Ochrona i bezpieczeństwo

Ostatnia modyfikacja: 01.06.2021

Spis treści

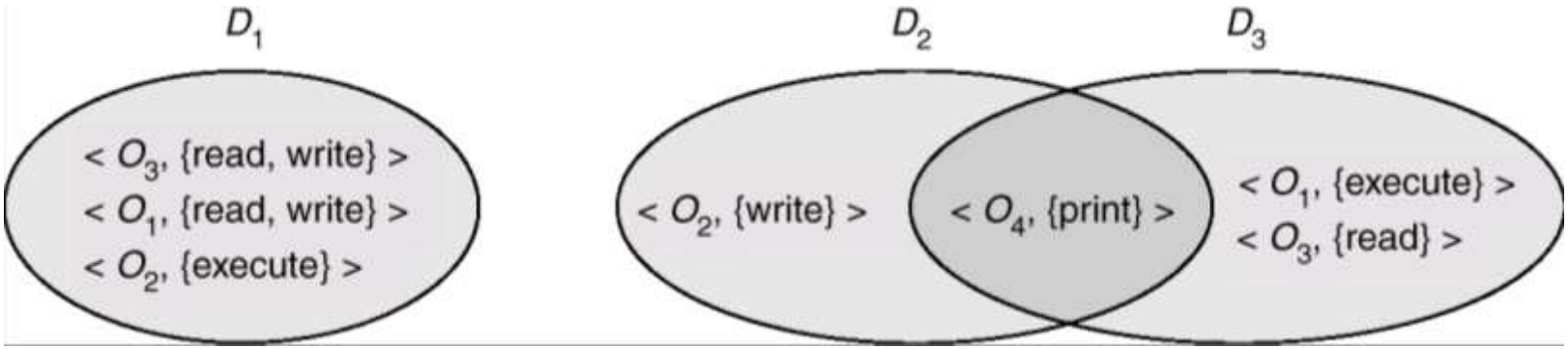
Ochrona

- Termin ochrona odnosi się do mechanizmu służącego do kontrolowania dostępu programów, procesów lub użytkowników do zasobów zdefiniowanych przez system komputerowy.
- Rolą ochrony jest dostarczania mechanizmu do wymuszania polityki rządzącej sposobem używania zasobów. **Mechanizmy** określają jak coś zrobić, a **polityka** - co ma być zrobione. Oddzielenie polityki od mechanizmu jest ważne ze względu na elastyczność systemu.
- System komputerowy jest zbiorem procesów i obiektów (sprzętowych i programowych). Każdy obiekt ma jednoznaczną (unikalną) nazwę; może być wykorzystywany za pomocą dobrze określonego zestawu operacji.
- **Problem ochrony** - zapewnić, by każdy obiekt w systemie komputerowym był używany w sposób właściwy i tylko przez uprawnione do tego procesy.
- **Zasada wiedzy koniecznej**: proces powinien mieć aktualnie dostęp tylko do tych zasobów, których aktualnie potrzebuje do zakończenia zadania.

  Uwaga: większość systemów nie dysponuje wystarczająco drobnoziarnistymi mechanizmami ochrony (przywilejami czy prawami dostępu), by realizować w/w zasadę dokładnie.

Domeny ochrony

- Proces działa w **domenie ochrony**, która określa dostępne dla niego zasoby. Możliwość wykonywania operacji na obiekcie jest **prawem dostępu**.
- **Prawo dostępu** = <nazwa_obiektu, zbiór_dostępnych_operacji>
  zbiór_dostępnych_operacji jest podzbiorem wszystkich możliwych operacji dla obiektu o danej nazwie.
- **Domena_ochrony** = zbiór praw dostępu



Realizacja domen ochrony

Domenę można realizować różnymi sposobami:

- Każdy **użytkownik** może być traktowany jako domena. Przełączanie domen dokonuje się wraz z rejestracją (logowaniem) użytkownika

- Każdy **proces** może być domeną. Zbiór dostępnych obiektów zależy od tożsamości procesu. Przełączaniu domen odpowiada wysyłanie przez proces komunikatu do innego procesu i oczekiwanie na odpowiedź

- Domeną może być **procedura**. Zbiór dostępnych obiektów, to zmienne lokalne procedury. Przełączanie domen – wskutek wywołania procedury.

Realizacja domen ochrony – c.d.

