POLITECHNIKA GDAŃSKA

Wydział Elektroniki, Telekomunikacji i Informatyki

Michał Wróbel

Metody zapewniania bezpieczeństwa systemów operacyjnych

Rozprawa doktorska

Promotor:
dr hab. inż. Jerzy Kaczmarek
Wydział Elektroniki, Telekomunikacji i Informatyki
Politechnika Gdańska

Gdańsk, 2010

Spis treści

Rozdział 1.
Wprowadzenie

Współcześnie wzrasta liczba zagrożeń dla systemów informatycznych. Coraz więcej przeprowadzanych ataków jest skutecznych, a straty spowodowane włamaniami systematycznie rosną. Dlatego problem zapewniania bezpieczeństwa systemów komputerowych jest jednym z najbardziej istotnych zagadnień współczesnej informatyki. W ośrodkach naukowych na całym świecie prowadzone są liczne badania dotyczące zapewniania bezpieczeństwa systemów operacyjnych i programów użytkowych. Wśród wiodących jednostek badawczych można wyróżnić MIT, Uniwersytet Kalifornijski w Berkley, Uniwersytet Harvarda, czy uniwersytety w Melbourne, Lizbonie, Sharjah (ZEA), Pekinie, Mannheim, oraz politechniki w Madrycie, Wiedniu, Katalonii i wiele innych [11, 39, 63, 68, 69, 74, 76, 80, 90, 96, 114].

Badania prowadzone są w dwóch głównych kierunkach. Jednym jest tworzenie mechanizmów do minimalizacji skutków włamań, drugi kierunek to tworzenie systemów ograniczających możliwości włamań i ingerencji w zapisane dane.

Należy postawić pytanie czy w ogóle istnieje możliwość całkowitego zabezpieczenia systemów komputerowych. Wydaje się, że osiągnięcie absolutnego bezpieczeństwa systemów informatycznych nie jest możliwe. Przyczyną tego stanu jest nierozwiązywalny w dziedzinie informatyki problem, jakim jest występowanie błędów w programach komputerowych i nieprzewidywalne możliwości wykorzystywania przez intruzów niektórych fragmentów ich kodu. W praktyce nie jest możliwe stworzenia dużego programu komputerowego, w tym również systemu operacyjnego, który byłby całkowicie pozbawiony błędów. Można wręcz powiedzieć, że każdy działający program komputerowy posiada fragmenty kodu, które w określonych warunkach mogą wygenerować błąd, co czasami może doprowadzić do uzyskania dostępu do zasobów komputera przez intruza.

W takiej sytuacji można postawić ważne pytanie badawcze. Jak tworzyć mechanizmy, które pomimo istnienia błędów w systemach operacyjnych czy programach użytkowych zapewnią bezpieczeństwo systemowi komputerowemu. Jest to bardzo ważne pytanie, które będzie szczegółowo rozpatrywane w niniejszej pracy. Należy założyć, że przy pomocy różnych znanych lub nieznanych obecnie mechanizmów intruz zawsze będzie w stanie przejąć kontrolę nad systemem operacyjnym. Jest to założenie pesymistyczne, ale prowadzi do daleko idących wniosków. Wydaje się, że jedynym skutecznym rozwiązaniem pozwalającym na zapewnienie bezpieczeństwa systemów komputerowych jest stworzenie takich mechanizmów, które pomimo przeprowadzenia skutecznego włamania przez intruza nie pozwolą na dokonanie poważnych uszkodzeń systemu, zmiany danych, czy innych wrogich działań. Jest to bardzo istotne założenie z punktu widzenia celów, jakie postawiono przed tą pracą. Jak zostanie wykazane, jest możliwe wykonanie tego typu mechanizmów, które mogą być dołączane do kodu jądra systemów operacyjnych jako działające w nim moduły

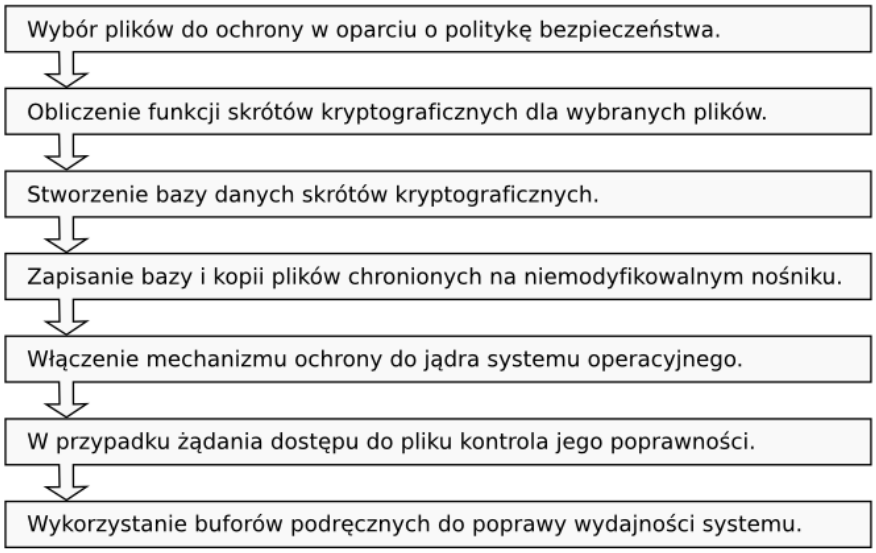
bezpieczeństwa. Należy zwrócić uwagę, że włamania do systemów operacyjnych są dokonywane nie tylko z wykorzystaniem znanych lub nieznanych błędów w kodzie systemu operacyjnego, ale również na bazie poprawnie działających mechanizmów zarządzania pamięcią komputera lub szeregowania zadań wykonywanych przez system operacyjny. Dlatego znalezienie skutecznych mechanizmów zabezpieczeń to zadanie trudne, choć wydaje się być rozwiązywalne.

Cele i zakres pracy

Celem pracy było opracowanie nowych metod zapewniania bezpieczeństwa systemów operacyjnych opartych o trwały zapis istotnych danych systemowych na niemodyfikowalnych nośnikach oraz wykonanie mechanizmów kontroli dostępu do plików z wykorzystaniem skrótów kryptograficznych. Mechanizmy powinny umożliwiać weryfikację poprawności treści plików oraz przywracania ich oryginalnej treści w przypadku nieuprawnionej modyfikacji. Celem pracy było również opracowanie skutecznej polityki bezpieczeństwa systemów operacyjnych, w której należało uwzględnić rodzaje plików, częstość ich modyfikacji i ich znaczenie z punktu widzenia poprawności działania systemu komputerowego.

Wśród znanych systemów ochrony integralności, opracowany mechanizm wyróżnia się dwoma innowacyjnymi rozwiązaniami. Po pierwsze umożliwia automatyczne przywracanie z kopii zapasowych zawartości plików zmodyfikowanych w sposób nieautoryzowany. Drugą niezwykle istotną cechą opracowanego mechanizmu jest przechowywanie na nośnikach niemodyfikowalnych kluczowych plików niezbędnych do działania systemu zabezpieczającego. Takie rozwiązanie zabezpiecza dane w sposób sprzętowy i uniemożliwia ich modyfikację podczas ataków intruzów.

Na rysunku 1.1 przedstawiono poglądowy proces zapewniania bezpieczeństwa zgodny z opracowanym w pracy modelem.



Rysunek 1.1. Proces wdrażania i działania proponowanego mechanizmu bezpieczeństwa

W pierwszym kroku, w oparciu o politykę bezpieczeństwa, następuje wybór plików przeznaczonych do ochrony spośród dużej liczby plików przechowywanych w systemie operacyjnym. Podczas selekcji należy uwzględnić rodzaje plików, częstość ich modyfikacji i znaczenia z punktu widzenia poprawności działania systemu komputerowego. Następnie obliczane są skróty kryptograficzne wybranych plików, które zostają umieszczone w bazie danych. Baza danych skrótów kryptograficznych i kopie zapasowe chronionych plików zapisywane są na niemodyfikowalnym nośniku, co zapewnia ich bezpieczeństwo na poziomie fizycznym.

Mechanizm ochrony jest dołączany jako moduł do jądra systemu operacyjnego w celu uniemożliwienia jego wyeliminowania przez intruza. W przypadku każdego żądania dostępu do pliku dokonywana jest kontrola poprawności jego zawartości. W przypadku detekcji nieuprawnionej modyfikacji opracowany mechanizm zapewniania bezpieczeństwa automatycznie przywraca oryginalną zawartość zmienionego pliku z kopii zapasowych.

Opracowany model zapewniania bezpieczeństwa został zaimplementowany w wykonanym systemie o nazwie ICAR (ang. *Integrity Checking And Restoring*). W celu poprawy wydajności zastosowano w systemie ICAR mechanizm buforów podręcznych, które eliminują konieczność ponownego obliczania skrótów dla już zweryfikowanych plików. Przeprowadzone pomiary wydajności wykazały skuteczność działania mechanizmu buforów podręcznych.

W pracy wykazano, że opracowany oryginalny model zapewniania bezpieczeństwa systemów operacyjnych może zostać zaimplementowany w postaci mechanizmu działającego na poziomie jądra i w skuteczny sposób zapewniać bezpieczeństwo systemu operacyjnego. Opracowany mechanizm jest użyteczny i może być powszechnie stosowany.

Tezy pracy

W pracy wykazano następujące tezy:

1. Skutecznym sposobem zapewniania bezpieczeństwa systemów komputerowych jest kontrola poprawności danych zapisanych w kluczowych plikach, z wykorzystaniem skrótów kryptograficznych.
2. Wykorzystanie niemodyfikowalnych nośników do przechowywania kluczowych danych dla działania systemu ochrony zwiększa jego odporność na ataki.
3. Automatyczne przywracanie poprawnej zawartości plików chronionych w momencie wykrycia nieautoryzowanej modyfikacji zapewnia poprawność działania systemu operacyjnego.
4. Zastosowanie funkcji skrótów kryptograficznych wraz z mechanizmem buforów podręcznych do zabezpieczania kluczowych danych nie zmniejsza znacząco wydajności systemu operacyjnego.

Główne rezultaty pracy zostały przedstawione w 10 publikacjach.

Układ pracy

Praca doktorska została podzielona na 6 rozdziałów. W rozdziale pierwszym, który jest wprowadzeniem, znajduje się opis celu i zakresu pracy, tezy pracy oraz poglądowy opis zaprojektowanego procesu zapewniania bezpieczeństwa.

W rozdziale drugim, w oparciu o dane literaturowe, został opisany aktualny stan zaawansowania prac badawczych w dziedzinie zapewniania bezpieczeństwa systemów komputerowych. W sposób usystematyzowany przedstawiono zagrożenia występujące na różnych warstwach systemu komputerowego. Następnie opisano cele i metody działania intruzów. W dalszej części rozdziału przedstawiono znane obecnie mechanizmy bezpieczeństwa podzielone na trzy grupy: mechanizmy filtrowania ruchu sieciowego, mechanizmy wykrywania włamań oraz mechanizmy ograniczające skutki włamań. W tym rozdziale zostały wyszczególnione zwłaszcza te systemy i artykuły, które były istotne z punktu widzenia prac badawczych autora nad stworzeniem systemu ochrony integralności plików.

Rozdział trzeci został poświęcony prezentacji opracowanego przez autora modelu zapewniania bezpieczeństwa dla kluczowych danych, niezbędnych do poprawnego działania systemu operacyjnego. W modelu założono, że do skutecznego działania systemu zabezpieczeń niezbędne jest automatyczne przywracanie zmienionych przez intruza plików oraz wykorzystywanie niemodyfikowalnych nośników do zapisu kluczowych dla systemu zabezpieczeń danych. Rozdział zawiera szczegółowy opis algorytmu działania mechanizmu zabezpieczeń zaprojektowany na podstawie przedstawionego modelu. W rozdziale zostały również opisane zasady wyboru plików do ochrony w zależności od wybranej polityki bezpieczeństwa.

W rozdziale czwartym przedstawiono zaimplementowany na podstawie opracowanego algorytmu system ochrony integralności plików o nazwie ICAR (*Integrity Checking And Restoring*). Zostały opisane założenia projektowe systemu bezpieczeństwa oraz problemy, jakie należało rozważyć przy jego tworzeniu. Wykazano w tym rozdziale, że zaprojektowany system może być wykorzystywany zarówno w systemach LiveCD, jak i służyć do zabezpieczania dowolnych dystrybucji systemu Linux zainstalowanych na dyskach twardych. Implementację systemu ICAR przeprowadzono w oparciu o dystrybucję cdlinux.pl, której autor rozprawy jest głównym twórcą. W dalszej części rozdziału zostały zaprezentowane testy wydajnościowe prototypowej implementacji, których analiza pozwala na stwierdzenie, że system ICAR nie powoduje znaczących spadków prędkości działania systemu operacyjnego. Testy wykazały również, że zależność czasu dostępu do plików od ich rozmiarów pozostaje liniowa. Na końcu rozdziału czwartego przedstawiono wyniki oceny wiarygodności systemu ICAR, które zostały przeprowadzone metodą Trust Case [38].

Rozdział piąty zawiera rozważania dotyczące dalszych możliwości rozwoju zaprojektowanego mechanizmu. Opisane zostały możliwości wykorzystania wirtualizacji do lepszego zabezpieczania systemów operacyjnych z wykorzystaniem opracowanego przez autora modelu bezpieczeństwa.

