Obowiązkowa kontrola dostępu w systemie Linux

Bartosz Brodecki, Piotr Sasak

Instytut Informatyki
Politechnika Poznańska
email{Bartosz.Brodecki,Piotr.Sasak}@cs.put.poznan.pl

15 maja 2007

Spis Treści

Zagrożenia

Wiele zagrożeń jest poważnych, niektóre są mniej znaczące, niestety wszystkich należy się wystrzegać.

Fakt

Systemy powinny być stale aktualizowane. Jednak sama aktualizacja nie jest wystarczająca.

Teza

Powszechnie stosowane zabezpieczenia systemów operacyjnych są niewystarczające.

Wniosek

Poprawki (aktualizacje) powinny być tworzone natychmiast po otrzymaniu informacji o błędzie, niestety czas stworzenia takiej poprawki jest dosyć długi, zwykle dłuższy niż czas opracowania skutecznej metody wykorzystania błędu.

Błąd “dnia zerowego”

Błąd “dnia zerowego”

Jest to błąd, którego nie można naprawić w danej chwili, gdyż jeszcze nie zostały stworzone odpowiednie poprawki.

Problemy “dnia zerowego” są bardzo ważne z punktu widzenia bezpieczeństwa. Należy postawić sobie następujące pytania:

- czy możemy się jakoś przed nimi zabezpieczyć?
- czy każdy taki problem może prowadzić do całkowitej kompromitacji systemu informatycznego?

Wniosek

Rozwiązaniem są systemy z obowiązkową kontrolą dostępu.

Założenia systemów z uznaniową kontrolą dostępu

Uznaniowa kontrola dostępu (DAC) (*ang. Discretionary Access Control*) jest modelem podstawowych zabezpieczeń zaimplementowanych w obecnie działających systemach operacyjnych. Podstawowe cechy modelu to:

- właściciel obiektu decyduje o uprawnieniach do tego obiektu
- brak globalnej kontroli nad przepływem informacji w systemie
- brak możliwości centralnego sterowania uprawnieniami do zasobów
- prosta implementacja

Wniosek

Obecne wymogi nakładane na systemy operacyjne sprawiają, że model DAC jest niewystarczający w wielu zastosowaniach. Stosowanie go nastręcza wiele problemów z bezpieczeństwem i konieczne jest stosowanie mechanizmów wspomagających ten model, np listy kontroli dostepu (ACL).

Geneza systemów MAC

Systemy wymuszające obowiązkową kontrolę dostępu mają związek z militarnymi i wywiadowczymi zastosowaniami systemów komputerowych. Do głównych cech tych modeli można zaliczyć:

- utrzymanie poufności i integralności danych ma priorytetowe znaczenie
- użytkownik otrzymuje przyznany typ dostępu i nie może decydować o uprawnieniach do obiektu
- istnieje globalnie zdefiniowana polityka bezpieczeństwa systemu operacyjnego określająca uprawnienia

W opracowaniu TCSEC w klasie systemów oznaczonych jako B1 jednym z wymagań jest właśnie stosowanie obowiązkowej kontroli dostępu, jako mechanizmu kontrolującego ochronę informacji.

Model Bella–LaPaduli

Model Bella–LaPaduli opracowany w latach 1973–74 przez D. E. Bella i L. J. LaPadulę.
Jest on modelem systemu z obowiązkową kontrolą dostępu.
Umożliwia on klasyfikowanie elementów systemu ze względu na ich wrażliwość.

Charakteryzuje się następującymi założeniami:

- poziom bezpieczeństwa (L) – jest odzwierciedleniem ważności danych, każdy poziom jest opisany przez parę
  - klasyfikacja (K) – hierarchiczny stopień ochrony (np. jawne, poufne, tajne, ściśle tajne),
  - zbiór kategorii (C) – nie ma struktury hierarchicznej, określa obszar zastosowań (np. policja, wojewoda, prezydent miasta).
- uprawnienia, które mogą być przypisane do każdego z obiektów.

Poziom bezpieczeństwa

Możemy porównywać dwa poziomy bezpieczeństwa L1 i L2, jeśli są spełnione następujące relacje:

$$\begin{matrix} K1 \geq K2 \\ C1 \supseteq C2 \end{matrix} \quad \text{wtedy} \quad L1 \geq L2$$

np. niech L1=(tajne, policja), L2=(poufne, KWP),
C1=(policja, KWP), C2=(KWP)

$$\text{jeśli} \quad \begin{matrix} tajne \geq poufne \\ C1 \supseteq C2 \end{matrix} \quad \text{wtedy} \quad L1 \geq L2$$

Nie wszystkie pary poziomów bezpieczeństwa są porównywalne.

Poziom autoryzacji

Poziom autoryzacji podmiotu (użytkownika) – PA podmiotu

Jest to najwyższy poziom do jakiego podmiot może uzyskać dostęp. Może on również pracować na każdym poziomie niższym względem jego poziomu autoryzacji.

Poziom bezpieczeństwa obiektu – PB obiektu

Obiekt (plik, dane) staje się aktywny w momencie przypisania jemu poziomu bezpieczeństwa.
Pozostaje nieaktywny jeśli nie posiada przypisanego poziomu bezpieczeństwa, jest wtedy niedostępny dla podmiotów.

