

Bezpieczeństwo systemów operacyjnych

Ochrona zasobów, wsparcie sprzętowe

Arkadiusz Chrobot

Katedra Systemów Informatycznych

16 października 2026

Plan

- 1 Wstęp
- 2 Procesor
 - Tryby pracy procesora
 - Ochrona procesora
- 3 Ochrona pamięci
- 4 Procesory kryptograficzne
- 5 Kontrola dostępu — system operacyjny

Wstęp

Ochrona w ujęciu technicznym jest kontrolą dostępu do zasobów. W systemie komputerowym jest ona zorganizowana na wielu poziomach, od aplikacji użytkowych do sprzętu. System operacyjny również spełnia istotną rolę w tym zagadnieniu, nadzoruje wykorzystanie zasobów fizycznych oraz logicznych, które dodatkowo tworzy (np. pliki, łącza komunikacyjne, procesy). Sam zaś musi polegać na mechanizmach sprzętowych. Warto zauważyć, że część z rozwiązań stosowanych na poziomie systemu operacyjnego w tej samej lub zmienionej postaci jest stosowana na wyższych poziomach, takich jak oprogramowanie pośrednie (ang. *middleware*) lub aplikacje użytkowe, celem zrealizowania polityki bezpieczeństwa dla określonej dziedziny zastosowania.

DAC vs. MAC

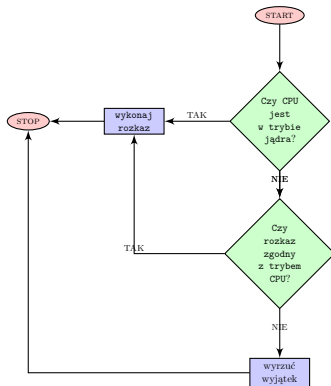
W informatyce istnieją dwa główne podejścia do kwestii ochrony. Oba są pochodną paradygmatu *wielowarstwowego bezpieczeństwa* (ang. *multilayered security*). Pierwsze jest nazywane *uznaniową kontrolą dostępu* (ang. *Discretionary Access Control* — DAC), natomiast to drugie *obowiązkową kontrolą dostępu* (ang. *Mandatory Access Control* — MAC). Model DAC zakłada, że osoba która ma uprawnienia administratora w systemie operacyjnym lub fizyczny dostęp do systemu komputerowego może wykonać w nim dowolną operację. W przypadku modelu MAC pełną kontrolę zachowuje dostawca lub właściciel sprzętu, dając administratorowi szersze prawa niż zwykłemu użytkownikowi, ale nie całkowitą władzę nad systemem.

Tryby pracy procesora

Podstawowy mechanizm ochrony zasobów zapewniany przez procesor, to *dualny tryb pracy*. Nie każde oprogramowanie uruchomione w systemie komputerowym powinno móc wykonać dowolny rozkaz z listy rozkazów CPU. Znajdują się na niej między innymi rozkazy operujące na urządzeniach wejścia-wyjścia, czy obsługujące inne sprzętowe mechanizmy bezpieczeństwa. Mogą one być wykonywane wyłącznie przez jądro systemu operacyjnego i powinny być niedostępne dla oprogramowania użytkowego. Dlatego w słowie binarnym każdego rozkazu jest wyróżniony bit. Jeśli w danym rozkazie jest on ustawiony na 1, to znaczy, że może on być wykonywany zarówno przez proces użytkownika, jak i przez jądro. Natomiast jeśli jego wartość wynosi 0, to dopuszczalne jest wykonanie tego rozkazu wyłącznie przez jądro. Aby ten podział był przestrzegany w *rejestrze stanu procesora* jest zapisana, na jednym bicie, informacja o bieżącym trybie pracy procesora. Wartość 1 oznacza *tryb użytkownika*, a 0 *tryb jądra*.

Tryby pracy procesora

Każde wykonanie dowolnego rozkazu poprzedzane jest *sprzętową* realizacją algorytmu opisanego poniższym schematem blokowym.



Rysunek: Weryfikacja zgodności wykonania rozkazu z trybem pracy procesora

Tryby pracy procesora — Aarch64

W procesorach o architekturze ARM Aarch64 zdefiniowano cztery tryby pracy nazwane *poziomami wyjątków* (ang. *exception levels*).