Stworzony w ramach badań prototyp systemu ICAR, po przeprowadzeniu testów, został udostępniony na licencji GPL (*GNU Public Licence*) i może być pobrany przez zainteresowanych ze strony projektu cdlinux.pl, spod adresu http://www.cdliux.pl/icar. System może być wykorzystany przez administratorów systemów operacyjnych do zapewniania bezpieczeństwa serwerów, którymi administrują. Powszechne wykorzystanie systemu ICAR będzie stanowiło dowód walidacji modelu i mechanizmu zabezpieczeń stworzonego w ramach niniejszej pracy.

Rozdział 2.
Bezpieczeństwo systemów operacyjnych

Systemy komputerowe składają się z dwóch nierozłącznych części: sprzętu elektronicznego (ang. *hardware*) i oprogramowania (ang. *software*). Współdziałanie tych dwóch części jest możliwie dzięki systemowi operacyjnemu. Bez jego obecności komputer nie jest w stanie poprawnie funkcjonować. Współcześnie dostępnych jest wiele systemów operacyjnych, z których najpopularniejszymi są systemy z rodziny Windows firmy Microsoft, MacOS firmy Apple i Linux.

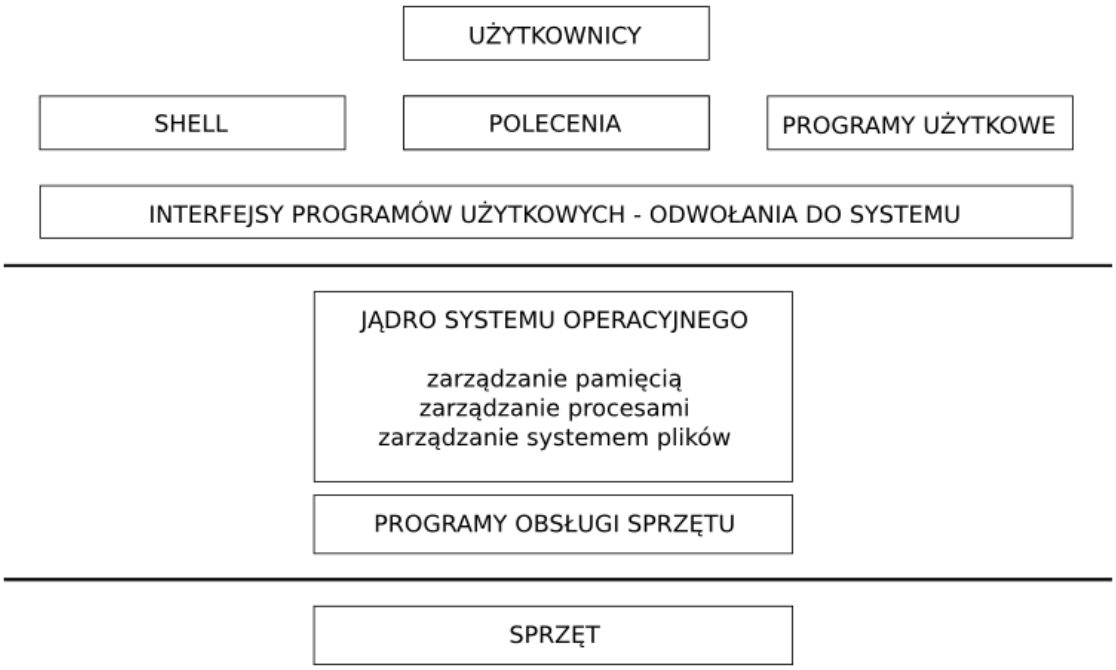
Według zbiorczej definicji A. C. Shaw [100] system operacyjny jest to zorganizowany zespół programów (systemów), które pełnią rolę pośredniczącą pomiędzy sprzętem a użytkownikiem, dostarczają użytkownikowi środków ułatwiających projektowanie, kodowanie, uruchamianie oraz eksploatację programów przetwarzających dane. W tym samym czasie system operacyjny steruje przydziałem zasobów sprzętowych i programowych dla zapewnienia efektywnego działania komputera. Definicja komercyjna mówi natomiast, że system operacyjny jest podstawowym oprogramowaniem dostarczanym przez producenta sprzętu komputerowego. Spośród kilku innych definicji jedna z nich podkreśla, że system operacyjny jest tylko oprogramowaniem jądra, ponieważ w rzeczywistości jest to najważniejsza część zbioru programów zarządzających komputerem. Z punktu widzenia użytkowników systemów komputerowych system operacyjny jest interfejsem, graficznym czy też tekstowym, za pomocą którego można uruchamiać programy na dowolnym, często znacznie różniącym się sprzęcie komputerowym.

Podkreślenia wymaga fakt, że system operacyjny należy do grona największych współczesnych programów komputerowych. Rozmiar nowoczesnych systemów operacyjnych jest liczony w milionach wierszy kodu (SLOC, ang. *Source lines of code*). Dla przykładu kod źródłowy systemu operacyjnego Windows XP firmy Microsoft składa się z około 40 milionów wierszy. Przy pracach nad tą wersją systemu Windows brało udział prawie 2 tysiące programistów [78]. Podobne rozmiary mają inne alternatywne systemy i tak np. najważniejsza część systemu operacyjnego Linux, jego jądro w wersji 2.6.8 składa się z ponad 4 milionów wierszy kodu [1].

Współcześnie różnice między popularnymi systemami operacyjnymi oprócz szczegółów implementacyjnych dotyczą tak naprawdę głównie interfejsów i wybranej funkcjonalności ocenianej z punktu widzenia użytkowników. Natomiast podstawowe mechanizmy w nich zawarte są bardzo podobne. Wszędzie występują programy do szeregowania zadań, do zarządzania pamięcią, takie jak stronicowanie na żądanie lub wstępne stronicowanie, a dane zorganizowane są w systemy plików [108].

W pracy rozpatrywane będą pewne problemy związane z zapewnianiem bezpieczeństwa systemu operacyjnego w ogólności. Natomiast implementacja opracowanych w pracy mechanizmów zostanie przeprowadzona na przykładzie systemu operacyjnego Linux. Najważniejszym powodem dla którego wybrano ten system operacyjny do wdrożenia i sprawdzenia skuteczności opracowywanych mechanizmów jest fakt, że kod systemu operacyjnego Linux jest znany i można go dowolnie modyfikować. Jądro systemu Linux zostało stworzone i jest rozwijane na zasadach Open Source, które pozwalają m. in. na prowadzenie prac eksperymentalnych nad nowymi mechanizmami, takimi jak te opracowywane w ramach prac badawczych autora. Dostępność kodu źródłowego jądra systemu Linux pozwala na dowolną jego modyfikację, wdrożenie i przetestowanie stworzonych mechanizmów. Zastosowanie do badań systemu Linux nie oznacza, że opracowane w pracy mechanizmy nie mogą być stosowane w innych systemach operacyjnych.

System komputerowy można podzielić na trzy zasadnicze warstwy [26], które zostały przedstawione na rysunku 2.1.



Rysunek 2.1. Podział systemu komputerowego

Pierwszą warstwę stanowi zbiór programów, do których użytkownik ma bezpośredni dostęp. W ramach tej warstwy może on korzystać z interfejsów graficznych lub tekstowych do uruchamiania aplikacji oraz komunikować się z systemem operacyjnym i przekazywać mu swoje polecenia poprzez rozbudowany język powłoki. W warstwie tej użytkownik ma możliwość modyfikowania sposobu działania uruchamianych przez siebie programów i wydawania poleceń organizujących pracę systemu operacyjnego. Oczywiście najważniejszym zbiorem programów dostępnych dla użytkownika są tzw. programy użytkowe, ponieważ głównym zadaniem systemu operacyjnego jest uruchamianie aplikacji i przydzielanie im zasobów niezbędnych do poprawnego przetwarzania danych zgodnie z instrukcjami przekazanymi przez użytkownika.

Drugą warstwę systemu komputerowego stanowi jądro systemu operacyjnego. Jest to zbiór programów do zarządzania procesami, pamięcią i systemami plików. Wewnątrz jądra znajdują się również bardzo ważne programy, które pozwalają na współpracę systemu operacyjnego ze sprzętem (hardware) stanowiącym trzecią warstwę systemu komputerowego. W przedstawionym podziale sprzęt jest rozumiany jako zbiór wszystkich urządzeń elektronicznych wchodzących w skład komputera, takich jak procesor, pamięć, dyski twarde, monitory, itp. Pomiędzy warstwą sprzętu, a warstwą jądra systemu operacyjnego występują czasami pewne problemy wynikające z braku odpowiednich sterowników dla urządzeń niedawno wyprodukowanych lub jeszcze mało popularnych. Ponieważ rozwój sprzętu komputerowego jest niezwykle dynamiczny, powoduje to konieczność częstych modyfikacji jądra systemu operacyjnego lub tworzenia nowych sterowników do obsługi urządzeń. Jest to szczególnie uciążliwe dla mniej popularnych systemów operacyjnych, takich jak np. Linux, dla których producenci sprzętu nie zadają sobie trudu tworzenia sterowników. W takich przypadkach są one tworzone przez niezależnych programistów, którzy często pracują bez pełnej specyfikacji urządzeń z wykorzystaniem inżynierii wstecznej.

Celem niniejszej pracy było stworzenie mechanizmów, które pozwolą na poprawę bezpieczeństwa systemu komputerowego. Żeby tego dokonać należało zidentyfikować zagrożenia, jakie występują na poszczególnych warstwach systemu operacyjnego. W niniejszej pracy rozpatrywane szczegółowo będą zagrożenia związane z celowymi atakami na system komputerowy. Nie będą natomiast rozpatrywane problemy związane z błędami generowanymi przez użytkowników lub przez sterowniki współdziałające z urządzeniami wejścia-wyjścia, a także błędy związane ze złym zaprojektowaniem programów użytkowych lub ich implementacją.

2.1. Rodzaje zagrożeń bezpieczeństwa w systemach operacyjnych

Na każdym z trzech poziomów, na jaki można podzielić system komputerowy, występują inne zagrożenia dla jego bezpieczeństwa. Najwięcej zagrożeń występuje na poziomie najwyższym. Programy użytkowe zazwyczaj pochodzą od różnych dostawców i w związku z tym mogą być różnej jakości. Błędy znajdujące się w aplikacjach mogą powodować ich niestabilną pracą, a w szczególnych przypadkach mogą prowadzić do wadliwego działania całego systemu operacyjnego. W niektórych sytuacjach mogą również umożliwiać nieautoryzowane przejęcie kontroli nad całym systemem operacyjnym, np. poprzez uzyskanie uprawnień administratora systemu. Programy użytkowe są również najczęściej atakowane przez intruzów. Ataki tego typu polegają między innymi na infekowaniu plików wirusami, instalacji programów typu koń trojański lub backdoor, które umożliwiają zdalne sterowanie systemem komputerowym.

2.1.1. Zagrożenia bezpieczeństwa w warstwie poleceń i programów użytkowych

Warstwa poleceń składa się z trzech najważniejszych elementów: z programów użytkowych, poleceń pozwalających na modyfikowanie działania systemu operacyjnego oraz z tak zwanej powłoki (ang. *shell*). Powłoka jest to niezwykle ważny program, tekstowy lub graficzny, umożliwiający użytkownikowi bezpośrednią ingerencję w działanie systemu operacyjnego. Od wielu lat toczy się nierozstrzygnięta dyskusja, czy powłoka jest częścią jądra systemu operacyjnego, czy jest programem użytkowym. Z punktu widzenia bezpieczeństwa systemów

komputerowych należy ją umieścić w jednej grupie z programami użytkowymi, a nie z jądrem systemu operacyjnego. Większość systemów operacyjnych posiada szeroki zbiór poleceń, które pozwalają użytkownikowi na zmianę działania systemu operacyjnego. Na przykład w systemach operacyjnych typu UNIX liczba poleceń powłoki może sięgać kilku tysięcy. Ponieważ system operacyjny Linux na początku był tworzony jako otwarta implementacja systemu UNIX dla komputerów PC, bazujących na architekturze i386, dlatego większość poleceń powłoki jest wspólna dla obu tych systemów. Podobieństwo pomiędzy programami powłoki polega na tym, że ich nazwy są takie same, funkcjonalność ich działania bardzo podobna, ale w zależności od producentów sam kod źródłowy programów może być różny. Niemniej z punktu widzenia użytkownika szczegóły implementacyjne nie mają znaczenia, dopóki wszystkie te programy działają w podobny sposób.

Polecenia powłoki można również zamiennie nazywać programami, gdyż tak naprawdę powłoka jest zbiorem programów, których skompilowane pliki wykonywalne są umieszczane w ściśle zdefiniowanej strukturze drzewa katalogów. Za pomocą tych programów użytkownik może bardzo skutecznie zarządzać większością funkcji wykonywanych przez systemem operacyjny. Dlatego zagrożenia związane z błędnym działaniem powłoki są bardzo duże i muszą być brane pod uwagę przy zapewnianiu bezpieczeństwa systemów operacyjnych. Jeżeli poszczególne programy tworzące środowisko powłoki, które zarządzają systemem operacyjnym, często na niskim poziomie, będą miały błędy, mogą one zostać wykorzystane przez intruza np. do przejęcia kontroli nad kontem administratora. Jest to jedna z najgroźniejszych sytuacji jaka może wystąpić z punktu widzenia bezpieczeństwa systemu operacyjnego. Przejęcie kontroli nad kontem administratora daje intruzowi praktycznie nieograniczone możliwości, nie tylko modyfikacji programów oraz plików systemowych i użytkowników, ale również pozwala na całkowity dostęp do zasobów sprzętowych, z których składa się komputer. Pozwala także na wydawanie poleceń zmieniających uprawnienia użytkowników, czy pozwalających na modyfikację konfiguracji systemu komputerowego.

