RSBAC

"Zaufanie jest dobre, a kontrola jeszcze lepsza."

Michal Purzyński
(michal@rsbac.org)

RSBAC Team

Wprowadzenie
plan działania

- Wprowadzenie
  - Historia
  - Dlaczego?
  - Ogólnie
  - Co to tak naprawdę jest?
  - Audyt

Wprowadzenie: Historia

- Praca magisterska – Amon Ott, listopad 1996
- Pierwsza wydana wersja – 0.9 dla Linux 2.0.30, styczeń, 9, 1998
- wersja aktualna – 1.2.6 dla Linux 2.4.32 oraz 2.6.16.14
- rozwojowa – 1.3, ogromny krok naprzód

Wprowadzenie: Dlaczego?

- Słaba kontrola dostępu w systemie Linux
  - mała dokładność
  - użytkownik uid “0”
    - pełny dostęp
    - zbyt często potrzebny
    - łatwo zdobyć ten poziom dostępu

Wprowadzenie: Dlaczego?

- DAC – Discretionary Access Control
- kontrola dostępu w gestii każdego użytkownika

- pełne zaufanie dla uid “0”?

- trojany, wirusy...

Wprowadzenie: Dlaczego

- Inne powody
  - Moduły do konkretnych i specyficznych zadań administracyjnych
    - (...bo istnieją takie zadania o których sie linusowi nie śniło ;)
  - Elastyczny dobór kombinacji modeli do aktualnych potrzeb
  - Przenośna architektura
    - nie tylko Linux ma problemy z bezpieczeństwem!

Ogólne spojrzenie na RSBAC

- Rozszerzenie jadra Linux na licencji OpenSource
  - niezależne od rządowa, agencji wywiadowczych, wielkich firm
  - kilka dobrze znanych modeli bezpieczeństwa (MAC, ACL, RC)
  - kilka nowych

Ogólne spojrzenie na RSBAC

- Kontrola nad dostępem każdego użytkownika oraz aplikacji do sieci
  - (...lubię cie. wiec cie sprawdzę.)
- Możliwa dowolna kombinacja modeli!
  - moduły są w pełni niezalewne
- Łatwość uzupełniania o nowe moduły – mechanizm REG

Co to jest RSBAC?

- Rozszerzenie jadra systemu o mechanizmy umożliwiające kontrole dostępu (framework)
- Bazuje na ideach General Framework for Access Control (GFAC)
  - Abrams i LaPadula
- Elastyczna architektura
  - pełna modularność
  - może obsługiwać praktycznie każdy model bezpieczeństwa

Potężne możliwości logowania

- Możliwy szczegółowy audyt wszelkich zdarzeń zachodzących w systemie
- Logowanie ze względu na
  - aplikacje
  - użytkownika
  - obiekt
  - zadanie i decyzje

Architektura i implementacja plan działania

- Architektura i implementacja
  - Obiekty i podmioty
  - Żądania i decyzje
  - Diagram architektury
  - Model REG
  - Kontrola dostępu do sieci

Architektura: Obiekty i podmioty

- Pomioty – aktywne
  - procesy uruchamiane przez użytkowników
    - użytkownik Nastia uruchamia program firefox razem z potrzebnymi bibliotekami
- Obiety – pasywne
  - FILE, DIR, FIFO, SYMLINK
    - pliki, katalogi, fifo, linki
  - DEV
    - urządzenia identyfikowane przez major:minor
  - IPC
    - komunikacja miedzy procesami