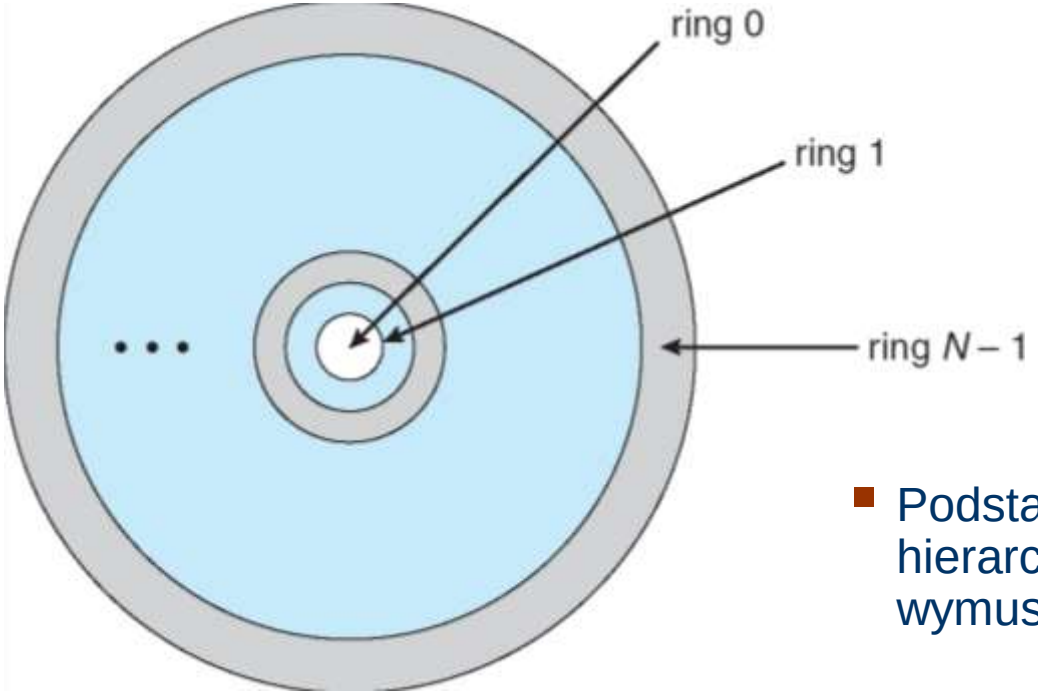
Przykłady

- System operacyjny działający w dwóch trybach (monitora i użytkownika) dysponuje dwoma domenami ochrony:
  - użytkownika
  - W takim systemie system operacyjny jest chroniony przed procesami użytkownika/użytkowników. Brakuje ochrony użytkowników przed sobą.
  - nadzorcy
- UNIX
  - domena jest związana z użytkownikiem (*user-id, UID*)
  - Przełączaniu domen odpowiada czasowa zmiana identyfikacji użytkownika. Możliwe jest przełączanie za pomocą wywołania funkcji systemowej (*setuid, seteuid*), albo wykonanie programu innego użytkownika, zawartego w pliku, którego i-węzeł ma zaznaczony bit ustanowienia identyfikatora użytkownika **(setuid)**.

Pisanie programów uprzywilejowanych wymaga wielkiej staranności. Zasada wiedzy koniecznej może być przybliżana przez przełączanie bieżącej grupy do której należy proces.

Realizacja domen ochrony – c.d.

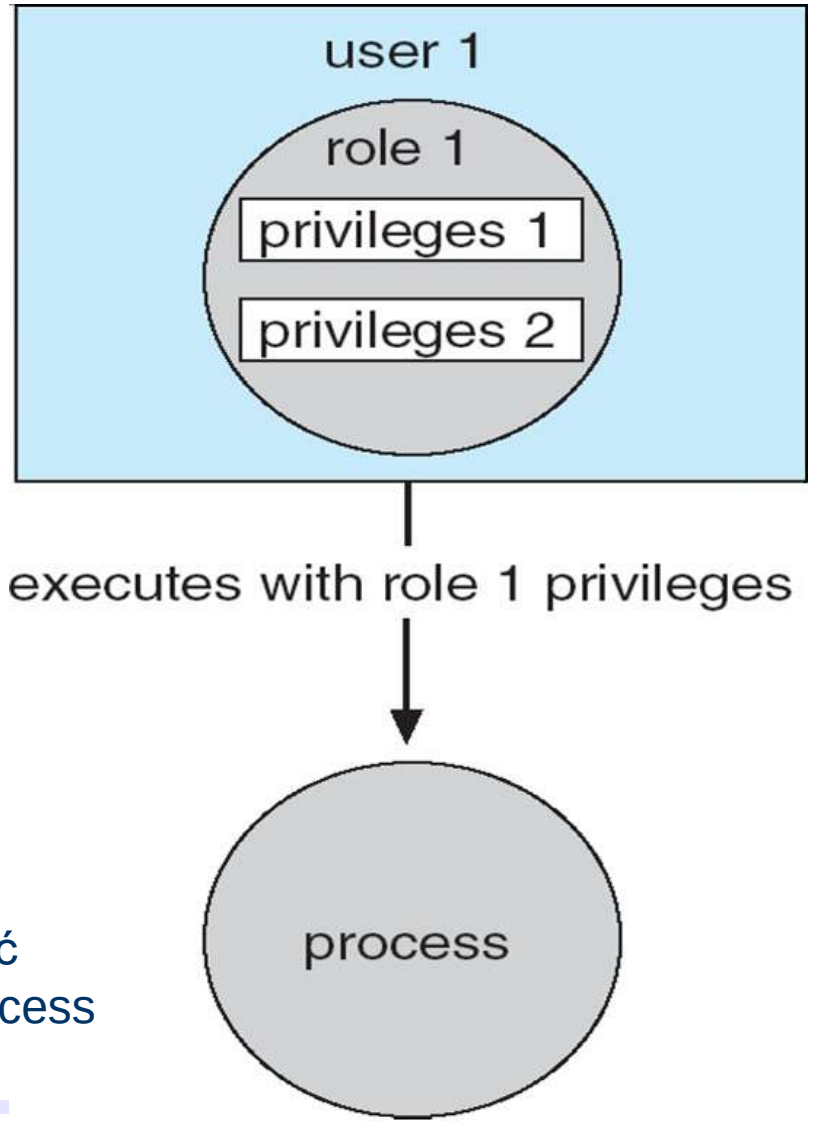
- W systemie MULTICS domeny ochrony są zorganizowane hierarchicznie w strukturę pierścieniową.
  - Niech D_i and D_j będą dwoma pierścieniami domen.
  - Jeśli $j < i \Rightarrow D_i \subseteq D_j$, tzn. więcej przywilejów ma proces w domenie o mniejszym indeksie.
  - Przełączanie domen występuje wskutek wywołania procedury z innego pierścienia.



