsăm pentru moment teoria complexității. Să presupunem că $P \neq NP$, căci altfel nu mai avem ce studia, şi să aruncăm o privire la modul în care poate fi folosită criptografia pentru a transmite secrete. Treaba asta este cu mult mai grea decît pare.

Pericole

Cînd se proiectează un protocol sigur (de pildă de autentificare) trebuie să ştim exact de ce anume vrem să ne păzim. Un model răspîndit, pe care îl vom folosi şi în textul de față face următoarele presupuneri:

- Agenții finali sunt siguri: terminalul pe care lucrez eu şi serverele care conțin informațiile sunt la adăpost în mod fizic şi nu pot fi controlate de ``duşmani'';
- Canalul de comunicații între agenții finali este complet nesigur;
- Un agent străin poate intercepta absolut toate mesajele din rețea şi le poate studia: ascultare pasivă; (passive wiretapping sau eavesdropping);
- Un agent străin poate injecta mesaje noi în rețea între mesajele existente (active wiretapping, tampering). Anume inamicul poate modifica mesajele existente, poate şterge unele dintre ele, poate modifica mesajele de pe rețea sau poate stoca mesaje transmise şi le poate retransmite mai tîrziu.

Este de asemenea important de ştiut că uşurința spargerii unui protocol creşte cu cantitatea de mesaje avute la dispoziție; cu cît mai multe cifruri sunt observate, cu atît mai mult succes pot fi făcute anumite atacuri statistice.

Protocolul Kerberos

Protocolul Kerberos a fost proiectat la Universitatea MIT (Massachusetts Institute of Techonology) în cadrul proiectului Athena, în jurul anului 1984. Scopul protocolului Kerberos este de a permite unui client să-şi demonstreze identitatea unui server aflat la distanță, undeva dincolo de o rețea complet nesigură. Protocolul garantează de asemenea că clientul nu poate conversa cu un calculator care se dă drept server; autentificarea se face în ambele direcții.

Protocolul în sine constă dintr-un schimb de mesaje între un client şi o serie de servere, fiecare cu o altă misiune. Idea de bază aparține lui Needham şi Schroeder care au publicat ideea inițială în 1978 în Communications of the ACM. Descrierea detaliată a protocolului se

găseşte în documentul numit RFC 1510, care este disponibil de pe Internet[1]. Cei care administrează Kerberos pot pune întrebări pe grupul de News comp.protocols.kerberos.

Protocolul Kerberos indică de fapt o serie de mesaje care trebuie schimbate între părțile care doresc să comunice; unele din mesaje sunt criptate. Ce funcție de criptare/decriptare se foloseşte teoretic nu contează prea tare, atîta vreme cît funcția este greu inversabilă. Implementările curente folosesc un algoritm standard de criptare numit DES (Data Encryption Standard).

Kerberos este un protocol; pentru a fi util anumite aplicații (cele care au nevoie de comunicație client-server) trebuie modificate pentru a folosi autentificarea oferită de Kerberos. (Aplicațiile modificate sunt atunci numite ``kerberized''). În mod normal într-un domeniu administrativ în care se foloseşte Kerberos programe ca telnet, rlogin, POP (post-office protocol), fişiere la distanță (AFS), etc. trebuie rescrise în aşa fel încît clientul să se autentifice serverului folosind noua metodă (toate aceste aplicații sunt de tip client-server).

Principii

Protocolul pare destul de complicat, dar dacă aveți răbdare o să înțelegeți tot ce se întîmplă; nu e mare ştiință la mijloc.

Premiza inițială este că există un server central, numit serverul de autentificare (AS), care cunoaşte identitățile tuturor clienților posibili. De asemenea, fiecare client a stabilit o parolă secretă pe care şi acest server o ştie. Parola ajunge la server printr-o cale sigură, de exemplu prin poştă sau printr-un mesager uman (mă rog, sigură cel puțin din punct de vedere software). Parola asta nu este cunoscută de nimeni altcineva, nici măcar de celelalte servere cu care clientul va comunica (de la care va obține serviciile care-l interesează de fapt). Toate celelalte servere din domeniul administrativ sunt şi ele clienți ai AS: au o parolă unică ştiută numai de AS şi de acel server.

Niciodată parola nu va circula în clar pe rețea; atunci oricine ar putea să o citească şi să o folosească în locul clientului de drept.

Pentru a se proteja împotriva duşmanilor care vor înregistra sau şterge mesaje, mesajele vor avea informații ca ora la care au fost trimise şi un număr de ordine (pentru a detecta omisiunile).

O altă regulă interesantă este că cheile folosite în comunicarea dintre client şi servere se schimbă frecvent. Implementarea standard expiră o cheie după 25 de ore. În felul acesta un atac criptanalitic nu va avea prea mult succes: dacă durează mai mult de 25 de ore atunci cheia descoperită este deja inutilă (desigur, eventualele mesaje deja interceptate şi stocate vor putea fi citite, dar stricăciunile sunt limitate). Cheia pe care un client şi un server o folosesc în comun se numeşte cheie de sesiune şi este generată aleator.

Mesajele (protocolul Needham-Schroeder)

Să presupunem că clientul nostru vrea să vorbească cu un server de disc. Protocolul esențial este compus din 4 mesaje (protocolul real este o simplă extensie a ideii pe care o prezint aici):

1. Un mesaj de la client spre AS, prin care se indică intenția de a comunica cu serverul de disc;
2. Un mesaj de răspuns de la AS pentru client, prin care AS îi trimite clientului noua cheie de sesiune şi un pachețel pentru serverul de disc;
3. Un mesaj de la client spre serverul de disc, în care este inclus pachețelul de mai sus, pentru a garanta faptul că clientul a discutat cu AS (pachețelul poate veni numai de la AS);
4. Un mesaj de răspuns de la serverul de disc prin care clientul este convins că serverul a putut deschide pachețelul, deci că discută într-adevăr cu serverul de disc.

După acest schimb inițial de mesaje clientul şi serverul de disc vor folosi cheia de sesiune generată de AS pentru a cripta cu DES toată comunicația dintre ei.

Vom vedea că mesajele sunt relativ încîlcite pentru a preveni toate atacurile de mai sus. Dacă veți încerca să simplificați protocolul aproape sigur îl veți face vulnerabil la anumite tipuri de atacuri.

Să notăm cele 3 entități care colaborează astfel: C clientul, AS serverul de autentificare şi S serverul de disc.

O tehnică importantă a protocolului, care este folosită pentru a contraataca folosirea unor mesaje vechi înregistrate este de a eticheta mesajele cu ora emiterii (într-un mod care nu poate fi contrafăcut) şi de a verifica la recepție dacă ora este rezonabilă. Sincronizarea ceasurilor este o problema extrem de grea în sistemele distribuite, aşa că pentru Kerberos ``rezonabil'' înseamnă că ceasul local la recepție arată plus/minus 5 minute de ora din mesaj (ora de transmitere).

Un alt truc interesant este folosirea a ceea ce se numeşte **nonce**: un obiect care este folosit o singură dată. Acesta este practic un număr aleator. Vom vedea mai jos cum este acesta folosit.

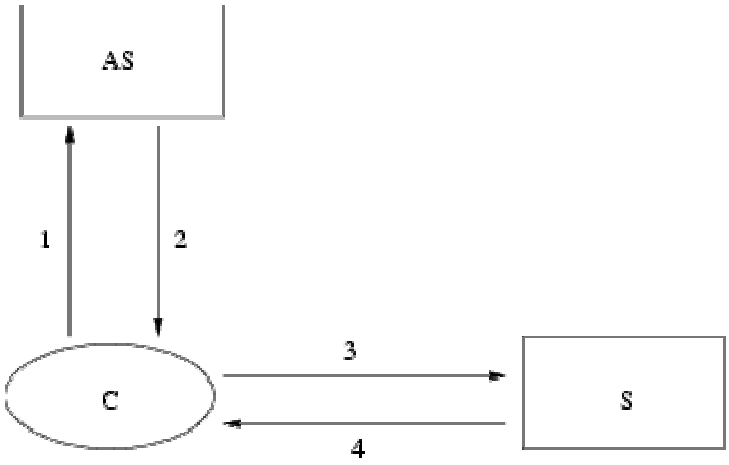


Figure 1: Protocolul de bază (simplificat).

Vom mai introduce următoarele notații:

- $K_{a,b}$ este cheia de sesiune pe care o folosesc pentru discuție a şi b; de exemplu $K_{C,S}$ va fi cheia folosită pentru criptare/decriptare între client şi serverul de disc;
- Cum am spus mai sus, AS este serverul de autentificare, care ştie parola fiecărei alte entități din sistem;
- K_z este cheia pe care clientul z şi AS o folosesc în comun (parola clientului z); de pildă K_s este parola serverului de disc (cunoscută numai de el şi de AS);
- Voi scrie $\{mesaj_1, mesaj_2\}_K$ pentru a indica faptul că mesajele 1 şi 2 sunt puse laolată într-un pachet care este apoi criptat cu cheia K.
- O abreviere utilă este cea de tichet (ticket): $T_{a,b} = \{ K_{a,b}, a, ora\ curenta \}$: un mesaj în care sunt împachetate 3 informații: o cheie de sesiune între a şi b, numele lui a şi ora curentă.

  Cu aceste notații putem scrie complet ne-ambiguu toate mesajele schimbate între C, S şi AS. Săgeata indică traseul fiecărui mesaj:

  C ---> AS:

  C, S, ora expirare, N (nonce, aleator).

  Clientul afirmă propria identitate, identitatea serverului cu care vrea să discute, trimite un număr aleator şi cît timp ar dori să converseze cu S. Pînă aici totul e simplu. AS va răspunde astfel:

  AS ---> C$:

  $\{K_{C,S}, S, ora\ expirare, N\}_{KC}, \{T_{C,S}\}_{KS}$.

  Acest complicat mesaj are mai două părți mari: prima este destinată clientului, şi este criptată cu cheia clientului KC, iar a doua este un tichet pentru S, criptat cu cheia lui S. Să vedem la ce folosesc feluritele părți din mesaj:
- Prima parte a mesajului este criptată cu cheia KC a lui C. Pentru că această cheie este secretă, nimeni altcineva nu poate decripta acest mesaj decît C; pentru restul lumii mesajul este gunoi. La fel stau lucrurile şi pentru partea a doua a mesajului, care fiind criptată cu KS este inteligibilă numai pentru S.
- KC,S este un număr aleator generat de AS, care va fi folosit ca cheie de sesiune între C şi S. Observați că cheia de sesiune apare în ambele părți ale mesajului, în aşa fel încît va fi cunoscută atît de către C cît şi de S.
- AS îi confirmă lui C identitatea serverului de disc S şi indică durata de validitate a lui KC,S (care poate fi diferită decît a dorit C în primul mesaj).
- Faptul că N apare în mesajul către C garantează că acest mesaj a fost criptat de AS: nimeni altcineva nu putea să-l bage pe N înauntru. De asemenea, acest mesaj nu putea fi un mesaj mai vechi dintr-o comunicație anterioară, pentru că N diferă.
- Partea a doua a mesajului este opacă pentru C; tot ce poate face C cu ea este să o înainteze lui S. C (sau oricine altcineva) nu o poate citi. Dacă cineva ar modifica partea a doua, S nu ar mai putea-o decodifica şi obține un mesaj corect, deci tichetul nu poate fi contrafăcut sau modificat.