Uprawnienia

Uprawnienia, które mogą być przypisane do każdego z obiektów:

- R *(ang. read)* – odczyt,
- A *(ang. append)* – dopisywanie danych bez możliwości odczytu,
- W *(ang. write)* – zapis danych,
- E *(ang. execute)* – wykonanie aplikacji,
- N *(ang. grant rights)* – nadawanie uprawnień.

Przykładowo twórca nowego pliku staje się jego właścicielem i ma uprawnienia do nadawania uprawnień innym podmiotom (użytkownikom), za wyjątkiem prawa N.

Wyróżniamy pojęcie podmiotu zaufanego, jest to podmiot, który gwarantuje utrzymanie bezpieczeństwa. Natomiast na podmioty nie będące zaufanymi nakłada się ograniczenia.

Aksjomaty I

Bezpieczeństwo systemu według tego modelu jest spełnione, jeśli zachowane będą następujące aksjomaty:

1. **Aksjomat bezpieczeństwa prostego** – podmiot może mieć uprawnienia R i W do obiektu, jeśli poziom autoryzacji podmiotu jest poziomem bezpieczeństwa wyższym lub równym niż poziom bezpieczeństwa obiektu.

2. **Aksjomat gwiazdki** – dla każdego podmiotu (p) nie będącego zaufanym oraz dla każdego obiektu (o), spełnione są trzy warunki:
   - p będzie posiadać uprawnienie R do o, jeśli
     PB obiektu $\leq$ aktualny PB podmiotu
   - p będzie posiadać uprawnienie W do o, jeśli
     PB obiektu $=$ aktualnemu PB podmiotu
   - p będzie posiadać uprawnienie A do o, jeśli
     PB obiektu $\geq$ aktualny PB podmiotu

Aksjomaty II

3. **Aksjomat stałości** – żaden podmiot nie ma prawa modyfikować klasyfikacji aktywnego obiektu.
   W nowszych wersjach tego modelu aksjomat ten został usunięty lub zmodyfikowany w zależności od rozpatrywanego systemu.

4. **Aksjomat bezpieczeństwa uznaniowego** – podmiot może wykonywać jedną z wielu operacji, do których posiada konieczną uprawnienie.

5. **Aksjomat niedostępności obiektu nieaktywnego** – żaden podmiot nie ma prawa R, ani W do obiektu nieaktywnego.

6. **Aksjomat niezależności stanu początkowego** – nowo aktywowany obiekt będzie mieć stan początkowy niezależnie od poprzednich aktywacji tego obiektu.

Aksjomaty III

Wniosek

Powyższe aksjomaty zostały przyjęte we wszystkich modelach stosujących obowiązkową kontrolę dostępu do informacji. Spełnienie ich zapewnia, że informacje klasyfikowane w systemie nie będą dostępne dla podmiotów, które nie otrzymały odpowiedniej autoryzacji.

RSBAC

W latach 1996-97 Amon Ott napisał pracę magisterską dotyczącą tworzonego przez niego projektu RSBAC [`http://www.rsbac.org`].

Początkowe założenia projektu:

- zaimplementowanie systemu zgodnego z MAC,
- wbudowanie dodatkowych zabezpieczeń (np. PAX).

Projekt aktualnie:

- znacznie rozszerzył bezpieczeństwo systemu Linux,
- rozszerzył obowiązkową kontrolę dostępu,
- istnieje kilkanaście modułów, które zapewniają różną funkcjonalność,
- możliwość prostego dodawania modułów dzięki Generalized Framework for Access Control (GFAC).

RSBAC II

Możliwości systemu RSBAC:

- stosowanie kilkudziesięciu uprawnień do obiektów,
- możliwość grupowania użytkowników względem określonych ról w systemie,
- przypisywanie uprawnień do ról,
- przyporządkowanie ról do aplikacji (czasem nawet kilku),
- możliwość zmiany roli (przez aplikację) w trakcie działania,

RSBAC III

RSBAC umożliwia:

- stosowanie bardzo dużych ograniczeń do każdej z ról,
- nadawanie uprawnień tylko do potrzebnych bibliotek i plików oraz gniazdek sieciowych,
- ograniczanie prawa zapisu do otwartych na początku plików,
- stosowanie innych – bardziej zaawansowanych ograniczeń.