- Podstawowa wada struktury hierarchicznej – nie pozwala na wymuszanie zasady wiedzy koniecznej

Realizacja domen ochrony – c.d.

- W systemie Solaris 10 wprowadzono kontrolę dostępu opartą na **rolach** (*role-based access control*, **RBAC**). Może ona dotyczyć nie tylko obiektów ale czynności (np. prawo do wykonania funkcji systemowej, czy użycia opcji w tej funkcji).
- Rola może być przypisana procesowi.
- Użytkownicy mogą mieć przypisane role po uwierzytelnieniu. Umożliwia im to wywoływanie programów z dodatkowymi uprawnieniami bez ryzyka związanego z pracą z prawami użytkownika *root* czy wykorzystaniem bitów *setuid/setgid*.
- RBAC może realizować funkcjonalność MAC/DAC (Mandatory/Disretionary Access Control – patrz s. 21.



Ogólny model ochrony - macierz dostępów

domain \ object	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Wiersze-reprezentują domeny, kolumny-obiekty.

Wpis w elemencie macierzy (i,j) oznacza dozwolone operacje na *j*-tym obiekcie dla procesu w *i*-tej domenie

Wykorzystanie macierzy dostępów

- Macierz dostępów dostarcza **mechanizmu**, który pozwala podejmować użytkownikowi **decyzje polityczne**, dotyczące tego kto co może w systemie. System gwarantuje, że proces działający w domenie D_i może mieć tylko dostęp tylko do obiektów wymienionych w wierszu *i* -tym, wyłącznie w sposób dozwolony przez wpisy w macierzy dostępów.
- Mechanizm macierzy dostępów wystarcza do definiowania i implementowania ścisłej kontroli zarówno statycznych, jak i dynamicznych związków pomiędzy procesami a domenami. Związki dynamiczne wymagają jednak dodatkowych operacji dodawania i usuwania praw dostępu. To z kolei wymaga ustanowienia reguł manipulowania domenami i polami tablicy dostępów - jako specjalnymi obiektami podlegającymi ochronie. Specjalne prawa dostępu:
  - **właściciela obiektu** O_i (proces może tworzyć nowe prawa i usuwać istniejące)
  - **kopiowania prawa dostępu w ramach jednej kolumny** (dalej ozn. *). Proces uzyskujący nowe prawo dostępu może mieć lub nie mieć prawa dalszego kopiowania uzyskanego prawa (decyduje *polityka*).
  - **kontroli** – proces z domeny D_i może zmodyfikować prawa dostępu w domenie D_k
  - **przekazania** – umożliwia procesowi zmianę domeny z D_i na D_k
- Prawa *kopiowania* i *właściciela* są przydatne do ograniczania propagacji praw dostępu. Nie wystarczają jednak, by zapobiec rozchodzeniu się informacji przechowywanej początkowo w obiekcie na zewnątrz jego środowiska wykonania (problem zamknięcia, ***confinment problem*** – w ogólnej postaci nie jest rozwiązywalny).

Przykład: macierz dostępów z domenami-obiektami

domain \ object	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			

switch - przełączanie przynależności do domeny (własność przekazania)

Przykład: kopiowanie praw dostępu

domain \ object	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute		

(a)

domain \ object	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute	read	

(b)

read*, write* - możliwość kopiowania prawa dostępu (odczyt, zapis) z jednej domeny do drugiej (w ramach tej samej kolumny macierzy)

Przykład: zastosowanie read* z kolumny F2 do wiersza D3 macierzy z rys. (a) zmieniło macierz dostępu jak pokazano na rys. (b). W tym przykładzie kopiowanie praw było ograniczone, gdyż nie dało prawa dalszego kopiowania procesom w domenie D3.