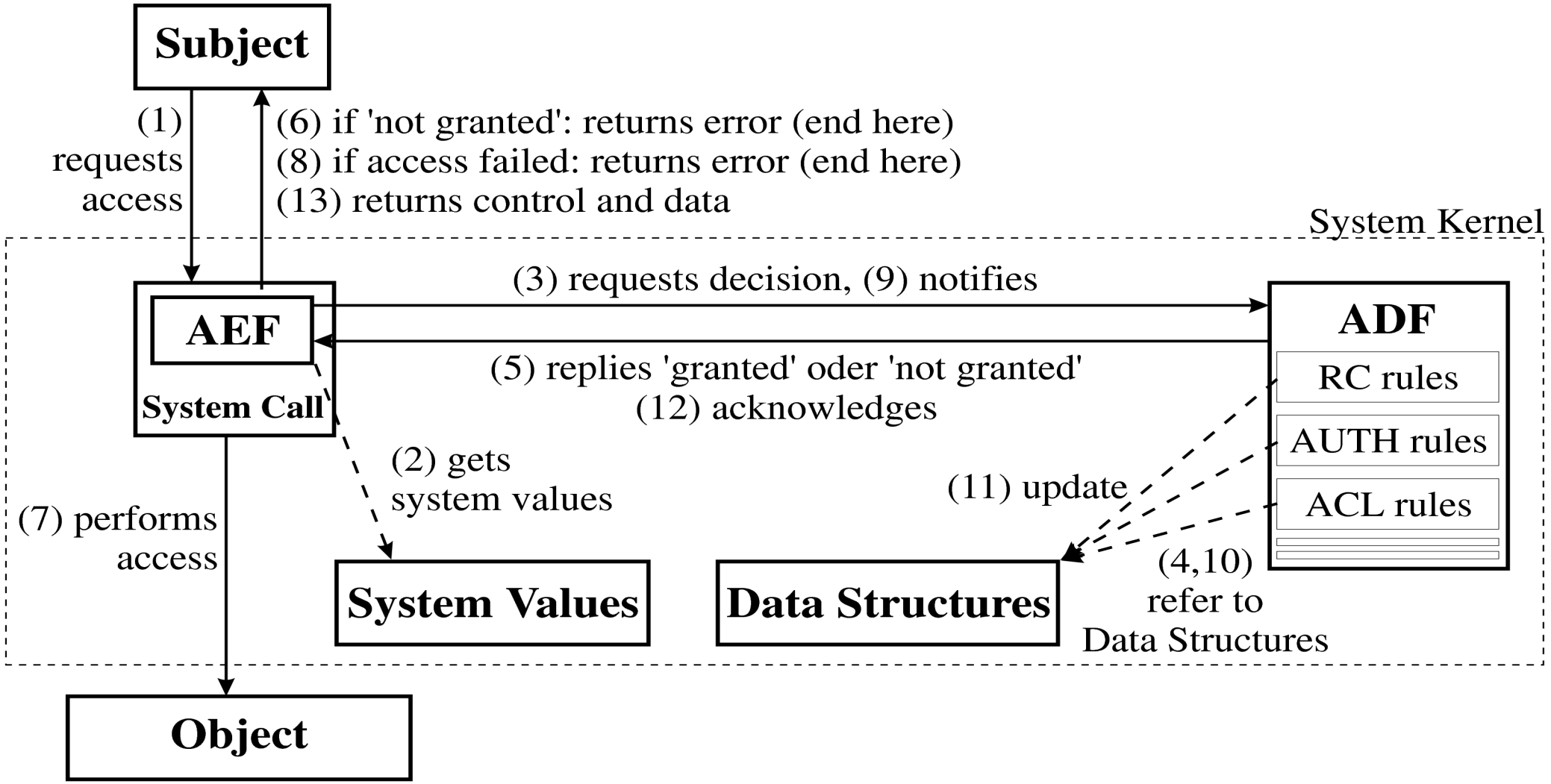
Architektura: Obiekty i podmioty

- ...obiekty
  - SCD
    - “zasoby” systemu
  - USER
    - użytkownicy (tak, również jako obiekty!)
  - PROCESS
    - procesy (jak wyżej)
  - NETDEV, NETTEMP, NETOBJ
    - urządzenia sieciowe, szablony, pakiety

Architektura: Żądania i decyzje

- Zadania
  - czyli co podmiot chce zrobić z obiektem
  - 49 typów
  - pogrupowane w 7 grup dla łatwiejszej administracji
  - przykłady
    - R_CREATE: DIR (gdzie?), IPC, NETTEMP, NETOBJ
    - R_MOUNT: FILE, DIR, DEV
    - R_READ_OPEN: FILE, FIFO, DEV, IPC
    - R_ADD_TO_KERNEL: FILE, DEV, SCD, NONE
    - ...i wiele innych

Architektura: Diagram



Architektura: Model REG

- Umożliwia pisanie własnych modułów kontroli dostępu
- Moduły jadra – (un)plug & play w trakcie pracy systemu
- Rejestracja modeli, syscalli, list do przechowywania danych (polityki)
- Wygodne API
- Przykładowe moduły w zestawie