Do pierwszej warstwy systemu komputerowego należy również zaliczyć wszystkie programy odpowiedzialne za autoryzację użytkowników oraz przydzielanie im dostępu do danych i programów. W przypadku błędów w implementacji np. programu odpowiedzialnego za zmianę haseł użytkowników może dojść do sytuacji, że przy odpowiednim wywołaniu polecenia, intruz będzie mógł zmienić nie tylko swoje hasło, ale również dowolnego innego użytkownika w systemie. Przejęcie kontroli nad systemem operacyjnym sprowadza się przede wszystkim do uzyskania możliwości dowolnej modyfikacji plików zapisanych w systemie operacyjnym. Po uzyskaniu takiego przywileju intruz może odczytywać prywatne dane użytkowników, a także usuwać dowolne pliki i katalogi, co w konsekwencji może prowadzić do trwałego uszkodzenia systemu operacyjnego. Atakujący jest również w stanie instalować dowolne wrogie programy typu backdoor, które pozwolą mu na późniejsze uzyskanie dostępu do systemu z ominięciem procedur zabezpieczeń [102].

Jak wykazały liczne, udokumentowane przypadki włamań do systemów operacyjnych głównym sposobem przejmowania kontroli nad systemem komputerowym są błędy w programach użytkowych, takich jak np. przeglądarki internetowe. W dobie powszechnego dostępu do sieci Internet, wykorzystując występujące w programach komputerowych błędy, zwane czasami dziurami (ang. *bugs*), można dokonywać ataków na systemy komputerowe w skali masowej, a wręcz globalnej. Błędami w programach użytkowych, które są najczęściej wykorzystywane do ataków na system komputerowy są tak zwane błędy przepełnienia bufora

(ang. *buffer overflow vulnerabilities*). Atak tego typu polega na wprowadzeniu do pamięci wykonywanego procesu wrogiego zbioru poleceń, a następnie wymuszenie wykonania skoku do wprowadzonego kodu. Wewnątrz tego wrogiego kodu znajdują się najczęściej polecenia umożliwiające przejęcie kontroli nad zaatakowaną maszyną [23]. Oznacza to, że każdy proces działający w systemie operacyjnym z prawami administratora może stanowić zagrożenie dla tego systemu i prowadzić do uzyskania przez intruza niepożądanych przywilejów. Błędy tego typu pojawiają się co jakiś czas praktycznie we wszystkich, również tych popularnych, programach komputerowych. Obecnie są również często używane do zdalnych ataków na serwisy internetowe.

Zagrożenia występujące w warstwie poleceń i programów użytkowych są praktycznie niemożliwe do wyeliminowania z uwagi na ogromną złożoność współczesnych aplikacji. Rozmiary kodów źródłowych niektórych programów przekraczają dziesiątki milionów wierszy, a pomimo zastosowania nawet najlepszych metod inżynierii oprogramowania nie można stworzyć oprogramowania pozbawionego całkowicie błędów. Nie można również całkowicie zabezpieczyć programów użytkowych, z uwagi na ogromną kreatywność ludzi zajmujących się atakami na systemy komputerowe, których kompetencje w zakresie znajomości współczesnych technologii informatycznych są często niezwykle wysokie. Z tych względów zabezpieczanie systemów operacyjnych musi odbywać się w inny sposób niż eliminowanie błędów w programach użytkowych, np. na poziomie jądra systemu operacyjnego, a nie na poziomie użytkowym.

2.1.2. Zagrożenia bezpieczeństwa w warstwie jądra systemu operacyjnego

Analizując największe możliwe zagrożenia bezpieczeństwa, jakie mogą wystąpić w warstwie jądra systemu operacyjnego, można wyróżnić przede wszystkim problemy związane z zapewnieniem prawidłowego zarządzania uruchomionymi w systemie procesami oraz ochroną obszarów pamięci operacyjnej przydzielanej poszczególnym procesom. Ponieważ współczesne systemy operacyjne działają w trybie wieloprocesowym i wielożytkownikowym, z punktu widzenia analizy bezpieczeństwa czynnikiem krytycznym jest konieczność całkowitej separacji obszarów pamięci przydzielonych poszczególnym procesom. Niedopuszczalna jest sytuacja, kiedy jeden proces może uzyskać dostęp do danych innego procesu przechowywanych w pamięci operacyjnej. Niestety, pomimo ciągłego rozwoju systemów operacyjnych, co jakiś czas odkrywane są błędy na poziomie jądra, które umożliwiają dokonywanie takich operacji.

Najgroźniejsze są błędy, które pozwalają nie tylko na odczyt danych procesów zapisanych w pamięci operacyjnej, ale również na ich modyfikację. Takie błędy są najczęściej wykorzystywane przez programy atakujące systemy operacyjne, powszechnie określane terminem *exploit*. Wykorzystując tego typu niedoskonałości w budowie systemów operacyjnych intruz może stworzyć program, który umożliwi mu uzyskanie przywilejów administratora, a tym samym przejęcie całkowitej kontroli nad systemem operacyjnym [75].

Co ciekawe, zdarza się, że takie błędy występowały niezauważone w popularnych systemach operacyjnych przez wiele lat. W lutym 2008 roku został wykryty poważny błąd w systemie operacyjnym Linux, który dotyczył wszystkich wersji jądra tego systemu począwszy od wersji 2.6.17, która została wydana w czerwcu 2006 roku. Oznacza to, że był on obecny w systemie Linux przez prawie dwa lata. Błąd ten umożliwiał modyfikację dowolnych obszarów

pamięci operacyjnej. Specjalnie przygotowany program pozwalał na uzyskanie praw administratora dzięki wykorzystaniu błędu w obsłudze funkcji *vmsplice*. Uruchomienie programu skutkowało przyznaniem każdemu użytkownikowi systemu operacyjnego pełnych praw administratora [138]. Poprawiona wersja jądra systemu Linux, która wyeliminowała ten błąd, została udostępniona w dwa dni po opublikowania informacji o zagrożeniu. Oznacza to, że usunięcie nawet tak poważnych błędów z kodu źródłowego nie jest problemem, w przeciwieństwie do ich odpowiednio wczesnego wykrywania. Nawet fakt, że kod źródłowy jądra systemu operacyjnego Linux jest publicznie dostępny i eksperymentuje na nim wielu ludzi, nie ułatwił szybkiego znalezienia tak poważnego błędu.

Innym typem błędów jaki występuje na poziomie jądra systemu operacyjnego jest niewłaściwa obsługa systemów plików. Wykorzystując błędy, które polegają na niewłaściwym zarządzaniu plikami intruz może uzyskać prawa odczytu lub nawet zapisu danych, do których w żadnym wypadku nie powinien mieć dostępu. W takiej sytuacji może dojść do kradzieży poufnych danych, ich usunięcia, a nawet całkowitego zniszczenia systemu plików.

Błędy znajdujące się na poziomie jądra różnych systemów operacyjnych, wykrywane są stosunkowo rzadko, zwykle co kilka miesięcy. Część z nich jest bardzo poważna, tak jak opisany powyżej błąd dostępu do pamięci, część może w wyniku dodatkowych agresywnych działań intruza doprowadzić do utraty danych, a część jest nieistotna z punktu widzenia bezpieczeństwa systemów operacyjnych. Analiza raportów błędów wykrywanych w systemach operacyjnych prowadzi do wniosku, że nie ma możliwości zaimplementowania tak dużego programu, jakim jest jądro, który byłby całkowicie pozbawiony błędów. W wyniku ciągłego rozwoju, zwłaszcza sprzętu komputerowego jądro musi być ciągle rozwijane, co oczywiście poprawia jego wydajność, umożliwia obsługę nowych urządzeń, zwiększa jego funkcjonalność, ale ciągła modyfikacja i rozbudowa kodu źródłowego jądra systemu operacyjnego z zasady niesie za sobą zagrożenie pojawiania się nowych błędów.

O ile atak intruza na jądro systemu operacyjnego jest znacznie trudniejszy do realizacji niż ma to miejsce w przypadku programów użytkowych, jednak negatywne skutki takich włamań są o wiele poważniejsze. Na poziomie jądra systemu operacyjnego również mogą rezydować wirusy lub programy typu backdoor. Najczęściej taka sytuacja ma miejsce przy instalacji sterowników nieznanego pochodzenie lub aktualizacji jądra z niepewnych źródeł.

2.1.3. Zagrożenia bezpieczeństwa w warstwie sprzętu

Sprzęt znajdujący się na ostatnim poziomie systemu komputerowego zasadniczo nie jest wykorzystywany do uzyskiwania dostępu do systemu komputerowego przez intruza. Zagrożenia bezpieczeństwa związane ze sprzętem dotyczą przede wszystkim jego niezawodności. Przykładowo fizyczne uszkodzenia nośników danych, takich jak dyski twarde, mogą prowadzić do trwałej utraty istotnych danych zapisanych w systemie komputerowym.

Największym problemem, jaki generuje dynamiczny wzrost liczby urządzeń wchodzących w skład systemów komputerowych, jest ich różnorodność oraz ciągły rozwój technologiczny. Pojawianie się nowych urządzeń wymusza na programistach jądra systemu operacyjnego wprowadzanie nowych sterowników oraz ciągłe rozwijanie modułów odpowiedzialnych za

obsługę sprzętu [108]. Jest to szczególnie widoczne w systemach przeznaczonych dla komputerów osobistych, w których użytkownicy mają niezwykle szerokie możliwości dostosowywania konfiguracji sprzętowych do własnych potrzeb.

W systemach komputerowych o dużym znaczeniu strategicznym polityka bezpieczeństwa zakłada brak aktualizacji komponentów sprzętowych bez wyraźnej potrzeby właśnie po to, żeby nie było konieczności aktualizacji jądra, bądź instalacji niesprawdzonych sterowników urządzeń. Zastosowanie jednej konfiguracji sprzętowej przez cały cykl życia systemu komputerowego jest jednym ze sposobów podnoszenia jego bezpieczeństwa.

Znane są jednak, wprawdzie nieliczne, ale występujące przypadki, że to sprzęt komputerowy umożliwia przełamanie zabezpieczeń systemu. Jako przykłady takich zdarzeń mogą posłużyć dwa błędy wykryte w 2008 roku. Pierwszy pozwala na dostęp do chronionych obszarów pamięci komputera wykorzystując złącze FireWire (IEEE-1394). Jako dowód (ang. *proof-of-concept*) wykrytej dziury został stworzony specjalny program komputerowy, który wykorzystując specyfikę protokołu FireWire pozwala na zalogowanie się na konto administratora na dowolnym komputerze z systemem MS Windows XP za pośrednictwem drugiego komputera podłączonego przez złącze IEEE-1394. Autor tego programu twierdzi, że wykorzystując tę metodę można włamywać się nie tylko do systemów operacyjnych z rodziny Windows, ale również do każdego innego systemu operacyjnego [124]. Jest to związane ze specyfiką urządzeń FireWire, które do szybszego działania wymagają bezpośredniego dostępu do pamięci operacyjnej, gdy tymczasem taki dostęp jest zawsze zagrożeniem dla bezpieczeństwa systemów komputerowych [88]. Drugi opisany w literaturze przykład dotyczy zabezpieczenia danych poprzez przechowywanie ich na szyfrowanych partycjach dysku twardego. Zostało udowodnione, że możliwe jest odczytanie danych, w tym odczytanie wartości klucza szyfrującego bezpośrednio z kości pamięci. Współczesne pamięci RAM tylko teoretycznie tracą dane od razu po zaniku zasilania, a faktycznie taki proces w temperaturze pokojowej trwa nawet do kilkunastu sekund. Obniżając temperaturę kości pamięci, czas jej czyszczenia zwiększa się nawet do kilku godzin w bardzo niskich temperaturach. W tym czasie można swobodnie odczytać wszystkie dane przechowywane w pamięci w momencie wyjęcia kości pamięci z komputera [46].

Awarie fizycznych elementów systemu komputerowego nie prowadzą zwykle do nieodwracalnych strat i nie są groźne dla użytkownika, z wyjątkiem systemów wymagających ciągłej pracy, jak np. sieciowych serwerów aplikacji. Najgroźniejsze w skutkach mogą być uszkodzenia nośników danych, takich jak dyski twarde, które mogą prowadzić do utraty plików. Jeśli jednak prowadzona jest rozsądna polityka wykonywania kopii bezpieczeństwa, groźba całkowitej utraty plików jest niska.

Na podstawie przeprowadzonej dotychczas analizy można wywnioskować, że na każdym poziomie, na jaki można podzielić system operacyjny, czyli zarówno na poziomie użytkownika, na poziomie jądra, jak i na poziomie sprzętu mogą występować błędy zmniejszające bezpieczeństwo systemu komputerowego. Ciągły rozwój zarówno programów użytkowych, jądra systemu operacyjnego, a także sprzętu komputerowego, nie tylko zwiększa funkcjonalność, ale może również prowadzić do pojawiania się nowych błędów zagrażających bezpieczeństwu. Należy zatem stwierdzić, że istnieje ciągłe zagrożenie bezpieczeństwa systemów komputerowych.

Znaczna ilość błędów, jakie występują w systemach operacyjnych, jak i innych produktach informatycznych, wynika z dużej złożoności kodów źródłowych tych programów, jak również z częstych zmian funkcjonalnych. Każde dodanie nowej funkcjonalności do programu niesie za sobą ryzyko wprowadzenia do kodu nowych błędów. Należy zwrócić uwagę, że również kod poprawiający błędy czy wykryte usterki może generować nowe błędy. Często zdarza się, że usunięty błąd w kolejnych wersjach ponownie pojawia się w programie, z uwagi na problemy z zarządzaniem kodem dużych projektów tworzonym przez wielu ludzi.