Można również definiować

- kopiowanie praw z uprawnieniem dalszego kopiowania oraz
- przekazywanie praw (przekazujący traci prawa)

Przykład: prawa właściciela

domain \ object	F_1	F_2	F_3
D_1	owner execute		write
D_2		read* owner	read* owner write*
D_3	execute		

(a)

domain \ object	F_1	F_2	F_3
D_1	owner execute		
D_2		owner read* write*	read* owner write*
D_3		write	write

(b)

owner - prawa właściciela obiektu, umożliwiające dodawanie i usuwanie dowolnych praw w danej kolumnie macierzy dostępów

Przykład: operacja *control*

object / domain	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			

control - prawo procesu działającego w danej domenie do usuwania praw dostępu w innej domenie.

← Przykład. Dodanie **control** do pola D2-D4 (z **switch**) może prowadzić do poniższej macierzy dostępów

object / domain	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch control
D_3		read	execute					
D_4	write		write		switch			

Implementacja macierzy dostępów

- Macierz dostępów jest rzadka.
- Najprostsza realizacja: globalna tablica uporządkowanych trójek *<domena, obiekt, zbiór_praw>.* Tablica ta może być bardzo duża.
- Każda kolumna macierzy dostępów może przybrać postać **wykazu dostępów** do danego obiektu (*Access-Control List*, ACL). Podejście można rozszerzyć, dodając *domyślny* zbiór praw dostępu.
- **Wykaz uprawnień** (*Capability List*) jest spisem obiektów i operacji dozwolonych na tych obiektach w danej domenie. Wykaz uprawnień jest chronionym obiektem związanym z domeną, dostępnym jedynie pośrednio dla procesów działających w tej domenie. Wykaz jest utrzymywany przez system operacyjny.
- **Mechanizm zamka-klucza** jest rozwiązaniem pośrednim pomiędzy wykazem dostępów, a wykazem uprawnień. Każdy obiekt ma wykaz jednoznacznych wzorców binarnych zwanych kluczami. Proces działający w domenie może mieć dostęp do obiektu tylko wtedy, gdy dana domena zawiera klucz pasujący do jednego z zamków danego obiektu. Wykazy kluczy domeny muszą być zarządzane przez system operacyjny na zamówienie danej domeny. Użytkownicy nie mają prawa bezpośrednio sprawdzać ani zmieniać zamków ani kluczy.

Cofanie praw dostępu

W systemie ochrony dynamicznej jest niezbędna operacja cofania praw dostępu do obiektów dzielonych przez różnych użytkowników.

Możliwość ta rodzi pytania: natychmiast czy z opóźnieniem, wybiórczo czy dla wszystkich użytkowników, częściowo czy całkowicie (wszystkie prawa), czasowy czy na stałe?

- Cofanie praw w wykazie dostępów jest proste i szybkie – wymaga usunięcia pozycji na liście kontroli dostępu.
- Cofanie praw dla list uprawnień jest trudniejsze. Jest kilka schematów:
  - **Wtórne pozyskiwanie** (system okresowo usuwa uprawnienia z każdej domeny; proces musi próbować uzyskiwać je na nowo, co wymaga sprawdzenia)
  - **Wskaźniki zwrotne** (od obiektów do wszystkich jego uprawnień)
  - **Adresowanie pośrednie** (uprawnienia wskazują na obiekty za pośrednictwem tablicy globalnej)
  - **Klucze** (wzorce binarne wiązane ze wszelkimi uprawnieniami; zmiana klucza głównego obiektu zmienia możliwość użycia kluczy posiadanych przez wybrane procesy).