EL0 poziom aplikacji,

EL1 poziom (jądra) systemu operacyjnego,

EL2 poziom hipernadzorcy,

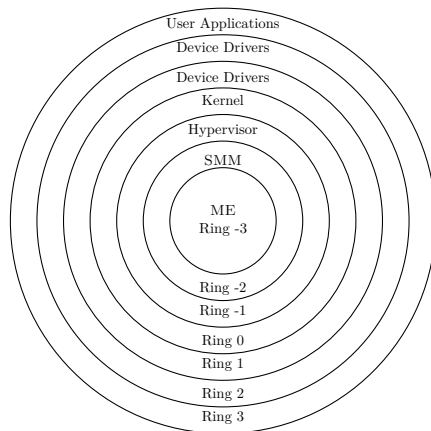
EL3 poziom oprogramowania sprzętowego/monitora bezpieczeństwa.

Uprzywilejowanie oprogramowania rośnie wraz z numerem poziomu wyjątku, zatem najwyższe ma oprogramowanie sprzętowe (ang. *firmware*), które może także pełnić rolę monitora bezpieczeństwa (ang. *security monitor*). Architektura Aarch64 uwzględnia również potrzeby hipernadzorców typu pierwszego — działających bezpośrednio na sprzęcie — nadzorujących maszyny wirtualne. Zmiana poziomu wyjątku zachodzi w jednej z pięciu sytuacji: przyjęcie obsługi wyjątku, powrót z obsługi wyjątku, reset procesora, w trakcie debugowania, przy wyjściu z debugowania.

Tryby pracy procesora — x86

Mechanizm ochrony w procesorach o architekturze x86, czyli produkowanych przez firmy Intel i AMD, oparty jest na rozwiązaniu, które po raz pierwszy zastosowano w systemie operacyjnym Multics, nazwanym *pierścieniami ochrony* (ang. *rings of protection*). W oryginalnej koncepcji i procesorach produkowanych do 2008 roku występowały 4 takie pierścienie. Potem, w CPU zwiększono ich liczbę (Rysunek 2) do siedmiu. Pierścień 3 przeznaczony jest dla aplikacji użytkowych, dwa kolejne (2–1) dla sterowników urządzeń, a pierścień 0 dla jądra systemu. Nieformalnie nowe pierścienie oznaczono liczbami ujemnymi — producenci nie wprowadzili własnych oznaczeń. Pierścień –1 przeznaczony jest dla hipernadzorcy maszyn wirtualnych, –2 dla oprogramowania nadzorującego nowoczesny sprzęt, np. zarządzającego energią. Ostatni pierścień (–3) jest przeznaczony dla oprogramowania, które w procesorach Intela nazywa się *Management Engine* (ME) lub *Converged Security and Management Engine* (CSME). W przypadku AMD nazywa się ono *Secure Technology*.

Tryby pracy procesora — x86



Rysunek: Pierścienie ochrony w procesorach x86 (źródło: <https://medium.com/swlh/negative-rings-in-intel-architecture-the-security-threats-youve-probably-never-heard-of-d725a4b6f831>)

Ochrona procesora

Warto zaznaczyć, że Intel ME jest niczym innym jak systemem operacyjnym, bazującym na MINIX 3, wbudowanym w procesor (!) i wykonywanym zawsze gdy płyta główna ma zasilenie, nawet jeśli komputer jest wyłączony. ME podobnie jak *Security Technology* nie jest wolne od podatności, które mogą być wykorzystane przez adwersarzy. Systemy operacyjne, takie jak Linux lub Windows korzystają jedynie z dwóch pierścieni ochrony nr 3 i nr 0.

Do ochrony procesora przez byciem zmonopolizowanym przez procesy użytkownika służy zazwyczaj *czasomierz* (ang. *timer*) w postaci licznika zliczającego w dół i generującego przerwanie, gdy jego wartość osiągnie 0. Rozkaz inicjowania takiego licznika może wykonać tylko system operacyjny, który może wykorzystać ten licznik do implementacji szeregowania rotacyjnego lub jako układ typu *watchdog*.

