FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

PROJEKT PRI PREDMETU KRIPTOGRAFIJA IN TEORIJA KODIRANJA 2

SELinux: varni operacijski sistem

Avtor:
Tadej JANEŽ

Mentor:
izr. prof. dr. Aleksandar JURIŠIĆ

Povzetek

Naše življenje postaja vse bolj odvisno od računalnikov in njihovega zanesljivega delovanja. Vendar pa različna programska oprema, ki teče na njih ni brez napak, zato je smiselno uporabiti MAC sistem kot je SELinux, ki na nivoju operacijskega sistema poskrbi, da "luknja" v programu ne more vplivati na delovanje ostalih programov in operacijskega sistema samega.

Po opisu razvoja varnih operacijskih sistemov sledi opis implementacije SELinuxa v jedru Linuxa. Nato je podan pregled delovanja SELinuxa. Z opisi pripomočkov je podan praktični vidik uporabe SELinuxa in podani so primeri težav, s katerimi se pogosto srečujemo. Seznanimo se še z ostalimi konkurenčnimi varnostnimi sistemi za Linux. V zaključku podamo priporočilo za uporabo SELinuxa za določene vrste aplikacij ter opišemo, kakšno mesto predstavlja SELinux v celotnem mozaiku računalniške varnosti.

29. avgust 2008

Kazalo

1 Uvod **2**

2 Razvoj varnih operacijskih sistemov **2**

3 Implementacija SELinuxa **4**

4 Delovanje SELinuxa **5**
4.1 Varnostni konteksti 6
4.2 Trije gradniki varnostnega konteksta: uporabnik:vloga:tip 6
4.3 Uveljavljanje tipov 6
4.4 Domene subjektov in prehajanje med njimi 7
4.5 Nadzor dostopa na podlagi vloge subjekta (uporabnika/procesa) 8

5 Uporaba SELinuxa v praksi **9**

6 Primerjava z drugi varnostnimi sistemi **13**
6.1 AppArmor 13
6.2 grsecurity 13

7 Zaključek **14**

1 Uvod

Za nastanek sistema SELinux (Securety Enhanced Linux) je ključna ugotovitev, da so varnostne "luknje" v današnjih programih neizogibne. To so leta 1998 v odmevnem članku [3] opisali kasnejši avtorji SELinuxa. Kljub čedalje večji ozaveščenosti računalniških uporabnikov o potrebi po računalniški varnosti, trenutne metode zagotavljanja varnosti nimajo možnosti za uspeh. Današnje metode zagotavljanja računalniške varnosti temeljijo na zgrešeni predpostavki, da je lahko zadostna varnost zagotovljena z obstoječimi mehanizmi danes najbolj razširjenih operacijskih sistemov (Linux, Windows, OS X). V resnici potreba po varnosti zaradi čedalje večje povezanosti računalnikov in izmenjave podatkov iz dneva v dan narašča. Zato je potreben razvoj varnostnih mehanizmov na nivoju operacijskega sistema, ki bodo poskrbeli, da varnostna "luknja" enega programa ne more vplivati na delovanje ostalih programov in spodaj ležečega operacijskega sistema.

Ena od možnosti za zagotavljanje varnosti na nivoju operacijskega sistema je tudi SELinux. Sistem SELinux je dodatek k običajnemu Linuxu, ki nudi različne varnostne politike, vključno s sistemom MLS (Multi-layer Security), ki se uporablja v vojaških in drugih ustanovah, kjer je potrebno zagotoviti kontroliran hierarhičen dostop do funkcij operacijskega sistema. Implementiran je z uporabo podsistema LSM (Linux Security Modules) jedra Linuxa, ki omogoča integracijo različnih varnostnih sistemov v Linuxovo jedro. Širše gledano je SELinux množica modifikacij in dodatkov, ki jih lahko uporabimo na kateremkoli UNIXu podobnem sistemu, kot sta npr. Linux in BSD. SELinux je izumila ameriška NSA (Nacional Security Agency), ki še vedno bedi nad njegovim razvojem.

Delovanje SELinuxa je zapleteno in podrobneje opisano v sledečih poglavjih. Na tem mestu pa je podan zgolj en primer njegove koristnosti in uporabe v praksi. V spletnem strežniku Apache odkrijejo novo varnostno "luknjo" za katero popravka še ni na voljo. Le-ta napadalcu z uporabo posebno formuliranega zahtevka, ki ga pošlje spletnemu strežniku, omogoča prevzem nadzora nad Apache strežnikom. Napadalec izkoristi to "luknjo" za pridobitev privilegijev, ki jih ima strežnik Apache. Ker ima proces, v katerem teče strežnik Apache skrbniške privilegije, to napadalcu omogoča, da dobi dostop do root uporabnika in s tem neomejen nadzor nad sistemom in njegovimi sredstvi. Napadalec lahko v sistem vgradi stranska vrata za ponoven dostop do sistema, počisti dnevniške datoteke in zabriše vse svoje sledi.

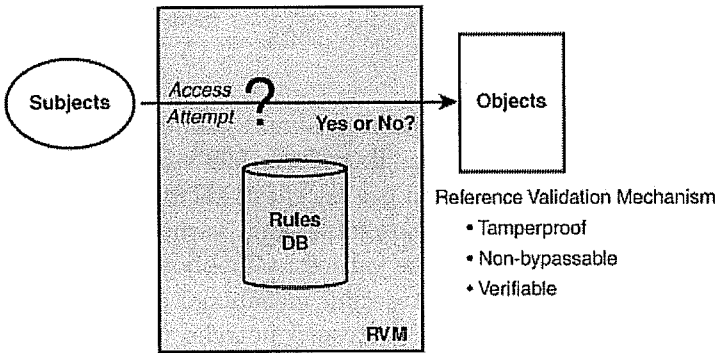
Če pa bi na strežniku tekel sistem SELinux, bi se zgodba odvijala drugače. Kot prej, bi napadalec prek varnostne "luknje" vdrl v Apache strežnik. Toda s tem bi pridobil samo pravice, ki pritičejo varnostni domeni pod katero je bil pognan Apache strežnik. Ta domena bi napadalcu onemogočila pisanje HTML datotek, ki sestavljajo spletno stran. Torej ne bi mogel spremeniti vsebine spletne strani kot tudi ne izvajati katerihkoli drugih programov. Tak napad bi torej povzročil veliko manjšo škodo kot prvi.

V nadaljevanju po opisu razvoja varnih operacijskih sistemov sledi opis implementacije SELinuxa v jedru Linuxa. Nato je podan pregled delovanja SELinuxa. Z opisi pripomočkov je podan praktični vidik uporabe SELinuxa in podani so primeri težav, s katerimi se pogosto srečujemo. Seznanimo se še z ostalimi konkurenčnimi varnostnimi sistemi za Linux. V zaključku je podano priporočilo za uporabo SELinuxa za določene vrste aplikacij ter opis mesta, ki ga predstavlja SELinux v celotnem mozaiku računalniške varnosti.