Porównanie implementacji macierzy dostępów

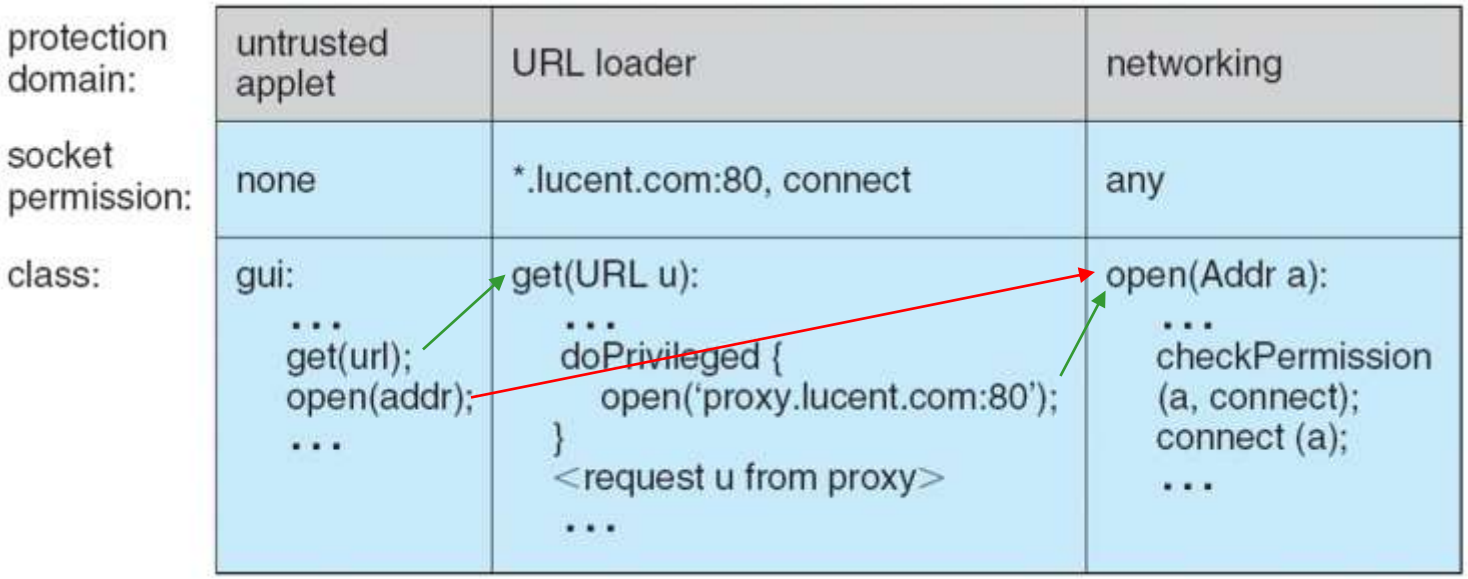
- Wykazy dostępów odpowiadają bezpośrednio zapotrzebowaniu użytkowników: tworząc obiekt określa które domeny mają mieć do niego dostęp i za pomocą jakich operacji. Informacja o prawach dostępu w poszczególnych domenach jest rozproszona, a przeszukiwanie długich wykazów dostępów może być kosztowne.
- Wykazy uprawnień są użyteczne dzięki lokalizowaniu informacji dotyczącej poszczególnych procesów. Proces usiłujący uzyskać dostęp do obiektu musi wykazać, że ma odpowiednie uprawnienia. Cofanie uprawnień bywa niewydajne.
- Mechanizm zamka-klucza może być zarówno efektywny jak i elastyczny - zależnie od długości klucza. Uaktualnianie przywilejów dostępu jest względnie łatwe przy użyciu techniki klucza głównego (*master key*) obiektu.
- W większości systemów stosuje się kombinację wykazów dostępów i uprawnień. Przy pierwszej próbie dostępu do obiektu przegląda się wykaz dostępów, a przy dostępie dozwolonym tworzy się wykaz uprawnień do obiektu, który dołącza się do procesu (np. jako pozycję w tablicy otwartych plików) . Wykaz jest wykorzystywany przy następnych dostępach. Nie zwalnia to jednak systemu ze sprawdzania prawa dostępu przy każdej próbie dostępu, ale pozwala odrzucać próby dostępu niezgodne z uprawnieniem (np. próbę zapisu do pliku otwartego do odczytu). Po ostatnim dostępie (np. po zamknięciu sesji plikowej) uprawnienie procesu jest usuwane.

Ochrona na poziomie języka programowania

- Określenie pożądanej kontroli dostępu za pomocą języka programowania wysokiego poziomu daje elastyczność - pozwalając na zmianę sposobu postępowania z zasobami w zależności od zastosowania i upływu czasu.
- Ochrona programowa umożliwia egzekwowanie ochrony tam, gdzie nie istnieje możliwość kontroli sprzętowej. Może interpretować reguły ochrony - zarówno bez kontroli sprzętowej jak i ze wsparciem sprzętu.
- Wymuszanie przez jądro daje jednak wyższy stopień ochrony samego systemu, aniżeli generowanie przez kompilator kodu sprawdzającego ochronę. W systemie ochrony tworzonym przez kompilator bezpieczeństwo zależy od poprawności translatora, mechanizmów zarządzania pamięcią, bezpieczeństwa plików. Ochrona sprzętowa jest także bardziej odporna na naruszenia wynikającego ze złego działania sprzętu lub oprogramowania systemowego.
- Jądro ochrony ogranicza wdrażanie polityki zdefiniowanej przez użytkownika. Za pomocą języka programowania można uzyskać większą elastyczność.
- Kontrola statycznego dostępu może odbywać się w czasie kompilacji, a nie w czasie działania programu.

Ochrona w języku Java 2

- Ochrona jest realizowana przez maszynę wirtualną Java (Java Virtual Machine, JVM)
- Domena ochrony określa które operacje może wykonywać klasa, a których nie może wykonywać.
- Domena ochrony jest przydzielona klasie w czasie jej ładowania przez JVM.
- Jeśli wywoływana jest metoda biblioteczna, wykonująca uprzywilejowaną operację – wówczas badany jest stos wywołań, aby upewnić się, że operacja może być zrealizowana przez bibliotekę.