Ochrona pamięci

Współcześnie ochrona pamięci polega na użyciu *pamięci wirtualnej*, najczęściej bazującej na koncepcji *stronicowania na żądanie* (ang. *on-demand paging*). Procesory x86 oferowały również wsparcie sprzętowe dla segmentacji, ale nie była ona powszechnie używana i producenci tych CPU zrezygnowali z tego rozwiązania. Zazwyczaj każda strona może być oznaczona w tablicy stron, jako przeznaczona do czytania, pisania lub wykonania. W trakcie wykonania procesów jednostka sprzętowa MMU (ang. *Memory Management Unit*) weryfikuje na bieżąco, czy proces przestrzega tego oznaczenia. Niektóre procesory o architekturze ARM posiadają dodatkowo mechanizm uwierzytelniania wskaźników (ang. *Pointer Authentication* ), takich jak zapisany na stosie adres powrotu lub wskaźniki funkcji. Systemy operacyjne również wprowadzają mechanizmy programowe, utrudniające przeprowadzenie niektórych ataków na dane przechowywane w RAM. Należy do nich *Address Space Layout Randomization*, stosowany zarówno w przestrzeni użytkownika, jak i w jądrze.

Ochrona pamięci

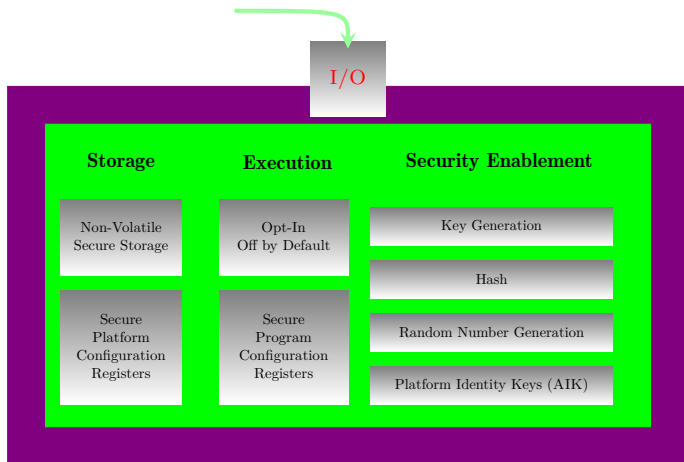
Powoduje on, że po każdym ładowaniu programu do pamięci operacyjnej jego kod i dane (np. funkcje i zmienne) znajdują się pod innymi adresami niż poprzednio. Odkryto również ataki na sprzętową ochronę pamięci oferowaną przez niektóre CPU, które bazują na kanałach bocznych (ang. *side channel*). Należą do nich [▶ Meltdown](#), umożliwiający odczyt pamięci jądra i [▶ Spectre](#), pozwalający odczytać pamięć dowolnego procesu użytkownika. Ich przeprowadzenie jest utrudniane na poziomie oprogramowania. Np. w przypadku Meltdown jądro stosuje różne odwzorowania przestrzeni pamięci wirtualnej w trybie użytkownika i trybie jądra. Dzięki temu, procesy użytkownika „widzą” tylko niewielki fragment zawartości obszaru pamięci jądra, który jest niezbędny do uruchamiania wywołań systemowych oraz obsługi wyjątków i przerw. W Linuksie ten mechanizm nazywa się *Kernel Page Table Isolation* (KPTI). Przed innym atakiem z tej rodziny, o nazwie [▶ Row Hammer](#) służą z kolei mechanizmy sprzętowe odświeżające wiersze w RAM.

Współczesne zastosowania sprzętu komputerowego wymusiły przeniesienie wykonania operacji związanych z bezpieczeństwem, takich jak np. generowanie kluczy do szyfrowania, z procesorów ogólnego przeznaczenia (CPU) do specjalnie skonstruowanych procesorów kryptograficznych, które stanowią *podstawę zaufania* (ang. *root of trust*) w systemach komputerowych. Na masową skalę zostały one zastosowane w urządzenia mobilnych, takich jak smartfony. Ich użycie w innych systemach wbudowanych wymusza na terenie Unii Europejskiej ustawa *Cyber Resilience Act* (CRA)¹.

W przypadku komputerów klasy PC te procesory mają postać modułu TPM (ang. *Trusted Platform Module*). Jest to rodzina mikroprocesorów (najnowszy ma oznaczenie 2.0) opracowana przez konsorcjum *Trusted Computing Group*, założone przez Intel, Microsoft, HP i IBM. Z założenia ten moduł ma umożliwiać realizację modelu MAC. Jego ogólny schemat znajduje się na kolejnym slajdzie.