  C ---> S:

  $\{C, ora\ curenta, suma\ de\ control\}_{KC,S}, \{T_{C,S}\}_{KS}$.

Mesajul are din nou două părți.

- Partea a doua este exact partea a doua a mesajului 2, aşa cum a fost primită de la AS.
- Prima parte a mesajului are scopul de a demonstra că acest mesaj este ``proaspăt'': S va extrage $K_{C,S}$ din partea a doua a mesajului şi o va folosi pentru a citi prima parte a mesajului. De acolo află ora la care a fost trimis mesajul, şi îl rejectează dacă ora diferă prea tare de ora locală. (Asta ar putea să însemne că mesajul a fost capturat de un inamic şi re-lansat). Suma de control ne asigură că mesajul nu a fost modificat de nimeni; este practic imposibil să modifici un mesaj criptat şi să repari şi suma de control dacă nu ştii cheia.
  S ---> C:
  $\{\text{ora curenta+1}\}_{KC,S}$.

Prin acest mesaj S în convinge pe C că a ajuns la destinația dorită: mesajul are ora curentă trimisă anterior de C plus 1. Dar ora putea fi extrasă numai de cel care avea $K_{C,S}$, care fiind o cheie de sesiune era ştiută numai de S. Nu era suficient să returneze aceeaşi valoare, pentru că atunci acest mesaj putea fi o copie a (unei părți a) mesajului anterior.

La sfîrşitul acestei comunicații atît C cît şi S sunt siguri de identitatea celuilalt şi în plus au la dispoziție o cheie de sesiune $K_{C,S}$ cu care pot cripta toate mesajele pe care le schimbă. Autentificarea a fost făcută.

Implementarea Kerberos

Între un protocol de autentificare pe hîrtie şi o implementare reală pe un calculator e o distanță considerabilă. Secțiunea următoare, consacrată slăbiciunilor lui Kerberos va ilustra şi mai pregnant acest lucru.
Implementarea lui Kerberos încearcă să mai țină cont de anumite particularități ale lumii reale care fac realizarea protocolului mai dificilă.

Prima întrebare spinoasă care se iveşte este: unde este stocată parola fiecărui client (K_C, K_S, etc.). AS trebuie să fie o maşină foarte sigură, undeva într-un subsol ferit, dar maşinile client vor fi probabil peste tot la-ndemînă. Dacă parola unui client este stocată pe disc atunci este mai la-ndemîna atacurilor asupra clientului.

Ca atare Kerberos nu memorează parola nicăieri! Utilizatorul este obligat să o tasteze de fiecare dată cînd vrea să fie autentificat. Observați că K_C este folosită în mesajele de mai sus numai pentru a decripta mesajul 2; în rest este inutilă. Deci procedura este următoarea: programul kerberizat al clientului va lua legătura cu AS, iar cînd răspunsul soseşte utilizatorul trebuie să tasteze parola K_C. Parola este imediat folosită pentru a decripta mesajul de la AS după care este complet ştearsă din memorie. În acest fel fereastra de vulnerabilitate este redusă la maximum.

Pe de altă parte asta poate fi foarte neplăcut, pentru că atunci utilizatorul va trebui să tasteze parola pentru fiecare nou server S pe care vrea să-l folosească (adică pentru fiecare nouă sesiune). Şi cum ştim că utilizatorii sunt leneşi, aşa ceva este inadmisibil. Kerberos a introdus atunci un al doilea server central numit ``Serverul care dă tichete'': Ticket Granting

Server, TGS. Ideea este TGS are de fapt toate parolele K_S. Clientul C se autentifică la TGS exact în acelaşi fel ca la oricare server S: obținînd un tichet de la AS. Odată autentificat la TGS şi folosind cheia de sesiune $K_{C,\ TGS}$ clientul poate solicita oricîte chei pentru alte servere S, S_1, etc.

Figura 2 şi tabela 1 arată situația reală: mesajele 1,2 sunt schimbate numai cînd utilizatorul face login. Mesajele 3,4 sunt schimbate de fiecare dată cînd utilizatorul vrea să contacteze un nou server (adică să deschidă o nouă sesiune). Mesajul 5 este folosit de client pentru a se autentifica fiecărui nou server, iar mesajul 6 este opțional, autentificînd serverul pentru client. În mesajul 5 apare un element nou, numit $K_{subsesiune}$: clientul poate alege aici o nouă cheie de sesiune care să fie folosită în locul cheii oferite de TGS, $K_{C,S}$. Asta nu schimbă prea tare natura protocolului.

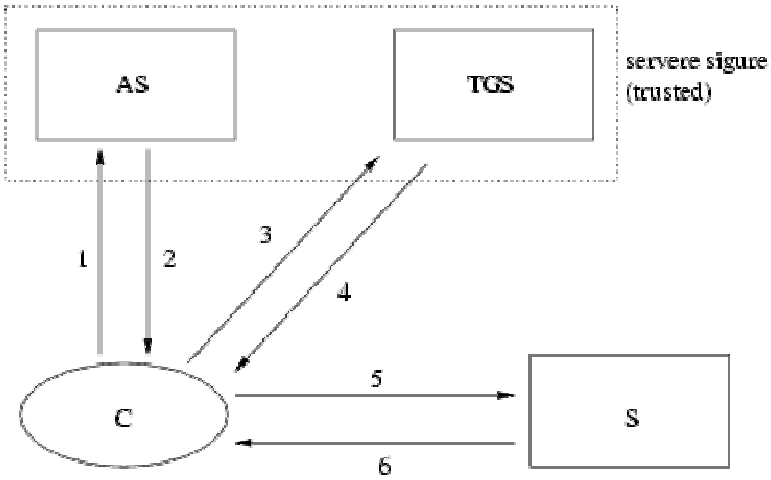


Figure 2: Protocolul Kerberos complet.

Table 1: Schimbul complet de mesaje în protocolul Kerberos.

Nr.	Între	Conținutul mesajului
1	C ---> AS	C, TGS, ora de expirare, N
2	AS ---> C	$K_{C,\ TGS}$, ora de expirare, N$\}_{K_C}$, $\{T_{C,TGS}\}_{KTGS}$
3	C ---> TGS	{ora locala$\}_{KC,TGS}$, $\{T_{C,TGS}\}_{KTGS}$, S, ora de expirare, N_1
4	TGS ---> C	$\{K_{C,S}$, S, ora de expirare, $N_1\}_{KC,\ TGS}$, $\{T_{C,S}\}_{KS}$
5	C ---> S	{ora locala, suma de control,$K_{subsesiune}\}_{KC,S}$, $\{T_{C,S}\}_{KS}$
6	S ---> C	{ora locala$\}_{KC,S}$

Kerberos este implementat sub forma unor procese server (AS, TGS) şi a unor biblioteci care se pot lega în programele clienților şi serverelor. Funcționează sub o mare

varietate de sisteme de operare: Unix şi Windows NT fiind cele mai notabile. Mai are tot felul de zorzoane, legate de pildă de autentificarea între domenii administrative diferite, transmiterea tichetelor, etc.

Slăbiciunile lui Kerberos

Chiar dacă în teorie Kerberos este minunat, implementarea lui în practică este cel puțin dificilă. Condițiile ideale existente pe hîrtie sunt greu de obținut într-o rețea de calculatoare reale.

La ora actuală nu există nici un fel de metodă complet riguroasă pentru a arăta ca un protocol criptografic nu scapă informații; există metode pentru a testa dacă un protocol rezistă la atacurile cunoscute, dar foarte adesea se publică algoritmi care mai tîrziu se dovedesc greşiți. În general, raționamentele cu astfel de protocoale sunt foarte complicate. Cercetarea în domeniu este în plină desfăşurare şi foloseşte tehnici foarte exotice, ca teoria informației, teoria complexității, logici speciale (ex. knowledge theory), etc.

Voi ilustra aici numai unele dintre deficiențe, pentru a da o idee despre natura lor.
Să observăm că clientul trebuie să păstreze undeva cheile de sesiune pentru a putea conversa cu serverele: fiecare mesaj după cele de autentificare va fi criptat cu aceste chei. Clientul trebuie să posede deci practic permanent $K_{C,\ TGS}$ şi $K_{C,S}$. E adevărat că aceste chei expiră în 25 de ore, deci sunt mai puțin importante decît o parolă care teoretic este folosită luni întregi. Întrebarea este însă: unde sunt ținute pe calculatorul clientului aceste chei?

Pe o stație obişnuită Unix lucrează în mod normal mai mulți utilizatori. Tichetele unuia ar trebui să fie ferite de ceilalți. Dar pe un sistem Unix practic nimic nu poate fi adăpostit împotriva administratorului (root). Administratorul unui sistem poate citi orice fişier, şi poate inspecta memoria fizică a oricărui proces. Acesta este un călcîi al lui Ahile al lui Kerberos; toate metodele cunoscute pentru a penetra un sistem Unix amenință siguranța întregului protocol. Ori securitatea unui sistem Unix, care este foarte complicat, este extrem de greu de controlat; există o sumedenie de breşe de care un atacator ar putea profita.

O altă mare problemă este cu stațiile de lucru fără disc (diskless); aceste stații de obicei importă discuri prin rețea. Deci de îndată ce o astfel de stație stochează un tichet pe disc, tichetul va călători prin rețea, care am stabilit că este expusă la tot felul de atacuri!

Nici păstrarea tichetului în memorie nu este neapărat mai sigură: algoritmii de paginare stochează paginile pe disc (în partiția de swap) atunci cînd calculatorul nu are destulă memorie, deci am revenit la aceeaşi problemă.

Iată încă un exemplu: am văzut că prospețimea unui tichet este verificată comparînd ora locală a serverului cu ora din tichet. Pentru un interval de 5 minute serverul memorează toate tichetele primite, pentru a depista duplicate, eventual rezultate dintr-un atac care re-transmite pachete vechi capturate. Un tichet mai vechi de 5 minute este considerat expirat şi ignorat. În felul acesta un server nu va primi niciodată acelaşi pachet de două ori. Asta presupune că serverul şi clientul au ceasuri relativ sincronizate.

Într-o rețea mare de calculatoare sincronizarea ceasurilor se face automat, folosind un protocol numit NTP: Network Time Protocol. Un atac foarte spectaculos este următorul: un atacator înregistrează o serie de mesaje de la un client care ştie că reprezintă o tranzacție importantă. Peste o săptămînă atacatorul infiltrează în rețea mesaje false NTP prin care setează ceasul unui server cu o săptămînă în urmă. După asta atacatorul retransmite mesajele capturate, care vor fi re-executate, pentru că serverului îi par proaspete.