Jedyną stosunkowo skuteczną metodą pozbawienia programów komputerowych luk związanych z bezpieczeństwem jest zaprzestanie wprowadzania nowej, często zbędnej funkcjonalności, a poprawianie jedynie wykrytych błędów. Taki cykl wytwarzania co prawda nie gwarantuje, że programy będą całkowicie pozbawione błędów, ale może znacznie zminimalizować ryzyko pojawiania się nowych. Jednym z większych projektów, który stosuje cykl wytwórczy polegający na całkowitym zamrożeniu rozwoju programu jest dystrybucja systemu operacyjnego Linux o nazwie Debian. Po dokładnym przetestowaniu wersji testowej i wprowadzeniu koniecznych poprawek podejmowana jest decyzja o wydaniu wersji stabilniej dystrybucji. Taki proces stabilizacji może trwać nawet kilka lat. Od tego momentu udostępniane są nowe wersje pakietów jedynie dla programów, w których zostały wykryte błędy o krytycznym znaczeniu. Z tego powodu Debian jest uważany za jeden z najstabilniejszych i najbezpieczniejszych systemów operacyjnych. Wadą takiego podejścia do wytwarzania oprogramowania jest niebezpieczeństwo nienadążania za rozwojem zarówno programów użytkowych jak i sprzętu komputerowego. W niektórych zastosowaniach korzystanie z przestarzałego oprogramowania, pozbawionego nowych funkcji jest niemożliwe. Z tego powodu projekt Debian udostępnia oprócz wersji stabilnej i testowej, wersję niestabilną przeznaczoną do instalacji w obszarach gdzie istotniejsze od bezpieczeństwa jest wykorzystywanie najnowszych wersji programów.

2.2. Cele ataków na systemy operacyjne

Współcześnie zdecydowana większość informacji wykorzystywanych przez ludzi jest przetwarzana przez systemy komputerowe i przechowywana w formie cyfrowej. Mamy również do czynienia z digitalizacją informacji, czyli przetwarzaniem tradycyjnych, papierowych nośników danych, takich jak książki czy dokumenty do postaci cyfrowej. Dane zapisane w postaci plików komputerowych są obecnie wykorzystywane powszechnie w wielu dziedzinach, zarówno przy prowadzeniu działalności gospodarczej, naukowej, technicznej, jak również w życiu codziennym. Firmy prowadzące działalność biznesową często wykorzystują systemy komputerowe do usprawnienia prac wewnątrz przedsiębiorstwa, a także do kontaktów z klientami i kontrahentami za pośrednictwem sieci Internet. W systemach komputerowych są przechowywane i przetwarzane, często bardzo ważne z punktu widzenia firmy, dane biznesowe, księgowe, finansowe, jak również plany dotyczące działalności gospodarczej, czy projekty techniczne, takie jak np. projekty samochodów stworzone całkowicie za pomocą oprogramowania komputerowego. Również osoby prywatne coraz częściej wykorzystują komputery w życiu codziennym np. do prowadzenia korespondencji czy robienia zakupów w wirtualnych sklepach. Skalę tego zjawiska można przedstawić na podstawie największej polskiej platformy aukcyjnej, *Allegro.pl*, na której w 2009 roku przeprowadzono transakcje na łączną kwotę około 6 mld złotych. Natomiast łączna wielkość transakcji konsumenckich przeprowadzonych drogą elektroniczną w tym samym

roku wyniosła ponad 13 mld złotych [132]. Komputery są coraz częściej wykorzystywane również jako centra multimedialne, na których przechowywane są dane o bardzo dużej wartości sentymentalnej z punktu widzenia użytkownika, jak np. prywatne zdjęcia czy filmy. Coraz większą popularność zdobywają obecnie usługi finansowe prowadzone za pośrednictwem sieci Internet, wśród których można wymienić bankowość elektroniczną. Z usług największego polskiego banku elektronicznego – *mBanku*, który prowadzi działalność prawie wyłącznie za pośrednictwem sieci, na początku 2008 roku korzystało ponad 1,5 miliona osób [131].

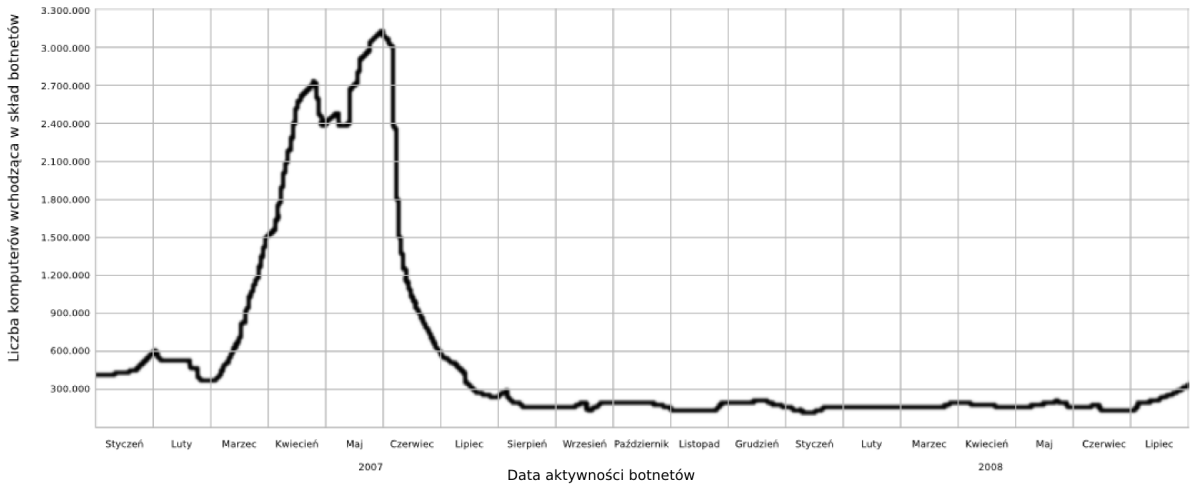
Tak duża skala zastosowania systemów komputerowych i cyfrowego przetwarzania informacji zwiększa znaczenie bezpieczeństwa i ochrony danych. Dla przykładu kradzież haseł dostępu do kont bankowych może mieć poważne konsekwencje dla użytkownika, łącznie ze stratą wszystkich środków finansowych zgromadzonych na jego rachunkach. Również usunięcie bądź zmiana danych zapisanych w systemie komputerowym, np. w bazie danych banku, może prowadzić do utraty pieniędzy księgowanych w sposób elektroniczny. Taka sytuacja może zostać spowodowana błędnym działaniem systemów informatycznych, jak również może wynikać z celowego działania intruza. W styczniu 2008 roku wystąpił błąd w programie współpracy systemu komputerowego *mBanku* z serwisem kart płatniczych *Visa*, który doprowadził do wielokrotnego księgowania tych samych operacji, co natomiast powodowało utratę środków płatniczych na rachunków bankowych klientów [133].

Jest oczywiste, że cyfryzacja danych jest procesem nieodwracalnym i implikuje konieczność stworzenia nowych, skutecznych mechanizmów zapewniania bezpieczeństwa danych cyfrowych. Pomimo licznych zalet przetwarzania i przechowywania danych w postaci cyfrowej, technologia ta ma również pewne wady. Dane cyfrowe mogą być, zgodnie ze swoją naturą, w bardzo prosty sposób rozpowszechniane, jak również kasowane i modyfikowane w niezauważalny dla użytkowników systemów komputerowych sposób. W przypadku danych przechowywanych na nośnikach tradycyjnych, np. papierowych, każda zmiana pozostawiała za sobą ślad, dzięki czemu można było ją zauważyć, zidentyfikować, a często również przywrócić. Natomiast po zmianie danych cyfrowych przez intruza często trudno jest zauważyć kiedy, kto i gdzie dokonał takiej zmiany. Konsekwencje takiego działania mogą być poważne, łącznie z wymiernymi stratami finansowymi, np. w przypadku modyfikacji danych finansowych w systemach komputerowych banków. Niestety włamania do systemów komputerowych są bardzo częste, a to przekłada się na wymierne straty finansowe, niejednokrotnie sięgające milionów dolarów.

Nasuwa się pytanie, dlaczego ludzie dokonują włamań do systemów komputerowych. Prawdopodobnie istnieje wiele różnych motywów. Jednym z nich jest chęć kradzieży poufnych danych, takich jak tajemnice firmowe, czy hasła dostępu do usług finansowych. Tego typu działalność prowadzi do wymiernych korzyści finansowych osób, którym udało się dokonać takiego włamania. Do tego typu procederu często wykorzystywane są specjalnie przygotowane programy, takie jak konie trojańskie. Są to programy powszechnie dostępne w Internecie, które oprócz funkcji pożądanych przez użytkownika zawierają złośliwy kod umożliwiający przejęcie przez intruza kontroli nad komputerem [102]. Często komputery, na które został przeprowadzony skuteczny atak, lub które zostały pomyślnie zainfekowane odpowiednimi wirusami i końmi trojańskimi grupowane są w tzw. botnety. Są to zbiory komputerów, które mogą być zdalnie kontrolowane przez intruza i działać jednocześnie po wydaniu odpowiedniego polecenia za pośrednictwem sieci Internet [51]. W 2007 roku doszło do skutecznego zainfekowania pokaźnej grupy komputerów na całym świecie. Dokonano tego

za pomocą załączników dołączanych do wiadomości email lub poprzez specjalnie spreparowane strony WWW. Stworzony w ten sposób system botnet otrzymał nazwę *Storm*. W szczytowej fazie jego działalności szacowano, że należało do niego od 1 do 50 milionów komputerów na całym świecie [22]. Ta rozbieżność w szacowanej liczbie przejętych przez przestępców komputerów wynika z zastosowania różnych metod badań tego typu zjawisk przez różne instytucje i różnych badaczy.

Jednym z ciekawszych narzędzi służących do szacowania ilości ataków w sieci Internet jest tzw. *honeypot*, czyli w wolnym tłumaczeniu „lep na włamywaczy". Honeypot jest to system bezpieczeństwa, którego zadanie polega na umożliwieniu włamywaczom badania systemu, jego atakowania, a nawet na przełamywaniu zabezpieczeń [107]. Na podstawie zgromadzonych danych można wykrywać nowe techniki włamań, identyfikować intruzów, czy oszacować liczbę ataków w sieci lokalnej bądź globalnej. W praktyce honeypot najczęściej jest wydzielonym komputerem, często pozbawionym zaawansowanych zabezpieczeń lub celowo wyposażonym w znane luki bezpieczeństwa. System taki ma bardzo dobrze opracowane mechanizmy zapamiętywania historii działań włamywaczy, gromadzi liczne szczegóły na temat ich aktywności i pozwala na dogłębną analizę przeprowadzanych ataków. Pomimo braku szczegółowych analiz naukowych przydatności i skuteczności narzędzi tego typu, honeypoty są często wykorzystywane w zastosowaniach komercyjnych [56]. Obecnie można zaobserwować znaczny rozwój tej technologii. Zostały stworzone liczne narzędzia, tzw. szkieletowe honeypoty (ang. *honeypot framework*), ułatwiające wykorzystanie takich systemów w sieciach lokalnych [106]. Powstają również całe grupy serwerów zwane farmami, które są wykorzystywane do badania aktywności włamywaczy w całej sieci Internet. Na podstawie zebranych w ten sposób danych publikowane są liczne statystyki oraz informacje o pojawiających się nowych rodzajach zagrożeń w sieci [91].



Rysunek 2.2. Liczba komputerów wchodząca w skład botnetów

Jedną z największych organizacji zajmujących się badaniem aktywności botnetów z wykorzystaniem technologii honeypot jest fundacja *Shadowserver* [139]. Organizacja ta opublikowała interesujące dane, pokazujące liczbę aktywnych komputerów wchodzących w skład różnych botnetów w okresie od początku 2007 roku do lipca 2008. Dane te zostały przedstawione na rysunku 2.2. Należy zastrzec, że przedstawione liczby dotyczą tylko tych sieci komputerowych, które są monitorowane przez komputery współpracujące z *Shadowserver Foundation* i nie uwzględniają danych z innych źródeł. Na przedstawionym

wykresie można zauważyć, że największa ilość komputerów wchodzących w skład botnetów była aktywna pomiędzy kwietniem a czerwcem 2007 roku. W tym okresie liczba komputerów, nad którymi przejęto kontrolę przekroczyła 3 miliony. Był to okres największego rozwoju botnetu *Storm*. W czerwcu liczba ta zaczęła spadać, głównie dzięki udostępnionej przez firmę Microsoft poprawce do systemu MS Windows, która usuwała wrogie oprogramowanie Storma z komputerów. Pomimo przeprowadzonych działań, w ciągu roku, od sierpnia 2007 do lipca 2008, liczba komputerów nad którymi intruzi utrzymywali kontrolę wahała się w granicach 300.000 maszyn.

Tak duży zbiór zainfekowanych komputerów, nad którym kontrolę sprawują przestępcy, jest często wykorzystywany do przeprowadzania innego typu ataków. Najczęściej przeprowadzane są ataki typu *DDoS* (ang. D*istributed Denial-of-Service*) na serwisy internetowe. Ataki takie polegają na wysyłaniu w krótkim czasie milionów zapytań do wybranego serwisu przez wszystkie komputery będące pod kontrolą atakującego [85]. Przeprowadzony na dużą skalę atak z wielu komputerów rozproszonych po całym świecie może skutecznie zawiesić działanie każdego serwisu internetowego [8]. Ataki *DDoS* są często wykorzystywane do szantażu właścicieli znanych stron internetowych. Możliwości obrony przed atakiem tego typu są bardzo ograniczone.