2 Razvoj varnih operacijskih sistemov

Prvi operacijski sistemi so imeli malo ali nič varnostnih mehanizmov. Uporabniki so lahko dostopali do katerekoli datoteke, če so poznali njeno ime. Kmalu zatem so se pojavili prvi sistemi nadzora dostopa, ki so nudili določen občutek varnosti.

V 1970-ih in 1980-ih letih je ameriški Department of Defence financiral prve raziskave na področju varnosti operacijskih sistemov. Rezultat tega dela je t.i. Andersonovo poročilo [1], v



Slika 1: Koncept referenčnega monitorja (Reference monitor concept). *Subjekti* so procesi, ki dostopajo do različnih sistemskih sredstev, ki jim v tem primeru pravimo *objekti.*

katerem je definiran prvi model za opisovanje nadzora dostopa v operacijskih sistemih. Ta model imenujemo Koncept referenčnega monitorja (Reference monitor concept) in je prikazan na Sliki 1. Pri referenčnem monitorju operacijski sistem izolira pasivna sredstva kot so npr. datoteke v različne *objekte* in aktivne entitete kot so procesi, ki tečejo na operacijskem sistemu, v *subjekte.* Referenčni monitor vsebuje validacijski mehanizem, ki preverja dostop med različnimi objekti in subjekti z uveljavljanjem varnostne politike v obliki množice pravil za nadzor dostopa. Na ta način lahko programom omejimo dostop do sistemskih sredstev. Odločitve nadzora dostopa temeljijo na varnostnih atributih, ki so povezani z vsakim objektom oz. subjektom. Pri standarnem Linuxu imajo subjekti (v tem primeru procesi) uporabniške identifikatorje (UID) in objekti (npr. datoteke) dovoljenja za dostop, ki določajo, kateri procesi lahko odpirajo datoteke.

Razen implementacije varnostne politike, so osnovni cilji implementacije referenčnega monitorja naslednji:

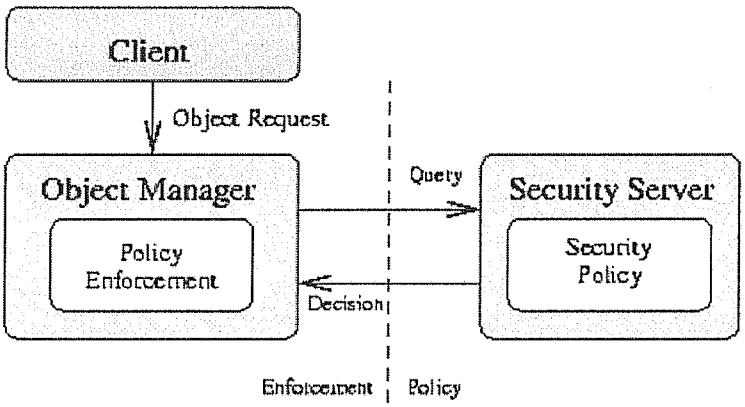
- Nespremenljivost (Tamper-proof): nadzor dostopa ne more biti zlonamerno spremenjen
- Neizbežnost (Nonbypassable): subjekti se ne morejo izogniti nadzoru dostopa
- Preverljivost (Verifiable): pravilnost implementacije varnostne politike je preverljiva

Skoraj vsi operacijski sistemi imajo neko obliko referenčnega monitorja in v njih lahko najdemo subjekte, objekte in varnostna pravila. V standardnem Linuxu so subjekti večinoma procesi in objekti razna sistemska sredstva za deljenje, hrambo in izmenjavo komunikacij (datoteke, direktoriji, vtiči, skupni pomnilnik). Pravila varnostne politike so uveljavljena skozi referenčni monitor (v tem primeru jedro) in so fiksna in nespremenljiva, medtem ko varnostne atribute, ki jih ta pravila uporabljajo za validacijo (npr. dovoljenja za dostop do datotek) lahko spreminjamo in dodajamo.

Standardni sistem varnosti na Linuxu spada med DAC (Discretionary Access Control) sisteme. DAC je oblika nadzora dostopa, kjer lastnik objekta določa nadzor dostopa po svoji volji. Lastnik lahko doda, spreminja ali odstrani nadzor dostopa za objekt. Tak sistem je prisoten v večini današnjih operacijskih sistemov (Linux, Windows, OS X). Poznamo več različic DAC sistemov. Večina jih temelji na identiteti uporabnika (user-identity-based), bolj splošni pa uporabljajo sezname nadzorov dostopa (access control list).

Največja hiba vseh DAC sistemov je v tem, da ne ločijo med človeškimi uporabniki sistema in računalniškimi programi. Ponavadi DAC posnema lastniški koncept, kjer ima npr. lastnik datoteke dovoljenje, da po svoji presoji določi dostop do datoteke in dodeli dostop samo tistim uporabnikom, ki jim zaupa [1]. Če predpostavimo, da lahko zaupamo človeškemu uporabniku (v

[1]Beseda *discretionary* (ang. *discretion* presoja) izvira iz tega, da lastnik dostop določa po svoji presoji.



Slika 2: Flask varnostna arhitektura. Komponente, ki uveljavljajo pravila varnostne politike so t.i. *upravljalci objektov* (Object managers). Komponente, ki upravljalcem objektov nudijo varnostne odločitve so t.i. *varnostni strežniki* (Security servers).

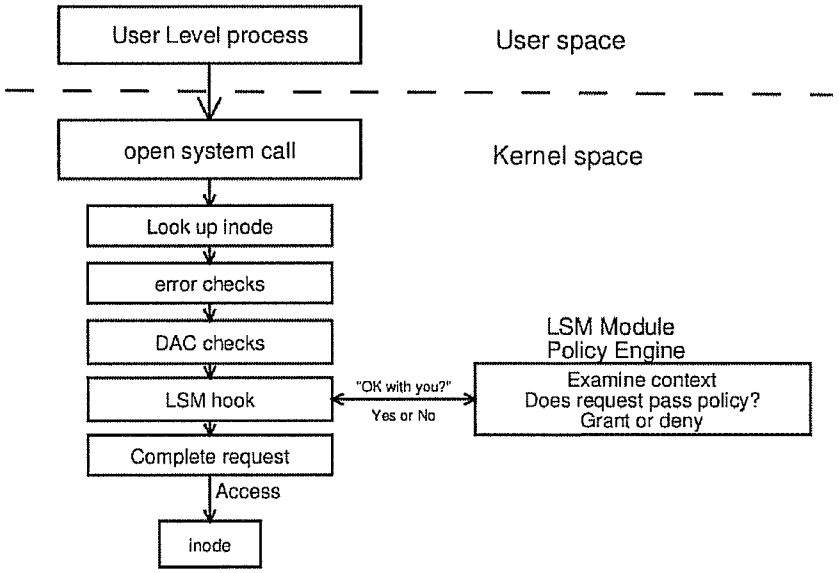
splošnem je to zelo napačna predpostavka), delovanje računalnikov ne posnema realnega sveta. Uporabniki se zanašajo na programsko opremo, ki je niso napisali sami, da za njih opravlja naloge. V resnici torej ne dajemo uporabnikom možnosti, da dodeljujejo in uporabljajo dostop, ampak njihovi programski opremi. Vendar so programi običajno polni (zlonamernih) napak, kar pa je v nasprotju z DAC predpostavko o poštenem okolju, kjer so vsi programi zaupanja vredni in brez napak. Torej, če je uporabniku dodeljen dostop, to v resnici pomeni, da imajo vsi njegovi programi ta dostop, tudi zlonamerni.