Asta face securitatea în calculatoare o problemă foarte spinoasă: adesea protocoalele propuse sunt eronate, dar nu există nici o metodă formală pentru a depista şi verifica asta. Chiar dacă un protocol este corect formal, se poate baza pe asumpții nerezonabile asupra mediului în care operează, cum ar fi ceasurile sincronizate. Şi chiar dacă se bazează pe asumpții rezonabile, implementarea scrisă de un programator uman poate să aibă bug-uri care o fac vulnerabilă.

Kerberos pentru un utilizator

Voi încheia acest articol ilustrînd cum se manifestă Kerberos pentru un utilizator obişnuit. Domeniul administrativ în care lucrez eu de obicei este complet kerberizat. Programul meu login a fost modificat ca imediat ce tastez parola să discute cu AS şi să obțină un tichet pentru serverul care dă tichete, TGS. Directorul meu casă este de pe un disc din rețea, aflat undeva departe, pe un server de disc (protocolul folosit este AFS: Andrew File System). Atunci cînd comunic prima oară cu serverul de AFS demonul local de pe maşina mea foloseşte tichetul obținut de la AS pentru a obține de la TGS un tichet pentru serverul de disc. După aceea se autentifică serverului de disc, exact ca în schema de mai sus.

După autentificarea cu serverul de disc toată comunicația între maşina mea şi serverul de disc se va face folosind un alt protocol, numit Secure RPC (Remote Procedure Call): apel sigur de procedură la distanță, care foloseşte inițial cheia de sesiune oferită de TGS.

Pentru mine ca utilizator final totul este aproape complet transparent. Singura neplăcere este că la fiecare 25 de ore tichetele pentru TGS expiră, şi atunci trebuie să-mi tastez din nou parola pentru a obține tichete proaspete. Asta poate fi neplăcut dacă vrei să rulezi o simulare mai îndelungată.

Mai am la dispoziție următoarele comenzi:

kinit

prin care pot să-mi schimb identitatea Kerberos (de pildă dacă un coleg vrea să lucreze pe calculatorul meu o să tasteze kinit numele-lui şi apoi parola lui personală;

kdestroy

prin care toate tichetele de pe maşina locală sunt distruse; foarte util dacă plec şi nu vreau ca cineva să poată lucra în numele meu;

kpasswd

prin care îmi pot schimba cheia (parola) stocată pe AS. Schimbarea parolei este sigură, pentru că parola va fi trimisă criptat la un server special care face managementul parolelor şi modifică baza de date din care citeşte AS. Autentificarea la managerul de parole se face tot folosind Kerberos. Asta înseamnă că odată ce am o parolă Kerberos (care trebuie introdusă manual în baza de date a lui AS) schimbarea o pot face fără să mai bat vreun administrator la cap;

kdb_init

este o comandă folosită de administrator pentru a crea o nouă bază de date Kerberos atunci cînd porneşte serviciul;

kdb_admin

este comanda prin care administratorul adaugă un nou utilizator în baza de date;

kdb_edit

este comanda prin care se pot adăuga noi administratori în baza de date AS: persoane care au dreptul să modifice baza de date.

În plus, în domeniul în care lucrez eu, majoritatea comenzilor care operează în rețea au fost kerberizate. De pildă demonul şi clientul de telnet (terminal virtual): cînd eu fac telnet pe o maşină la distanță clientul îmi cere parola după care obține un tichet de la TGS pentru demonul telnetd de pe maşina de la distanță; în acest fel parola mea nu circulă niciodată prin rețea (cum ar fi fost cazul dacă telnet nu era kerberizat).

Concluzie

Securitatea în calculatoare este o problemă de mare importanță economică, mai ales acum cînd tot mai multe tranzacții se fac prin Internet. Proiectarea unui protocol de securitate este o treabă foarte complicată, iar implementarea nu este deloc simplă. Domeniul este în plină cercetare în continuare. Dificultățile sunt amplificate de faptul că lanțul este tot atît de slab cît cea mai slabă verigă, iar verigi sunt destul de multe. Fiți deci cu ochii-n patru.

V. Cele mai intalnite probleme in securitatea sistemelor Linux (probleme si solutii) – partea I

(Langa Mihai)
(Clabescu Catalin)

PORTURI DE RETEA DESCHISE

(Langa Mihai)

Asa cum fiecare cont de utilizator din sistemul tau este o potentiala cale pentru un haker the parole sa-ti compromita securitatea, asa si fiecare serviciu de retea a sistemului poate reprezenta o vulnerabilitate de aceea e bine ca serviciile care nu sunt folosite sa fie dezactivate sau dezinstalate. Majoritatea variantelor de Linux tind sa instaleze odata cu instalarea sistemului de operare o gramada de programe si servicii inutile pentru o buna parte din utilizatorii obisnuiti pentru ca distribuitorul prefera o metoda usoarea de distribuite in detrimentul uneia care sa asigure cat de cat securitate sistemului. E bine ca atunci cand sistemul de operare este instalat, utilizatorul sa fie atent la aceste pachete de software si sa le deselecteze instalarea.

Pentru a afla la orice moment de timp care sunt serviciile ce ruleaza activ in sistemul de operare se poate folosi comanda `netstat –atuv`**.** Care si un sistem “de casa” poate avea o gramada de port-uri deschise, iar serverele Web mari pot avea mult mai multe.

Multi distribuitori precum Red Hat sau Mandriva ofera utilizatorului acces la un panou de control prin care acesta poate dezactiva cu usurinta serviciile pe care nu doreste sa le foloseasca.

NFS, finger, the shell, exec, servicii login r* (rsh, rexec si rlogin), FTP, telnet, sendmail, DNS si linuxconf sunt doare cateva dintre cele mai populare servicii instalare automat de catre distribuitorii Linux. E bine ca cel putin o parte din acestea sa nu fie activate pentru majoritatea sistemelor. Majoritatea sunt controlate de deamon-ul xinetd. Acestea pot fi dezactivate prin editarea scripturilor `/etc/xinetd.d/*`.

De asemnea, e bine ca dupa editarea unui script de start-up sau a altor fisiere de configurare precum si a patch-urilor de securitate, utilizatorul sa reboot-eze sistemul de cateva ori pentru a se asigura ca modificarile au fost salvate corespunzator si sistemul este stabil la pornire si utilizare.

Este important de retinut ca nu este nevoie de deamon-uri FTP sau telnet pentru ca un client sa se poata conecta la alte sisteme. Acelasi lucru este valabil si in cazul daemon-ului sendmail care nu este necesar pentru a trimite un e-mail prin port-ul 25, utilizatorilor locali sau pentru a downloada mail-ul prin POP sau IMAP. DNS-ul este folosit doar atunci cand alt sistem

interogeaza sistemul utilizatorului in legatura cu acest tip de date. Majoritatea programelor care ruleaza activ pe sisteme vor citii fisierul `/etc/resolv.conf` si vor interoga principalul server DNS al ISP-ului sau organizatiei in loc sa contacteze procesul cu acelasi nume de pe sistemul clientului.

Toate aceste servicii cu exceptia instalarii normale a NFS, DNS si sendmail-ului sunt pornite la cerere de catre xinetd. Ele pot fi dezactivate prin commentarea randului pe care se afla in `/etc/xinetd.d`. Multi distribuitori offera aceasta optiune prin intermediul panoului de control sau a Linuxconf-ului.

Serviciile independente de sistemul de operare se pot dezactiva prin modificarea lor din `/etc/rc.d` sau a altor fisiere de configurare.

Intr-un sistem bazat pe Red Had se pot folosii urmatoarele comenzi pentru inchiderea portmap-ului si prevenirea reinitializarii acestuia:

```
/etc/rc.d/init.d/portmap stop
chkconfig –del portmap
```

O solutie alternativa il reprezinta programul ntsysv cu meniu bazat pe ASCII. Ca si chkconfig, ntsysv manipuleaza legaturile simbolice din **/etc/rc.d/rc[0-6].d** de aceea va trebui explicit sa fie si acest serviciu inchis. Pentru a face aceste lucruri se scriu comenzile:

```
/etc/rc.d/init.d/portmap stop
ntsysv
```

In cadrul altor versiuni care folosec script-uri de start-up in stil System V (direcotare `/etc/rc.r/rc[0-6].d` pentru versiunile Red Had si **/etc/rc.[0-b].d** pentru cele Debian) trebuiesc redenumite anumite script-uri sub rcX.d care incep cu majuscula S si care contin in numele lor denumirea serviciului. Spre exemplu:

```
cd /etc/rc.d/rc4.d mv S11portmap K11portmap
```

Asa cum doar script-urile ale caror nume incep cu S sunt invocate atunci cand se intra in nivelul respectiv de functionare asa si cele care incep cu K sunt invocate cand se iese din nivel. Acest lucru se intampla pentru a inchide deamon-urile care trebuiesc rulate doar pe acel nivel. Spre exemplu, acest mecanism va inchide sshd, deamon-ul ssh atunci cand are loc comutarea de la nivelul de rulare 3(multiuser cu networking) la nivelul s (mod cu utilizator unic).

Kernel-ul Linux 2.6 precedent versiunii 2.6.17.4 are o vulnerabilitate mare a root-ului local ce permite oricarui utilizator cu cont shell sa se transforme in root prin folosirea ssh-ului sau abuzand de un program CGI de server Web.(A se vedea CVE-2006-2451). O posibila solutie la aceasta problema o reprezinta comanda:

```
chmod 700 /etc/cron*/
```

O solutie mai buna este scrierea unui modul ce poate fi incarcat in kernel ce previne folosirea apelului de sistem prctl() in afara root-ului. Desigur, cea mai totala solutie o reprezinta upgradarea kernel-ului. Daca sistemul este un statie la distanta sau o se afla la o locatie unde nu sunt administratori de sistem experimentati exista riscul ca kernel-ul nou sa nu boot-eze si problema sa devina mai grava.

In Slackware sau sisteme similare se poate pur si simplu comenta liniile din /etc/rc.d/* ce initializeaza serviciile nefolosite. In acest caz se poate folosi programul grep pentru identificarea acestora. Trebuie insa ca dupa modificarea fisierelor de configurare sa se si inchida serviciile care ruleaza deja dinainte de modificarea rutinei sau sa se reboot-eze sistemul si sa se verifice ca fisierele de configurare au fost modificate corect (in acest caz este recomandata si crearea unui back-up).

Dezactivarea acestor servicii se face in general prin package managerul oferit de distrubuitor: RPM in cazul Red Had; dpkg in cazul Debian; YAST la sistemele SuSe si pkgtool la cele Slackware.

VERSIUNILE VECHI DE SOFTWARE

(Langa Mihai)

Linux si Unix nu sunt sisteme de operare perfecte. Lunar utilizatorii reusesc sa gaseasca diverse vulnerabilitati in structura lor. In cazul sistemului Linux viteza cu care aceste probleme sunt identificate si remediate este cea mai mare din lume de aceea una din principalele atributii ale unui administrator de sistem este sa tina pasul cu schimbarile si solutiile noi.