Inną często podejmowaną akcją, za pomocą komputerów wchodzących w skład botnetu, jest rozsyłanie reklam za pośrednictwem poczty elektronicznej, czyli tzw. spamu. Autorzy spamu zarabiają, gdy odbiorca wiadomości odwiedzi reklamowaną stronę. Oszuści mogą również zarabiać wykorzystując technikę, która otrzymała umowną nazwę „Nigeryjski szwindel". Polega ona na wyłudzeniu pieniędzy od nieświadomych użytkowników sieci. Oszuści wysyłają wiadomość pocztą elektroniczną, w której oferują otrzymanie części gigantycznego majątku zgromadzonego w jakimś egzotycznym kraju. Zadaniem odbiorcy jest pomoc w przelewie tych pieniędzy na rachunek w jednym z europejskich krajów. Pomoc taka miałaby polegać na wpłaceniu drobnych w porównaniu z oferowanymi profitami kwot na wskazane rachunki bankowe. Po dokonaniu kilku transakcji kontakt z przestępcami się urywa [104]. Pomimo że jest to ewidentny przykład niezbyt wyrafinowanego oszustwa, to wielu ludzi daje się nabrać na takie nielogiczne praktyki. Znane są również przypadki, że przestępcy rozsyłając za pośrednictwem botnetów setki milionów wiadomości email ze spreparowanymi informacjami o spółkach aukcyjnych doprowadzali do manipulacji ich kursami giełdowymi [47].

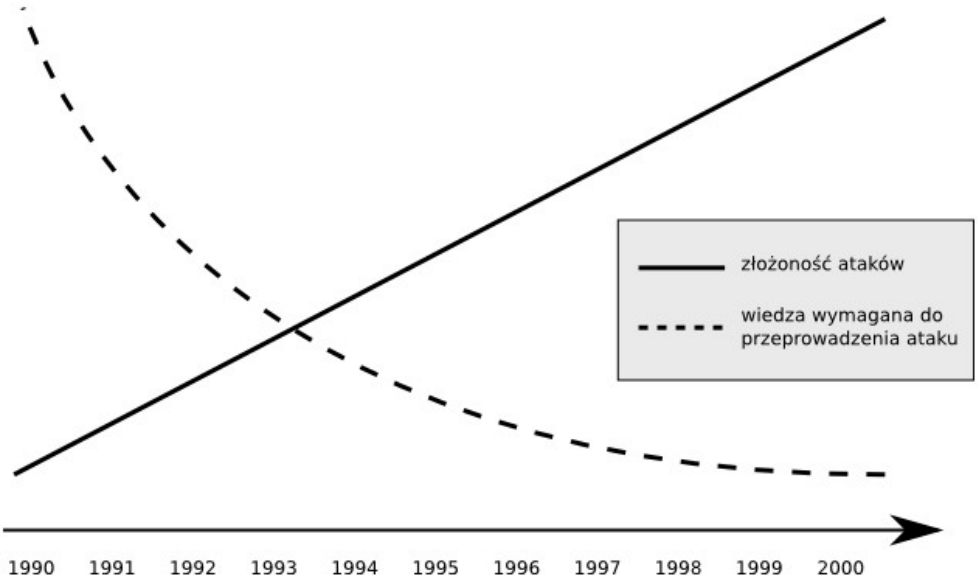
Należy stwierdzić, że oszustwa z wykorzystaniem spamu wysyłanego za pośrednictwem poczty elektronicznej to poważny i trudny do wyeliminowania problem wynikający z powszechnego wykorzystywania sieci Internet. Według różnych szacunków w chwili obecnej spam stanowi od 80 do 95% wszystkich wiadomości email, a liczba wysyłanych niechcianych wiadomości jest oceniana na około 100 miliardów dziennie. Sytuacja taka generuje nie tylko problemy prawne czy też techniczne, ale również stawia przed ośrodkami naukowymi zadania, by poszukiwać metod eliminacji tego typu zjawisk, które mają istotne znaczenie z praktycznego punktu widzenia. Najskuteczniejsze w chwili wydają się mechanizmy filtrowania wiadomości elektronicznych z wykorzystaniem sztucznej inteligencji [55].

Zapewnienie bezpieczeństwa informacji przesyłanych w sieci Internet sprowadza się głównie do zapewniania bezpieczeństwa serwisów internetowych, gdy tymczasem włamania do systemów komputerowych, na których działają serwisy internetowe są zjawiskiem powszechnym. Wraz ze zwiększaniem się liczby, zarówno serwisów, jak i internautów, ataki

tego typu są coraz częściej przeprowadzane. Czasami serwisy, na które został przeprowadzany skuteczny atak, są wykorzystywane do udostępnianie złośliwych programów (np. koni trojańskich). Odwiedzenie takiej strony może skutkować instalacją na niezabezpieczonych komputerach oprogramowania, które może być wykorzystane np. do kradzieży haseł dostępu do serwisów banków [116].

W ostatnim czasie popularne i zarazem bardzo niebezpieczne są ataki na serwisy internetowe, których celem jest tzw. kradzież tożsamości, czyli przeważnie uzyskanie danych osobowych klientów takiego serwisu. Tego typu ataki mogą mieć bardzo poważne skutki zarówno finansowe, jak i prawne, np. zdobyte hasła do serwisów aukcyjnych mogą służyć do dokonania oszustwa polegającego na oferowaniu nieistniejących towarów.

Niepokojący jest fakt, że z biegiem czasu, przy rosnącej liczbie komputerów podłączonych do sieci globalnej, rośnie złożoność zabezpieczeń komputerów, ale jednocześnie, paradoksalnie wiedza intruzów niezbędna do przeprowadzenia skutecznego ataku jest coraz niższa. Badania nad tego typu zależnościami zostały przeprowadzone w ośrodku CERT przy uniwersytecie Carnegie Mellon w Pittsburghu [81]. Uzyskane wyniki zostały zilustrowane na wykresie przedstawionym na rysunku 2.3.



Rysunek 2.3. Złożoność ataków w porównaniu do wymaganej wiedzy intruzów [81]

Przedstawiona zależność wynika z faktu, że powszechnie dostępne są narzędzia, które umożliwiają wykonywanie automatycznych ataków na wybrane komputery nawet przez osoby o niskich kwalifikacjach. W żargonie środowiska zajmującego się bezpieczeństwem systemów komputerowych osoby korzystające z takich narzędzi nazywane są pogardliwie *script-kiddies*, co w wolnym tłumaczeniu oznacza dzieci korzystające z gotowych skryptów [7].

Analizując podłoże udanych ataków na systemy komputerowe można dojść do wniosku, że jedną z głównych przyczyn ich skuteczności jest niska jakość wytwarzanego oprogramowania. W dziedzinie inżynierii oprogramowania występuje obecnie znane zjawisko zwane kryzysem jakości. Ponadto około 80% projektów informatycznych kończy się fiaskiem bądź to z powodu niedotrzymania założonych terminów lub znacznego przekroczenia

zaplanowanych budżetów. Należy też stwierdzić, że błędy w programach użytkowych były, są i będą, ponieważ nie ma możliwości stworzenia dużego programu komputerowego całkowicie ich pozbawionego. Również w kodzie systemów operacyjnych występują często poważne dziury, które mogą w nim występować przez wiele lat zanim zostaną wykryte i poprawione.

Istniejące błędy ułatwiają ataki na systemy komputerowe, zwłaszcza gdy zostaną wykryte przez osoby działające z niskich pobudek. Osoby zajmujące się wykrywaniem słabych punktów zabezpieczeń można podzielić w zależności od ich motywacji na dwie grupy. Istnieje pewna grupa osób, tzw. *Etyczni Hackerzy* (ang. *ethical hackers*), która zajmuje się wykrywaniem błędów zawodowo, pracując w specjalnie do tego celu powołanych jednostkach badawczych, czy też hobbystycznie. Osoby te po znalezieniu potencjalnej dziury zwykle powiadamiają producentów programów o zagrożeniach i miejscu ich występowania. Bardzo często są to osoby o najwyższych kompetencjach. Ich praca jest niezwykle cenna i przyczynia się do poprawy jakości oprogramowania, ponieważ wykryte błędy są z reguły szybko usuwane przez producentów. Jedną z najbardziej cenionych osób z tej grupy, zwanej czasami również „Białymi Kapeluszami" (ang. *white hats*), jest Polak Michał „lcamtuf" Zalewski, który wykrył m. in. błędy w protokołach SSH [136], TCP/IP [134], a obecnie koncentruje się na bezpieczeństwie przeglądarek internetowych pracując w firmie Google.

Motywacją działania innej grupy, tzw. Crackerów (ang. cracker, black hat) jest osiąganie określonych korzyści, najczęściej finansowych. Działalność tego typu osób na przestrzeni lat doprowadziła do ogromnych strat finansowych. Wprawdzie włamywanie się do systemów komputerowych jest obecnie przestępstwem i często osoby trudniące się tym ponoszą karę, łącznie z pozbawieniem wolności, lecz zjawisko to nadal istnieje, a wręcz się rozwija. W niektórych krajach sąd może wydać osobie skazanej za przestępstwo komputerowe zakaz korzystania z komputerów.

Jakość oprogramowania zależy głównie od poziomu jakości procesu wytwarzania oprogramowania. W tym aspekcie należy wyróżnić specjalną grupę programów komputerowych powstających w oparciu o zasady Open Source, które polegają głównie na udostępnianiu wraz z programem jego kodu źródłowego. W ten sposób tworzone i rozpowszechniane jest między innymi jądro systemu operacyjnego Linux. Dzięki powszechnej dostępności kodów źródłowych liczba osób, które mają możliwość testowania i kontrolowania tego typu programów jest bardzo duża. Dzięki temu proces znajdowania błędów, zwłaszcza takich, które ułatwiają ataki na systemy komputerowe, jest bardzo wydajny. O znalezionych błędach informowani są autorzy oraz społeczność zorganizowana w różne grupy dyskusyjne, a nawet proponowane są gotowe modyfikacje kodu, które eliminują błędy. Ponieważ liczba, ludzi którzy zajmują się kodowaniem programów Open Source jest bardzo duża i duże jest ich zaangażowanie, programy tego typu są coraz wyższej jakości. Często poziom jakości programów Open Source jest wyższy niż programów tworzonych w tradycyjny sposób przez firmy komputerowe.

Bardzo ciekawym i dokładnie analizowanym zjawiskiem socjologicznym i ekonomicznym są motywacje poszukiwania i wykrywania błędów, jak również motywacje dokonywania ataków na systemu komputerowe. Część ataków zarówno na systemy komputerowe osób prywatnych, jak i firm oraz na serwisy internetowe jest dokonywana bez złych intencji, np. jedynie w celu wykrycia błędów w oprogramowaniu lub po prostu w celu udowodnienia swojej wiedzy informatycznej. Jednak tego typu ataki, które w założeniu nie mają powodować szkód, również mogą prowadzić do groźnych sytuacji, np. włamywacz może, nawet przypadkowo

usunąć ważne dane lub doprowadzić do ich nieautoryzowanego opublikowania. Bardzo niebezpieczną motywacją, jaka często towarzyszy włamywaczom są różnego rodzaju działania destrukcyjne, które mogą polegać na usuwaniu plików czy podmienianiu ich zawartości.

Podsumowując, intruzów i włamywaczy można podzielić na tych, którzy czerpią zyski ze swojej działalności oraz na tych, którzy dokonują agresywnych działań z pobudek innych niż finansowe. Niestety, w ostatnich latach można zaobserwować znaczny wzrost ataków dokonywanych w złych intencjach, najczęściej z zamiarem uzyskania korzyści finansowych, pomimo że taka działalność jest przestępstwem.

Stworzenie mechanizmów zabezpieczających przed atakiem na systemy komputerowe to zagadnienie zarówno z naukowego, jak i technicznego punktu widzenia bardzo złożone, nad którym pracuje wiele ośrodków naukowych na całym świecie. Metody zabezpieczeń mogą iść w dwóch kierunkach – poprawy jakości wytwarzanego oprogramowania oraz stworzenia specjalnych mechanizmów uniemożliwiających dokonywanie ataków na system. W niniejszej pracy zostanie przedstawiony opracowany przez autora mechanizm zabezpieczania systemów operacyjnych przed modyfikacjami dokonywanymi przez osoby nieuprawnione.

2.3. Metody zabezpieczania systemów komputerowych

Bezpieczeństwo systemów informatycznych może być rozpatrywane w trzech aspektach: poufność (ang. *confidentiality*), integralność (ang. *integrity*) oraz dostępność (ang. *availability*). Poufność definiowana jest jako zabezpieczenie lub ukrycie informacji przed niepowołanym dostępem. Integralność odnosi się zazwyczaj do zabezpieczania danych przed niepowołaną modyfikacją, tak aby użytkownik miał zaufanie do informacji i źródła jej pochodzenia. Dostępność natomiast oznacza możliwość używania pożądanych zasobów zgodnie z ich przeznaczeniem [13].