Wniosek

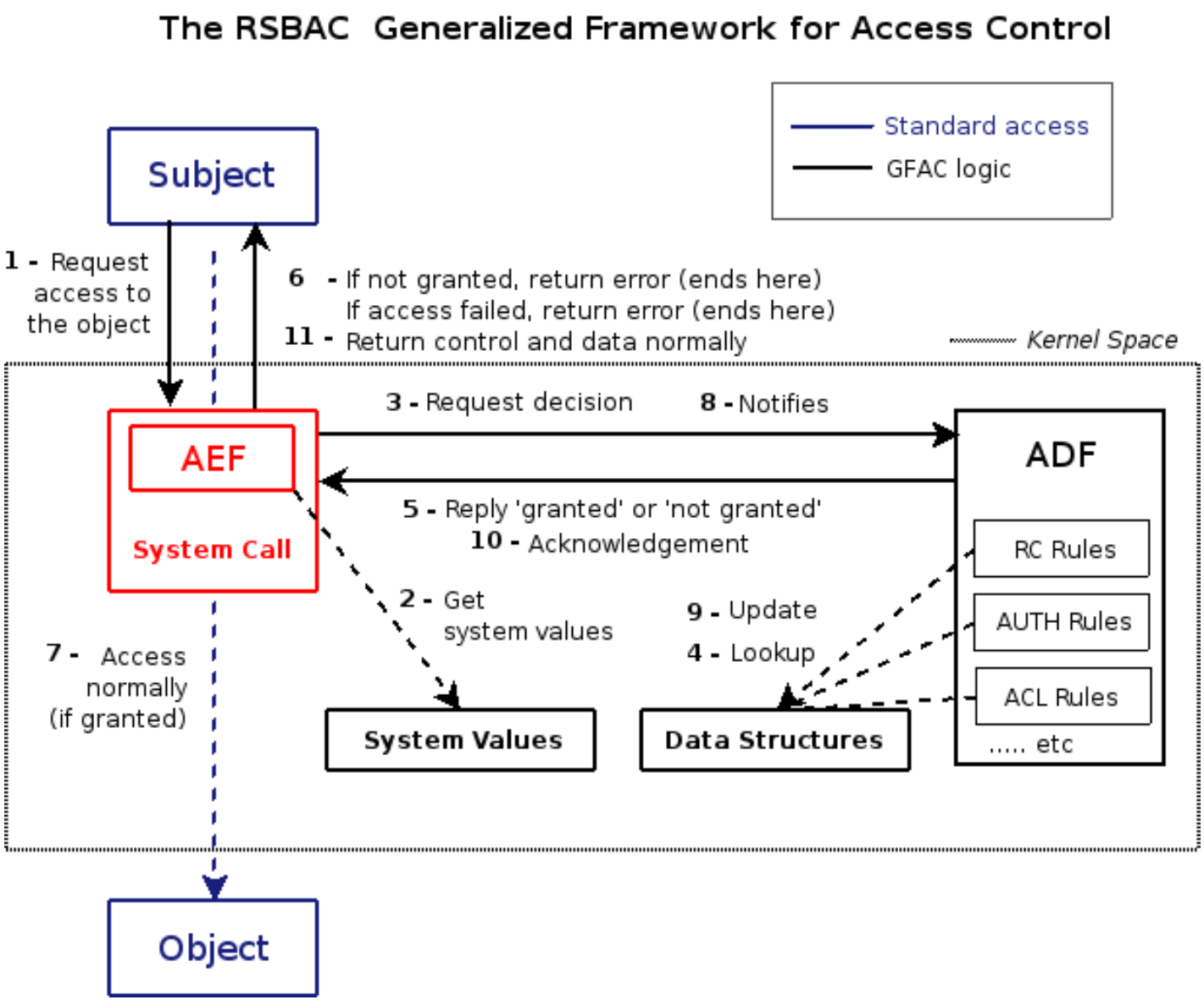
Każda aplikacja ma swoje środowisko pracy i nie może ona zwiększyć swoich uprawnień.

RSBAC IV

Projekt RSBAC charakteryzuje się:

- zapewnieniem bezpieczeństwa przed błędami “dnia zerowego”,
- niewielkim zespołem programistycznym,
- brakiem nacisku ze strony jakiegokolwiek rządu,
- dosyć dynamicznym rozwojem,
- dość szybkimi aktualizacjami (przygotowania łatek do nowych wersji jąder),
- niewielkimi środkami finansowymi.

RSBAC V - budowa GFAC



RSBAC VI

Konfiguracja:

- jest bardzo skomplikowana,
- wymaga dużo czasu i wielu testów,
- jej tworzeniem zajmuje się specjalny użytkownik - oficer bezpieczeństwa.

Polityka bezpieczeństwa:

- powinna być stosowana dla całego systemu,
- stosowanie jej jedynie dla jednej aplikacji spowoduje zablokowanie pozostałych aplikacji,
- może być tworzona dla poszczególnych aplikacji, ale implementacja polityki powinna być całościowa.

RSBAC podczas pierwszego startu tworzy podstawową politykę bezpieczeństwa, która jest dosyć restrykcyjna.

RSBAC VII

Tworzenie polityki dla konkretnej aplikacji powinno być poprzedzone analizą:

- określenie potrzebnych plików/katalogów (obiekty)
- określenie wymaganych gniazdek (obiekty)
- określenie czy uprawnienia będą zależne od stanu w którym jest proces (role - podmioty)
- określenie uprawnień łączących podmioty z obiektami