Fiecare versiune are o lista de mail prin care sunt anuntate noi pachete de imbunatarire a securitatii precum si locatia de unde pot fi downloadate. Exista de asemenea si liste independente la fel de eficiente precum Bugtraq sau Alert de la X-Force. Alte surse bune de informatii la zi legate de securitatea Linux pot fi gasite pe http://www.lwn.net sau http://www.linuxtoday.com. Aceste site-uri tind sa aibe o atitudine neutra fata de toti distribuitorii si sa ofere cele mai optime solutii in functie de de distribuitorul ales.

Un alt avantaj al sistemului Linux este faptul ca odata ce este gasita o solutie la o problema instalarea dureaza foarte putin iar in general sistemul nu trebuie sa stea inactiv mai mult de cateva secunde sau minute, cu exceptia problemelor ce tin de kernel, rareori fiind necesar un reboot.

PROGRAME NESIGURE SAU CONFIGURATE DEFECTUOS

(Langa Mihai)

Una din principalele probleme de securitate o reprezinta in continuare folosirea programelor nesigure cum ar fi PHP, FTP, rsh, NFS si portmap in situatii necontrolate cum trebuie si sub configuratii ineficiente.

Majoritatea administratorilor de sistem cunosc faptul ca POP si IMAP(cu exceptia incluziunii in SSL), telnet si FTP transmit datele importante cum ar fi parolele fara sa le mai encripteze. Ei stiu ca PHP, NFS si portmap au o istorie destul de mare de probleme de securitate precum si de defecte de design a sistemului de autentificare. Cu toate acestea multi continua sa le foloseasca si sa fie surprinsi atunci cand integriatea sistemului de siguranta este periclitata. Este recomandat ca in locul acestor programe sa se foloseasca spop, simap, ssh si scp-ul sau sftp-ul ssh-ului, sa se ataseze un firewall bun la subnet sau sa foloseasca retele VPN restritionate intre facilitati. Daca totusi este nevoie si de PHP e bine ca acesta sa fie la zi cu patch-urle de securitate si supravegheat constant.

Multe programe sunt sigure doar daca sunt configurate corespunzator. Multi administratori de sistem le configureaza defectuos datorita lipsei de experienta si cunostinte asupra potentialelor riscuri.

Inainte de luarea deciziei de folosire a unui serviciu e bine ca administratorul sistemului sa se documenteze asupra riscurilor de securitate si sa inteleaga cum ar putea face astfel incat programul sa ruleze intr-un mod sigur. In cazul in care acest lucru nu este posibil e bine sa fie cautate alternative sigure. Firewall-ul trebuie configurat prin subnet-uri separate pe interfete separate pentru diferite categorii de utilizatori cum ar fi studenti, personal al facultatii, vanzari, HR sau inginerie. In interiorul Firewall-ului trebuie interzisa posibilitatea existentei unei retele wireless ale carui trafic nu este encriptat cu IPsec sau cu un protocol alternativ. WEP-ul (Wired Equivalent Privacy) sau succesorii sai nu sunt o solutie buna din acest punct de vedere.

Serverele WEB si programele CGI sunt cele mai vulnerabile din punct de vedere a mentinerii securitatii sub sistem Linux si Unix. Programele CGI sunt printele cele mai usor de folosit pentru infiltrarea in sisteme pentru ca sunt programe care ruleaza pe calculatoarele utilizatorilor la cererea altor clienti si sunt capabile sa manipuleze o multime de date importante (ex: numerele card-urilor de credit, tranzactionarea intre diverse conturi, etc). Desi aceasta vulnearabilitate in securitate se poate observa si la tipuri de servere cum ar fi sendmail riscurile rezultate din accesul extern sunt mult mai mici mai ales daca sunt la zi cu patch-urile de securitate.

Iata cateva reguli ce trebuiesc respectate pentru ca un site Web sa fie securizat:

- Casutele text trebuie sa aibe o limita inferioara si una superioara in ceea ce priveste lungimea textului introdus;
- Fiecare camp de text trebuie sa aibe o anumita lista de caractere care pot fi folosite. Utilizatorii malitiosi vor trimite in geneal caractere de control si octeti non-ASCII. Majoritatea haker-ilor folosesc codarea % sau alte seturi alternative pentru a genera aceste caractere. De aceea verificarea trebuie facuta atat inainte cat si dupa codare;
- Fiecare valoare introdusa trebuie verificata de doua ori. Multe pachete de tip shopping-cart pun preturile direct in componenta chestionarelor de alegere a prodsuelor. Unii clienti pot folosi aceasta vulnerabilitate si modificand codul pot sa schimbe pretul produsului;
- In locul folosirii intervalelor pentru valorile numerice e bine, pe cat posibil, sa fie indicate valorile exacte ce pot fi folosite;
- E bine ca limbajul de programare folosit sa fie unul securizat. Datele introduse de client nu trebuie sa ajunga direct la script-ul shell pentru ca sunt multe moduri prin care un haker poate sa exploateze sau sa blocheze shell-ul. Limbajele care sunt recomandate pentru securitatea lor sunt C, C++, Perl, Java sau Python.

INSUFICIENTA RESURSELOR SI ALOCAREA PROASTA A PRIORITATILOR

(Clabescu Catalin)

In multe organizatii exista problema resurselor pentru securitate, fiind importanti foarte multi factori pentru a oferi securitate de nivel inalt: educatie, design, implementare corecta, instruirea utilizatorului, mentenanta si vigilenta continua. Deseori, securitatea este limitata de ceea ce administratorul de sistem este dispus sa faca in timpul pe care il are la dispozitie, insa problema resurselor nu reprezinta responsabilitatea administratorului. Acesta pur si simplu nu are la indemana resursele necesare pentru a implementa un sistem de securitate viabil.

Desi aceasta nu este o problema „tehnica”, reprezinta totusi cauza esecurilor de securitate pentru numeroase organizatii. Lipsa resurselor reprezinta de obicei rezultatul unei erori de prioritati. Spre exemplu, conducatorii organizatiilor care inca nu au fost victimele unor atacuri considera ca „ mass-media exagareaza proportiile unui pericol, dincolo de riscul real”. In astfel de situatii este important ca managerul respectiv sa ia la cunostiinta cazuri documentate de spargeri ale sistemelor de securitate: ca exemplu, cazul firmei TJX Cos., companie care a suferit un astfel de atac in Decembrie 2006.

In ciuda faptului ca deseori clientii sunt avertizati in legatura riscurile de securitate, acestia neglijeaza de multe ori astfel de probleme. Urmarile pot fi dezastruoase, intrucat

pagubele produse de o bresa de securitate pot fi de 10 ori mai mari decat costurile impuse de un sistem de securitate fiabil. Mai mult, acest factor de 10/1 nu ia in considerare si pierderile suferite din cauza intarzierii in productie provocate deseori de astfel de atacuri, si nu ia in considerare nici pierderea investitorilor si publicitatea negativa.

Ce poate fi facut pentru a rezolva problema resurselor insuficiente si a prioritatilor gresite? Importanta este in primul rand evidentierea slabiciunilor retelei: demonstratii cu firewall Linux, server Web, VPN. Administratorul trebuie sa arate clar cat de usor se pot sparge parolele, sau cat de simplu se poate ataca o retea Wi-Fi. Numarul atacurilor poate fi evidentiat cu ajutorul Snort si PointSentry, instalate in afara firewall-ului.

BOOT-AREA SECURIZATA (SOLUTIE)

(Clabescu Catalin)

Nici microkernel-urile si nici tunelarea nu rezolva problema. „Ceea ce vezi este ceea ce primesti”. Este usoara emularea unei interfete sau a unui dispozitiv complet care poate sa faca un utilizator sa dezvaluie date confidentiale. Un atac recent asupra PGP s-a folosit exact de acest fenomen. Tehnica de rezolvare a acestei probleme este cunoscuta de peste 10 ani sub numele de „Boot-are securizata”. Aceasta asigura faptul ca un echipament hardware specific, cu un sistem de operare specific, cu interfata grafica specifica si o aplicatie specifica ruleaza intr-adevar pe dispozitivul identificat. Echipamentul Hardware trebuie sa stabileasca identitatea unui incarcator de boot, pe incarcatorul de boot al sistemului de operare, pe sistemul de operare al interfetei cu utilizatorul, s.a.m.d, formand astfel un lant de autentificare de sus in jos cu radacina in hardware, iar un protocol de autentificare poate fi folosit pentru a verifica acest lant de la un calculator aflat la distanta. Acest calculator poate fi un server aflat la mii de kilometri distanta sau un smart card folosit pentru identificarea locala a unui dispozitiv.

Atacurile recente asupra terminalelor mobile pot fi atribuite unor asemenea probleme. Astfel, telefoane ieftine au fost vandute cu preconditia ca pentru o anumita perioada de timp doar un anumit operator de telefonie mobila poate fi folosit. Aplicarea acestei reguli intra in functionalitatea sistemului de operare. Atacatorii furau telefoanele mobile si le inlocuiau sistemele de operare. Astfel, interlocutorii nu aveau de unde sa stie daca telefonul folosea sau nu sistemul de operare original. Exista doua limitari care trebuiesc mentionate. Una este cea de protectie fizica a dispozitivelor,mai ales rezistenta la accesul neautorizat si libertatea canalelor laterale. Canalele laterale sunt mijloace de extragere a datelor confidentiale precum cheile criptografice folosit interfete nedorite. Un exemplu de interfata nedorita este consumul de putere care variaza in functie de cifra binara care este prelucrata in acest moment intr-o cheie criptografica. O interfata nedorita evidenta este accesul direct la magistrala de memorie a unui dispozitiv, de exemplu cu emulatoare in interiorul circuitului. Rezistenta la sabotaj necesita faptul ca modificarile neanuntate sunt imposibile (sau foarte putin probabile).

Obtinerea simultana a protectiei la sabotaj si a evitarii interfetelor nedorite este greu de obtinut, depinzand in primul rand de eforturile atacatorului.

Cealalta limitare impotriva careia boot-area securizata nu ofera protectie este un ataca deseori numit Mafia Fraud. In acest caz, un atacator inlocuieste un dispozitiv util cu unul fals care directioneaza toate comunicatiile catre dispozitivul original, realizand astfel toti pasii din protocolul de boot-are securizata.

Apoi acesta poate arata o interfata arbitrara cu utilizatorul, care il poate insela pe acesta, facandu-l sa divulge date confidentiale. Mai multe tehnici pentru rezolvarea acestei probleme au fost propuse, iar cea mai promitatoare pare sa fie bazata pe salturi frecvente, controlate de chei, intre un numar mare de canale.