W rozdziale tym zostaną przedstawione znane mechanizmy zabezpieczeń, pokrywające wszystkie trzy aspekty bezpieczeństwa. Analizując dotychczas znane mechanizmy zabezpieczania systemów komputerowych można wyróżnić trzy grupy. Do pierwszej należy zaliczyć mechanizmy filtrowania ruchu sieciowego, które pozwalają na blokowanie zdalnego dostępu do sieci lokalnych i komputerów. Tego typu mechanizmy mają szerokie zastosowanie z uwagi na duży udział włamań do systemów komputerowych poprzez sieć Internet. Do drugiej grupy należą mechanizmy pozwalające na ograniczanie szkód jakich może dokonać intruz po udanym włamaniu do systemu. Tego typu mechanizmy mają bardzo duży potencjał i dlatego są rozwijane w wielu ośrodkach naukowo-badawczych na całym świecie. W trzeciej grupie mechanizmów zabezpieczania można umieścić metody pozwalające na wykrywanie włamań oraz kontrolę plików pod kątem ich nieuprawnionych modyfikacji przez intruzów [61].

W tabeli 2.1. przedstawiono pokrycie aspektów bezpieczeństwa przez systemy wchodzące w zdefiniowane wyżej grupy mechanizmów. Dodatkowo, z grupy systemów wykrywania włamań zostały wyodrębnione systemy ochrony integralności plików, a z mechanizmów ograniczania skutków włamań, wirtualizacja.

Tabela 2.1. Podział mechanizmów bezpieczeństwa

Mechanizmy	Poufność	Integralność	Dostępność
Filtrowanie ruchu sieciowego	x		
Ograniczanie skutków włamań	x	x	
Wykrywanie włamań		x	
Ochrona integralności plików		x	x
Wirtualizacja			x

W dalszej części rozdziału przedstawione zostaną systemy, które zostały zaimplementowane w systemie operacyjnym Linux. System ten został wybrany do badań z uwagi na jawność jego kodu i możliwość dokonywania dowolnych modyfikacji przez każdego zainteresowanego. Jednak większość omawianych mechanizmów bezpieczeństwa jest lub może być zaimplementowana w dowolnych systemach operacyjnych.

2.3.1. Mechanizmy filtrowanie ruchu sieciowego

Bardzo skutecznym sposobem zapewnienia bezpieczeństwa systemu komputerowego jest całkowite odizolowanie go od sieci Internet. Taka praktyka jest z powodzeniem stosowana w wielu instytucjach np. finansowych i wojskowych. Pomimo oczywistych wad takiego rozwiązania i zmniejszenia funkcjonalności systemu, w pewnych przypadkach takie działania są sensowne i całkowicie uzasadnione. Oczywiście obecnie tylko niewielka część komputerów działa w ten sposób. Zdecydowana większość maszyn jest podłączona do sieci globalnej. Dotyczy do zarówno komputerów domowych, firmowych czy np. serwerów obliczeniowych.

Zamiast całkowitego odłączenia komputera od sieci Internet powszechnie stosuje się rozwiązanie mniej restrykcyjne, choć podobne w swoim działaniu. Filtruje się ruch sieciowy wychodzący i przychodzący do systemu komputerowego według określonych zasad. Narzędzia służące do tego celu noszą nazwę zapór sieciowych (ang. *firewall*). Zapora sieciowa jest to system lub grupa systemów, które kontrolują ruch pomiędzy dwoma sieciami komputerowymi i na podstawie zdefiniowanych reguł przepuszczają lub blokują połączenia [18]. W ten sposób systemy komputerowe chronione są przed niepowołanym dostępem do zasobów [110].

Zabezpieczenie typu firewall pozwala na filtrowanie ruchu sieciowego pomiędzy siecią globalną i lokalną. W ten sposób można ograniczyć dostęp do zasobów lub usług tylko dla wybranej grupy komputerów działających poza siecią lokalną. Istnieje również możliwość kontroli dostępu do komputerów zewnętrznych przez użytkowników znajdujących się wewnątrz sieci. Zapory sieciowe są również skutecznym sposobem na ukrywanie architektury sieci lokalnych, pozwalają na autoryzowanie użytkowników, a także na śledzenie ruchu sieciowego.

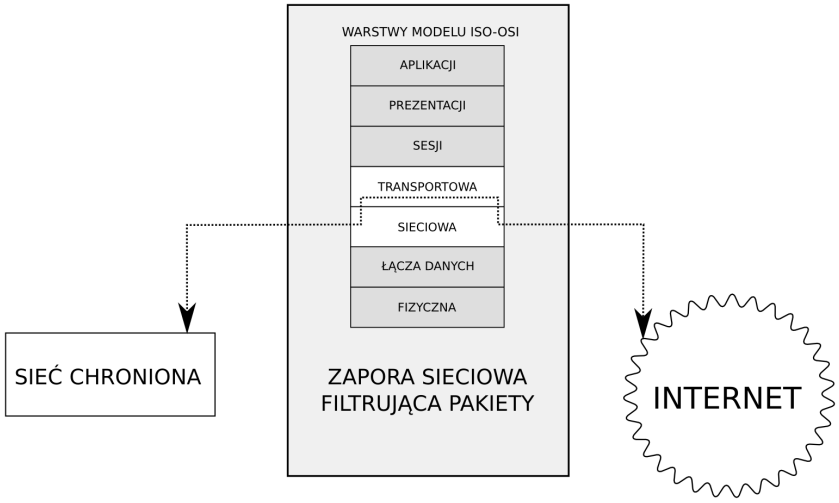
Funkcje zapory sieciowej pełnią specjalnie do tego celu stworzone programy, które mogą działać zarówno na dedykowanych urządzeniach, jak i na zwykłych komputerach. Poszczególne rozwiązania realizujące funkcje zapór sieciowych mogą się znacząco różnić w sposobie implementacji poszczególnych zasad filtrowania ruchu sieciowego.

Istnieją dwie powszechnie stosowane polityki filtrowania ruchu sieciowego. Definicja pierwszej mówi, że wszystko, co nie jest dozwolone, jest zabronione. Druga polityka opiera się na odwrotnym założeniu, czyli wszystko, co nie jest zabronione, jest dozwolone. Pomimo podobieństwa tych sformułowań działanie zapór sieciowych opartych o te dwie zasady jest zupełnie różne. Pierwsze podejście zakłada, że domyślnie blokowany jest cały ruch sieciowy. W ten sposób realizowana jest zasada całkowitego odcięcia systemu komputerowego od sieci Internet. Dopiero w dalszej kolejności tworzone są reguły, które pozwalają na ograniczenie tego bezwzględnego zakazu. W ten sposób odblokowywane są niektóre często używane usługi, jak na przykład dostęp do poczty elektronicznej czy pewnych precyzyjnie zdefiniowanych adresów w sieci Internet [44]. Taka polityka zapewnia bardzo duże bezpieczeństwo systemów komputerowych, jednak odbywa się to kosztem pewnych oczywistych ograniczeń dla użytkowników sieci lokalnych. W niektórych instytucjach tego typu ograniczenia wykonywane są również w celu zwiększenia wydajności pracowników poprzez odcięcie im dostępu do treści nie związanych z pracą.

W drugim podejściu domyślnie cały ruch sieciowy jest odblokowany i dozwolony. Wdrażanie reguł bezpieczeństwa sprowadza się do blokowania dostępu do określonych zasobów, usług sieci Internet lub do wybranych komputerów. Zapewnia to większą elastyczność, jednak kosztem znacznego nakładu pracy koniecznej do zapewnienia pełnego bezpieczeństwa chronionej sieci lokalnej lub systemu komputerowego [79]. W celu zapewnienia wysokiego bezpieczeństwa systemu administrator musi zidentyfikować wszystkie usługi udostępniane przez komputer oraz dla każdej z nich podjąć decyzję, czy jej udostępnienie jest niezbędne i jeśli tak to komu należy przydzielić dostęp. Dopiero na podstawie przeprowadzonej analizy można stworzyć zbiór reguł zapory sieciowej, który będzie blokował niepożądany ruch sieciowy. Z uwagi na fakt, że zaprojektowanie całkowicie bezpiecznych reguł zapory sieciowej działającej według tej polityki jest niezwykle trudne, a dla niektórych usług wręcz niemożliwe, zalecane jest stosowanie polityki bardziej restrykcyjnej, działającej według reguły „wszystko, co nie jest dozwolone, jest zabronione” [111].

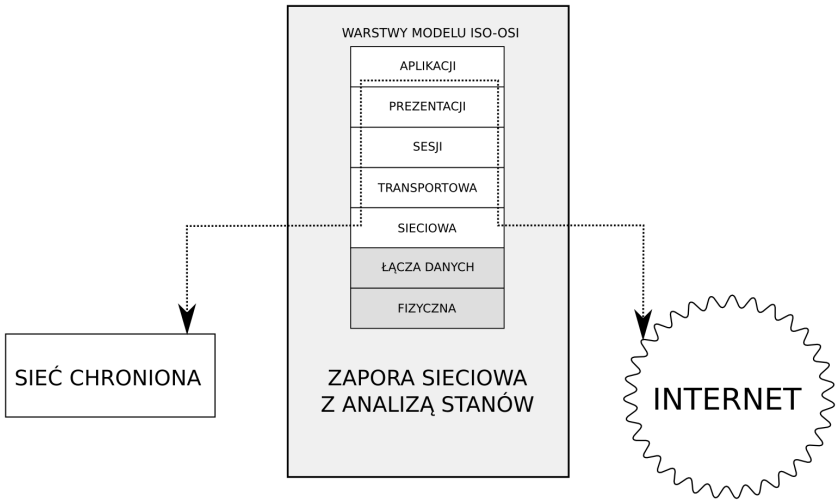
Zapory sieciowe wykorzystują do filtrowania ruchu sieciowego informacje zapisywane w różnych warstwach modelu ISO-OSI, który opisuje podstawową strukturę architektury sieciowej [53]. Systemy typu firewall można podzielić na kilka grup pod względem rodzaju informacji, które są używane do filtrowania ruchu sieciowego.

Najpopularniejsza i zarazem najprostsza zapora sieciowa kontroluje ruch na podstawie danych zawartych w opisie pakietów sieciowych i jest nazywana zaporą sieciową filtrująca pakiety (ang. *packet filtering firewall*) . Jak pokazano na rysunku 2.4, zapory sieciowe tego typu kontrolują dane zapisane w warstwie sieciowej oraz transportowej modelu ISO-OSI. Pozwala to na filtrowanie pakietów pochodzących z określonych adresów IP, wyróżnionych sieci lub podsieci, czy też konkretnych portów TCP lub UDP. Tego rodzaju zapory pozwalają np. na udostępnianie usług działających na wskazanych portach tylko ściśle określonym komputerom. Umożliwiają również całkowite zablokowanie ruchu sieciowego pochodzącego z komputera znajdującego się w określonej podsieci [17].



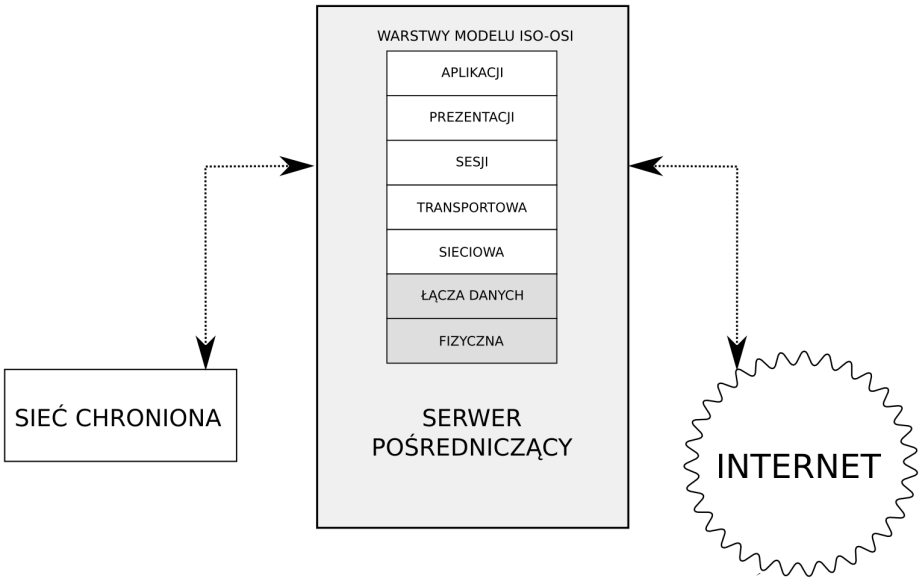
Rysunek 2.4. Zasada działania zapory sieciowej filtrującej pakiety

Filtr kontekstowy, zwany również zaporą sieciową z analizą stanów (ang. *stateful packet inspection firewall*), pozwala na analizowanie i filtrowanie ruchu sieciowego w kontekście całych sesji komunikacyjnych. W tym celu tworzone są tablice przechowujące informację o wszystkich sesjach TCP oraz połączeniach UDP zestawionych w kontrolowanej podsieci. Na podstawie zgromadzonych danych pakiety sieciowe są filtrowane według ustalonych reguł. Systemy tego typu umożliwiają również identyfikację użytkowników, a także kontrolę przesyłanych danych, np. przepuszczając na porcie 80 tylko ruch sieciowy zgodny z protokołem HTTP. Działanie tego typu zapory polega na badaniu wszystkich warstw modelu OSI począwszy od warstwy sieciowej, a skończywszy na warstwie aplikacji co pokazano na rysunku 2.5 [95].



Rysunek 2.5. Filtr kontekstowy

Innym typem zapory sieciowej jest tzw. serwer pośredniczący (ang. *proxy server*), który działa na najwyższej warstwie modelu OSI, warstwie aplikacji. Jego działanie polega na całkowitym odseparowaniu chronionej sieci lokalnej od Internetu. W rezultacie komputer w sieci wewnętrznej nie może nawiązać bezpośredniego połączenia z żadnym zewnętrznym serwerem. Wszelki kontakt dokonywany jest poprzez serwer proxy, który nawiązuje połączenie z komputerem docelowym.