Architektura: NETTEMP

- Sposób na opis zakończeń sieciowych
  - numer porządkowy
  - nazwa
  - “rodzina” adresu (INET, IPX, UNIX, inne...)
  - adres (np 192.168.1.1 albo “/dev/log”)
  - długość adresu (np 24 bity, 8 bajtów)
  - typ gniazdka (ANY, STREAM, DGRAM, inne...)
  - protokół (ICMP, UDP, TCP, inne...)
  - urządzenie sieciowe (np eth0)
  - zakres portów (np 1024-65535, albo 80)

Architektura: NETTEMP

- atrybuty przyporządkowane do każdego NETTEMP
- wartości domyślne “out of box”
- ADF dopasowuje od najniższego do najwyższego numeru porządkowego!
- może opisywać lokalne zakończenie sieciowe bądź zdalne, zależnie od zadania

Architektura: NETTEMP

- Przykłady co możemy zrealizować
  - apache może działać jedynie na porcie 80
    - kontrolowane jest wywołanie “bind”
  - proxy może łączyć sie tylko ze światem zewnętrznym, nie z LAN
  - proxy może tylko akceptować połączenia z LAN
  - użytkownicy mogą łączyć sie tylko do serwera mail i proxy
  - użytkownicy (oczywiście dotyczy również uid 0!) mogą używać tylko UNIX i INET
  - wiele innych...

Modele bezpieczeństwa

- Authentication Enforcement – AUTH
- User management - UM
- Role Compatibility – RC
- Mandatory Access Control - MAC
- Access Control Lists – ACL
- Privacy Model - PM
- File Flags – FF
- Linux Capabilities – CAP
- Process Jails – JAIL
- Resource Control – RES
- Pageexec Support – PAX
- Dazuko - DAZ

Modele bezpieczeństwa: AUTH

- Kontroluje procesy – CHANGE_OWNER (setuid)
  - opcjonalnie także gid
- Każdy plik ma listę uid(/gid)
  - dziedziczone do procesu
  - auth_may_setuid i auth_may_set_cap
- Opcjonalnie – demon przeprowadzający autentykacje
  - proces – (autoryzacja + autentykacja) -> demon
  - demon ustawia “auth capability” procesu
  - proces wywołuje setuid

Modele bezpieczeństwa: AUTH

- Dodatkowa funkcjonalność
  - TTL w ustawieniach auth capabilities
    - przydatne do zdalnej administracji polityka bezpieczeństwa
  - kontrola wszystkich rodzajów uid/gid
  - tryb uczenia!
  - opcjonalnie setuid(getuid()) -> GRANTED
    - desktop - środowiska kde/gnome

Modele bezpieczeństwa: UM

- Słaby standardowy mechanizm zarządzania informacjami o autoryzacji i autentykacji użytkowników
  - atak przeciwko PAM
  - mała dokładność
  - atak na hasła
  - brak poprawnej kontroli nad setuid()
  - trudności z ochrona bazy danych użytkowników

Modele bezpieczeństwa: UM

- Rozwiązanie – RSBAC UM
  - nikt nie ma dostępu do danych użytkowników
  - autentykacja w obszarze jądra
    - atak przeciwko PAM nie działa
  - prawa dostępu do danych każdego użytkownika osobno
    - umożliwia separacje przywilejów
  - szczegółowa kontrola nad tymi prawami
    - wiele typow praw dostępu dla różnych danych
  - setuid() tylko jeśli pomyślnie przeszedł autentykację
  - łatwość ochrony danych

Modele bezpieczeństwa: RC

- Model bazujący na rolach i typach
  - proces <- rola, obiekt <- typ
  - wiele atrybutów – domyślna rola, aktualna rola, wymuszona, początkowa, typ obiektu
  - Kompatybilność rol – możliwość przełączania miedzy rolami – kontrolowana!
- Sprawdzenie praw dostępu
  - proces(aktualna rola) - zadanie -> obiekt (typ)
  - jeśli kompatybilne -> GRANTED

Modele bezpieczeństwa: RC

- Wspiera separacje przywilejów
  - admin role
  - assign roles
  - admin, assign, access control, supervisior, read_attribute, modify_attribute
- TTL dla wszystkich uprawnień!
- Zaawansowany mechanizm dziedziczenia

Modele bezpieczeństwa: ACL

- Jaki podmiot może uzyskać dostęp do jakiego obiektu z jakimi prawami?
- podmioty
  - użytkownicy, RC role (!), grupy ACL
- Grupy ACL
  - każdy użytkownik może mieć własną hierarchie
  - grupy prywatne i publiczne
- Dziedziczenie z maskami (podobnie do Netware 3.xx)
- Domyślne wpisy ACL

Modele bezpieczeństwa: ACL

- Specjalne prawa do administracji
  - Access Control
  - Forward
  - Supervisor
- TTL dla każdego wpisu!
  - prawa dostępu
  - członek grupy
- Tryb uczenia!