Zato je bilo v 1970-ih in 1980-ih letih posvečeno veliko energije problemu zlonamerne in hroščate programske opreme. Cilj je bil doseči sistem MAC (Mandatory Access Control), kjer o nadzoru dostopa ne določa lastnikova presoja, temveč organizacijska varnostna politika, ki je ne morejo spreminjati posamezni programi. Večino teh raziskav je financirala vojska, ki se je osredotočila na zaščito zaupnosti zaupnih vladnih podatkov. Zato je tudi večina MAC sistemov, ki so bili implementirani do sedaj, reševala problem *večnivojske varnosti* (multilayer security).

3 Implementacija SELinuxa

Korenine SELinuxa segajo v raziskave operacijskih sistemov z visokim zagotovilom varnosti ter raziskave o mikrojedrih iz 1980-ih let. Obe veji raziskav sta se združili v projektu DTMach (Distributed Trusted Mach), ki je izšel iz projektov LOCK, ki je vseboval obliko uveljavljanja tipov v operacijskem sistemu z visokim zagotovilom varnosti, in Trusted Mach, ki je temeljil na implementaciji večnivojske varnosti v mikrojedru Mach. Pri projektu DTMach in podobnih projektih, ki so mu sledili, je sodeloval tudi raziskovalni oddelek ameriške NSA. Sad tega dela je bila nova varnostna arhitektura z imenom Flask, ki je podpirala fleksibilno in dinamično uveljavljanje tipov. Osnovni princip delovanja Flask arhitekture kaže Slika 2, podrobneje pa je opisan v [6].

Različne platforme, na katerih je bilo narejeno to delo, so bila raziskovalna mikrojedra z zelo majhnim številom uporabnikov. Zato je NSA prišla do spoznanja, da je za demonstracijo uporabnosti te tehnologije in povečanje njene podpore potrebno to tehnologijo približati širši skupnosti. Poleti leta 1999 je NSA začela implementirati varnostno arhitekturo Flask v jedro Linuxa. V decembru 2000 pa je NSA prvič javno objavila svoje delo pod imenom Security Enhanced Linux. To je bila povsem nepričakovana poteza za poznavalce varnosti in te ameriške agencije, saj je NSA izdala celotno izvorno kodo za delovanje SELinuxa. Poleg tega pa še dokumentacijo, ki je zelo nazorno in jasno opisovala to varnostno arhitekturo. SELinux je tako



Slika 3: Delovanje ogrodja LSM. Podan je primer dostopa do datoteke (vsaka datoteka ima svoj *inode*). Najprej se zgodi preverjanje standardnih Linuxovih dovoljenj DAC, nato sledi še preverjanje SELinuxovih dovoljenj s pomočjo sistema *LSM hook*.

kmalu doživel veliko zanimanje Linuxove skupnosti. Prva izdaja SELinuxa je bila kolekcija popravkov za jedro 2.2.x.

Po konferenci Linux Kernel Summit leta 2001 v Ottawi, pa so v Linuxovi skupnosti začeli projekt LSM (Linux Security Module) [8]. Njegov cilj je bil, da zgradi fleksibilno ogrodje, ki bo omogočalo dodajanje različnih varnostnih dodatkov v jedro Linuxa. Delovanje ogrodja LSM opisuje Slika 3. NSA in skupnost SELinuxa sta bila eden glavnih akterjev pri razvoju LSM in sta preko svojih potreb veliko prispevala k njegovi arhitekturi. Sočasno sta tudi prilagodila SELinux, da je začel uporabljati ogrodje LSM. Glavne funkcije ogrodja so bile integrirane v prevladujoče jedro avgusta 2002 in so tako izšle kot del jedra 2.6. Do leta 2003 je NSA s pomočjo odprtokodne skupnosti končala migracijo SELinuxa na ogrodje LSM in rezultat tega dela je bila vključitev SELinuxa v glavno jedro 2.6.

Mnoge distribucije Linuxa so začele uporabljati funkcionalnosti SELinuxa v jedru 2.6, vsaka na svoj način, vendar orodij za delo s SELinuxom takrat še bilo. Levji delež pri pripravi sistemskih knjižnic in programov za uporabo, nadzor in upravljanje SELinuxa je prispeval Red Hat preko svoje odprtokodne distribucije Fedora. S pomočjo NSA je omogočil, da je Fedora postala prva širokouporabljena distribucija z integrirano podporo za SELinux. Do takrat je bil SELinux vedno samo dodatek, ki je zahteval veliko znanja in spretnosti za svojo namestitev in uporabo. Tako je SELinux in MAC končno doseg trg prevladujočih operacijskih sistemov.

SELinux je še vedno relativno nova in kompleksna tehnologija, ki se razvija naprej. Več o tem v zaključnem poglavju.

4 Delovanje SELinuxa

Podrobnosti delovanja SELinuxovega mehanizma nadzora dostopa so zapletene in obširne ter so podrobneje opisane v [5] in [4]. Vendar so osnovni koncepti in cilji SELinuxa dokaj preprosti. V tem poglavju bodo opisani varnostni konteksti, trije glavni gradniki varnostne politike, uveljavljanje tipov, nadzor na podlagi vloge subjekta, domene objektov in prehajanje med njimi.

4.1 Varnostni konteksti

Vsi nadzori dostopa v operacijskih sistemih temeljijo na atributih za nadzor dostopa, ki pripadajo objektom in subjektom. V SELinuxu tem atributom pravimo *varnostni kontekst* (security context). Vsem objektom (datoteke, komunikacijski kanali, vtiči, omrežni naslovi, itd.) in subjektom (procesi) pripada en sam varnostni kontekst. Vsak varnostni kontekst ima tri elemente: uporabnik (user), vloga (role), identifikator tipa (type identifier). Standarden način za definicijo in prikaz varnostnega konteksta je:
`user:role:type`
Nizi, ki določajo identifikatorje za vsak element so definirani v SELinuxovem jeziku varnostne politike.

Na tem mestu lahko naredimo primerjavo med standardnim Linuxom in SELinuxom. Pri standardnem Linuxu so atributi nadzora dostopa subjektov dejanski ID uporabnika in skupine, ki je dodeljen vsem procesom s procesno strukturo v jedru. Te atribute jedro zaščiti in njihovo spreminjanje je nadzorovano. V primeru datotek inode vsake datoteke hrani množico bitov za nadzor dostopa in ID uporabnika in skupine, ki ji datoteka pripada. Prvi nadzira dostop na podlagi treh skupin bitov branje/pisanje/izvajanje, po ena skupina za lastnika datoteke, skupino datoteke in vse ostalo. Drugi pa na podlagi lastnika datoteke in skupine odloči, katero skupino bitov uporabiti pri dostopu do datoteke.