Rysunek 2.6. Serwer pośredniczący

Mechanizm ten zilustrowano na rysunku 2.6. W ten sposób połączenie klient-serwer jest całkowicie kontrolowane poprzez zaporę pośredniczącą co daje administratorom większą kontrolę nad ruchem sieciowym, można np. kontrolować połączenia wykonywane przez aplikacje. Można powiedzieć, że serwer pośredniczący jest umieszczony na pierwszej linii ataku, jednak w związku z tym, że nie działają na nim żadne aplikacje, ani nie są przechowywane ważne dane, straty wynikające z udanego włamania lub jego zniszczenia są stosunkowo niskie. Uszkodzenie serwera proxy nie powoduje automatycznego uszkodzenia żadnych zasobów komputerowych znajdujących się w sieci lokalnej, co mogłoby się przekładać na znaczące straty dla użytkowników.

W celu zabezpieczania sieci lokalnych przed atakami sieciowymi stosowane są również metody ukrywania istniejących sieci lokalnych przez zapory sieciowe, takie jak NAT (ang. *Network Address Translation*) [121].

W celu zwiększenia skuteczności zapór sieciowych stosuje się połączenie różnych metod i mechanizmów filtrowania pakietów. Jednak należy stwierdzić, że nawet najbardziej złożone zapory sieciowe nie są w stanie całkowicie ochronić systemu komputerowego. Pozwalają jedynie na częściowe lub całkowite odseparowanie sieci lokalnej od środowiska zewnętrznego. Zapory sieciowe nie chronią przede wszystkim przed następującymi zagrożeniami:

- wirusami komputerowymi,
- końmi trojańskimi,
- atakami dokonywanymi wewnątrz sieci lokalnej,
- atakami wykorzystującymi błędy w aplikacjach.

Z tych powodów, aby zapewnić pełne bezpieczeństwo systemów, oprócz zastosowania zapór sieciowych, konieczne jest wykorzystywanie innych metod zabezpieczania systemów komputerowych, które zostaną przedstawione w dalszej części pracy.

2.3.2. Mechanizmy wykrywania włamań

Ważną grupę produktów podnoszących bezpieczeństwo systemów komputerowych stanowią programy pozwalające na wykrywanie włamań. W najnowszych rozwiązaniach granica pomiędzy systemami wykrywania włamań a zaporami sieciowymi zaczyna zanikać. Powstają systemy prewencji (ang. *Intrusion Prevention System, IPS*), które łączą filtrowanie ruchu sieciowego z aktywnym systemem wykrywania włamań [4].

Systemy detekcji włamań są również związane z systemami sprawdzania integralności systemów plików. Rozwiązania tego typu nie zabezpieczają bezpośrednio systemów komputerowych, ale pomagają wykrywać niebezpieczeństwa związane z podejrzaną aktywnością w sieci czy nieautoryzowanymi próbami dostępu do systemów operacyjnych i zmian w plikach systemowych.

Działanie systemów detekcji włamań (ang. *Intrusion Detection System, IDS*) polega na monitorowaniu ruchu sieciowego i wykrywaniu zdarzeń, bądź sekwencji zdarzeń, które zgodnie ze zdefiniowanymi procedurami analizy są traktowane jako podejrzane. Mechanizmy reakcji na pozytywnie zidentyfikowane zagrożenie mogą być różne w zależności od poszczególnych systemów. Może to być bardzo prosta procedura polegająca jedynie na wysyłaniu informacji do administratora o wystąpieniu zdarzenia. Bardziej rozbudowane systemy IDS mogą podejmować o wiele bardziej złożone działania obronne, np. automatycznie blokować podejrzane adresy IP czy wyłączać atakowane usługi.

Pod względem sposobu działania systemów IDS można wyróżnić te, które działają w ramach sieci lokalnych (ang. *Network Intrusion Detection System, NIDS*) lub takie, które działają w obszarze systemów operacyjnych zainstalowanych na chronionym komputerze (ang. *Host Intrusion Detection System, HIDS*). Natomiast w zależności od metod wykrywania podejrzanych zdarzeń, systemy detekcji włamań można podzielić na działające na bazie sygnatur przypisanych do określonych rodzajów ataków lub takie, które potrafią wykrywać anomalie w ruchu sieciowym. Niestety, działania na bazie sygnatur znanych ataków są mało skuteczne, ponieważ rodzaje ataków mogą być bardzo różne, a najgroźniejsze z nich to takie, które dotychczas były nieznane i pojawiły się niedawno. Systemy IDS, które tylko wykrywają zagrożenie i informują o tym administratorów zwane są czasami systemami pasywnymi. Natomiast systemy, które mogą podejmować określone akcje blokujące ataki nazywane są systemami aktywnymi. Tego typu systemy uznawane są za bardziej skuteczne w ochronie systemów komputerowych [112].

Systemy detekcji włamań działają skutecznie pod warunkiem dostosowania ich konfiguracji do konkretnego systemu komputerowego i jego środowiska sieciowego. Jest to czynność czasochłonna i wymaga dużej kompetencji administratorów. Należy znaleźć kompromis, aby z jednej strony nie generować zbyt wielu fałszywych alarmów (ang. *false positives*), a z drugiej wykrywać jak największą ilość faktycznych groźnych prób włamania.

W systemie operacyjnym Windows dodatkowo zalecane jest stosowanie programów chroniących komputer przed wirusami. Tego typu programy można również zaliczyć do grupy systemów wykrywania włamań. Programy antywirusowe mogą działać na dwa sposoby: albo są uruchamiane przez system operacyjny lub administratora okresowo i sprawdzają wszystkie pliki w systemie, czy nie zostały zainfekowane, albo monitorują na bieżąco pliki używane przez wszystkie aplikacje. Wykrywanie wirusów obydwa się zwykle na podstawie

tzw. sygnatur wirusów. Ta metoda jest skuteczna tylko wtedy, gdy producent oprogramowania już zidentyfikował wirusa, którym został zainfekowany komputer. Dlatego programy działające w ten sposób okresowo muszą uaktualniać bazę danych ze wzorcami sygnatur znanych wirusów. Istnieją również bardziej zaawansowane metody identyfikacji infekcji plików, tzw. metody heurystyczne, które pozwalają na wykrywanie nieznanych jeszcze wirusów [34]. Dla innych systemów operacyjnych również są dostępne programy antywirusowe, jednak z uwagi na małą ilość wirusów na tego typu platformy są one rzadko stosowane.

2.3.3. Mechanizmy ograniczające skutki włamań do systemów komputerowych

Projektując mechanizmy poprawiające bezpieczeństwo systemów komputerowych i systemów operacyjnych należy przyjąć założenie, że nie ma możliwości całkowitej eliminacji włamań. Z tego założenia wynikają pewne wnioski praktyczne dotyczące architektury systemów zabezpieczeń. Systemy takie powinny ograniczać możliwości szkodliwego działania intruza nawet wtedy, gdy uzyska on dostęp do systemu, również z uprawnieniami administratora. Takie podejście do zagadnienia zabezpieczania systemów komputerowych jest stosunkowo nowe i obecnie nie ma zbyt wielu systemów działających w ten sposób. Założenie to stało się również podstawą podczas projektowania nowych, skutecznych mechanizmów zabezpieczania systemów komputerowych w trakcie prac badawczych autora.

W celu realizacji założenia ograniczania możliwości destrukcyjnych działań intruzów opracowano szereg mechanizmów redukujących skutki włamań. Wśród stworzonych rozwiązań można wyróżnić takie, które zmierzają do kontroli i ograniczania przywilejów uruchamianych wewnątrz systemu operacyjnego procesów. Takie systemy pozwalają na minimalizację zagrożeń wynikających z zastosowania ataków wykorzystujących błędy w działających aplikacjach, np. w przeglądarkach internetowych czy w serwerach usług sieciowych. Mechanizmy redukujące skutki błędnego wykonywania programów pozwalają m. in. na wykrycie i uniemożliwienie wykorzystania błędów przepełnienia stosu, które są jednym z najczęstszych źródeł ataków na systemy komputerowe. Nową grupą mechanizmów ograniczającą skutki ataków są systemy, który umożliwiają całkowite odseparowanie od siebie usług działających w systemie operacyjnym.

Jeden z najważniejszych projektów ograniczających skutki włamań został stworzony przez Narodową Agencję Bezpieczeństwa USA (ang. *National Security Agency, NSA*). Nadano mu nazwę *Security-Enhanced Linux* (w skrócie *SELinux*), czyli Linux o zwiększonym bezpieczeństwie. Mechanizm ten został zaimplementowany jako rozszerzenie jądra systemu operacyjnego Linux i udostępniony publicznie na zasadach Open Source w 2000 roku. Rozwiązanie to powstało, aby zapewnić bezpieczeństwo krytycznych zasobów informatycznych w organizacjach rządowych i militarnych.

W systemie *SELinux* rozwiązano skutecznie problem kontroli dostępu do zasobów. W rodzinie systemów UNIX domyślną i powszechnie stosowaną polityką kontroli dostępu do zasobów jest tzw. swobodna kontrola dostępu (ang. *Discretionary Access Control*, *DAC*). Oznacza to, że właściciel może innym użytkownikom dowolnie udostępniać własne zasoby. W projekcie *SELinux* zaimplementowano znacznie bezpieczniejszy model kontroli dostępu do zasobów zwany modelem obowiązkowej kontroli dostępu (ang. *Mandatory Access Control*, *MAC*). Konfiguracja praw dostępu w modelu MAC przypomina reguły zapór sieciowych.

Oznacza to, że administrator definiuje filtry dostępu do zasobów, które nie mogą być przez nikogo innego zmienione. Oznacza to, że nawet właściciel nie może zmienić praw dostępu do swoich zasobów [41]. Podobnie jak w przypadku zapór sieciowych nadrzędna zasada głosi, że to, co nie jest dozwolone, jest zabronione [103].

Wykorzystana w *SELinux* polityka obowiązkowej kontroli dostępu powoduje, że po przeprowadzeniu przez intruza skutecznego włamania do systemu wykorzystującego np. błąd w aplikacji działającej z uprawnieniami użytkownika uprzywilejowanego root, powstałe szkody będą się ograniczały tylko do tych zasobów, do których dostęp miała dana aplikacja. Przy zastosowaniu klasycznego modelu swobodnej kontroli dostępu intruz uzyskałby pełen dostęp do całego systemu operacyjnego.

W systemie *SELinux* zaimplementowano również szereg innych mechanizmów bezpieczeństwa, m.in. wielopoziomowy model bezpieczeństwa (ang. *Multi-Level Security, MLS*), który jest powszechnie stosowany w systemach wojskowych. Tego typu model bezpieczeństwa oparty jest o zasadę, że przesyłanie informacji odbywa się z tzw. dolnych poziomów bezpieczeństwa do górnych, czyli np. z poziomu poufnego do tajnego, czy z tajnego do ściśle tajnego. Przesyłanie informacji w drugą stronę, z góry na dół, nie jest możliwe [41]. Ta polityka bezpieczeństwa z uwagi na swą restrykcyjność jest bardzo bezpieczna, lecz mało elastyczna i z tego powodu stosowana jedynie w systemach przeznaczonych do przechowywania najbardziej tajnych danych.

W systemie *SELinux* wdrożono również kontrolę dostępu opartą na rolach (ang. *Role-Based Access Control, RBAC*). Takie podejście upraszcza zarządzanie dostępem do zasobów, gdyż odpowiednie prawa przydzielane są do ról, a nie do poszczególnych użytkowników. Użytkownikom są natomiast nadawane role w zależności od piastowanego stanowiska, odpowiedzialności służbowej, czy nawet umiejętności [33].

Pomimo znacznych zalet zwiększających bezpieczeństwo, *SELinux* nie jest powszechnie stosowany poza organizacjami rządowymi czy militarnymi. Jego podstawową wadą jest skomplikowana konfiguracja, co powoduje, że system jest trudny do wdrożenia i utrzymania, a do jego administracji konieczna jest wysoko wykwalifikowana kadra.

Innym projektem, w którym częściowo zaimplementowano model obowiązkowej kontroli dostępu do zasobów (MAC), jest system *AppArmor*, stworzony w 1998 roku przez Immunix, a następnie rozwijany przez firmę Novell. W programie tym model MAC nie jest stosowany, jak w rozwiązaniu *SELinux* dla całego systemu, lecz jedynie dla określonych, krytycznych aplikacji [10]. System *AppArmor* jest zdecydowanie łatwiejszy do wdrożenia, konfiguracji i utrzymania.

System *AppArmor* bardzo dobrze nadaje się do ochrony serwerów sieciowych, które oprócz dostarczania stron internetowych, są coraz częściej stosowane m.in. do udostępniania publicznie usług w technologii cienkiego klienta. W celu zabezpieczenia takich systemów za pomocą *AppArmor* należy poddać ochronie serwer HTTP i serwer bazy danych. Po udanym ataku na tak chroniony system, przy wykorzystaniu dziury w serwisie internetowym, intruz nie będzie w stanie odczytywać dowolnych plików z całego systemu czy uruchamiać programów w powłoce. Możliwe będzie jedynie uzyskanie dostępu do plików, które były używane przez procesy zaatakowanych usług sieciowych.