Modele bezpieczeństwa: MAC

- MAC – model Bell & LaPadula
- Poziomy zaufania
- No write down, no read up
- Opcjonalnie zaawansowane dziedziczenie
- Wiele dodatków “ułatwiających życie”
- Dodatkowa możliwość - “MAC Light”

Modele bezpieczeństwa: PM

- Simone Fischer-Hübner's Privacy Model
- Składowanie i przetwarzanie danych osobowych
  - zgodny z normami i wytycznymi EU
- najważniejsza jest prywatność
- zapewnia poufność, bezpieczeństwo i dostępność
- dostęp w ramach transakcji
- nacisk na cel przetwarzania i obszar danych który obejmie

Modele bezpieczeństwa: FF

- File Flags – atrybuty obowiązkowe dla każdego (z pewnymi wyjątkami)
- Dziedziczenie
  - domyślnie aktywne
- Przykładowe atrybuty – no_execute, secure_delete, append_only i wiele innych...

Modele bezpieczeństwa: CAP

- Linux Capabilities
  - min i max zbiór “capabilities” dla użytkownika lub procesu
  - aktywowane w momencie CHANGE_OWNER i EXECUTE
  - Pierwszeństwo min nad max
  - Pierwszeństwo polityki programu nad użytkownika
  - Przykład – odebrać mail serwerowi wszystkie capabilities poza potrzebnymi

Modele bezpieczeństwa: JAIL

- Zamkniecie procesu w odizolowanym środowisku
  - opcjonalnie chroot()
- Odebranie możliwości komunikacji z procesami z zewnątrz
- Wiele innych restrykcji, większość opcjonalna
- Ogranicza możliwość “administracji” i dostęp do sieci

Modele bezpieczeństwa: RES

- min i max limity zasobów systemu (rlimits)
- Aktywowane przy CHANGE_OWNER (setuid) i EXECUTE
- Pierwszeństwo min nad max ustawieniami
- Pierwszeństwo polityki programami nad użytkownika
- Wszystkie atrybuty istniejące w systemie
  - np max rozmiar pliku, ilość procesów, rozmiar zaalokowanej pamięci

Modele bezpieczeństwa: PAX

- Administracja flagami PaX
- Przechowuje ustawienia w ACI
- Łatwość backupu
- Niepotrzebna modyfikacja maglownika ELF