4.2 Trije gradniki varnostnega konteksta: uporabnik:vloga:tip

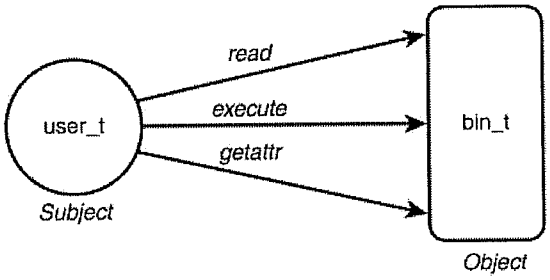
Kot je bilo prej omenjeno, v SELinuxu atribute za nadzor dostopa predstavlja trojček varnostnega konteksta. Medtem ko standardni Linux uporablja ID uporabnika/skupine procesa, datotečna dovoljenja dostopa in ID uporabnika/skupine datoteke za odločitve ali dostop dovoli ali zavrne, SELinux uporablja varnostna konteksta procesa ter objekta, do katerega proces dostopa. Ker je glavni mehanizem SELinuxovega nadzora dostopa uveljavljanje tipov, se za ugotavljanje dostopa uporablja identifikator tipa iz varnostnega konteksta.

Varnostni kontekst je preprost, konsistenten atribut nadzora dostopa. V SELinuxu je primarni del, ki ugotavlja dostop, identifikator tipa. Zaradi zgodovinskih razlogov je tip procesa ponavadi poimenovan *domena*. Identifikatorja uporabnika in vloge v varnostnem kontekstu imata malo vpliva v politiki nadzora dostopa pri uveljavljanju tipov, razen za uveljavljanje omejitev (contraint enforcement). Za procese sta identifikatorja uporabnika in vloge bolj zanimiva, ker se uporabljata za nadzor povezave med tipi in uporabniškimi identifikatorji in s tem z Linuxovim sistemov uporabniških računov. Za objekte pa skoraj ne igrata nobene vloge. Konvencija je, da je vloga objekta tipično `object_r` in uporabnik tisti uporabniški identifikator, ki ga ima proces, ki je objekt ustvaril. Vendar pa nimata vpliva na nadzor dostopa.

4.3 Uveljavljanje tipov

V SELinuxu morajo biti vsi dostopi eksplicitno dovoljeni. SELinux privzeto ne dovoli nobenega dostopa, ne glede na ID uporabnika/skupine. To pomeni, da privzeto pri SELinuxu ni superuporabnika kot je root uporabnik v standardnem Linuxu. Način, kako je dodeljen dostop za subjektov tip (t.j. domeno) in objektov tip, je z uporabo `allow` pravil. `allow` pravilo ima 4 elemente:

- *izvorni tip (source type)* Običajno tip procesa, ki želi dobiti dostop
- *ciljni tip (target type)* Tip objekta, do katerega dostopa proces
- *razred objekta (object class)* Razred objekta za katerega je željeni dostop dovoljen
- *dovoljenja (permissions)* Tipi dostopa, ki so dovoljeni izvornemu tipu za ta ciljni tip in razred objekta



Slika 4: Shematski prikaz `allow` pravila varnostne politike. Proces tipa `user_t` lahko bere, izvaja ali dobi atribute datotečnega objekta tipa `bin_t`.

Kot primer vzemimo naslednje pravilo:

```
allow user_t bin_t : file {read execute getattr};
```

Ta primer kaže osnovno sintakso allow pravila. To pravilo ima dva identifikatorja tipa: izvorni (subjektov) tip `user_t` in ciljni (objektov) tip `bin_t`. Identifikator `file` je ime razreda objekta definiranega v politiki in v tem primeru predstavlja navadno datoteko. Dovoljenja v zavitih oklepajih so podmnožica dovoljenj, ki so za veljavna za objekt razreda navadnih datotek. Prevedba tega pravila so torej glasi:
Proces tipa `user_t` lahko bere, izvaja ali dobi atribute datotečnega objekta tipa `bin_t`.
Shema tega pravila je prikazana na Sliki 4.

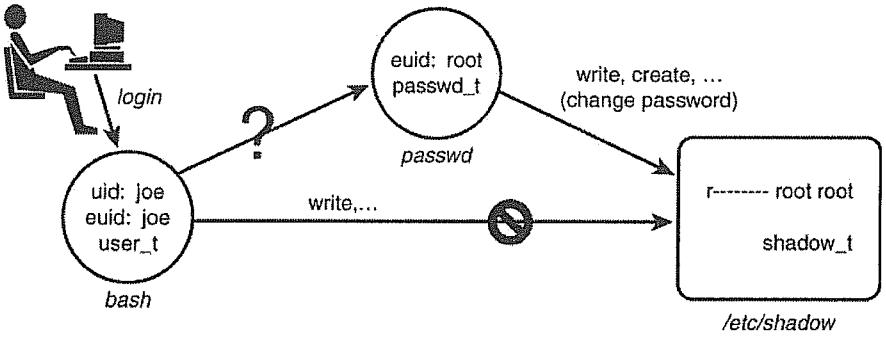
Dovoljenja SELinuxa so torej veliko bolj granularna od tistih v standardnem Linuxu, kjer imamo samo tri (`rwx`). V tem primeru sta `read` in `execute` dokaj običajna, medtem ko je `getattr` manj pogost. Dovoljenje `getattr` omogoča klicočemu subjektu, da vidi atribute kot je datum, čas, in DAC nadzorna dovoljenja datoteke. Pri standardnem Linuxu lahko klicoči subjekt vidi te atribute že če ima samo dovoljenje za iskanje po direktoriju, kjer se nahajajo te datoteke in nima bralnega dostopa do datoteke.

4.4 Domene subjektov in prehajanje med njimi

Če bi bilo treba pri pisanju politike uveljavljanja tipov napisati le `allow` pravila za dostope procesov do objektov kot so datoteke, potem bi bilo pisanje take politike enostavno. Vendar moramo najti način, kako varno poganjati programe v procesih s pravo domeno. Npr. nočemo, da lahko programi, ki jim ne zaupamo, dostopajo do `shadow` datoteke z uporabniškimi imeni in gesli v procesu z domeno `passwd_t`. To bi pomenilo, da bi ta program lahko dostopal do vseh uporabniških imen in gesel sistema. Ta primer nas vodi k problemu *prehajanja domen (domain transitions)*. Za lažjo ilustracijo problema poglejmo shemo na Sliki 5.