RSBAC VII - przykład

```
auth_set_cap FILE add /usr/sbin/httpd2-prefork 30
rc_copy_type FD 0 4
(...)
rc_copy_type FD 0 10
rc_copy_type NETOBJ 0 4

rc_set_item TYPE 4 type_fd_name WWW_Config
(...)
rc_set_item TYPE 10 type_fd_name TempFiles
rc_set_item TYPE 4 type_netobj_name HTTP_NETOBJ
rc_set_item TYPE 5 type_netobj_name UNIX_NETOBJ

attr_set_file_dir RC DIR /etc/apache2 rc_type_fd 4
attr_set_file_dir RC FILE /etc/localtime rc_type_fd 4
(...)
attr_set_file_dir RC DIR /tmp

rc_type_fd 10
rc_copy_role 0 4
rc_copy_role 0 5
rc_set_item ROLE 4 name WWW_Server
rc_set_item ROLE 5 name WWW_User

attr_set_user RC wwwrun rc_def_role 5
attr_set_file_dir RC FILE /usr/sbin/httpd2-prefork rc_initial_role 4

net_temp new_template 10000 UNIX
net_temp set_address_family 10000 UNIX
net_temp set_type 10000 STREAM
attr_set_net RC NETTEMP rc_type 5 10000
net_temp new_template 20000 HTTP
net_temp set_address 0.0.0.0/0
net_temp set_address_family 20000 INET
net_temp set_valid_len 20000 32
net_temp set_type 20000 STREAM
net_temp set_protocol 20000 TCP
net_temp set_min_port 20000 80
net_temp set_max_port 20000 80
attr_set_net RC NETTEMP rc_type 4 20000

rc_set_item ROLE 4 type_comp_fd 0 R EXECUTE
MAP_EXEC
rc_set_item ROLE 4 type_comp_fd 4 R
rc_set_item ROLE 4 type_comp_fd 5 R
APPEND_OPEN WRITE CREATE
(...)
rc_set_item ROLE 4 type_comp_netobj 5 CREATE
CLOSE CONNECT

rc_set_item ROLE 5 type_comp_fd 0 R EXECUTE
MAP_EXEC
rc_set_item ROLE 5 type_comp_fd 4 R
rc_set_item ROLE 5 type_comp_fd 5 R
APPEND_OPEN WRITE
(...)
rc_set_item ROLE 5 type_comp_netobj 4
GET_STATUS_DATA WRITE ACCEPT
NET_SHUTDOWN
```

RSBAC VIII

Tworzenie polityki bezpieczeństwa w systemie RSBAC może być uproszczone dzięki wykorzystaniu systemu uczenia się przez niektóre moduły.
Automatyczne uczenie się nie jest oczywiście idealne, gdyż jego zadanie jest stworzyć taką politykę, która nie będzie blokowała danej aplikacji, mogą wystąpić następujące problemy:

- nieścisła polityka,
- polityka dające zbyt duże możliwości,
- brak rozróżnienia poziomów pracy aplikacji oraz uprawnień potrzebnych na każdym z tych poziomów.

Moduły, które posiadają możliwość automatycznego uczenia się to:

- AUTH
- ACL

RSBAC IX - ciekawsze rozszerzenia

Poniżej zaprezentowano kilka ciekawszych modułów systemu RSBAC:

- **ACL** (*ang. Access Control Lists*) - umożliwia bardzo szczegółowe opisywanie uprawnień do obiektów
- **UM** (*ang. User Management*) - zmienia sposób zarządzania użytkownikami (przechowywanie informacji o użytkownikach w zastrzeżonej przez RSBAC części systemu plików)
- **DAZ** (*ang. Dazuko*) - umożliwia wykorzystywanie skanera antywirusowego dla obiektów podczas próby dostępu
- **CAP** (*ang. Linux Capabilities*) - zarządzanie systemowymi uprawnieniami, w tym ograniczanie uprawnień administratora (np. bindowanie do niskich portów)
- **RES** (*ang. Linux Resources*) - umożliwia ograniczanie zasobów dla konkretnych użytkowników/procesów (np. max liczba procesów dla użytkownika)
- **FF** (*ang. File Flags*) - ustawia dodatkowe flagi dla plików (np. bezpieczne kasowanie plików)

SELinux I

SELinux jest przykładową implementacją koncepcji obowiązkowej kontroli dostępu w systemie Linux. Rozwiązanie to zostało zaprezentowane publicznie w grudniu 2000 roku przez Amerykańską Agencję Bezpieczeństwa Narodowego *(ang. National Security Agency)*. SELinux został zaprojektowany w celu ochrony przed:

- nieautoryzowanym odczytywaniem danych,
- nieautoryzowanymi modyfikacjami danych i programów,
- możliwymi próbami obchodzenia mechanizmów bezpieczeństwa oprogramowania,
- niekontrolowanym zwiększaniem uprawnień *(ang. privilege escalation)*,
- niekontrolowanym przepływem informacji.

SELinux II

W SELinuxie występują trzy byty. Są to obiekty, podmioty i akcje.

- podmioty – elementy aktywne (procesy)
- obiekty – elementy pasywne (m.in pliki, katalogi)
- akcje – typy realizowanego dostępu przez podmioty do obiektów (zapis, odczyt, wykonanie)

Zadaniem SELinuxa jest podejmowanie decyzji czy dany podmiot może wykonać żądaną akcję na obiekcie. Decyzja jest podejmowana na podstawie utworzonej polityki bezpieczeństwa.

SELinux III

Zasada działania systemu SELinux opiera się na kliku mechanizmach:

- Tworzenie domen. Każda z domen posiada zbiór uprawnień, które ściśle określają funkcjonalność domeny.
- Każdy obiekt i podmiot jest skojarzony z domeną. W tym celu wykorzystywana jest procedura etykietowania podczas której obiekty zyskują tzw. kontekst bezpieczeństwa (ang. *security context*).
- Mechanizm type–enforcement (TE) wykorzystywany do wiązania obiektów i podmiotów z domenami.
- Mechanizm RBAC (Role Based Access Control) wspomaga działanie TE poprzez przypisanie użytkowników do ról dając w ten sposób możliwość wejścia do domeny.