Instalacja RSBAC - Kernel

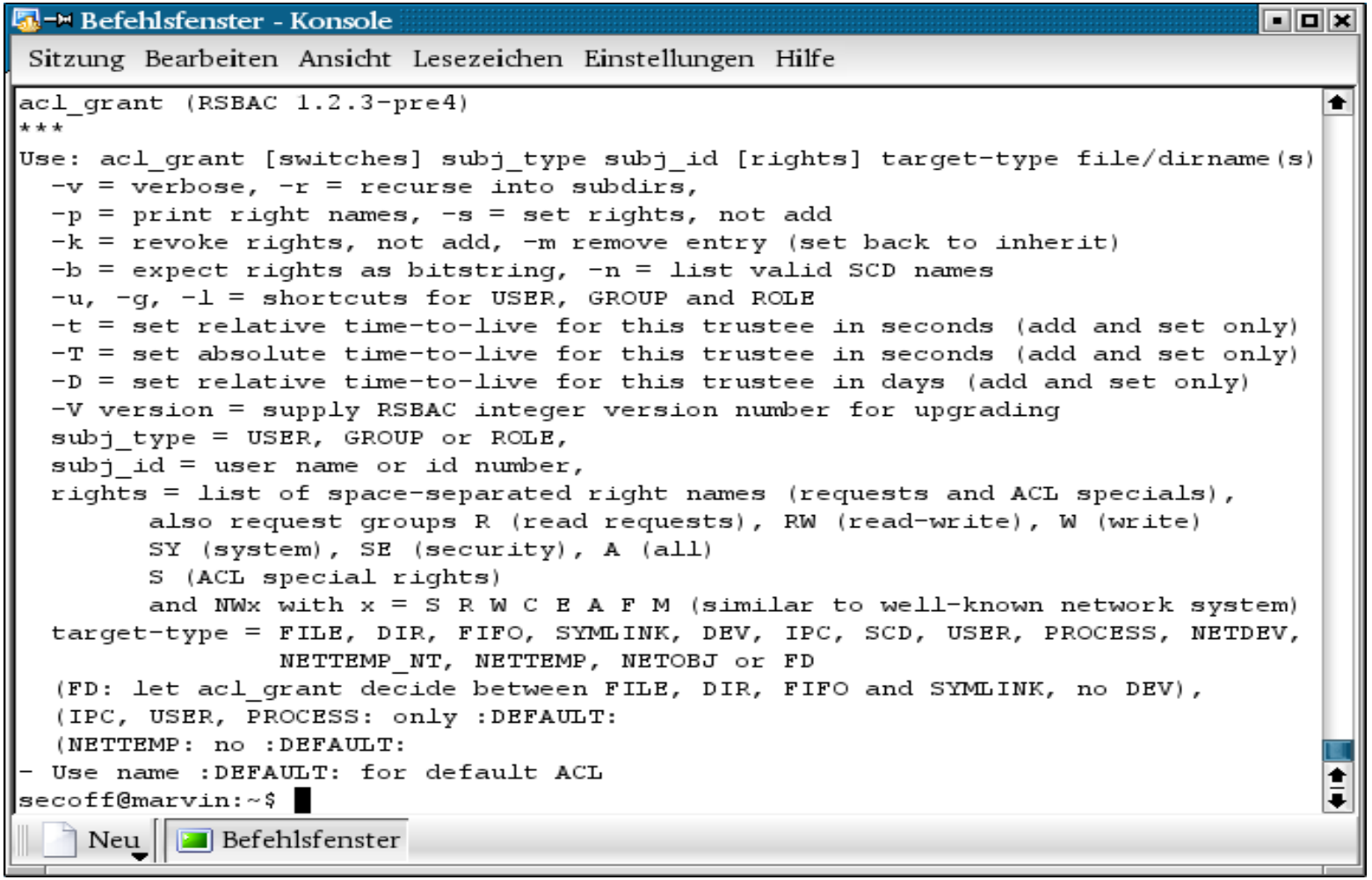
- Dwie możliwości
  - classic -> gotowe źródła jądra Linux + rsbac
  - enhanced -> Linux + rsbac + pax
    - wystarczy rozpakować, skonfigurować, skompilować i gotowe
  - samodzielne patchowanie kernela vanilla
    - rozpakować vanilla
    - rozpakować archiwum rsbac
    - nałożyć patch
    - konfiguracja, kompilacja i gotowe

Instalacja RSBAC - Narzędzia

- Archiwum rsbac-admin
  - narzędzia
  - rklogd
  - moduły pam i nss
  - przykładowe skrypty (m.in. do backupu polityki)
  - strony man
- Ważne – podczas pierwszego startu najlepiej dodać parametr rsbac_softmode lub rsbac_enable_login!
- Jadro Maintenance – brak ADF i AEF, aktywne tylko ACI