V tipičnem sistemu se uporabnik Joe prijavi v sistem in preko `login` procesa se zanj ustvari proces lupine `bash`. Pri standardni Linuxovi varnosti sta resnični in dejanski ID uporabnika (v tem primeru `joe`) enaka. V naši vzorčni SELinux politiki ima proces lupine tip `user_t`, ki je tip za običajne nepriviligirane uporabniške procese. Ker Joe-jeva lupina poganja ostale programe, procesi teh programov obdržijo tip `user_t`. Vprašanje je torej, kako lahko Joe zamenja svoje geslo?

Gotovo nočemo, da nepriviligiran tip `user_t` omogoča branje in pisanje `shadow` datototeke. To bi pomenilo, da lahko katerikoli program (vključno z Joe-jevo lupino) lahko vidi in spreminja vsebino te kritične datoteke. V resnici hočemo, da ima le program `passwd` ta dostop in še to samo takrat, ko teče kot proces tipa `passwd_t`. Pojavi se novo vprašanje, kako lahko nudimo varno in nevsiljivo metodo za prehod iz Joe-jeve lupine, ki teče kot tip `user_t`, v proces, ki izvaja program `passwd` s tipom `passwd_t`.



Slika 5: Problem prehajanja domen. Uporabnik Joe poganja procese s tipom `user_t`, rad pa bi spremenil svoje geslo v datoteki `shadow`. Za to rabi program `passwd`, ki teče kot tip `passwd_t`

Rešitev tega problema je vpeljava prehajanja domen, ki jo opisuje Slika 6. Glede na prejšnji primer je na tej shemi dodan tip izvršljive datoteke `passwd_exec_t`. Sedaj lahko v politiko uveljavljanja tipov dodamo prvo pravilo iz Slike 6:

```
allow user_t passwd_exec_t : file {getattr execute};
```

To pravilo lupini uporabnika Joe (`user_t`) dovoli izvedbo sistemskega klica `execve()` na izvršljivi datoteki `passwd` (`passwd_exec_t`). SELinuxovo dovoljenje `execute` je v bistvu enako kot `x` dostop v standardnem Linuxu. Najprej lupina izvede sistemski klic `fork()` in naredi še eno kopijo svojega procesa z identičnimi varnostnimi atributi. Ta kopija še vedno ohrani prvotni tip lupine (`user_t`). Torej mora biti dovoljenje `execute` namenjeno originalni domeni. Zato je izvorni tip `user_t`.

Naslednje pravilo iz Slike 6 je:

```
allow passwd_t passwd_exec_t : file entrypoint;
```

To pravilo nudi *vstopno točno dostopa (entrypoint access)* za domeno `passwd_t`. Dovoljenje `entrypoint` je precej pomembno dovoljenje SELinuxa. S tem dovoljenjem določimo, katere izvršljive datoteke oz. programi lahko "vstopajo" v domeno. Za prehajanje domene mora imeti nova domena (v tem primeru `passwd_t`) dostop `entrypoint` za izvršljivo datoteko za prehod na novo domeno. V tem primeru, če predpostavimo, da ima samo `passwd` datoteka labelo `passwd_exec_t` in ima samo tip `passwd_t` dostop `entrypoint` za `passwd_exec_t`, potem imamo konfiguracijo, kjer se lahko samo program za upravljanje gesel teče v domeni `passwd_t`. To je močno sredstvo računalniškega nadzora.

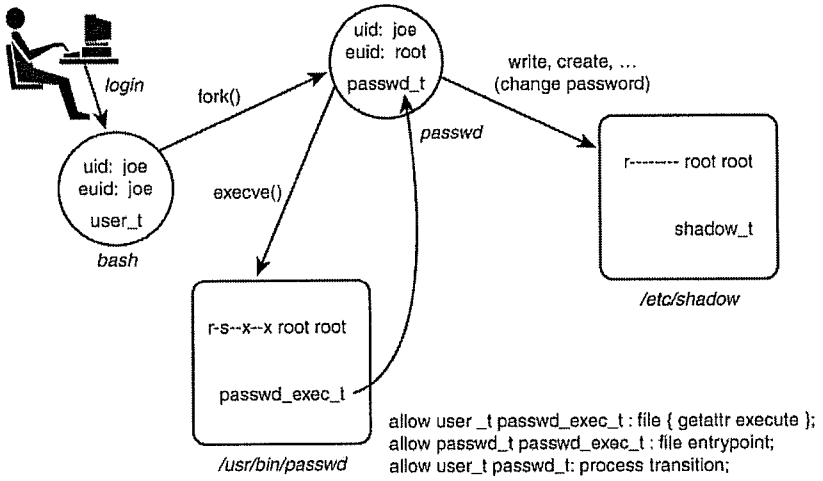
Zadnje pravilo na Sliki 6 je sledeče:

```
allow user_t passwd_t : process transition;
```

To `allow` pravilo je prvo, ki se ne nanaša na dostop do datotek. V tem primeru je razred objekta proces (`process`). V tem pravilu je dovoljenje dostop `transition`. To pravilo je potrebno, da se lahko tip procesovega varnostnega konteksta spremeni. Da je prehajanje domene možno, mora imeti prvotni tip (`user_t`) dovoljenje za prehod na novi tip (`passwd_t`).

4.5 Nadzor dostopa na podlagi vloge subjekta (uporabnika/procesa)

SELinux ponuja tudi obliko *nadzora dostopa na podlagi vloge subjekta (role-based access control, RBAC)*. Tudi ta funkcionalnost SELinuxa temelji na uveljavljanju tipov. Vloge omejujejo tipe, do katerih lahko proces prehaja na podlagi identifikatorja vloge v procesovem varnostnem kontekstu. Na ta način lahko pisec politike ustvari vlogo, ki lahko prehaja v množico domen (ob



Slika 6: Primer prehajanja domen. Uporabnik izvede program `passwd` in pri tem izvrši prehod programa iz domene `user_t` v domeno `passwd_t`. Slednje omogočajo podana tri `allow` pravila.

predpostavki, da pravila uveljavljanja tipov tako prehajanje dovoljujejo) in tako definira meje vloge.

Če vzamemo primer datoteke z gesli iz prejšnjih poglavij, mora biti sedaj tudi Joe-jevi vlogi (`user_r`) omogočeno, da se poveže z novo domeno. Shema novega primera je prikazana na Sliki 7.