SELinux IV

Zadaniem szczególnego użytkownika określanego mianem oficera bezpieczeństwa *(ang. security officer)* jest wyspecyfikowanie globalnej polityki bezpieczeństwa dla całego systemu. W szczególności:

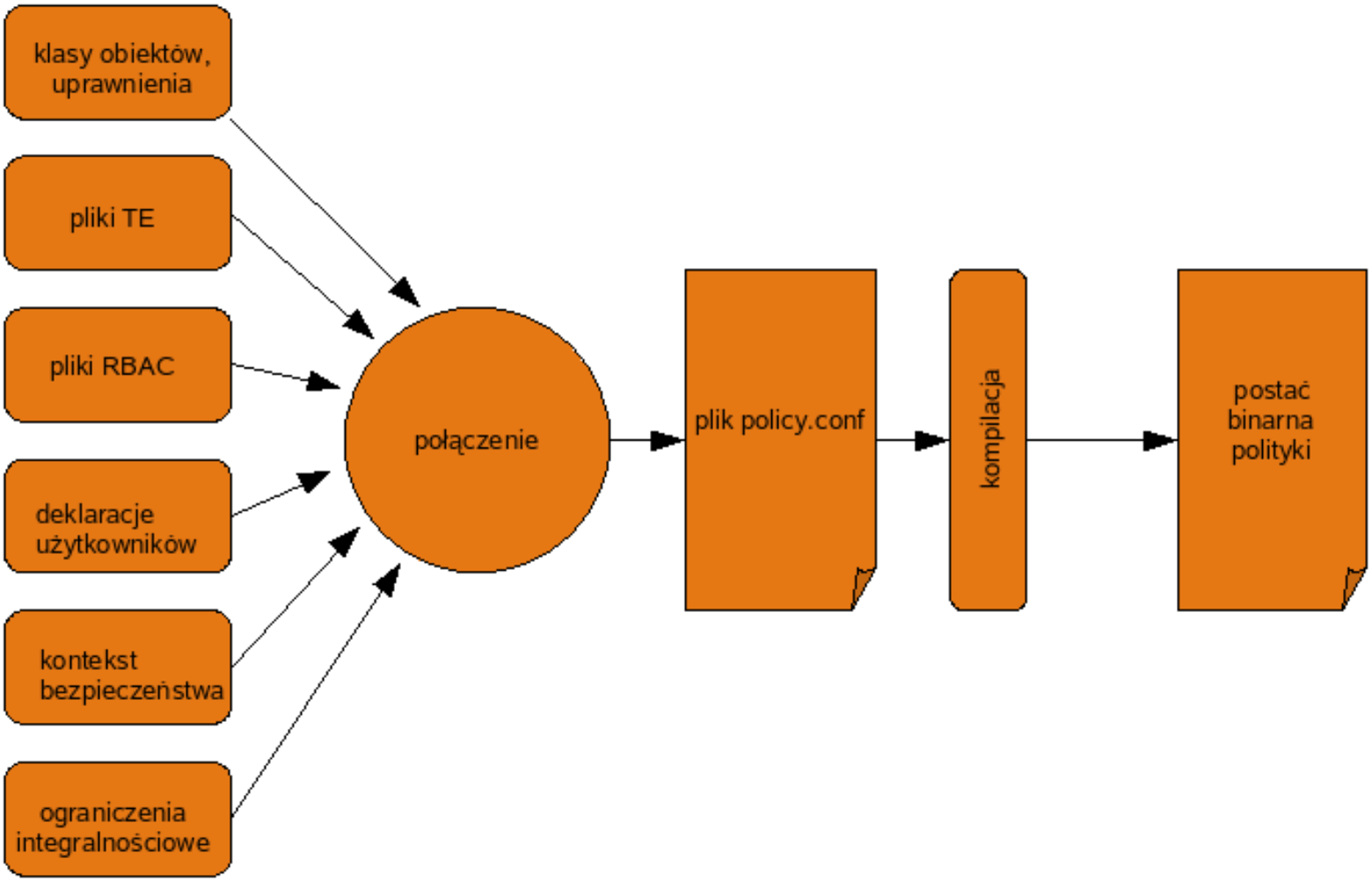
- określenie zbioru domen
- zdefiniowanie zestawu ról
- zdefiniowanie możliwych przejść między domenami
- przygotowanie bądź modyfikacja kontekstu bezpieczeństwa dla plików
- przeprowadzenie operacji etykietowania systemu plików

SELinux V I

W SELinuxie nastąpiła duża zmiana koncepcji budowy polityki bezpieczeństwa. Od początku istnienia SELinuxa zarządzanie polityką było znacznie utrudnione przez jej skomplikowaną strukturę. Poniżej zaprezentowano poglądowe rysunki przedstawiające logiczną budowę polityki monolitycznej oraz propozycji firmy Tresys Technology.