TIPURI DE VIRUSI

(Langa Mihai)

Trojan Horse:
Un Trojan Horse este un program care este aparent folositor si sigur. Desi poate sa indeplineasca functioneaza pe care utilizatorul doreste sa o foloseasca, in paralel programul poate rula si alte instructiuni ce sunt malitioase pentru sistem si utilizator. Acest tip de program nu se autoreplicheaza si are in general doar scopul de a compromite securitatea sistemului. Ele trebuiesc trimise de catre alti utilizatori sau trebuiesc atasati diverserlor programe. Multe astfel de program sunt aparent niste glume sau software de circulatie.

Worm:
Un worm este un program care se autoreplicheaza inclusiv de pe un disc pe altul sau cu ajutorul email-ului sau al altei interfete de transport. In general acesta patrunde in sistem folosindu-se de vulnerabilitatile sale de securitate (spre exemplu printr-un email infectat)

Virus de bootsector:
Este un tip de virus care se ataseaza de sectiunea initiala a hard disk-ului care este citita la boot-area sistemului. Acesti virusi sunt cel mai des raspanditi prin floppy disk.

MacroVirus:
Acesti virusi se folosesc de limbajul de programare a altor programe pentru a se raspandii. Ei infecteaza documente cum ar fi cele MS Word sau Excel si astfel se raspandesc de la un utilizator la altul.

Virus de tip memory resident:
Acest tip de virus se gaseste de obicei in memoria volatila a sistemului (RAM). Este in general initiat de un virus care se afla stocat in calculator si ramane in memorie care daca programul mama a fost inchis.

Virus Rootkit:

Este un tip nedetectabil de virus care are ca scop permiterea controlului sistemului din exterior. Acest tip de virus este in general instalat de catre un virus Trojan si in general este “deghizat” intr-un fisier propriu sistemului de operare.

Virus polimorfic:

Acest tip de virus nu numai ca se poate autoreplica dar isi poate si schimba amprenta digitala la fiecare replicare facand detectia sa destul de grea pentru un software de securitate nu prea sofisticat.

Bomba logica de timp:

Acesti virusi sunt programati sa fie initiati la o data specifica sau atunci care apare un anumit eveniment.

VI. Concepte fundamentale: exemple si comparatii intre Linux si Windows, UID si SETUID la liunx, SID si jetonul de acces la Windows

(Stancescu Mihai Alexandru)

I. Comparatii LINUX/WINDOWS

Utilizatorii care au decis sa treaca de la Windows la Linux au intampinat diferente majore intre cele doua sisteme de operare si de multe ori au renuntat la o solutie gratuita in favoarea unei solutii comerciale pentru ca nu au stiut aceste diferente fundamentale dintre cele doua sisteme de operare. Procesul de migrare de la Windows la Linux s-ar putea realiza mai usor daca utilizatorii ar cunoaste exact aceste diferente:

1. Pret

Tinand cont de starea actuala a economiei pretul joaca un rol major in alegerea sistemului de operare. Din ce in ce mai multe persoane acorda o atentie bine meritata sistemului de operare Linux. In aceasta zona supremația Linux-ului nu poate fi contestata deoarece majoritatea distributiilor sunt oferite gratuit. Poate va trece prin minte intrebarea “De ce este gratuit?”, raspunsul e simplu: Linux este creat si dezvoltat de o comunitate de programatori care nu lucreaza pentru aceeasi companie, Linux a fost gratuit inca de la inceputurile sale. Majoritatea programelor care au fost create pentru Linux sunt gratuite. Exista alternative la toate programele comerciale care ruleaza doar pe Windows iar faptul ca sunt gratuite nu le face mai putin calitative. In unele cazuri aceste programe gratuite si open source sunt mai bune decat alternativele comerciale.

2. Libertate

Folosind sistemul de operare Linux aveti libertatea de a alege, nu putem spune acelasi lucru despre Windows care va blocheaza la modul in care compania Microsoft considera ca ar trebui sa functioneze un sistem de operare. Microsoft considera ca daca pune la dispozitia utilizatorilor o bara de activitati, un buton Start, icoane si un system tray este suficient. Pentru unii poate asa este, dar majoritatea utilizatorilor vor sa aiba ceva diferit, personalizat sau cu mai multe functionalitati. Folosind Linux poti face sistemul de operare sa arate exact cum iti doresti, singurele limite sunt timpul si imaginatia.

3. Ierarhia fisierelor de sistem

In Linux se foloseste un sigur sistem ierarhic, totul incepe in directorul root “/”. Unitatile de stocare fiind etichetate /dev/sda, /dev/sdb etc. In Windows sistemul ierarhic este multiplu si depinde de numarul unitatilor de stocare, se foloseste un root pentru fiecare unitate de stocare. Sub Linux doar o unitate de stocare contine directorul root, celelalte unitati de stocare prezente vor fi montate in directorul /media/.

4. Suport Hardware

Aici lucrurile sunt un pic complicate deoarece sistemul de operare Windows are un segment de piata mult mai mare (chiar urias) iar majoritatea producatorilor de componente hardware vor ca produsul lor sa fie compatibil 100% cu Windows. Sub Linux suportul hardware depinde de modul in care producatorul este convins de catre dezvoltatori sa predea specificatiile. Se pot intalni cazuri in care specificatiile nu sunt eliberate de producator iar respectivele

componente hardware nu vor functiona corespunzator sub Linux. Totusi in ultimii ani a fost acordata o atentie din ce in ce mai mare de producatorii de hardware sistemului de operare Linux iar cazurile in care o componenta hardware nu functioneaza pe Linux sunt destul de izolate.

5. Securitate

Acest subiect este foarte dezbatut de ambele parti. Poate din cauza cotei de piata uriase, a vulnerabilitatilor si a atentiei acordate sistemului de operare Windows il fac mult mai slab la acest capitol decat Linux-ul. Principala vulnerabilitate a Windows-ului o reprezinta accesul la root. Pentru a face pagube pe un sistem Linux trebuie neaparat sa stii parola de acces la root. Asta nu inseamna ca Linux-ul este sigur 100%, sunt multe gauri de securitate si in Linux. In momentul cand este descoperita o vulnerabilitate in Linux aceasta este rezolvata de catre comunitate foarte repede pe cand Microsoft au demonstrat de multe ori ca au nevoie de prea mult timp pentru a rezolva o problema.

Ce este UNIX ?

Mediu interactiv care permite comunicarea directa cu calculatorul si primirea imediata a raspunsurilor si mesajelor. Mediu multiuser care imparte resursele calculatorului prin tehnica "time sharing". Mediu multitasking care permite executia mai multor programe in acelasi timp. Sistemul contine: un Kernel (nucleu) care coordoneaza functionarea interna a calculatorului (de ex. alocarea resurselor) si mai multe Shell-uri (la noi fiecare user are ca shell BASH) actioneaza ca o legatura intre utilizator si kernel prin interpretarea si executia comenzilor introduse interactiv.

Conectarea in sistem

Pentru a se conecta, un utilizator trebuie sa tasteze: numele utilizatorului (login: nume utilizator) si parola (password:parola)

Sistemul nu lucreaza cu nume ci el lucreaza doar cu numere ca de exemplu: cu numar de utilizator (UID), cu numar de grup de lucru (GID), cu director de intrare ($HOME). Deci sistemul stie doar ca ulilizatorul xxx are numarul 512 (UID) si in continuare va lucra numai cu numarul 512 pentru utilizatorul xxx.

Un utilizator conectat poate sa ceara numele sau de utilizator (prin comanda), sa intre ca alt utilizator definit.

II. Super-User, UID, GID si Setuid in Linux

Orice sistem de operare este administrat de catre useri care au conturi. Si in linux exista acest concept. De aceea, astazi o sa incercam sa ne aplecam asupra unui concept destul de cunoscut si anume, acela de: User, super-user(root), grup, ca antecamera pentru intelegerea mai bine a notiunilor urmatoare (permisiuni, permisiuni speciale, modificare permisiuni).

Spre exemplu, in linux fiecare proces care ruleaza este administrat de catre un un user.

Exista un super-user numit root, administrator al sistemului, care are drepturi depline asupra resurselor.

De aici putem vorbi de mai multe tipuri de useri:

-useri de sistem

-useri persoane

Userul de sistem este unul virtual, care administreaza dupa anumiti algoritmi felul in care un proces functioneaza in background sau foreground. Spre exemplu procesul "**wwwhttpd**" ruleaza sub userul "**wwwuser**".
Userii persoane, daca le putem spune asa, sau userii reali sunt cei care au conturi personale, care se autentifica, in general prin username si parola.

Fiecarui cont de user ii este alocat un username, o parola, un home director si un enviroment.

Administrarea utilizatorilor este una dintre activitatile *defacto* a administratorului de sistem UNIX/Linux.
Utilizatorii pot fi ori persoane reale, ori utilizatori logici. Acestia din urma sunt rezervati pentru anumite aplicatii care efectueaza activitati specifice (cum ar fi utilizatorul *apache* utilizat de serverul *httpd).* Fiecare utilizator are asociat câte un identificator *(User ID)* si un identificator de grup *(Group ID),*ambii luând valori numerice. Un grup poate contine mai multi utilizatori. Acesti doi identificatori sunt atribuiti în momentul crearii utilizatorului, însa pot fi modificati si ulterior. Folositi împreuna, *UID* si *GID* determina drepturile de acces la fisiere si la alte resurse ale fiecarui utilizator.

Fisierul care memoreaza informatiile despre utilizatori este /etc/passwd, iar cele despre grupuri este /etc/group.
Crearea unui utilizator cuprinde urmatorii pasi:

- atribuie noului utilizator un nume, un identificator de utilizator, precum si un identificator de grup;
- creeaza o intrare noua în fisierele /etc/passwd, /etc/group, eventual /etc/shadow si /etc/gshadow;
- atribuie o parola noului utilizator;
- creeaza directorul home al utilizatorului, de regula /home/nume;
- copiaza fisierele implicite aflate în /etc/skel în directorul home.

Cea mai importanta cerinta pentru pastrarea securitatii sistemului este ca toti utilizatorii sa aiba parola. Este obligatorie si stabilirea unei parole initiale pentru utilizator, care, de preferinta, va trebui sa fie schimbata de catre acesta la prima intrare în sistem.
A doua cerinta importanta este utilizarea de parole cât mai sigure. Laintroducerea parolelor, sistemul va face o comparatie cu baza de date de asa-zise "cuvinte de dictionar", adica cuvinte frecvent folosite, stocate în/usr/share/dict

Tot fiecarui user i se aloca un UID, care este, in general, pe 16 sau 32 de biti. Userul (super-userul) **Root** are UID=0.
Cu toate acestea pe acelasi UID pot fi pusi mai multi useri.
Userii fac parte din "group". Nu este obligatoriu ca un user sa faca parte numai dintr-un grup, ci poate face parte din mai multe grupuri.
Fiecarui user i se poate aloca un UID, iar fiecarui grup i se aloca un numar de indentificare numit GID.
Exista trei fisiere specifice legate de useri si group.