Problem bezpieczeństwa

- **Ochrona** jest problemem ściśle wewnętrznym (jak umożliwić kontrolowany dostęp do programów i danych przechowywanych w systemie komputerowym?). System jest idealnie bezpieczny jeśli dostęp do zasobów spełnia wymagania polityki bezpieczeństwa w każdych warunkach pracy systemu.
- Nie można oczekiwać bezwzględnego **bezpieczeństwa**, ale można uczynić koszt naruszenia bezpieczeństwa tak wysokim, że zniechęci do prob. Pożądana jest analiza ryzyka, biorąca pod uwagę **własności środowiska**, w którym ma działać system.
- Zapewnianie bezpieczeństwa dotyczy 4 poziomów:
  - **Bezpieczeństwo fizyczne**
  - **Czynnik ludzki**
  - **Sieć komputerowa**
  - **System operacyjny i oprogramowanie użytkownika**

  Stopień bezpieczeństwa systemu określa najsłabszy element systemu.
- **Naruszenia bezpieczeństwa** mogą być złośliwe, bądź przypadkowe. Łatwiej jest chronić się przed nadużyciami przypadkowymi.
- Do złośliwych nadużyć należą:
  - nieuprawniony dostęp do danych (kradzież informacji)
  - nieupoważniona modyfikacja lub niszczenie danych, wprowadzanie niespójności w danych.

Koncepcje bezpieczeństwa

- **Identyfikacja (*identification*)** - reprezentacja podmiotu (użytkownika) za pomocą nazwy (identyfikatora)
- **Uwierzytelnianie (*authentication*)** – weryfikacja zadeklarowanej wcześniej tożsamości użytkownika.
- **Autoryzacja (*authorization*)** – sprawdzanie, czy podmiot jest uprawniony do uzyskania dostępu
- **Kontrola dostępu (*Access Control*)** – pośredniczy w dostępie użytkownika do zasobów systemu. Mediates. Uznaniowa kontrola dostępu (***Discretionary Access Control*****,** *DAC*) – określa identyfikatory tych użytkowników i grup, którzy mogą mieć dostęp do obiektu dla wybranych operacji. Prawa dostępu określa właściciel. Obowiązkowa kontrola dostępu (***Mandatory Access Control****, MAC*) określa na podstawie atrybutów bezpieczeństwa podmiotów i obiektów, a także na podstawie zestawu reguł tworzących politykę bezpieczeństwa, czy podmiot może uzyskać dostęp do obiektu. Użytkownik nie ma wpływu na atrybuty bezpieczeństwa jak i na reguły (jest to domena administratora bezpieczeństwa).
- **Audyt (*audit*)** logowanie zdarzeń związanych z bezpieczeństwem
- **Rozliczalność (*accountability*)** – wymaga identyfikacji, uwierzytelniania i audytu.
- **Powtórne użycie obiektu (*object reuse*)** wymaganie, by wszystkie nowe obiekty przydzielone użytkownikowi nie zawierały żadnych informacji o poprzednim użyciu.
- **Pewność (*assurance*)** – gwarancja (wewnętrzna producenta bądź zewnętrzna)
- **Baza bezpieczeństwa komputerowego (*Trusted Computing Base*****,** *TCB***)** – minimalny zespół wszystkich systemów ochronnych (sprzęt, oprogramowanie), które realizują w pełni założone zasady bezpieczeństwa.

Klasyfikacja bezpieczeństwa systemów

Wg **Trusted Computer Systems Evaluation Criteria** („The orange book", DoD USA) są 4 główne kategorie bezpieczeństwa systemów:

- **D** – minimalne bezpieczeństwo.
- **C** – Podkategoria **C1** (ochrona uznaniowa, *discretionary security protection*),**C1** = **D** + **separacja użytkowników i danych**. Podkategoria **C2** (ochrona z kontrolą dostępu, *controlled access protection*). **C2** =**C1 + kontrola dostępu przez użytkownika** (user-level access control). **+ audyt** (rejestrowanie zdarzeń istotnych dla bezpieczeństwa).. Standardowy poziom bezpieczeństwa dla zastosowań komercyjnych
- **B** – **C + etykietowanie obiektów**. Podklasa **B1** (ochrona z etykietowaniem, *labeled security protection*) wymusza pierwszeństwo etykiety (*confidential, secret, top secret*) nad prawami dostępu nadanymi przez właściciela obiektu. Podklasa **B2** (ochrona strukturalna, *structured protection*) definiuje wymagania jądra podsystemu ochrony, interfejsu, testowalności, narzędzi administrowania bezpieczeństwem. **Operator != Administrator** . Podklasa **B3** (ochrona przez podział, *security domains*). Części systemu istotne ze względu na bezpieczeństwo przetwarzania powinny być oddalone od części zapewniających użytkownikowi pewne użyteczne dla niego funkcje, ale nie mające związku z bezpieczeństwem przetwarzania.
- **A** – kategoria ta (*verified design*) wymaga formalnych dowodów poprawności projektu i realizacji

W Europie opracowano standardy **ITSEC** (Information Technology Security Evaluation Criteria), które rozróżniają wymagania funkcjonalne od jakościowych (gwarancji).