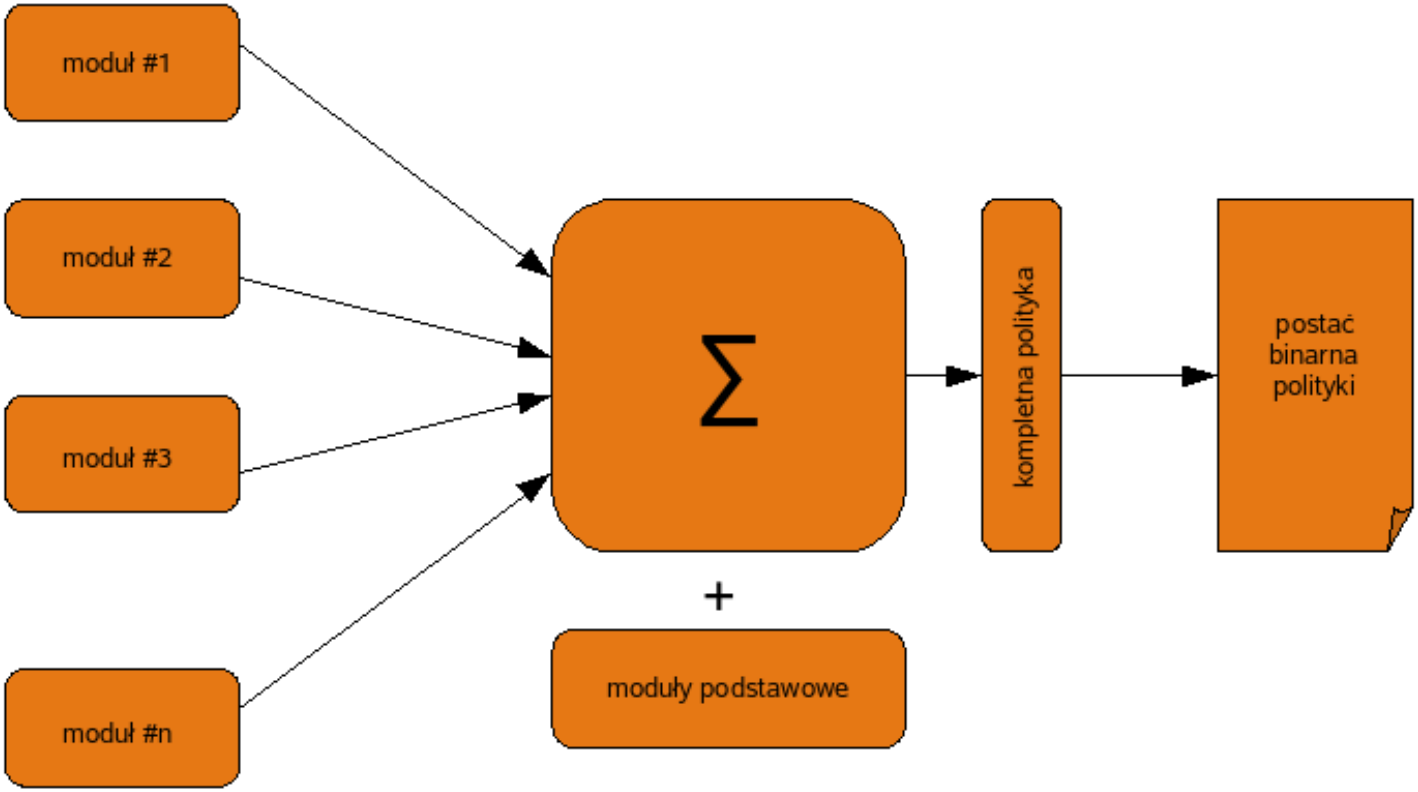
- polityka monolityczna

SELinux V II


klasy obiektów, uprawnienia
pliki TE
pliki RBAC
deklaracje użytkowników
kontekst bezpieczeństwa
ograniczenia integralnościowe
połączenie
plik policy.conf
kompilacja
postać binarna polityki

SELinux V III

- polityka modularna



SELinux VI

Każdy moduł składa się z trzech części: część prywatna zawierająca definicje typów i atrybutów(.te), zbiór interfejsów(.if) oraz zbiór definicji etykiet(.fc)

```
#przykładowy plik *.te
policy_module(bind,1.1.0)
type named_t;
type named_exec_t;
init_daemon_domain(named_t,named_exec_t)
type named_cache_t;
files_type(named_cache_t)
type named_conf_t;
files_type(named_conf_t)
type named_zone_t;
files_type(named_zone_t)
allow named_t named_cache_t:file manage_file_perms;
allow named_t named_conf_t:file r_file_perms;
allow named_t named_zone_t:file r_file_perms;
```

```
#przykładowy plik *.if
##<summary> Berkeley internet name domain (DNS) server.</summary>
##################
<summary>
Execute bind in the named domain.
</summary>
<param name="domain">
Domain allowed access.
</param>
interface ('bind_domtrans', 'gen_require('type named_t, named_exec_t;')
domain_auto_trans($1, named_exec_t, named_t)
allow $1 named_t:fd use;
allow named_t $1:fd use;
allow named_t $1:fifo_file rw_file_perms;
allow named_t $1:process sigchld;
')
```

```
#przykładowy plik *.fc
/etc/rndc.*              gen_context(system_u:object_r:named_conf_t,s0)
/usr/sbin/named          gen_context(system_u:object_r:named_exec_t,s0)
/var/named(/.*)?         gen_context(system_u:object_r:named_zone_t,s0)
/var/named/slaves(/.*)?  gen_context(system_u:object_r:named_cache_t,s0)
/var/named/data(/.*)?    gen_context(system_u:object_r:named_cache_t,s0)
```

Koncepcja modularnej polityki jest bardzo podobna do rozwiązania AppArmor firmowanego przez firmę Novell.

SELinux VII

W dystrybycji Fedora Core 5 udostępiono narzędzie policygentool, które pozwala utworzyć wstępną politykę dla danego programu (podobne narzędzie zastosowano w AppArmor). Poniżej zaprezentowano przykład użycia tego narzędzia.