/etc/passwd
/etc/shadow
/etc/group
In etc/passwd se gasesc userii de sistem:
Ex: bog: x:500:500:User normal:/home/stud:/bin/bash
username: parola:uid:gid_principal:comentariu: directorul_home: shell
Obs: fiecare camp este separat prin doua puncte.
La campul parola **x**-ul ne arata ca parola userului respectiv se gaseste in /etc/shadow.
Si aici avem campuri asemanatoare cu cele din /etc/passwd, dar diferite din punct de vedere al continutului.
Ex: bog:fbL5SW3kI2legWuZ:1462:0:999:7:::
adica username : parola(hash-ul ei): nr. de zile de la 1 ian. 1970 pana in ziua cand a avut loc ultima schimbare a parolei: nr. minim de zile intre doua schimbari: nr. maxim de zile intre doua schimabari: **warn**(nr. de zile inainte de expirarea parolei in care userul este avertizat):**inactive**(nr. de zile de la expirarea parolei si pana cand contul este dezactivat): nr. de zile de la 1 ian 1970 de cand contul este dezactivat.
Obs: aici observati ca ultimele campuri sunt inactive, deoarece intre delimitatori nu exista nimic completat.
Si fisierul din **/etc/group** este structurat pe acelasi principiu:
nume_grup: parola:GUID: userii care fac parte din acest grup

Pentru a administra conturile userilor exista trei metode:
O prima metoda o reprezinta comenzile specifice shell.
Cea mai simpla, este editarea directa, de catre superuser (root) a fisierelor /etc/passwd, /etc/shadow si /etc/group.
O alta metoda o reprezinta interfata grafica (kde, gnome, xfce, etc).
Noi o sa ne aplecam asupra primei metode (comenzile specifice shell).
Pentru administrare useri exista: **useradd, usermod, userdel,** (la care se adauga optiuni: -c ‘comentariu’, -d (/home/director), -g (grup principal), -G (grup secundar), -s (shell-ul(bash)), -u (uid) -m (creaza home_director) -k(skeleton- se foloseste numai impreuna cu m)) ‘numele utilizatorului’.
-**userdel -r** (se foloseste pentru stergerea unui utilizator, iar optiunea r este folosita pentru stergerea home-directory)
Obs: comanda userdel sterge si grupul corespunzator userului doar daca numai exista si alti useri atasati acestui grup.
Comanda usermod este folosita pentru a modifica un user. Se folosesc optiuni asemanatoare cu cele de la useradd.
passwd (numele_user) – modifica parola userului.
Când se pune problema drepturilor de acces a proceselor la diverse fişiere se foloseşte un set de patru numere de identificare. Aceste sunt cunoscute sub numele de utilizatorul real şi identificatorul de grup, şi utilizatorul efectiv şi identificatorul lui de grup. Identificatorul real al unui proces este de fapt UID(*user ID*) şi GID(*group ID*) ale utilizatorului care rulează procesul. Identificatorul efectiv este de obicei acelaşi cu cel al utilizatorului real cu excepția cazului în care sunt setate opțiunile *setuid* şi *setgid*. Dacă una sau amândouă opțiunile sunt activate atunci UID sau GID efectiv al procesului vor primi valoarile UID sau GID asociate fişierului din care rulează procesul.

Să presupunem că avem un utilizator cu UID 200 şi GID 20 care doreşte să ruleze programul */usr/bin/passwd*. Acest program are UID 0 (root) şi GID 1 şi are de asemenea set bitul *setiud*. Când este rulat programul procesul asociat va avea ID-ul utilizatorului real 200, ID-ul grupului real la 20, ID-ul utilizatorului efectiv 0, iar ID-ul grupului efectiv 20.
Id-ul real est folosit pentru a identifica utilizatorul pentru care este rulat un proces. ID-urile efective sunt folosite pentru a putea realiza o sortare a permisiunilor şi privilegiilor pe care procesele le au la accesarea unui fişier. Acest lucru se realizează pe baza următorului algoritm:
Dacă ID-ul utilizatorului efectiv al procesului este 0 (root) atunci procesului nu i se aplică nici o restricție, în caz contrar treci la pasul 2.
Dacă ID-ul utilizatorului efectiv al procesului este acelaşi cu ID-ul utilizator al fişierului atunci procesului i se acordă accesul la fişier în conformitate cu drepturile de acces ale proprietarului acelui fişier. Dacă nu treci la pasul 3.
Dacă ID-ul grupului efectiv al procesului este acelaşi cu ID-ul de grup al fişierului atunci procesului i se acord accesul la fişier în conformitate cu drepturile de acces ale grupului fişierului. Dacă nu treci la pasul 4.
Acesul la fişier este acordat pe baza dreprturilor de acces ale celorlalți utilizatori (*others*).

Toate aceste numere de identificare se pot obține cu ajutorul următoarelor funcții:
uid_t getuid(void) // obține ID-ul utilizatorului real
uid_t getgid(void) // obține ID-ul grupului real
uid_t geteuid(void) // obține ID-ul utilizatorului efectiv
uid_t getegid(void) // obține ID-ul grupului efectiv
uid_t getpid(void) // obține ID-ul procesului
uid_t getppid(void) // obține ID-ul procesului părinte
uid_t getpgrp(void)// obține ID-ul de grup al procesului

III. SID in Windows

Un identificator de securitate (SID) este o valoare unică de lungime variabilă, care este utilizat pentru a identifica un principal de securitate sau un grup de securitate în sistemele de operare Windows. SIDs bine cunoscute sunt un grup de Sid care identifică generic utilizatori sau grupuri generice. Valorile lor rămâne constantă în toate sistemele de operare.

Această informație este utilă pentru depanare probleme care implică securitate. Este, de asemenea, util pentru potențiale probleme de afi□are, care poate fi văzut în editorul ACL. Un SID pot fi afişate în editorul ACL în loc de utilizator sau nume de grup.

SIDs cunoscute:
SID: S-1-0
Nume: Null autoritatea
Descriere: Un identificator de autoritate.
SID: S-1-0-0
Nume: nimeni nu
Descriere: Nici O securitate principal.

SID: S-1-1
Nume: Lume autoritatea
Descriere: Un identificator de autoritate.
SID: S-1-1-0
Nume: toată lumea
Descriere: Un grup care include toți utilizatorii, utilizatorii anonimi chiar şi oaspeții. De membru este controlată de sistemul de operare.

IV. Jetonul de acces in Windows

Conturile de utilizator de domeniu permit accesul într-un domeniu de rețea şi accesarea resurselor din rețea. În momentul accesării, prin nume de utilizator şi parolă, controlerul de domeniu utilizează aceste informații pentru autentificarea identității utilizatorului şi pentru definirea unui jeton de acces care conține informațiile despre utilizator şi configurările de securitate. Jetonul de acces vă identifică pe stațiile din domeniu pe care încercați să le accesați. Jetonul de acces e valid pe durata sesiunii de accesare. Puteți defini conturi de utilizator de domeniu doar în cadrul unui domeniu, adică, în cazul în care există în rețea o stație pe care este instalat un sistem de operare Windows 2000 Server sau versiuni ulterioare şi această stație e configurată ca şi controler de domeniu (ceea ce înseamnă că pe acea stație este instalat şi configurat serviciul Active Directory).

Un jeton de acces este un obiect care incapsuleaza descriptorii de securitate ai unui proces.Atasati unui proces,descriptorii de securitate identifica proprietarul unui obiec,in acest caz ai unui proces,si lista de acces care specifica drepturile pe care proprietarul le are asupra obiectului.Jetonul de acces este folosit de catre windows cand un proces sau un thread incearca sa interactioneze cu un obiect al carui descriptor de securitae forteaza lista de acces.

Jetonul de acces este generat de serviciul de logon cand un anumit utilizator se logheaza pe statie si credentialele oferite de catre utilizator sunt verificate in baza de autentificare,specificand drepturile pe care utilizatorul le are in descriptorul de securitate incapsulat de jeton.Jetonul este atasat fiecarui proces pornit de catre utilizator .Oricand un proces acceseaza resurse care sunt sub protectia listei de acces,Windows se uita in descriptorii de securitate ai jetonului de acces daca proprietarul acelui proces este eligibil sa acceseze acea informatie,si daca are dreptul,ca operatii poate executa (citire,scriere,executie).

VII. Functiile principale din Win32 API legate de securitate

(Radulescu Vlad Alexandru)

Windows API este o interfa ă destinată programării aplica iilor pentru sistemul de operare Microsoft Windows (API este acronimul din limba engleză pentru Application Programming Interface). Windows API este cunoscută, în general, cu numele de Win32 API, însă denumirea Windows API reflectă mai precis capacită ile i utilitatea sa, suportul atât pentru Windows 32 bi i, cât i pentru Windows 64 biti.
Microsoft Windows SDK (en. Software Development Kit) con ine documenta ia i unelte necesare programatorilor pentru a realiza aplica ii folosind Windows API. Prin Win32 API programatorul are acces direct la o mare parte a func iilor de nivel de bază (en. low-level) ale sistemului de operare, putând crea aplica ii într-un mod foarte flexibil. Windows API con ine o ofertă de servicii pentru toate aplica iile bazate pe ferestre grafice (en. windows). Această interfa ă de programare permite utilizatorilor să realizeze o interfa ă grafică propriilor aplica ii, să acceseze sistemul computerului, memoria acestuia, dispozitivele (fie de intrare, fie de ie ire), să implem zenteze sunete, imagini, video sau func iuni de re ea (respectiv Internet) în aceste aplica ii. Programarea cu Windows API înseamnă primirea, interpretarea, trimiterea de „mesaje” către „ferestre”, sau „controale” (obiecte controlabile - ex. ToolBox, EditBox, Button, Text, CheckBox). Func ia principală (en. main - WINAPI WinMain) a unei aplica ii Win32 con ine o „buclă” (structură iterativă) (ex. while() în limbajul C++) în care sunt apelate func iile de preluare i traducere a mesajelor trimise de către utilizator (prin intermediul dispozitivelor de intrare, apoi al sistemului de operare, în cazul de fa ă Windows). Mesajele sunt interpretate I trimise mai departe ferestrei active. Mesajele pot fi trimise atât de sistem, asemenea unor mesaje din "subcon tient", dacă ar fi să facem o compara ie cu creierul uman, cât i explicit, "con tient", prin func ia SendMessage(). Fiecărei ferestre i se asociază o procedură responsabilă cu interpretarea mesajelor primite. Spre exemplu, dacă unei ferestre i se trimite un mesaj de „distrugere”, aceasta dispare. Dacă unui control de tip CheckBox i se trimite mesajul de validare, în dreptul său apare binecunoscutul marcaj de validare.