Vsem procesom na sliki smo dodali vlogo `user_r`. Pravtako smo dodali novo pravilo, t.i. *stavek vloge (role statement)*:

```
role user_r type passwd_t;
```

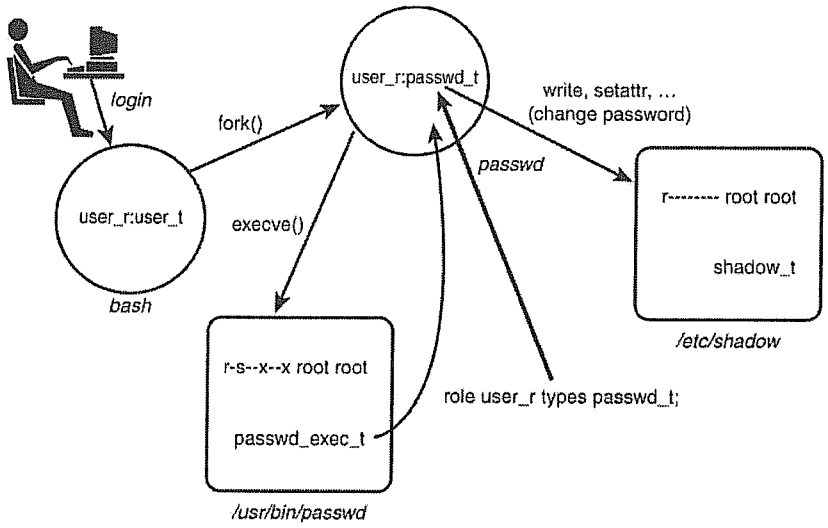
Stavek vloge deklarira identifikator vloge in ga poveže s tipi, ki so deklarirani v vlogi. Prejšnji stavek deklarira vlogo `user_r` in jo poveže s tipom `passwd_t`. To pomeni, da je tipu `passwd_t` dovoljeno sobivanje v varnostnem kontekstu z vlogo `user_r`. Brez tega stavka vloge, kontekst `joe:user_r:passwd_t` ne bil mogel biti ustvarjen, sistemski klic `execve()` bi odpovedal, čeprav politika uveljavljanja tipov dovoljuje Joe-jevemu tipu (`user_t`) ves potreben dostop.

5 Uporaba SELinuxa v praksi

Dandanes obstaja že veliko število distribucij Linuxa, ki imajo podporo za SELinux že vgrajeno. Komercialna podpora za SELinux je del Red Hat Enterprise Linux v4 in kasnejših različic. Med distribucijami, ki jih podpira odprtokodna skupnost pa je SELinux podprt v Fedori od različice 2 naprej, v Debianu od izdaje etch-a naprej, v Ubuntuju od 8.04 Hardy Herona naprej, v Hardened Gentoo-ju, in v Yellow Dog Linuxu. Obstajajo tudi popravki za SUSE in Slackware, toda njihov razvoj se je, kot kaže, ustavil.

SELinux je zelo fleksibilna arhitektura in med njegovim razvojem so se pojavili trije različni tipi varnostnih politik: *striktna (strict)*, *ciljna (targeted)* in *večnivojska (multi-layer security, MLS)*.

Prvotna varnostna politika, ki jo je objavila NSA je bila striktna. Njen cilj je bil, da zaklene celoten operacijski sistem, kontrolira ne samo sistemske demonske procese, ampak tudi uporabniške programe. Taka politika ima največ domen in pomeni največje breme za uporabnike. To varnostno politiko je imela privzeto Fedora Core 2. Striktna varnostna politika je povzročila velko problemov, saj je zapovedovala, kako naj uporabniki uporabljajo svoj sistem. Pokrivati je morala vse možne postavitve sistema. Seveda je imela zato veliko hroščev in napak in ljudje



Slika 7: Primer prehajanja domen kot je opisan na Sliki 6, le da je sedaj dodana še vloga uporabnika Joe in pravilo `role statement`, ki tipu `passwd_t` dovoljuje sobivanje v varnostnem kontekstu z vlogo `user_r`.

so jo v večini želeli izklopiti. Striktna varnostna politika se najbolje obnese takrat, ko imamo kontroliran uporabniški prostor. Npr. lahko bi postavili sistem, kjer bi varnostna politika uporabnikom dodeljevala le dostop to spletnega brskalnika za ogledovanje datotek na internetu in shranjevanje le-teh le na določene direktorije.

Po slabih izkušnjah s prvotno striktno varnostno politiko je bil potreben razmislek, kaj sploh je cilj SELinuxa. Cilj SELinuxa je, da zaščiti uporabnika pred sistemskimi programi, ki so dostopni prek mrežnega vtiča. Ti programi so bili vrata in okna, kjer so hekerji vstopali v sistem. Zato je padla odločitev, da se cilja samo določene domene in se jih zaklene, medtem ko uporabniški programi še vedno tečejo nemoteno v domeni `unconfined_t`. Procesi v tej domeni imajo enak dostop do sistema kot če SELinuxa ne bi bilo. To politiko ima na voljo Fedora od verzije 3 naprej.

Zadnji tip varnostne politike je večnivojska varnostna politika oz. krajše MLS. MLS je namenjen samo strežnikom njegov cilj pa je, da omogoči Linuxu, da dobi EAL4+ certifikat. EAL (Evaluation Assurance Level) je standard za varnost operacijskih sistemov, višja številka pomeni večjo varnost. Ta varnostna politika uporablja še četrto polje varnostnega konteksta, ki predstavlja varnostni oz. zaupnostni nivo.

SELinux shranjuje svoje konfiguracijske datoteke v mapi `/etc/selinux`. V datoteki `/etc/selinux/config` je opredeljena varnostna politika, ki jo bo SELinux izvrševal ter način, na katerega jo bo izvrševal: *enforcing* (varnostna politika je vedno uveljavljena) in *permissive* (varnostna politika samo beleži varnostna opozorila). V nujnih primerih lahko nastavitve v `/etc/selinux/config` tudi razveljavimo z uporabo parametrov jedra ob njegovem zagonu. Parameter `selinux=0` zažene jedro s SELinuxom izklopljenim, `enforcing=0` pa zažene jedro v *permissive* načinu.

Za delovanje s SELinuxom morajo biti prilagojeni tudi standardni Linuxovi pripomočki, ki ponavadi za prikaz informacij o SELinuxu uporabljajo stikalo `-Z`. Primeri:

```
ls -Z
id -Z
ps auxZ
lsof -Z
netstat -Z
find / -context=
```

Pravtako so za delo s SELinuxom prilagojeni programi `cp`, `mv`, `install`, programi za prijavo v sistem (`sshd`, `login`, `xdm`), programi za delo z gesli (`passwd`, `useradd`, `groupadd`), `rpm`, programi za arhiviranje (`tar`, `zip`, `rsync`, `star`, `amanda`). Vsi ti programi imajo vgrajeno podporo za varnostne kontekste in uveljavljanje tipov.