Krok 1 - utworzenie wstępnej polityki(pliki: .te, .if, .fc).

```
policygentool daemond /usr/sbin/deamond
```

Krok 2 - kompilacja i utworzenie pakietu polityki(plik deamond.pp).

```
make -f /usr/share/selinux/devel/Makefile
```

Krok 3 - załadowanie pakietu polityki i ustawienie odpowiednich kontekstów bezpieczeństwa.

```
semodule -i daemond.pp
restorecon -v /usr/sbin/daemond
```

Krok 4 - ponowne uruchomienie programu i wyłączenie trybu wymuszania w SELinuxie.

```
setenforce 0
service daemond restart
```

SELinuxVIII

Oprócz zmian w samej strukturze polityki bezpieczeństwa wprowadzono możliwość wyboru polityki pod kątem profilu wykorzystania całego systemu. Są to następujące polityki:

1. Polityka o nazwie targeted – domyślnie uruchamiana polityka. Pozwala na zabezpieczanie tylko wybranych programów. Nie obejmuje całego systemu.
2. Polityka o nazwie Strict – ma za zadanie zabezpieczać cały system operacyjny. Potecjalnie może powodować problemy z stabilonością działania systemu. Nie jest zalecana dla nowych użytkowników.
3. Polityka o nazwie MLS (ang. Multi Level Security) – zawansowana postać polityki bezpieczeństwa integrująca w sobie teoretyczny model Bella-LaPaduli. Przeznaczona do zastosowań militarnych. Ma również pomóc systemowi Linux w certyfikacji do dawnego poziomu B1 (obecnie LSPP według Common Criteria).

SELinux IX

MCS (ang. Multi-Category Security) - nowy mechanizm bezpieczeństwa dla nieuprzywilejowanego użytkownika.

- Silnie bazuje na koncepcji MLS.
- Pozwala zwykłym użytkownikom korzystać z idei MLS.
- Wykorzystuje czwarte pole w kontekscie bezpieczeństwa

```
[chris@amd tmp]# ls -Z testfile.txt
-rw-r–r– chris users chris:object_r:tmp_t:Marketing testfile.txt
```

- Poziom bezpieczeństwa składa się z poziomu czułości/upoważnienia i kategori.

```
# chcat -L
s0
s0-s0:c0.c255 SystemLow-SystemHigh
s0:c0.c255 SystemHigh
s0:c0 CompanySecrets
s0:c1 Marketing
s0:c2 Finance
s0:c3 NO_Team
```

SELinux X - MCS przykłady

Nadanie kategori dla pliku:

```
[chris@amd tmp]# ls -Z newproduct.txt
-rw-r–r– chris users chris:object_r:tmp_t:Marketing newproduct.txt
# chcat     +Marketing newproduct.txt
```

Plik może posiadać wiecej niż jedną kategorię:

```
# chcat     +CompanySecrets newproduct.txt
[chris@amd tmp]# ls -Z newproduct.txt
-rw-r–r– chris users chris:object_r:tmp_t:Marketing,CompanySecrets newproduct.txt
# chcat     +Marketing newproduct.txt
```

Przypisanie użytkownikom odpowiednich kategori:

```
# semanage login -l
Login Name          SELinux User          MLS/MCS Range
__default__         user_u                  s0
chris               user_u                  s0
paul                user_u                  s0
root                root                    SystemLow-SystemHigh

# chcat -l     +Marketing     chris

# semanage login -l
Login Name          SELinux User          MLS/MCS Range
chris               user_u                  s0-Marketing
paul@amd# cat newproduct.txt
cat: newproduct.txt: Permission denied
```

Inne implementacje

Istnieją również inne implementacje obowiązkowej kontroli dostępu, poniżej kilka najbardziej popularnych:

- GRSecurity – zestaw łatek na jądro systemu Linux, [http://www.grsecurity.net]
- AppArmor – zestaw modułów i narzędzi dla systemu Linux, rozwijany przez firmę Novell, pełne wdrożenia w systemie SuSE Linux Enterprise Server 9 Service Pack 3 i OpenSuSE od wersji 10.1. [http://en.opensuse.org/Apparmor]
- LIDS – projekt powstał w 2001 roku, projekt składa się z łatek na jądro systemu Linux. [http://www.lids.org]

Porównanie DAC i MAC I

cechy	DAC	MAC	SELinux	RSBAC
łatwość zarządzania	duża	mała	mała (obecnie)	mała (obecnie)
podatność na niekontrolowany przepływ informacji	duża	mała	mała	mała
znajomość modelu wśród użytkowników	duża	mała	mała	mała
ochrona przed błędami dnia zerowego	brak	duża	duża	duża

Porównanie DAC i MAC II

globalna polityka bezpieczeństwa	brak	występuje	dedykowany język	dedykowane prg. adm.
eskalacja przywilejów	brak kontroli	podlega ścisłej kontroli	ograniczana przez domeny	ograniczona przez typy
konto “root”	występuje	brak	istnieje	istnieje
separacja procesów	nie występuje	?	występuje	występuje
dojrzałość implementacji	duża	mała	średnia	średnia
klasyfikacja według TCSEC	$< B1$	$< B3$	$\leq B1$ (?)	$\leq B1$ (?)