Securitate
In vederea asigurarii protectiei documentelor dintr-un sistem, s-a recurs la alocarea de drepturi utilizatorilor/grupurilor de utilizatori pe baza utilizarii unui username (nume de utilizator) si a unei parole. Utilizatorii care incearca sa acceseze documentele protejate sunt autentificati conform unei liste de autorizare. Autentificarea – este procesul de verificare a identitatii digitale a unui emitator din cadrul unei comunicatii (de exemplu, o cerere de logare in sistem). Emitatorul ce se doreste a fi autentificat poate fi reprezentat de un utilizator al unui calculator, un calculator propriu-zis sau un program. Autorizarea – este procesul (component a unui sistem de operare) care protejeaza resursele unui calculator permitand utilizarea acestora numai de catre consumatorii care au primit drepturi de utilizare. Resursele sunt reprezentate de fisiere individuale, programe, componente ale calculatorului sau functionalitati oferite de catre anumite aplicatii. Exemple de consumatori de resurse pot fi – utilizatori umani, programe sau alte componente ale calculatorului.

SDK(Software Development Kit) pentru managementul de alocare a drepturilor asupra unui director activ

SDK poate fi utilizat de aplicatii pentru a impune termenii folositi in criptari digitale, indiferent de formatul si continutul lor. Scopul directorului activ – Active Directory (DA) este asigurarea serviciilor de autentificare si autorizare centralizate pentru computerele ce utilizeaza medii Windows. Directorul activ stocheaza informatii si setari intr-o baza de date centrala.

Managementul de alocare a drepturilor – RMS (Rights Management Services) Microsoft Professional Services consultativ este o opțiune de sprijin care oferă support pe termen scurt, proactivă, consultativ dincolo de necesitatea de între inere produs pauzăfixa. Acest sprijin cuprinde lucrul cu acelaşi tehnician pentru a vă ajuta cu probleme cum ar fi migrația de produs, cod revizuire sau noi programul de dezvoltare. Această opțiune de sprijin este o opțiune de la distanță, bazate pe telefon. Acest serviciu este de obicei folosit pentru angajamente mai scurte pentru dezvoltatori şi pentru profesionişti IT care nu necesită consultarea onsite tradiționale sau cont susținută servicii de management care sunt disponibile la Microsoft alte acceptă opțiuni.

Pentru mai multe informații despre serviciile Microsoft consultativ, astfel cum pe cum să se angajeze, vizitați următorul site Web Microsoft: http://support.Microsoft.com/GP/AdvisoryService

Acest scenariu de servicii de consiliere a asista clienții cu o revizuire a lor setup serviciile de domeniu Active Directory (AD DS) şi cerințele pentru a ajuta dimensiunea şi distribui Active Directory Rights Management Services (AD RMS). Dislocarea AD RMS este o procedură foarte sensibile. Dacă dislocarea este executat incorect, clienții pot găsesc într-o stare ireversibile şi sunt inutilizabile în viitor. Pentru a proteja potențial mii de bucăți de conținut în timp, desfăşurarea trebuie să fie corectă. Şi, într-o implementare care nu se efectuează în conformitate cu cele mai bune practici, acest conținut va fi la riscul. Clienții care sunt noi pentru AD RMS poate decide să urmeze documenta ia online.

În acest caz, acei clienți, de asemenea, pot fi amenin ate. Acest lucru este deoarece clienții înțelegere se bazează pe documentația on-line pentru un singur-server specifice de instalare într-un mediu de încercare. Cu toate acestea, nu recomandăm acest tip de instalare pentru o produc ie mediu de implementare: http://technet.Microsoft.com/en-us/library/cc753531 (WS.10) .aspx

Puțini oameni au experiența să ştiu ce lucruri să fie conştienți de. Dislocarea AD RMS ar trebui considerate o problemă consultativ pentru clienții Microsoft Customer Service şi asistență (CSS).

Acest scenariu Pro consultativ necesită ca un inginer de asistență CSS clienți corespunzător parametrizării curente a AD DS. Scopul acestei analize este de a înțelege cerințele şi de a ajuta dimensiunea desfăşurarea noilor servicii RMS (Rights Management). Specialistul în asistență apoi va lucra cu clientul pentru a configura RMS şi asigurați-vă că toate caracteristicile şi scenarii sunt acum de lucru.

Criptarea

Criptarea reprezinta utilizarea unor coduri cu scopul convertii datei astfel incat numai un anumit rezipient sa fie capabil sa o descifreze, folosind o cheie. Tehnologiile de criptare Microsoft include CryptoAPI 2.0, Cryptographic Service Providers (CSP), CryptoAPI Tools,

CAPICOM, WinTrust, administrarea certificatelor si dezvoltarea unei infrastructuri de chei publice perzonalizate. Data codata si decodata:Pentru a putea transmite datele de-a lungul unui mediu de comunicatie cum ar fi de exemplu o linie telefonica, data trebuie serializata, adica trebuie convertita intr-un sir de unuri (“1”) si zerouri(“0”) care se transmit serial de-a lungul liniei. Acest lucru trebuie facut astfel incat computerul care receptioneaza data sa o poata converti inapoi in formatul sau original. Serializarea este realizata cu ajutorul protocolului de comunicare, si este controlata atat software cat si prin transmisia hardware. Sunt mai multe nivele la care se face coversia datelor. Un exemplu este in figura:

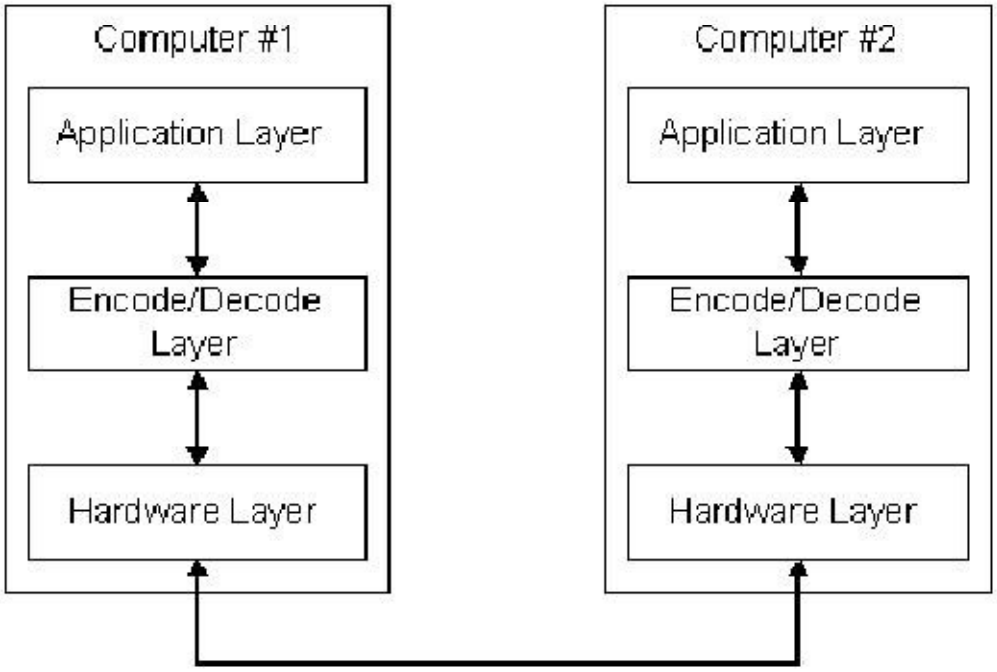


Fig. 1

In figura se exemplifica cum nivelul aplicatie al calculatorului #1 transmite data ce urmeaza sa ajunga la calculatorul #2 catre nivelul de codare/decodare. Acesta codeaza data intr-un sir de biti. La cel mai jos nivel, nivelul hardware, se convertesc bitii intr-un sir serial de unuri si zerouri care este transmis de-a lungul liniei catre calculatorul #2. Nivelul hardware al calculatorului #2 converteste unurile si zerourile inapoi in biti, si ii transmite mai sus catre nivelul de codare/decodare pentru a fi decodati, abia poi fiind transmisi catre nivelul aplicatie, aflat mai sus.Un principiu de design software acceptat este utilizarea abstractizarii, adica descrierea unei probleme sau a unui obiect in termeni generali ai acestuia (parametri generali) mai degraba decat detalierea resurselor necesare rezolvarii problemei, sau a detaliilor unui obiect.Majoritatea protocoalelor de comunicatie de folosesc de abstractizare. Obiectele de la nivelele superioare sunt definite in mod abstract si sunt destinate implementarii folosindu-se obiecte de la nivelele inferioare. De exemplu, un serviciu de la un nivel ar putea cere transferarea unor obiecte abstractizate intre calculatoare. Un nivel inferior poate utiliza reguli de codare pentru a transforma obiectul abstractizat intr-un sir de unuri si zerouri.

Criptarea si decriptarea datelor:

Criptarea este procesul translatarii datei sub forma textuala (plain text data) intr-o forma ce pare a fi aleatoare si fara un inteles aparent. Decriptarea este procedeul inver, de transformare a datei criptate inapoi in forma sa textuala.Pentru criptarea unei date de dimensiuni mai mari, este utilizata criptarea simetrica. Cheia simetrica este utilizata atat in timpul procesului de criptare cat si in timpul decriptarii. Scopul oricarui algoritm de criptare este acela de a face textul cat mai greu de descifrat. Chiar si utilizarea unei chei de 40 biti presupune existenta a mai mult de un miliard de variante posibile de criptare.

Cryptographic Service Providers:
Un CSP(Cryptographic Service Provider) contine implementarea unor standarde si a unor algoritmi de criptare. In forma sa minima, contine un DLL care implementeaza functii in CryproSPI (SPI – System Program Interface). Majoritatea CSP contin implementarea tuturor functiilor sale.
Criptarea API
Generatia urmatoare:Criptarea API – noua generatir (CNG – Cryptography API, Next Generation) va inlocui pe termen lung forma actuala, CryptoAPI. CNG se doreste a fi extensibila la mai multe nivele. Se considera ca ea va fi utila dezvoltatorilor de aplicatii care vor permite utilizatorilor crearea si schimbarea documentelor intr-un mediu sigur, cu atat mai mult utilizandu-se un mediu de transmisiune mai putin sigur cum este Internetul. Acesti dezvoltatori ar trbui sa fie familiari cu limbajul de programare C si C++ si mediul de programare de baza al Windows-ului.CNG poate fi momentan utilizat pe Windows Server 2008 si Windows Vista.