Common Criteria (ISO 15408) - norma pozwalająca w sposób formalny weryfikować bezpieczeństwo systemów teleinformatycznych.

Ochrona i bezpieczeństwo w SO Linux

- Rodzaje dostępu
  - Czytanie (*Read*)/pisanie (*Write*)/wykonywanie pliku (*Execute*) bądź przeszukiwania katalogu
- Najczęściej uzależnia się dostęp do plików od identyfikacji użytkownika, grupy bądź jego roli (*role-based access controll*)
- Gniazda w dziedzinie UNIX umożliwiają przekazywanie dostępu do otwartego pliku użytkownika serwerowi bez nadawania procesowi serwera takich praw dostępu do innych plików tego użytkownika.
- Linux realizuje rozszerzenie UNIXowego mechanizmu **setuid**, implementując POSIX-owy zachowany identyfikator użytkownika (***saved user-id*****)**, który umożliwia przełączanie efektywnego UID pomiędzy dwoma wartościami
- ***Linux control groups*** *(cgroups)* – jądro ogranicza dostęp do wybranych zasobów dla grup procesów. Procesy różnych cgrup mogą widzieć różne składniki fizycznego systemu.
- **Izolacja przestrzeni nazw** (*namespace isolation*) – własność jądra, polegająca na tym że procesy jednej cgrupy nie widzą takich zasobów innej cgrupy procesów jak: nazwa hosta, PID, UID, sterowniki sieciowe, obiekty IPC, zamontowane systemy plików itd..

Wykazy dostępów (ACL) w SO Linux

- Najogólniejsza metoda implementacji dostępu zależnego od tożsamości polega na skojarzeniu z każdym plikiem i katalogiem **wykazu dostępów** (*access control list, ACL*), zawierającego identyfikatory użytkowników i dozwolone rodzaje dostępu. Dla zmniejszenia wykazu wprowadza się klasy (grupy) użytkowników.
- **setfacl** -polecenie systemowe umożliwiające modyfikację wykazu dostępów do pliku/katalogu. Przykład:

```
$ setfacl –m user:student:r shared1.txt
```

- **getfacl** -polecenie systemowe umożliwiające wyświetlenie aktualnych wykazów dostępów dla podanego pliku/katalogu. Przykład:

```
$ getfacl shared1.txt
# file: shared1.txt
# owner: lopalski
# group: lopalski
user::rw-
user:student:r--
group::r--
mask::r--
other::---
```

Kategoria bezpieczeństwa C2 w SO Linux

- Typowo systemy Unix/Linux realizują domyślnie mechanizmy klasy C1

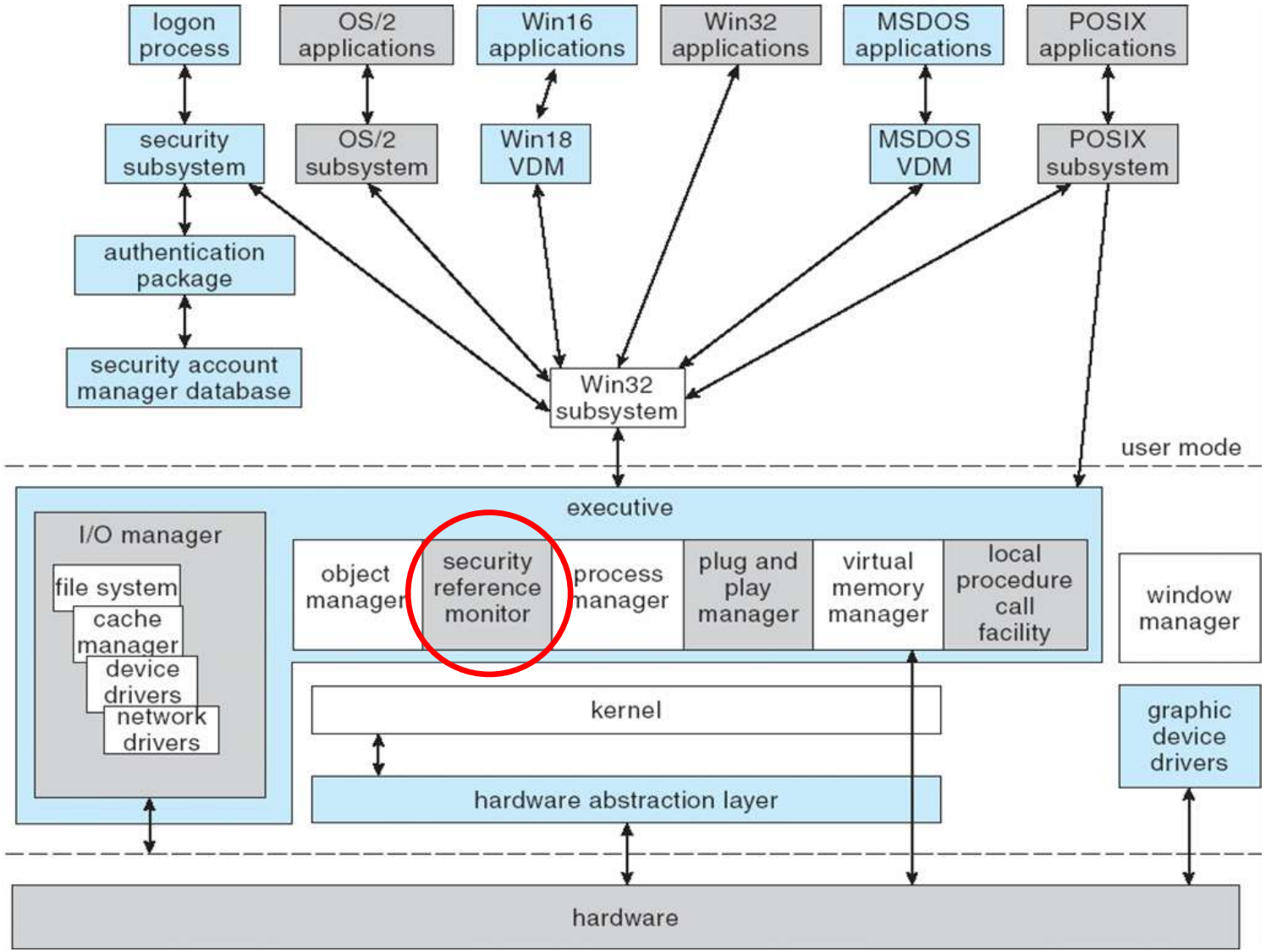
- Bezpieczeństwo C2 wymaga rozszerzeń systemu. Przykładowo:
  - HP-UX (i inne komercyjne systemy UNIX) mogą działać w modelu standardowym (tradycyjne mechanizmy DACL) oraz w Trusted Mode (spełniającym wymagania C2).

