Ochrona i bezpieczeństwo

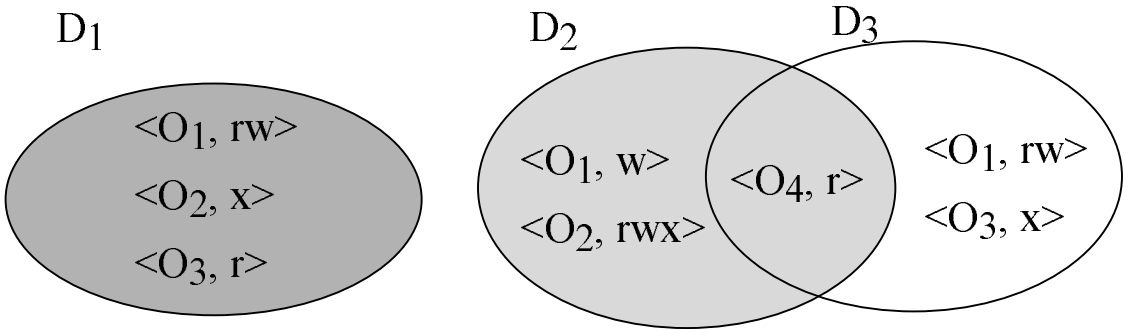
Ochrona: mechanizm służący do kontrolowania dostępu procesów lub użytkowników do zasobów systemu komputerowego

Bezpieczeństwo: miara zaufania, że zostanie zachowana nienaruszalność systemu i jego danych

Cel ochrony: zapewnić, że każdy obiekt w systemie jest wykorzystywany prawidłowo i tylko przez upoważnione do tego procesy

Domeny ochrony

- System komputerowy jest zbiorem obiektów (sprzętowych i programowych)
- Każdy obiekt ma jednoznaczną nazwę i można na nim wykonywać określony zbiór operacji
- Domena ochrony określa zasoby dostępne dla procesu:
  zbiór-praw = podzbiór zbioru operacji wykonywanych na obiekcie
  prawo-dostępu = <*nazwa-obiektu*, *zbiór-praw*>
  domena = zbiór *praw-dostępu*

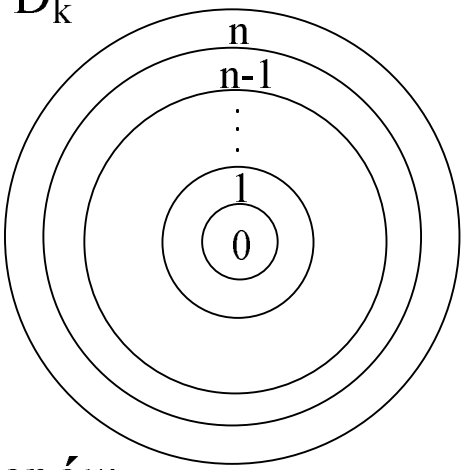


Implementacja domeny

- System składa się z dwóch domen: użytkownika i nadzorcy
- Unix
  - domena = id-użytkownika
  - przełączanie domeny realizowane przez system plików: z każdym plikiem jest związany bit domeny (set-uid bit); kiedy wykonuje się plik i set-uid bit=1, to czasowo identyfikatorem użytkownika staje się id właściciela pliku; po zakończeniu powrót do poprzedniego id
- Pierścienie w Multicsie

  niech D_i i D_k będą dwoma pierścieniami domen

  $k < i \Rightarrow D_i \subseteq D_k$



Macierz dostępów

- wiersze - domeny
- kolumny - obiekty i domeny
- każda pozycja - prawa dostępu, nazwy operacji

obiekt / domena	F_1	F_2	F_3	czytnik	drukarka
D_1	czytaj		czytaj		
D_2				czytaj	drukuj
D_3	czytaj pisz	wykonuj	czytaj pisz		

- jeśli proces w domenie D_i próbuje wykonać operację „op” na obiekcie O_k, to „op” musi należeć do macierzy dostępu
- macierz dostępu pozwala oddzielić *mechanizm* ochrony od realizowanej *polityki* ochrony
- można realizować dynamiczną ochronę
  - potrzebne operacje dodawania i usuwania praw dostępu
  - specjalne prawa dostępu:
    - *właściciel* obiektu O_i (dodawanie i usuwanie praw) - zmiana w kolumnie
    - *kopiowanie* (*ograniczone kopiowanie* lub *przekazanie*) operacji z O_i do O_k - zmiana w kolumnie
    - *kontrola* domeny - zmiana w wierszu
    - *przełączenie* z domeny D_i do D_k

obiekt / domena	F_1	F_2	D_1	D_2	D_3
D_1	czytaj*	właściciel		przełącz	
D_2			przełącz		przełącz kontroluj
D_3	czytaj pisz	wykonuj			