Administrarea securitatii

Tehnologiile de administrare pot fi utilizate pentru administrarea politei LSA si a politei de filtrare a parolei (password filter), cerrea de abilitate a programelor din surse externe ai a serviciul de securitate al atasamentelor care extinde sunctionalitatea uneltei de configurare a securitatii.Tehnologiile de administrare Microsoft includ polita LSA API, Filtrarea Parolelor API, si Serviciul de Securitate al Atasamentelor API. Aceste tenologii permit programatorilor dezvoltarea de aplicatii care administreaza sisteme si aplicatii.
Polita LSA – LSA este un subsistem protejat al Windows-ului care pastreaza informatii despre toate aspectele referitoare la securitatea locala a unui sistem. LSA ofera servicii de translatare intre nume si identificatori de securitate (SID – Security Identifiers).Pentru variantele de Windows Me/95/98 LSA nu face parte din sistem.
Filtrele de parole – ofera un mod de implementare de polite pentru parole si de schimbare a notificarii. Cand se efectueaza o cerere de schimbare a parolei, LSA face un apel catre filtrul de parole inregistrat in sistem. Fiecare filtru de parole este apelat de doua ori, prima data pentru a valida noua parola apoi, dupa ce toate filtrele au validat noua parola, pentru a notifica filtrele ca schimbarea s-a produs. Procesul este ilustrat in figura:

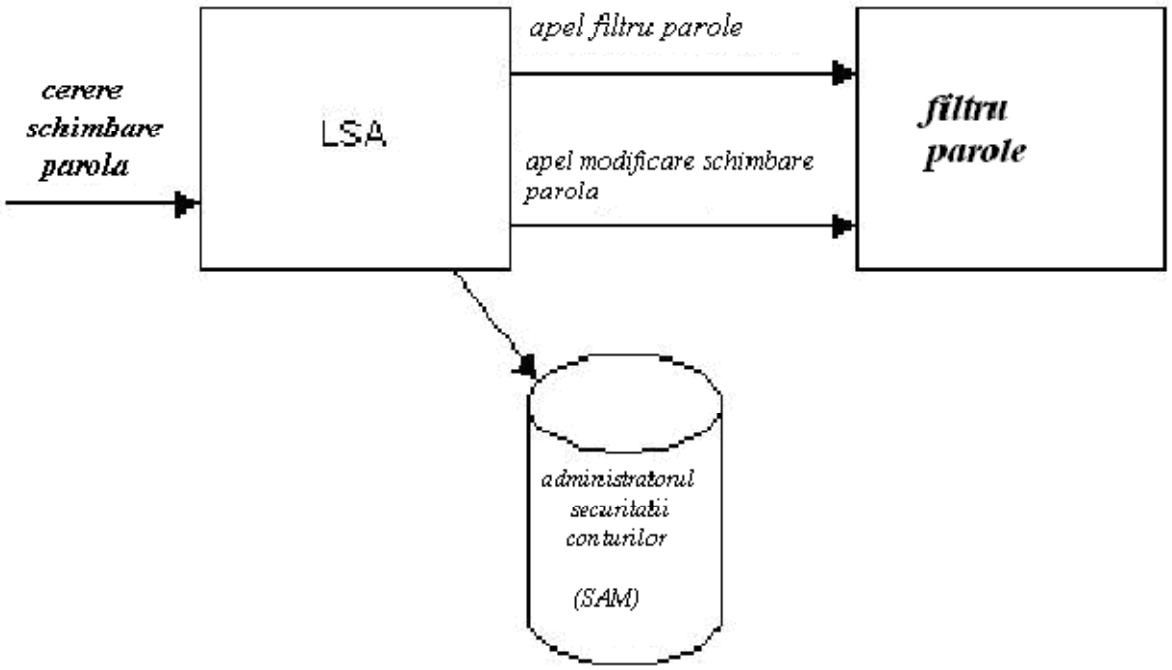


Fig. 2

Notificarea de schimbare a parolei este utilizata pentru sincronizarea schimbarii parolei cu conturile din baze de date externe. Pentru Windows NT 3.5 si versiuni anterioare acesteia, notificarea de schimbare a parolei nu este disponibila. Filtrele valideaza noua parola si indica daca aceasta respecta forma standard mentionata de polita.Functiile din Safer API ofera oricarei aplicatii care lanseaza un program dintr-o sursa externa posibilitatea de a cere permisiunea inainte de a fi lansat in executie programul respectiv. Aceste functii pot fi apelate inaintea incarcarii sau rularii executabilului. Pentru Windows Me/95/98/2000/NT aceste functii din Safer API nu sunt disponibile.

Serviciul de securitate a atasamentelor – este un set de unelte al Consolei de Administrare Microsoft (MMC – Microsoft Management Console) care simplifica configurarea si analiza securitatii sistemului. Ttusi, unele servicii au specializat cerintele de configurare care depasesc setarile de securitate oferite in mod standard. Pentru manevrarea acestor cerinte se poate extinde funtionalitatea uneltelor prin scrierea unui atasament care se ocupa de taskuri specifice de securitate. De exemplu, Spooler este un serviciu Windows NT care defineste obiecte private, care trebuie securizate, de exemplu imprimantele. Aceasta functionalitate nu este sub managementul setului standard de unelte si necesita un atasament care sa configureze si analizeze obiectele de tip imprimanta. Cand un utilizator schimba configuratia existenta, snap-in –ul Securitatii Configuratiei stocheaza noua informatie si apoi transmite cererea catre Motorul Securitatii Configuratiei. Motorul proceseaza cererea si seteaza serviciile in noua configuratie. Daca cererea afecteaza o setare standard de securitate, este procesata de catre motor. Daca respectiva cerere are un specific de servicii, atunci motorul apeleaza motorul atasament potrivit operatiei respective:

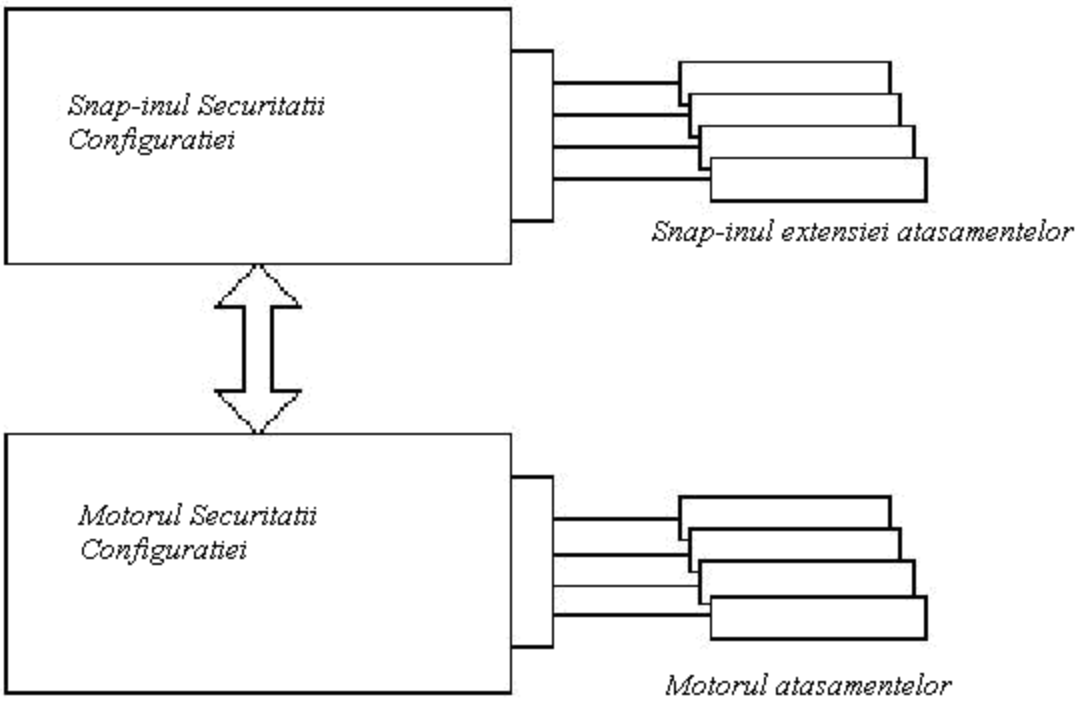

Snap-inul Securitatii Configuratiei
Snap-inul extensiei atasamentelor
Motorul Securitatii Configuratiei
Motorul atasamentelor

Fig 3.

Bibliografie

I. Implementarea securitatii la Linux si Windows: comparatie

Articole :

1.http://www.pcworld.com/businesscenter/article/202452/why_linux_is_more_secure_than_windows.html

2.http://www.pcworld.com/businesscenter/article/253787/why_switching_os_platforms_is_not_a_security_fix.html

3.http://www.pcworld.com/businesscenter/article/240262/15_essential_open_source_tools_for_windows_admins.html

4. http://www.pcworld.com/article/104693/linux_vs_windows_the_rematch.html

Imaginile au referinte pe fiecare pagina.

Carti:

Tanenbaum, "Sisteme de Operare" moderne, ediția 2-a, Ed. Biblos, Bucureşti, 2004

Site-uri:

http://www.linux.org

windows.microsoft.com

II. SELinux

a. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4755795&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F4755313%2F4755314%2F04755795.pdf%3Farnumber%3D4755795
b. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5342408&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F8013%2F5470945%2F05342408.pdf%3Farnumber%3D5342408

III. Apelurile de sistem legate de securitate la Linux

a. Tanenbaum, S. Andrew(2004), Sisteme de operare moderne, Byblos, pp 39 – 50
b. Improving Host Security with System Call Policies
   Niels Provos
   Center for Information Technology Integration
   University of Michigan
   provos@citi.umich.edu
   pp 1-3.
c. Linux man pages – Linux Programmer's Manual, pp syscall
d. Bach, Maurice J. (1986), The Design of the UNIX Operating System, Prentice Hall, pp. 15-16.
e. http://tldp.org/HOWTO/Security-HOWTO/password-security.html

f. http://www.advancedlinuxprogramming.com/alp-folder/alp-ch08-linux-system-calls.pdf pp 2-3, 5-6
g. http://security.stackexchange.com/
h. http://linux.die.net

IV. Kerberos
http://www.cs.cmu.edu/~mihaib/articole/kerberos/kerberos-html.html

V. Cele mai intalnite probleme in securitatea sistemelor Linux (probleme si solutii)
The Seven Deadly Sins of Linux Security, by Bob Toxen | May 1, 2007

VI. Concepte fundamentale: exemple si comparatii intre Linux si Windows, UID si SETUID la liunx, SID si jetonul de acces la Windows
a. http://www.islavici.ro/cursuri/Sisteme%20de%20operare%20-%20laborator/Cap6-Procese.pdf
b. http://www.scritube.com/stiinta/informatica/linux/Administrarea-utilizatorilor-s173122147.php
c. http://support.microsoft.com/kb/243330/ro
d. http://pierdutinspania.wordpress.com/tag/superuser/
e. http://sursa.md/product_info.php?products_id=225
f. http://books.google.ro/books?id=_JFGzyRxQGcC&pg=RA1-PA299&lpg=RA1-PA299&dq=ce+este+setuid+in+linux&source=bl&ots=JRXDDDE41F&sig=eW8wbE49l31EUooycY3zhxlKmhw&hl=ro&sa=X&ei=aSfOT4frAenP4QShkoG4DA&ved=0CGgQ6AEwBw#v=onepage&q=ce%20este%20setuid%20in%20linux&f=false
g. http://blog.weebo.ro/despre-useri-si-group-in-linux/

VII. Functiile principale din Win32 API legate de securitate
a. http://ro.wikipedia.org/wiki/Windows_API
b. http://stst.elia.pub.ro/news/SO_2008/SO_11_prosif/Tema%2011%20(Protectie%20si%20siguranta%20in%20functionare).pdf
c. http://support.microsoft.com/kb/2249169/ro