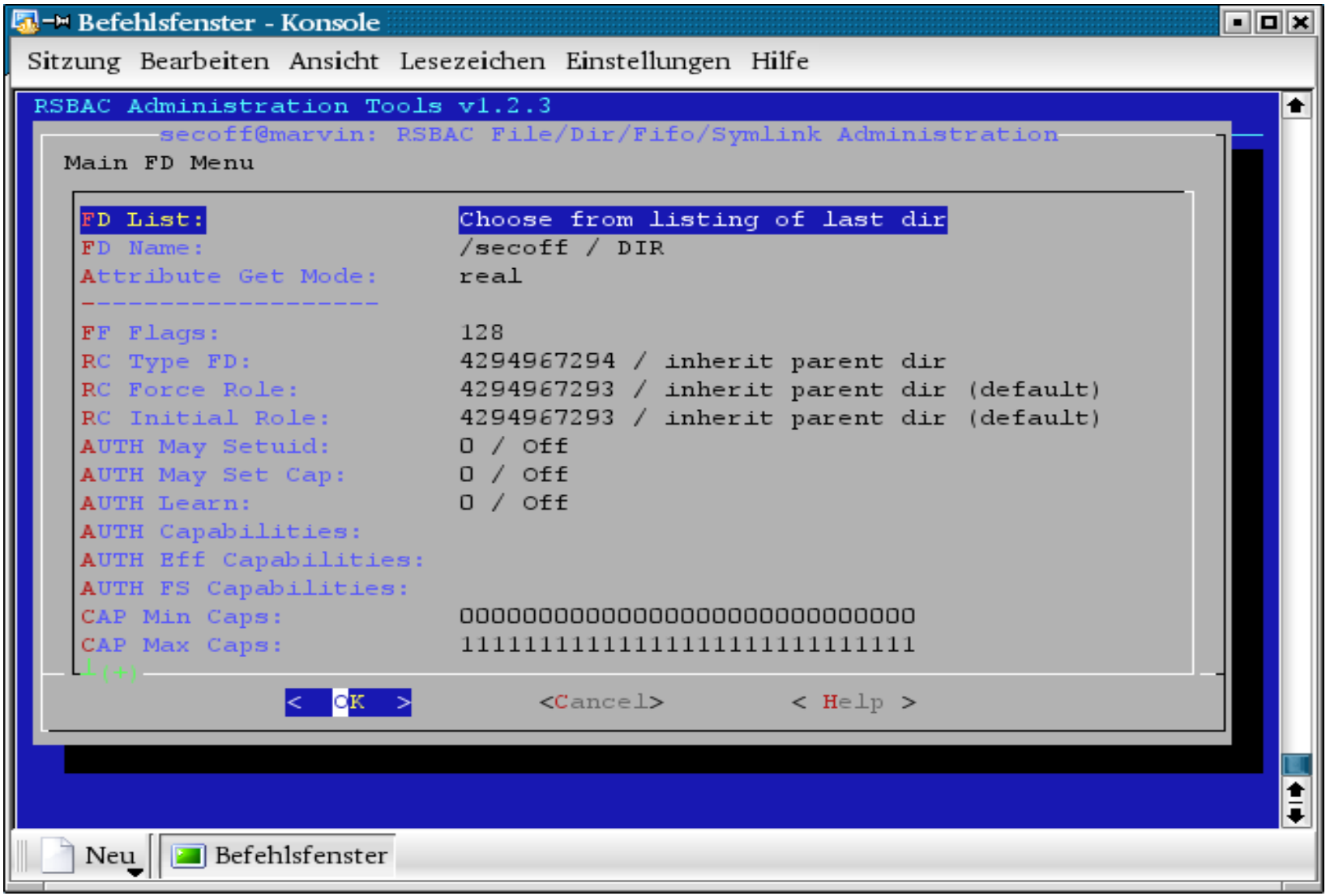
Narzędzia administracyjne

- Ustawienia ogólne i specyficzne dla modelu
- Narzędzia linii poleceń
  -



Narzędzia administracyjne

- Zarządzanie polityka za pomocą menu



Obszar zastosowań

- Stacje robocze, desktop
  - ochrona przed (nie)przypadkowa zmiana konfiguracji
  - ochrona przed trojanami, robakami, wirusami
    - ważne w przypadku wymiany danych z innymi systemami
  - ochrona systemu i danych użytkownika(!) przed skutkami błędow w źle napisanych programach
    - np przejecie kontroli nad przeglądarka www

Obszar zastosowań

- Serwery
  - odizolowanie aplikacji “serwerów”
  - ochrona przed wirusami, robakami, trojanami
  - ogólna ochrona całego systemu
    - m.in. danych, konfiguracji i innych...
  - ograniczenie zbędnych praw programów (CAP)
    - po co pingowi komplet capabilities jeśli potrzeba tylko jedno?
  - wspomaga ochronę przeciwko DoS (RES)
- Firewalle - hardening systemu operacyjnego

Obszar zastosowań

- (wirtualne) Web Serwery
  - ochrona systemu przed błędami w aplikacjach
  - ochrona każdej domeny z osobna, danych klienta
  - zabezpieczenie przed skutkami błędow w aplikacjach pisanych w cgi, php czy innych językach
- Serwery poczty
  - ochrona systemu operacyjnego
  - osobna ochrona każdego obsługiwanego klienta

Obszar zastosowań

- Serwery plików
  - ...oczywiście ochrona systemu i danych (zawsze!)
  - separacja przechowywanych danych, szczegółowa kontrola dostępu do nich wraz z możliwością dokładnego audytu
- Serwery aplikacji
  - oprócz ochrony systemu, danych
  - dokładnie rozdzielenie użytkowników, możliwość tworzenia wirtualnych serwerów
  - zabezpieczenie przed atakami od wewnątrz!
- Wszystkie inne typy serwerów

RSBAC w praktyce: Kto?