¹Katarzyna Zalewska „Cyber Resilience Act: co oznacza dla programistów systemów wbudowanych?”, Magazyn Programista, nr 123, 2026

Trusted Platform Module



Rysunek: Komponenty TPM (źródło: <https://trustedcomputinggroup.org/resource/trusted-platform-module-tpm-summary/>)

Trusted Platform Module

TPM umożliwia bezpieczne przechowywanie informacji, takich jak np. klucze szyfrujące/deszyfrujące pamięć masową, hasła, certyfikaty poświadczające tożsamość i metryki uwierzytelniające (ang. *authentication measurements*). Może on również wykonywać operacje kryptograficzne, takie jak szyfrowanie symetryczne (AES) i asymetryczne (RSA, ECC), a także wyznaczanie wartości funkcji skrótu (SHA). Ten moduł używany jest także do generowania kryptograficznie bezpiecznych liczb pseudolosowych.

Umożliwia on także realizację *bezpiecznego rozruchu* (ang. *secure boot*), czyli procesu uruchamiania systemu komputerowego, w którym sprawdzana jest integralność oprogramowania i konfiguracji. TPM może zweryfikować oprogramowanie sprzętowe oraz system operacyjnych, wyliczając ich skróty i porównując z zapisanymi. Może też weryfikować podpisy cyfrowe i blokować wykonanie oprogramowania, które nie przechodzi takiej weryfikacji.

Trusted Platform Module

Należy jednak pamiętać, że TPM może zawierać również podatności sprzętowe. W systemie Linux ten moduł może być obsługiwany przy pomocy narzędzi z pakietu `tpm2-tools`. Z kolei w Windows do jego obsługi można użyć PowerShell lub CMD. To pierwsze narzędzie udostępnia polecenie `Get-Tpm`, a to drugi `tpmtool`.

Kontrola dostępu — system operacyjny

System operacyjny musi sprawować także kontrolę nad wykorzystanie zasobów logicznych, które tworzy, takich jak pliki czy łącza komunikacyjne. Ich ochronę można modelować przy użyciu macierzy dostępu, takich jak ta znajdująca się poniżej, zapożyczona z książki Rossa Andersona:

Użytkownik	System Operacyjny	Program Księgowy	Dane Księgowe	Ślad audytowy
Sam	rwX	rwX	r	r
Alice	rx	x	—	—
Program księgowy	rx	rx	rw	w
Bob	rx	r	r	r

Macierze i domeny ochrony

Alternatywnie, polityka dostępu przedstawiona przez macierz może być opisana *trójkami dostępowymi* (ang. *access triples*), formalnie nazywanymi *domenami ochrony* (ang. *protection domain*). Taka trójka zawiera, zgodnie z nazwą, trzy wartości (*użytkownik, program, plik*).

Można użyć macierzy kontroli dostępu bezpośrednio do implementacji ochrony, ale sprawdzają się one w przypadku tylko małych systemów, o ograniczonej liczbie użytkowników. W praktycznych zastosowaniach nie skalują się dobrze.

Aby uzyskać efektywny system ochrony należy wprowadzić kilka zmian. Przede wszystkim należy wprowadzić *grupy i role*. Często te dwa pojęcia są uważane za tożsame, ale istnieje między nimi rozróżnienie. Grupy może być rozumiana jako lista podmiotów (ang. *principles*), czyli użytkowników, procesów i/lub wątków, a rola to po zestaw uprawnień, które podmiot może posiadać przez określony czas.

Listy kontroli dostępu

Większość mechanizmów ochrony w popularnych systemach operacyjnych bazuje na listach kontroli dostępu (ang. *Access Control Lists*), które są powiązane z określonymi zasobami. Możemy o nich myśleć jako o kolumnach z macierzy dostępu. Są one proste w implementacji i sprawdzają się w systemach, gdzie ochrona jest zorientowana na dane. W systemach z dużą i dynamiczną liczbą użytkowników, a także tam gdzie użytkownicy często muszą delegować uprawnienia są one mniej efektywne.