Implementacja macierzy dostępu

- Każda kolumna = *lista praw dostępu* do jednego obiektu; można rozszerzyć o *standardowy* zbiór praw dostępu
- Każdy wiersz = *lista dojść* (ang. *capability*) domeny
- Przykład : system plików w Unixie
  - z każdym plikiem jest związana lista praw dostępu
  - w chwili otwarcia pliku sprawdza się listę praw i kopiuje je do tablicy plików (dołącza do listy dojść procesu)
  - podczas dostępu do pliku sprawdza się listę dojść

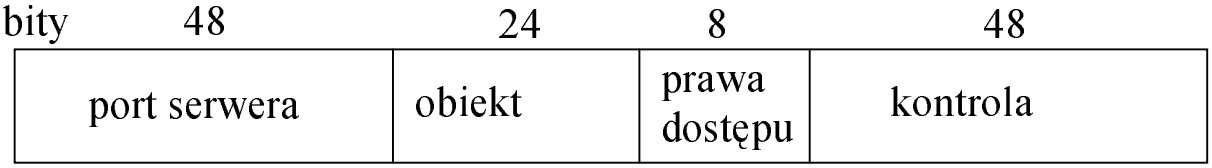
- podczas zamykania pliku usuwa się dojście do pliku
- gdy plik otwierany do czytania, do tablicy plików wstawia się jedynie dojście umożliwiające czytanie

Unieważnianie praw dostępu

- Lista praw dostępu: proste, natychmiastowe
- Lista dojść: trudniejsze, gdyż dojścia są rozproszone w systemie
  - wtórne pozyskiwanie (okresowo usuwa się dojścia z każdej domeny)
  - wskaźniki zwrotne - od obiektu do jego dojść (Multics)
  - dowiązanie pośrednie (w globalnej tablicy)
  - klucze: z każdym obiektem jest związany klucz główny (modyfikowalny); tworząc dojście przypisuje się mu bieżącą wartość klucza głównego (proces nie może go sprawdzić ani zmienić); operację można wykonać, gdy klucze pasują; unieważnienie polega na zmianie klucza głównego