Poleg tega pa je bilo razvitih še nekaj novih pripomočkov, ki so namenjeni za delo s SELinuxom. Osrednja knjižnica, ki jo lahko uporabljajo programi, ki delajo s SELinuxovimi atributi, je `libselinux`. Pod to knjižnico spadajo še orodja:

- `getenforce` - pove, v katerem načinu delovanja enforcing/permissive/disable je sistem
- `setenforce 1/0` - nastavi sistem na način enforcing/permissive
- `selinuxenabled` - uporabljen v skriptah za ugotavljanje ali je SELinux omogočen
- `matchpathcon` - pove privzeti kontekst datoteke/direktorija
- `avcstat` - pokaže statistiko SELinuxovih AVC-jev

Drugi paket pripomočkov pa je `policycoreutils`. Sem spadajo orodja:

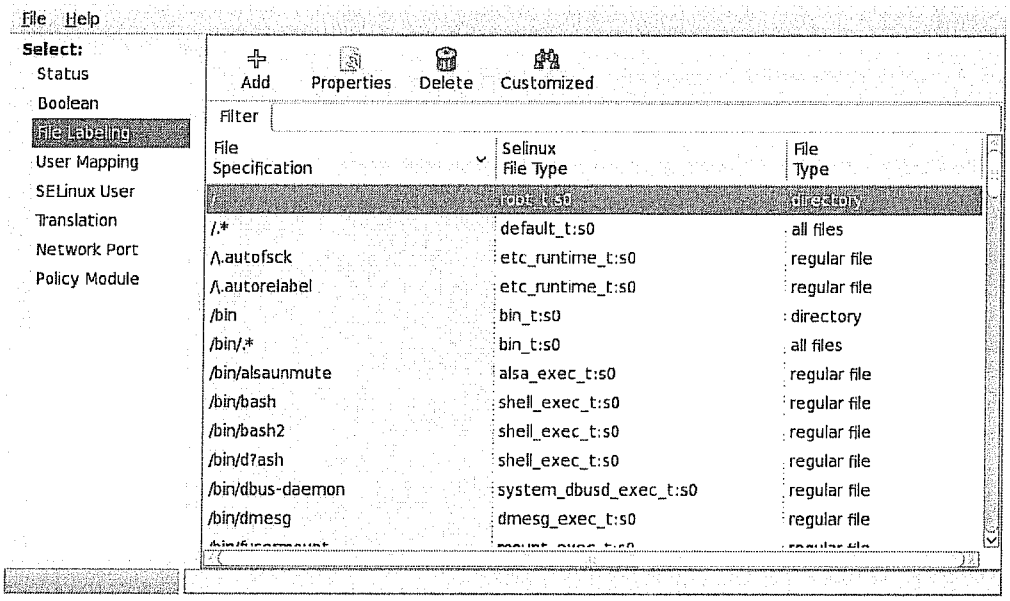
- `genhomedircon`, `fixfiles`, `restorecon`, `restorecond`, `setfiles`, `chcon`, `chcat` - orodja za označevanje datotek in delo z njihovimi konteksti
- `audit2allow`, `audit2why` - dodajanje novih pravil na podlagi dnevniških zapisov o zavrnjenih dostopih, razumevanje dnevniških zapisov o zavrnjenih dostopih
- `secon` - pokaže SELinuxov kontekst iz datoteke, programa ali uporabniškega vhoda
- `load_policy` - naloži novo SELinuxovo varnostno politiko v jedro
- `run_init` - požene init skripto v pravem SELinuxovem kontekstu
- `semanage`, `system-config-selinux` - tekstovni oz. grafični program za upravljanje nastavitev SELinuxa (delovanje `system-config-selinux`-a prikazuje Slika 8)
- `setsebool`, `getsebool` - prilagoditev varnostne politike z uporabo boolovih vrednosti
- `newrole` - požene lupino z novo SELinux vlogo

Pri uporabi SELinuxa se hitro pojavi veliko sporočil o zavrnjenih dostopih v datoteki `/var/log/messages` oz. `/var/log/audit/audit.log`. Primer sporočila, ko uporabnik želi zagnati Apache strežnik `httpd`, ki posluša na vratih 21 namesto na vratih 80:

```
host=tlinux-stable type=AVC msg=audit(1219995135.736:428):
avc: denied name_bind for pid=13842 comm="httpd" src=21
scontext=unconfined_u:system_r:httpd_t:s0 tcontext=system_u:object_r:ftp_port_t:s0
tclass=tcp_socket

host=tlinux-stable type=SYSCALL msg=audit(1219995135.736:428):
arch=40000003 syscall=102 success=no exit=-13 a0=2 a1=bfd80a00
a2=3d21cc a3=b9dca8c8 items=0 ppid=13841 pid=13842 auid=500 uid=0
gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=1
comm="httpd" exe="/usr/sbin/httpd" subj=unconfined_u:system_r:httpd_t:s0 key=(null)
```

Branje in razumevanje takih sporočil je precej naporno, zato so pri Fedori razvili nov grafični pripomoček z imenom `setroubleshoot`, ki to delo bistveno olajša. Ta program bi namesto zgornjih dveh sporočil izpisal:



Slika 8: Grafični program za upravljanje nastavitev SELinuxa `system-config-selinux`

```
Summary:
SELinux is preventing the http daemon from acting as a ftp server.
Detailed Description:
SELinux has denied the http daemon from listening for incoming connections on
the ftp port. This means that SELinux will not allow httpd to run as a ftp
server. If you did not setup httpd to run as a ftp server, this may signal a
intrusion attempt.
Allowing Access:
If you want the http daemon to listen on the ftp port, you need to enable the
httpd_enable_ftp_server boolean: "setsebool -P httpd_enable_ftp_server=1"
Fix Command:
setsebool -P httpd_enable_ftp_server=1
Additional Information:
Source Context                unconfined_u:system_r:httpd_t
Target Context                system_u:object_r:ftp_port_t
Target Objects                None [ tcp_socket ]
Source                        httpd
Source Path                   /usr/sbin/httpd
Port                          21
Host                          tlinux-stable
Source RPM Packages           httpd-2.2.9-1.fc9
Target RPM Packages
Policy RPM                    selinux-policy-3.3.1-84.fc9
Selinux Enabled               True
Policy Type                   targeted
MLS Enabled                   True
Enforcing Mode                Enforcing
Plugin Name                   httpd_enable_ftp_server
Host Name                     tlinux-stable
Platform                      Linux tlinux-stable 2.6.25.14-108.fc9.i686 #1 SMP
```

```
                                Mon Aug 4 14:08:11 EDT 2008 i686 athlon
Alert Count                     4
First Seen                      Sat 09 Aug 2008 02:18:45 PM CEST
Last Seen                       Fri 29 Aug 2008 09:32:15 AM CEST
Local ID                        eb3e1d58-3ed3-48fc-90ae-bae39edb5b1d
Line Numbers
```

S časom postaja uporaba SELinuxa čedalje bolj preprosta in lažje razumljiva tudi za širši krog uporabnikov Linuxa.

6 Primerjava z drugi varnostnimi sistemi

LIDS, RSBAC

SELinux predstavlja enega od možnih pristopov k problemu omejevanja delovanja programske opreme.

6.1 AppArmor

Sistem AppArmor, podrobneje opisan v [7], v splošnem uporablja podoben pristop kot SELinux. Gre za sistem, ki ga je v letih 2005-2007 razvijala družba Novell in pravtako dopušča upravljalcu sistema, da vsakemu programu priredi varnostni profil, ki omejuje zmožnosti, ki jih poseduje ta program. Pravtako nadgrajuje tradicionalni Linuxov DAC model.