W przypadku systemu Linux (i innych wywodzących się z Uniksa) listy są zrealizowane w postaci atrybutów `rwX`, gdzie litera `r` oznacza odczyt w zapis, a `x` wykonanie. Są trzy zestawy takich atrybutów: dla właściciela zasobu, dla grupy i dla reszty użytkowników.

Listy kontroli dostępu

Dodatkowo istnieją trzy specjalne atrybuty: *setuid* — jeśli jest ustawiony dla pliku binarnego, to ten plik będzie wykonywany z uprawnieniami jego właściciela, a nie użytkownika, który go uruchomił, *setgid* — ustawiony ten bit powoduje, że plik binarny jest wykonywany z uprawnieniami grupy użytkowników, której jest przypisany. W końcu atrybut *sticky bit* włącza prawo ograniczonego usuwania — tylko właściciel pliku może go usunąć, nawet jeśli ten plik znajduje się w katalogu, gdzie wszyscy użytkownicy mają wszelkie uprawnienia.

Listy kontroli dostępu

Z każdym plikiem/zasobem jest powiązany właściciel i grupa użytkowników. Obie wartości można zmienić za pomocą, odpowiednio, poleceń `chown` i `chgrp`. Uprawnienia dla danego pliku/zasobu można zmodyfikować poleceniem `chmod`. Z kolei informacje o tym, do których grup należy bieżący użytkownik można uzyskać poleceniem `id`. Z kolei istniejącego użytkownika można dodać do grupy za pomocą polecenia `usermod`. Nowego użytkownika w systemie można stworzyć przy pomocy `useradd` lub `adduser`, które także może go przypisać do grupy. Nową grupę tworzą polecenia `groupadd` i `addgroup`.

Listy kontroli dostępu

Linux posiada także inną implementację list kontroli dostępu, która nazywa się po prostu ...*access control list* (*acl*). Umożliwia ona bardziej elastyczną kontrolę dostępu do zasobów lepiej nadaje się do realizacji *zasady wiedzy koniecznej*, nazywanej również *zasadą najmniejszego uprzywilejowania*. Atrybuty **rw**x nadal są używane w tym rozwiązaniu, ale pozwala ono wprost wskazać, który konkretny użytkownik lub która konkretna grupa dysponuje jakimi uprawnieniami. Innymi słowy mechanizm *acl* umożliwia powiązanie praw do zasobu nie tylko z właścicielem i jedną z grup, ale także z kilkoma innymi grupami lub kilkoma konkretnymi użytkownikami. Jest on obsługiwany za pomocą poleceń **getfacl** i **setfacl**.

Możliwości

O możliwościach możemy myśleć jako o wierszach macierzy dostępu. W przeciwieństwie do list kontroli dostępu umożliwiając one łatwe delegowanie uprawnień na określony czas oraz ich weryfikacja w trakcie wykonania jest prostsza. Z drugiej strony trudniej jest zmieniać uprawnienia do zasobów albo przeprowadzić audyt nadużyć uprawnień, bo nie jest łatwo sprawdzić, kto w danym czasie dysponował określonymi uprawnieniami. Opracowano prototypowe systemy badawcze, gdzie możliwości były implementowane w postaci typów danych rozpoznawanych na poziomie sprzętowym lub przechowywane w specjalnych obszarach (segmentach) pamięci. Współcześnie jednak bardziej popularnym rozwiązaniem jest użycie do ich realizacji narzędzi kryptograficznych, np. certyfikatów. Zarówno Linux, jak i Windows stosują możliwości. W przypadku systemu Linux są one zrealizowane przy pomocy poleceń `sudo` i `su`.

Możliwości

Warto również wspomnieć, że w Linuksie zaimplementowano mechanizm o nazwie *capabilities*, który nie ma wiele wspólnego z możliwościami opisanymi na poprzednim slajdzie. Umożliwia on jedynie bardziej precyzyjne, drobnoziarniste kontrolowanie uprawnień jakie mają procesy do poszczególnych zasobów. Przykładowo programowi `ping` jest nadawana możliwość korzystania z surowych gniazd (ang. *raw sockets*), aby mógł obsługiwać protokół ICMPv6. Do obsługi takich możliwości służą między innymi polecenia `getcap` i `setcap`. Więcej informacji o możliwościach można znaleźć na stronie *capabilities* podręcznika *man* w sekcji 7.

Pytania

?

KONIEC

Dziękuję Państwu za uwagę!