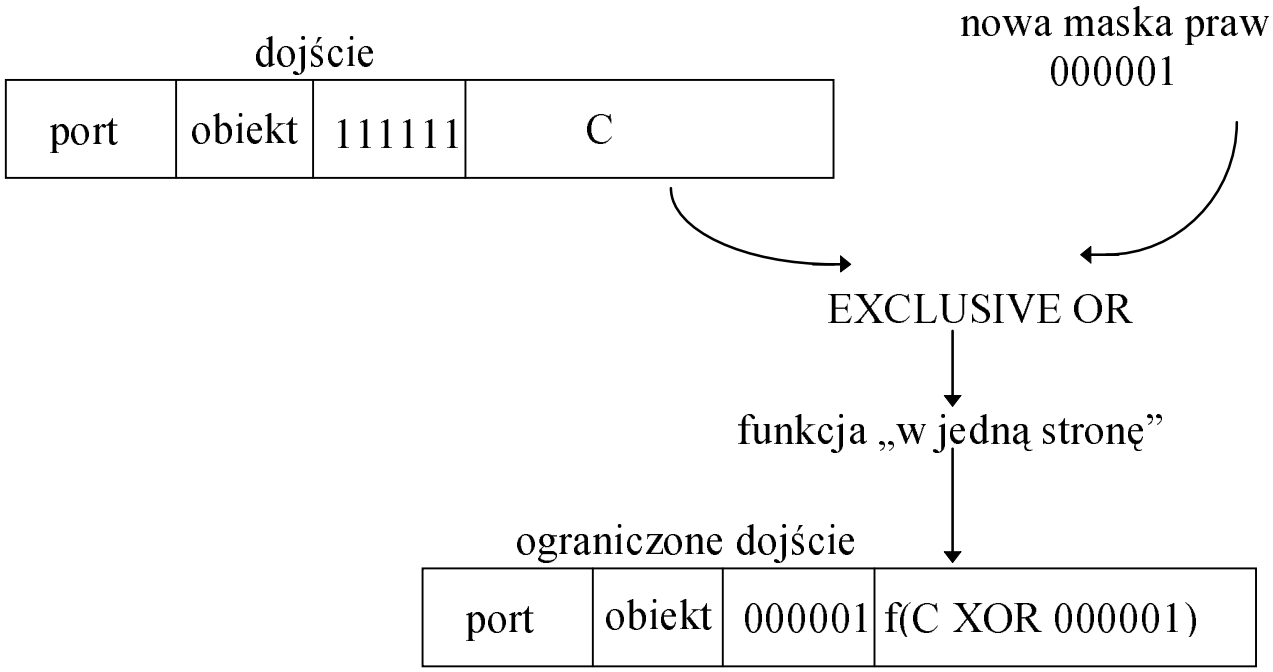
Przykład: obiekty i dojścia w Amoebie

- Wykonanie operacji na obiekcie wymaga wywołania zdalnej procedury (RPC) serwera
- Klienci nie znają położenia obiektów ani serwerów
- Klient tworzący obiekt wywołuje RPC, dostaje od serwera dojście do nowego obiektu



- Chcąc wykonać operację na obiekcie, klient musi dostarczyć dojście

- Jądro usuwa z dojścia port serwera (lokalizuje maszynę), resztę przesyła serwerowi
- Podczas tworzenia obiektu serwer losowo ustala wartość *pola kontrolnego*, wstawia ją do dojścia i zapamiętuje w swoich tablicach
- *Właściciel dojścia* ma wszystkie bity praw dostępu ustawione na 1
- Utworzenie *ograniczonego dojścia* wymaga przesłania do serwera dojścia i maski bitowej nowych praw



- Otrzymawszy ograniczone dojście serwer wykonuje XOR pola kontrolnego z tablicy z prawami dostępu i porównuje wynik z polem kontrolnym dojścia
- Co się stanie, gdy użytkownik „ręcznie” rozszerzy swoje prawa dostępu umieszczone w dojściu?

Problem bezpieczeństwa

- Bezpieczeństwo systemu wymaga uwzględnienia zewn. środowiska systemu i chronienia systemu przed:
  - nieautoryzowanym dostępem
  - złośliwą modyfikacją lub niszczeniem
  - przypadkowym „rozspójnieniem”
- Powody utraty informacji:
  - zdarzenia losowe
  - błędy sprzętowe
  - błędy ludzkie

  Można temu zapobiegać poprzez archiwizowanie
- Intruzi
  - pasywni („podglądacze”)
  - aktywni (ciekawscy, włamywacze dla sportu, włamywacze dla pieniędzy, szpiedzy)
- Kontrola „tożsamości” użytkownika - hasła (wymuszanie częstej zmiany hasła, kontrola „odgadywalności” hasła, rejestrowanie błędnych prób dostępu)
- Słynne przykłady błędów dotyczących ochrony w znanych systemach

Unix

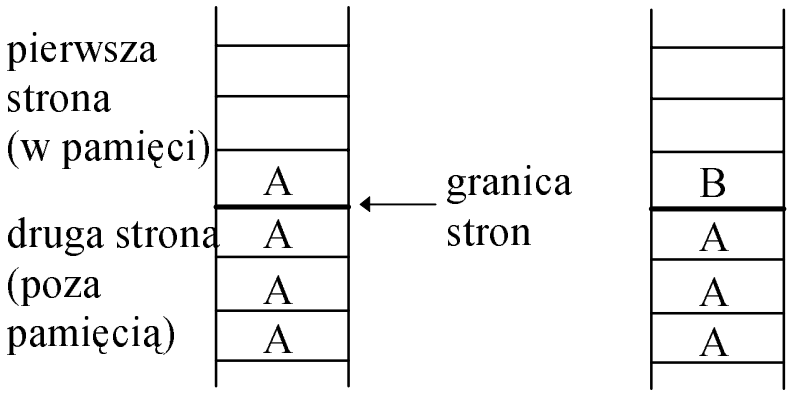
- Program lpr ma opcję usuwania pliku po jego wydrukowaniu; wczesne wersje systemu zezwalały na wydrukowanie przez każdego użytkownika (i późniejsze usunięcie) pliku systemowego z hasłami
- Kilka innych sposobów dotarcia do pliku z hasłami

Multics

- Ochrona dla pracy w trybie podziału czasu była świetna, a dla trybu wsadowego prawie nie istniała
  - zmodyfikowanie edytora, aby poza edycją kradł pliki
  - wstawienie skompilowanej wersji do katalogu bin ofiary

Tenex

- Pliki były chronione hasłami; aby odgadnąć hasło ofiary wykorzystywało się stronicowanie i pamięć wirtualną
- Intruz umieszczał hasło na stronie w określonej pozycji:



- Jeśli prawdziwe hasło zaczynało się na A, to pojawiało się przerwanie braku strony (wyłapywane przez intruza), wpp sygnał ILLEGAL PASSWORD

„The Internet Worm"

- 2.11.1988 student Cornell Univ, Robert Tappan Morris, wpuścił do sieci program z robakiem
- Unieszkodliwił tysiące komputerów na świecie
- Opisane w CACM, vol. 32, June 1989, pp. 678-687
- Morris wykrył dwa błędy w Berkeley Unix, które pozwoliły mu bez autoryzacji mieć dostęp do komputerów w Internecie
- Napisał program, który rozmnażał się w każdym komputerze, do którego dotarł

- Nie wiadomo, czy 2.11 miał być test, czy prawdziwy atak
- Program składał się z dwóch części: programu ładującego (l1.c, 99 wierszy w C) i właściwego robaka
- l1.c był kompilowany i wykonywany na atakowanej maszynie; podczas działania ściągał z maszyny, z której przyszedł, właściwego robaka
- Zamazując za sobą ślady, patrzył dokąd może się udać dalej
- Trzy metody infekowania kolejnych komputerów
  - próba wykonania rsh - niektóre komputery wpuszczają
  - finger - finger daemon nie sprawdzał przepełn. bufora)
  - wykorzystanie błędu w sendmail
- Gdy robak dotarł na docelowy komputer, starał się złamać hasło (artykuł ojca i Kena Thompsona z 1979 roku)
- Robak uciekał, gdy widział już swoją kopię, ale raz na 7 zostawał - to spowodowało epidemię
- Moriss wpadł, gdy jego przyjaciel powiedział dziennikarzowi (John Markoff), że robak jest niewinny, wszystko było żartem i autorowi jest przykro; zdradził przy tym login autora: rtm
- Morris był sądzony (sąd federalny):
  - grzywna 10,000 $
  - 3 lata opieki kuratora
  - 400 godzin pracy społecznej
  - ponadto zasądzono od niego w sumie 150,000 $
- Opinie na temat kary były podzielone

Szyfrowanie

- Szyfrowanie jest powszechnie stosowaną metodą ochrony informacji przesyłanej przez niepewne łącza
- Cechy dobrej metody szyfrowania:

  k - klucz

  E_k - ogólny algorytm szyfrowania kluczem k

  D_k - ogólny algorytm deszyfrowania kluczem k

  m - komunikat

  1. $D_k\,(E_k(m)) = m$
  2. E_k i D_k można obliczyć efektywnie
  3. Bezp. systemu zależy od tajności k, a nie E lub D
- Standard Szyfrowania Danych (*DES*)
  wymaga (bezpiecznej!) dystrybucji klucza do autoryzowanych użytkowników - kłopotliwe
- Szyfrowanie z *kluczem publicznym*: każdy użytkownik ma dwa klucze:
  - *klucz publiczny* - do szyfrowania danych
  - *klucz prywatny* - znany jedynie pojedynczemu użytkownikowi; do deszyfrowania danych

  (e, n) - klucz publiczny

  (d, n) - klucz prywatny

  (e, d, n - dodatnie liczby całkowite)

  m - komunikat, liczba całkowita z przedziału [0, n-1]

  $$E(m) = m^e \bmod n = C$$

  $$D(C) = C^d \bmod n$$

  $n = p \cdot q$ (p, q - duże, losowe liczby pierwsze, liczba cyfr>100)

  n - znane publicznie, ale p i q nie (rozłożenie liczby n na czynniki pierwsze jest b. trudne)

d = duża, losowa liczba całkowita, względnie pierwsza wobec $(p - 1) \cdot (q - 1)$: nww $[d, (p - 1) \cdot (q - 1)] = 1$
e: $e \cdot d \bmod (p - 1) \cdot (q - 1) = 1$
(d i e też trudno odgadnąć)

Przykład: $p = 5$, $q = 7$
$n = 35$, $(p - 1) \cdot (q - 1) = 24$
$d = 11$, $e = 11$
dla $m = 3$:
$C = m^e \bmod n = 3^{11} \bmod 35 = 12$
$C^d \bmod n = 12^{11} \bmod 35 = 3 = m$

PGP (*Pretty Good Privacy*, Philip Zimmermann)

Ogólnie dostępny pakiet programowy do kodowania i „podpisywania" przesyłanych danych

Nadawca:
- koduje podpis własnym kluczem prywatnym
- koduje podpisany komunikat kluczem publicznym odbiorcy

Odbiorca:
- dekoduje komunikat własnym kluczem prywatnym
- dekoduje podpis kluczem publicznym nadawcy