Porównanie DAC i MAC III

klasyfikacja rozwiązań opartych na:	C	brak	?	EAL4+ (cyberguard)
stopień integracji z systemem operacyjnym	całkowity	?	średni	średni
wymagana modyfikacja oprogramowania użytkowego	nie	?	tak	nie
współdziałanie MAC z DAC	brak (tylko DAC)	?	wymagane	opcjonalne
ochrona pamięci	brak	powinien (?)	brak	tak (PAX)

Porównanie DAC i MAC IV

ukryte kanały komunikacyjne	występują	nie powinny występo-wać	?	?
wsparcie dla X-window	tak	?	tak (?)	tak (?)

Porównanie wsparcia dla modelu Bella-LaPaduli

cechy	MAC	SELinux	RSBAC
klasyfikacja	bezograniczeń	1023	ograniczona do 64 klas
poziomy autoryzacji	bezograniczeń	1(MCS)	ograniczona do 253
uprawnienia	5	ok. 100	ok. 30

Porównianie możliwości

możliwość definiowania uprawnień do	SELinux	RSBAC
plików/katalogów	tak	tak
urządzeń (devices)	tak	tak
procesów	tak	tak
gniazd sieciowych	tak	tak
innych gniazd	tak	tak
interfejsów sieciowych	tak	tak
IPC	tak	tak
limity (ulimits)	nie	tak
capabilities	tak	tak

Obowiązkowa kontrola dostępu czy nadszedł czas, aby ją używać?

Przed podjęciem decyzji o wdrożeniu modelu MAC należy odpowiedzieć sobie na szereg pytań:

- jaki poziom bezpieczeństwa chcemy osiągnąć,
- w jaki sposób użytkownicy uzyskują dostęp do systemu,
- na jakie potencjalne zagrożenia system jest narażony,
- czy model obowiązkowej kontroli dostępu jest zrozumiały,
- czy możliwe jest określenie potrzeb każdego użytkownika, aplikacji działającej w systemie.

Wniosek I

Należy dobrze się zastanowić na ile mamy możliwości stosować system z obowiązkową kontrolą dostępu oraz na ile go faktycznie potrzebujemy.

Obowiązkowa kontrola dostępu, czy nadszedł czas, aby ją używać? II

Wniosek II

Jeżeli zależy nam na wysokim poziomie bezpieczeństwa i dysponujemy precyzyjnymi odpowiedziami na wyżej postawione pytania to warto wziąć pod uwagę wdrożenie modelu kontroli MAC. W przeciwnym wypadku wdrożenie modelu kontroli MAC może zakończyć się porażką.

Wniosek III

Przedstawione implementacje obowiązkowej kontroli dostępu są trudne w zarządzaniu, jednak znacząco wpływają na podwyższenie poziomu bezpieczeństwa.

Dziękujemy

Dziękujemy za uwagę

i

prosimy o pytania

Literatura I


J. Stokłosa, T. Bilski, T. Pankowski, Bezpieczeństwo danych w systemach informatycznych, Wydawnictwo Naukowe PWN, 2001


D. E. Bell, L. J. LaPadula, Secure computer systems: mathematical fundations. MTR-2547, vol. I-II, MITRE Corp., Bedford, MA, 1973


D. E. Bell, L. J. LaPadula, Secure computer systems: A refinement of the mathematical model. MTR-2547, vol. III, MITRE Corp., Bedford, MA, 1974


D. E. Bell, L. J. LaPadula, Secure computer systems: mathematical fundations and model. M74-244, MITRE Corp., Bedford, MA, 1974


`www.rsbac.org`


Peter A. Loscocco, Stephen D. Smalley, Patrick A. Muckelbauer, Ruth C. Taylor, S. Jeff Turner, John F. Farrell (National Security Agency), The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments, National Information Systems Security Conference, 1998


DOD 5200.28-STD. Department of Defense Trusted Computer System Evaluation Criteria, December 1985.


The Distributed Trusted Operating System, `http://www.cs.utah.edu/flux/fluke/html/dtos/HTML/dtos.html`


Flask: Flux Advanced Security Kernel, `http://www.cs.utah.edu/flux/fluke/html/flask.html`


Security-Enhanced Linux, `http://www.nsa.gov/selinux/`


Bill McCarty, SELINUX NSA's Open Source Security Enhanced Linux, O'Reilly, 2005

Literatura II


Ravi Sandhu, Edward Coyne, Hal Feinstein and Charles Youman, Role-Based Access Control Models, IEEE Computer, Volume 29, Number 2, February 1996


Marek Jawurek, RSBAC - a framework for enhanced Linux system security, RWTH Aachen


Tresys Technology, Refence Policy, `http://oss.tresys.com/projects/refpolicy`


SELinux, Fedora Core 5, `http://fedoraproject.org/wiki/SELinux`


James Morris, LIVEJOURNAL, A Brief Introduction to Multi-Category Security (MCS), `http://james-morris.livejournal.com/5583.html`


James Morris, LIVEJOURNAL, `GettingStartedwithMulti-CategorySecurity(MCS)`


Karl MacMillan, Tresys Technology, 2005 SELinux Symposium, Core Policy Management Infrastructure for SELinux, `http://selinux-symposium.org/2005/presentations/session3/3-2-macmillan.pdf`


Bartosz Brodecki, Piotr Sasak, MAC kontra DAC, co wybrać?, Pingwinaria 2007, ISBN 978-83-920463-5-6