Glavna razlika v primerjavi s SELinuxom je v tem, da ima AppArmor t.i. *učni način (learning mode)*, v katerem so kršitve politike zabeležene v dnevnik, vendar niso blokirane. Ta dnevnik lahko potem spremenimo v profil, ki opisuje tipično obnašanje programa. Zagovorniki AppArmor-ja to lastnost podpirajo, saj omogoča lažjo postavitev in upravljanje varnostnega sistema. Vendar je to tudi velik problem AppArmor-ja, saj se namreč zanaša na upravljalca sistema, da le-te točno pozna, kaj je varna uporaba programa in kaj ni. Dokaj lahko bi se zgodilo, da bi AppArmor naučili slabega obnašanja in sistem nas ne bi opozoril, da obstaja kakšen problem. Na primer, če konfiguracija strežnika vsebuje varnostne napake bomo AppArmor naučili, da je to standardno obnašanje in v prihodnosti nikoli ne bomo dobili opozoril o tem. V SELinuxu pa je že vključena preddefinirana varnostna politika, ki zagotavlja privzeto varnost.

Septembra 2007 je Novell odpustil večino razvijalcev AppArmor-ja in od takrat naprej je njegova prihodnost vprašljiva.

6.2 grsecurity

Sistem grsecurity je množica popravkov za jedro Linuxa s poudarkom na izboljšanju njegove varnosti. Tipično se uporablja za strežnike, ki sprejemajo povezave iz nepreverjenih lokacij. Primerjava med njim in SELinuxom je podana v članku [2].

Glavna komponenta grsecurity je Pax, ki med drugim označuje podatkovni del pomnilnika, kot npr. tisti na skladu, za ne-izvršljivega, in programski del pomnilnika kot ne-pisljivega. Cilj tega je preprečiti, da bi bil pomnilnik prepisan in s tem odpraviti velik del varnostnih lukenj, kot so npr. prekoračitve medpomnilnika (buffer overflows). Pax poleg tega nudi randomizacijo naslovnega prostora (address space layout randomization, ASLR), ki randomizira pomembne naslove pomnilnika in s tem prepreči napade, ki se zanašajo na fiksne naslove. Pax je na voljo tudi neodvisno od sistema grsecurity.

Paket? grsecurity ima še nekaj funkcij, kot so izboljšanje nadzora jedra Linuxa, blokiranje poganjanja programov, katerih lastnik ni root uporabnik ali katere lahko spreminjajo vsi uporabniki (in tako vsebujejo potencialno zlonamerno kodo), izboljšanje delovanja `chroot` "zaporov".

Medtem ko se SELinux uporablja MAC sistem, se grsecurity zanaša na *sezname nadzorov dostop (access control lists, ACL)*. Avtorji grsecurity-ja kot prednosti pred SELinuxom navajajo lažjo uporabo in nekatere funkcije, ki so ekskluzivne v tem sistemu, kot je zaščita naslovnega

prostora in omejevanje sredstev. Po drugi strani pa je grsecurity precej nerazširjen in slabo podprt. Tudi o kvaliteti njegove implementacije ni bilo narejenega še nobenega varnostnega poročila.

7 Zaključek

Napake v računalniški programski opremi bodo tudi v bližnji prihodnosti ostali stalnica, zato je potreben razvoj operacijskega sistema, ki bo sam po sebi omejil vpliv, ki ga imajo te (zlonamerne) varnostne "luknje" na celoten sistem. Eden od načinov, kako to zagotoviti je z uporabo varnostne politike in sistema obveznega nadzora dostopa znanega pod imenom MAC. Implementacija takega sistema, ki je na voljo za najbolj razširjene operacijske sisteme je SELinux. V osmih letih, odkar je NSA izdala prvi SELinux, odprtokodna skupnost v njem še ni našla kakšnih "skritih vrat" oz. česa podobnega, zato lahko z veliko verjetnostjo trdimo, da teorije zarote okoli NSA v povezavi s SELinuxom ne držijo.

V SELinuxu so varnostni atributi predstavljeni s trojčkom uporabnik:vloga:tip, ki mu skupaj pravimo varnostni kontekst. Od vseh je najpomembnejši zadnji, ki mu včasih namesto tip pravijo tudi domena, saj se uporablja ta t.i. uveljavljanje tipov. Uveljavljanje tipov je eden glavnih mehanizmov zagotavljanja varnosti v SELinuxu, poznamo pa še nadzor dostopa na podlagi vloge uporabnika (RBAC). Varnostna politika je preprosto množica `allow` pravil, ki zapovedujejo, kaj subjekti oz. objekti določenega tipa lahko izvajajo.

Dandanes ima večina distribucij podporo za SELinux že vgrajeno, vendar se predvsem v grafičnih pripomočkih za upravljanje in pomoč pri uporabi še vedno precej razlikujejo. Najbolje je SELinux integriran v distibucijo Fedora in njenega bratranca Red Hat Enterprise Linux, kar pa niti ne preseneča, glede na to, da Red Hat zaposljuje vodilne razvijalce na področju SELinuxa.

Obstaja tudi nekaj konkurenčnih varnostnih sistemov, kot sta npr. AppArmor in grsecurity, vendar noben od njiju nima tako široke podpore in tako zvenečih imen, ki so sodelovala v njegovem nastanku.

Od standardnega Linuxa se SELinux razlikuje predvsem v tem, da lahko zaščiti veliko več kot samo branje datotek. SELinux je samo še en nivo varnosti naložen nad običajni Linuxov DAC varnosti sistem in kot tak poskuša slednjega nadgraditi tam, kjer je to potrebno. SELinux še vedno ni nepremagljiv, je pa zelo zmogljiv in prosto dostopen varnostni sistem, ki bo okrepil varnost vsakega Linux sistema.

Literatura

[1] J.P. Anderson. Computer Security Technology Planning Study. Volume 2. 1972.

[2] M. Fox, J. Giordano, L. Stotler, and A. Thomas. SELinux and grsecurity: A Side-by-Side Comparison of Mandatory Access Control and Access Control List Implementations.

[3] P.A. Loscocco, S.D. Smalley, P.A. Muckelbauer, R.C. Taylor, S.J. Turner, and J.F. Farrell. The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments. *Proceedings of the 21st National Information Systems Security Conference*, 10:303–314, 1998.

[4] Frank Mayer, Karl MacMillan, and David Caplan. *SELinux by example.* Prentice Hall, 2007.

[5] B. McCarty. *SELinux: NSA's Open Source Security Enhanced Linux.* O'Reilly Media, Inc., 2004.

[6] R. Spencer, S. Smalley, P. Loscocco, M. Hibler, D. Andersen, and J. Lepreau. The flask security architecture: system support for diverse security policies. *Proceedings of the 8th conference on USENIX Security Symposium-Volume 8 table of contents*, pages 11–11, 1999.

[7] Wikipedia. Apparmor — wikipedia, the free encyclopedia, 2008. [Online; accessed 10-May-2008].

[8] C. Wright, C. Cowan, J. Morris, S. Smalley, and G. Kroah-Hartman. Linux Security Module Framework. *Ottawa Linux Symposium*, 2002.