- Dystrybucje wspierające RSBAC – Adamantix i Gentoo Hardened
- RSBAC używany produkcyjnie w m-privacy TightGate-Pro
  - serwer aplikacji – bezpieczny dostęp do internetu
  - używa większości wymienionych modeli
  - szczegółowa polityka dla wszystkich aplikacji działających na serwerze
- www.rsbac.org -> ochrona całego serwera

RSBAC w praktyce: Kto?

- www.cyberguard.com
  - używa RSBAC do zabezpieczenia systemu, danych oraz aplikacji działających na ich platformie
    - firewall/vpn, content filtering (layer 7)
  - produkt certyfikowany EAL 4 +
  - www.cyberguard.com/download/white_paper/en_cg_secure_to_the_core.pdf
- Wiele innych systemów produkcyjnych na całym świecie

RSBAC w praktyce: Stabilność

- Wysoka stabilność system pracujących pod kontrola RSBAC
  - uptime prawie rok na serwerach m-privacy oraz www.rsbac.org
    - restart wyłącznie z powodu upgrade
    - Obsługuje zarówno maszyny UP jak i SMP
    - Testowany od embedded to potężnych instalacji NUMA
    - Działa stabilnie na mainframe IBM zSeries
    - Działa stabilnie na maszynach IBM pSeries

RSBAC w praktyce: Architektury

- W kolejności od najbardziej do najmniej przetestowanych
  - x86 32 i 64 bit -> cały zespól RSBAC oraz maszyny produkcyjne
  - Power PC G4 (32 bit) -> moja główna platforma deweloperska
    - Oprócz tego RSBAC jest testowany i działa stabilnie na:
      - Power PC G5 (64 bit)
      - Sparc 32 bit i 64 bit
      - MIPS
      - ARM, SH, zSeries
    - Pilnie potrzeba maszyny na Itanium 2 SMP – anybody ?

RSBAC w praktyce: Wydajność

- Bardzo dobra wydajność, niskie dodatkowe obciążenie systemu kontrola dostępu
  - przemyślana architektura – tablice hashowane, ACI O(1), modularny kod, profiling
  - Na wydajność wpływa ilość i typ włączonych modułów, aktywne opcje, ustawienia logowania (poza nasza kontrola!)
  - Testy:
    - RSBAC bez modułów - + 0.68%
    - standardowa konfiguracja, obsługa sieci - + 2.30%
    - wszystkie moduły włączone - + 4.21%

Plany na przyszłość

- 1.3 juz blisko. a w nim:
  - virtual UM – dowolna liczba oddzielnych baz danych UM! atrybut dla każdego użytkownika, procesu, pliku z którego zbioru ma korzystać
  - ukrywanie obiektów
  - zabezpieczenie przed zmiennymi LD_*
  - automatyczna replikacja list do innych systemów!
  - dynamiczna zmiana rozmiaru list ACI w zależności od liczby obiektów
  - tryb uczenia dla CAP
  - jeszcze potężniejszy system logowania, podobny w założeniach do RC
  - mnóstwo marych zmian, poprawek
  - jeszcze większa wydajność
  - port RSBAC na XEN oraz kontrola nad administracja

Plany na dalsza przyszłość

- 1.4 – model kontroli nad XEN “per domain”
- obsługa ipv6
- odświeżenie PM
- oczywiście wiele innych ciekawych zmian
- a kiedyś.
  - kontrola dostępu dla X
  - SELinux jako moduł REG (nie żartuje:)
  - GRSecurity RBAC jako moduł REG (naprawdę)
  - automatyczne śledzenia atrybutów
  - pełny tryb uczenia dla wszystkich modułów
  - ...i oczywiście władza nad światem !
  - the show must go on...

RSBAC w sieci

- Strona projektu -> www.rsbac.org
- Lista mailingowa, dostępne archiwum
- IRC -> #rsbac@irc.freenode.net
-
- Adamantix -> www.adamantix.org
- Gentoo -> hardened.gentoo.org/rsbac

Dziękuje!

- Rule Set Based Access Control -> RSBAC
  - “Zaufanie jest dobre, a kontrola jeszcze lepsza”
  -
  -
  - Michal Purzynski (michal@rsbac.org)