- **Security-Enhanced Linux** (SELinux) – zestaw modyfikacji jądra Linuksa oraz sposobu przydzielania zasobów dla aplikacji. Implementuje m.in.:
  - Obowiązkową kontrolę dostępu (Mandatory Access Control, MAC)
  - Kontrolę dostępu w oparciu o role (Role-based access control, RBAC)
  - Wielopoziomowy model bezpieczeństwa

Ochrona i bezpieczeństwo w SO MSWin

- Każdy użytkownik i grupa mają unikalny **identyfikator zabezpieczeń** (SID)
- Podczas uwierzytelniania użytkownika procesowi użytkownika przypisuje się obiekt **żetonu dostępu** (*security access token*). Żeton zawiera SID użytkownika i grup do których należy, a także uprawnienia dostępne dla użytkownika. Żeton jest inicjalizowany z wyłączonymi uprawnieniami, dynamiczne włączanie uprawnień przez proces w potrzebie realizuje postulat minimalnych praw dostępu.
- Podmioty proste dziedziczą uprawnienia zalogowanego użytkownika. Podmioty serwerowe mogą dziedziczyć kontekst zabezpieczeń użytkownika –klienta usługi.
- Działania na obiektach odbywają się za pomocą standardowego zbioru metod: `create, open, close, delete, query name, parse, Security`.
- Z każdym obiektem, dla którego jest dostęp międzyprocesowy, jest skojarzony **deskryptor zabezpieczeń** (*security descriptor*). Zawiera SID właściciela i grupy, listy kontroli dostępu: DACL (ACL ustanawiany przez właściciela i SACL (uprawnienia audytu ustanowione przez administratora).
- **Zarządca obiektów** (*object manager*) nadzoruje użytkowanie wszystkich obiektów systemu. **Monitor bezpieczeństwa odniesień** (*security reference monitor*) sprawdza żeton bezpieczeństwa procesu i ACL obiektu przed zgodą na dostęp, odpowiada za modyfikacje przywilejów w żetonach oraz prowadzi dziennik zdarzeń bezpieczeństwa.
- Microsoft począwszy od Windows Vista i Windows Server 2008 stosuje Mandatory Integrity Control (MIC), aby móc ograniczyć prawa dostępu dla mniej zaufanych kontekstów. Możliwa jest więc np. izolacja procesów (typu sandboxing).

Architektura MSWin 7



Bezpieczeństwo tomu NTFS

- Bezpieczeństwo tomu NTFS wywodzi się z obiektowego modelu systemu NT.
- Każdy obiekt pliku ma w rekordzie tablicy MFT przechowywany atrybut deskryptora bezpieczeństwa (*security descriptor attribute*).
- Atrybut deskryptora bezpieczeństwa zawiera żeton dostępu właściciela pliku oraz listę kontroli dostępu, wykazującą prawa dostępu udzielone użytkownikom, którym pozwolono na korzystanie z pliku.