AppArmor z uwagi na większą szybkość działania i prostszą konfigurację jest dobrą alternatywą dla bardziej rozbudowanego systemu *SELinux* i jest częściej stosowany, zwłaszcza w serwerach sieciowych. Prostsza konfiguracja i ograniczony zakres ochrony ma również swoje negatywne konsekwencje, ponieważ w systemach wieloużytkownikowych zakres oferowanych zabezpieczeń może nie być wystarczający do eliminacji skutków bardziej wyrafinowanych ataków.

Obecnie wzrasta zainteresowanie innym mechanizmem bezpieczeństwa, który polega na wykorzystaniu tzw. wirtualizacji systemów operacyjnych. Wirtualizacja jest techniką pozwalającą na uruchamianie wielu systemów operacyjnych na jednym komputerze.

Monitor maszyny wirtualnej (ang. *Virtual Machine Monitor, VMM*), inaczej zwany również hipernadzorcą (ang. *hypervisor*), jest programem komputerowym pozwalającym na uruchamianie jednocześnie wielu systemów operacyjnych na jednym komputerze. Kiedy zainstalowany w maszynie wirtualnej system operacyjny (ang. *guest operating system*) odwołuje się do sprzętu komputerowego, żądanie takie jest przechwytywane, a następnie obsługiwane przez hipernadzorcę. Monitor maszyny wirtualnej umożliwia równoległy dostęp do sprzętu komputerowego przez wiele działających równocześnie systemów operacyjnych.

Zastosowanie wirtualizacji pozwala na odseparowanie środowisk roboczych. Oznacza to, że atak, włamanie czy zainfekowanie wirusem systemu gościa nie jest groźne dla systemu gospodarza. Również inne uruchomione wirtualne systemy pozostają bezpieczne z uwagi na ich odseparowanie.

Wyróżniane są dwa główne typy wirtualizacji, wirtualizacja sprzętu i wirtualizacja programów. Wirtualizacja sprzętu polega na stworzeniu programowego emulatora komputera, w którym instalowany jest system operacyjny gościa. Przy wirtualizacji programów system gościa może się w pewnej przestrzeni odwoływać do warstwy sprzętowej komputera gospodarza. Najnowsze rozwiązania wykorzystują specjalne procesory, które umożliwiają dokonywanie prawdziwej, sprzętowej wirtualizacji. Z punktu widzenia bezpieczeństwa nie ma znaczenia czy wykorzystuje się wirtualizację sprzętu czy wirtualizację programów. Niemniej z punktu widzenia wydajności wirtualizacja programów, przy wykorzystaniu odpowiednich procesorów jest rozwiązaniem lepszym.

Wykorzystując wirtualizację można na przykład w serwerach oferujących usługi internetowe uruchomić na osobnym serwerze wirtualnym usługę WWW, a na innym serwerze wirtualnym usługę ftp, a jeszcze na innym pocztę elektroniczną. Istnieje też możliwość utworzenia osobnych, dedykowanych systemów wirtualnych do obsługi różnych domen, usług czy klientów. W takim przypadku udane włamanie na serwer konkretnej domeny nie pociągnie za sobą zagrożenia dla bezpieczeństwa innych domen.

Wirtualizacja jest również coraz częściej wykorzystywana w procedurach pozwalających na przywracanie systemu operacyjnego po jego awarii (ang. *disaster recovery*). Zwykle w przypadku sprzętowej awarii, przeniesienie zainstalowanego na dysku twardym systemu operacyjnego na komputer o innej konfiguracji sprzętowej jest zadaniem skomplikowanym, a często wręcz niemożliwym. W związku z tym, że wirtualne systemy nie są związane z konfiguracją sprzętową, istnieje możliwość łatwego przenoszenia obrazu wirtualnego systemu na inny komputer z całkowicie różną konfiguracją sprzętową.

Wirtualizacja jest obecnie również wykorzystywana do automatycznego tworzenia tzw. migawek (ang. *snapshot*), które są rodzajem kopii zapasowej systemu gościa na stacjach roboczych. W takiej postaci można zapisać nie tylko stan systemów plików, ale również stan procesów. Po wykryciu włamania lub infekcji wirusem plików systemowych migawka może stać się alternatywą działającego już systemu i pozwolić na przywrócenie bezpiecznej konfiguracji tego systemu.

Wirtualizacja jest stosunkowo nową, ale bardzo skuteczną metodą zapewniania bezpieczeństwa systemów operacyjnych, o dużym potencjale i możliwościach rozwoju. Pozwala przede wszystkim na zarządzanie bezpieczeństwem systemów operacyjnych w sposób kompleksowy. Posiada jednak pewne wady, z których najważniejszymi są stosunkowo duże wymagania sprzętowe. Rola komputera, na którym zainstalowano wiele wirtualnych serwerów znacznie wzrasta, a jego sprzętowa awaria może pociągać za sobą bardzo poważne konsekwencje. Dlatego ze względów bezpieczeństwa stosuje się zrównoleglenie tego typu serwerów, co pozwala na uniknięcie zjawiska zwanego pojedynczym punktem awarii (ang. *single point of failure*).

Dynamiczny rozwój technologii, pojawianie się szybszych i tańszych procesorów z wieloma rdzeniami sprzyja rozwojowi wirtualizacji. W związku z tym zastosowanie tej techniki do podnoszenia bezpieczeństwa systemów komputerowych będzie z czasem coraz bardziej popularne, znacznie prostsze i wydajniejsze. Dlatego należy się spodziewać wzrostu zainteresowania wykorzystywaniem wirtualizacji do zapewniania bezpieczeństwa systemów operacyjnych.

Pomimo tego, że nad problemem bezpieczeństwa prowadzone są badania od wielu lat, nie udało się dotychczas stworzyć całkowicie skutecznego systemu zabezpieczeń. Prawdopodobnie każde zabezpieczenie programowe zostanie prędzej czy później złamane. Jest to spowodowane opracowywaniem coraz to nowszych metod przeprowadzania ataków, ciągłym wzrostem mocy obliczeniowych komputerów, a także znajdowaniem wielu dotychczas nieznanych błędów zawartych w praktycznie wszystkich programach komputerowych. Dlatego należy przyjąć założenie, że jeśli nie można całkowicie i skutecznie zabezpieczyć systemów komputerowych, powinno się zastosować takie mechanizmy, które ograniczą i będą przeciwdziałać skutkom udanych włamań do systemów komputerowych. Przedstawione wcześniej rozwiązania, takie jak model obowiązkowej kontroli dostępu, czy wirtualizacja ograniczają możliwości wrogiego działania intruzów, jednak nie eliminują całkowicie szkód, które mogą wystąpić po udanym ataku.

Rozdział 3.
Systemy ochrony integralności plików

Jednym z najskuteczniejszych sposobów wykrywania włamań do systemów komputerowych jest kontrola integralności systemu plików, która polega na wykrywaniu zmian w określonej grupie chronionych plików. Dla każdego wybranego do ochrony pliku, na podstawie jego zawartości, generowany jest unikalny wzorzec. Wzorzec ten jest przechowywany w bazie danych. Wszelkie zmiany zawartości każdego z chronionych plików będą mogły zostać wykryte przez odpowiednie oprogramowanie [58].

System ochrony integralności systemu plików (ang. *File System Integrity Checker*) jest programem komputerowym, który pozwala na zabezpieczenie systemu operacyjnego, poprzez kontrolę zawartości wybranej grupy plików, uznanej za kluczową dla poprawnego działania systemu komputerowego [5]. Wykrywanie zmian w plikach jest skuteczne, jeżeli baza danych wzorców chronionych plików zostaje wygenerowana zaraz po instalacji systemu operacyjnego. W przypadku konieczności aktualizacji plików systemowych, administrator może stworzyć nową bazę wzorców, jeśli jest pewien, że system nie został zaatakowany. Kluczową kwestią jest również stworzenie specjalnych mechanizmów chroniących bazę wzorców, ponieważ kontrola integralności polega na wygenerowaniu i porównaniu wzorca dla każdego chronionego pliku z jego wzorcem zapisanym w bazie danych.

Dla skuteczności systemów ochrony integralności systemów plików kluczową kwestią jest metoda wyznaczania wzorca, która musi się charakteryzować następującymi właściwościami:

1. Każda, nawet najmniejsza zmiana w pliku prowadzi zawsze do wygenerowania innego wzorca.
2. Wygenerowania zawartości pliku na podstawie znanego wzorca jest zadaniem obliczeniowo niewykonalnym.
3. Operacja wyznaczania wzorca musi być wydajna, nawet dla dużych plików.

Najczęściej stosowaną metodą wyznaczania wzorców dla plików jest funkcja skrótu kryptograficznego (ang. *cryptographic hash function*). Jest to funkcja *h*, która jako argument wejściowy przyjmuje ciąg znaków *M* dowolnej długości i na jego podstawie tworzy tzw. skrót (ang. *hash*) *h(M)*. Ważną właściwością takich funkcji jest to, że skrót jest ciągiem znaków o stałej długości [21]. W związku z tym, że długość skrótu jest stała, a wejściowy ciąg znaków dowolnie długi, funkcja skrótu nie jest funkcją różnowartościową. Oznacza to, że dla kilku wejściowych ciągów znaków może zostać wygenerowany taki sam skrót. Zjawisko to nosi nazwę kolizji (ang. *collision*) [93].

Dla zapewnienia bezpieczeństwa bardzo istotnym jest, aby funkcje kryptograficzne były odporne na ataki polegające na znajdowaniu przeciwobrazów (ang. *preimage attack*), drugich przeciwobrazów (ang. *second preimage attack*) oraz na generowanie kolizji (ang. *collision attack*) [98]. Poszczególne ataki są definiowane następująco [82,32]:

Znajdowanie przeciwobrazów funkcji kryptograficznej h: znając tylko skrót *h(M)* znajdź obraz *M*.

Znajdowanie drugich przeciwobrazów funkcji kryptograficznej h: znając skrót *h(M)* i obraz M znajdź obraz *M′* taki, że *M≠M′* i *h(M)=h(M′)*.

Generowanie kolizji funkcji kryptograficznej h: znajdź parę obrazów *M* i *M′* takich, że *M≠M′* i *h(M)=h(M′)*.

Stosowane w systemach ochrony integralności plików funkcje skrótu przyjmują jako argument zawartość pliku, a zwracają skrót będący wzorcem pliku, który jest zwany czasem cyfrowym odciskiem palca (ang. *digital fingerprint*).

Dotychczas najczęściej stosowanymi funkcjami skrótu kryptograficznego w systemach ochrony integralności plików była funkcja MD5 (ang. *Message-Digest algorithm 5*) [92] oraz funkcja SHA1 (ang. *Secure Hash Algorithm*) [30]. Jednak w związku z tym, że w 2004 roku został opublikowany algorytm generowania kolizji dla funkcji MD5 [113] a w 2005 dla SHA1 [112], funkcje te nie są już uznawane za bezpieczne. Wśród bardziej zaawansowanych rozwiązań obecnie najczęściej stosuje się nowsze, rozszerzone wersje funkcji SHA, takie jak SHA-224, SHA-256, SHA-384 czy SHA-512. Zwiększenie bezpieczeństwa chronionych plików za pomocą wymienionych funkcji niestety pociąga za sobą większą czasochłonność wykonywania operacji tworzenia skrótu.

Procedura ochrony integralności systemu plików polega na wyborze grupy plików, które mają być chronione, stworzeniu dla nich skrótów kryptograficznych, zapisaniu ich w bazie danych i co pewien czas uruchamianiu programu, który kontroluje integralność systemu plików. Program taki oblicza skrót kryptograficzny dla chronionych plików, a następnie porównuje uzyskane ciągi znaków z ich odpowiednikami zapisanymi w bazie danych. W przypadku, gdy występują różnice w skrótach uruchamiana jest procedura alarmowa. Podejmowane działania zależą od możliwości oferowanych przez program kontrolujący. Dodatkowo domyślne działania mogą być modyfikowane za pomocą ustawień konfiguracyjnych zdefiniowanych przez administratora systemu. Najczęściej podejmowane są następujące kroki:

- powiadomienie administratora systemu o zaistniałej sytuacji,
- zablokowanie wykonywanej przez potencjalnego intruza operacji na pliku,
- całkowite zablokowanie dostępu do zagrożonego pliku,
- natychmiastowe zamknięcie systemu operacyjnego.

System ochrony może jednocześnie wykonać kilka z tych operacji zapewniających bezpieczeństwo systemu [86]. Praktycznie zawsze, oprócz np. zablokowania dostępu do plików, powiadamiany jest administrator systemu.

Systemy ochrony integralności plików można podzielić na programy, które działają w przestrzeni użytkownika (ang. *userspace*) z uprawnieniami superużytkownika oraz na programy działające na poziomie jądra systemu operacyjnego (ang. *kernelspace*). Programy w przestrzeni użytkownika działają podobnie jak inne aplikacje. Są zwykle uruchamiane okresowo i mogą wykryć ewentualne zmiany w systemie plików dokonane przez intruza pomiędzy kontrolami zgodności wzorców. Programy takie nie są jednak w stanie zapobiec samemu włamaniu, a zmiany dokonywane w plikach nie są wykrywane automatycznie, a jedynie w czasie wykonywania procedury kontrolnej. W celu zapewnienia ciągłej kontroli integralności systemu plików konieczne jest zaimplementowanie mechanizmów ochrony w

jądrze systemu operacyjnego. Tak działające programy mogą dokonywać kontroli automatycznie np. podczas każdej próby odczytu pliku, a także wykonywać bardziej zaawansowane procedury alarmowe, takie jak natychmiastowe zablokowanie dostępu do zmienionego pliku.

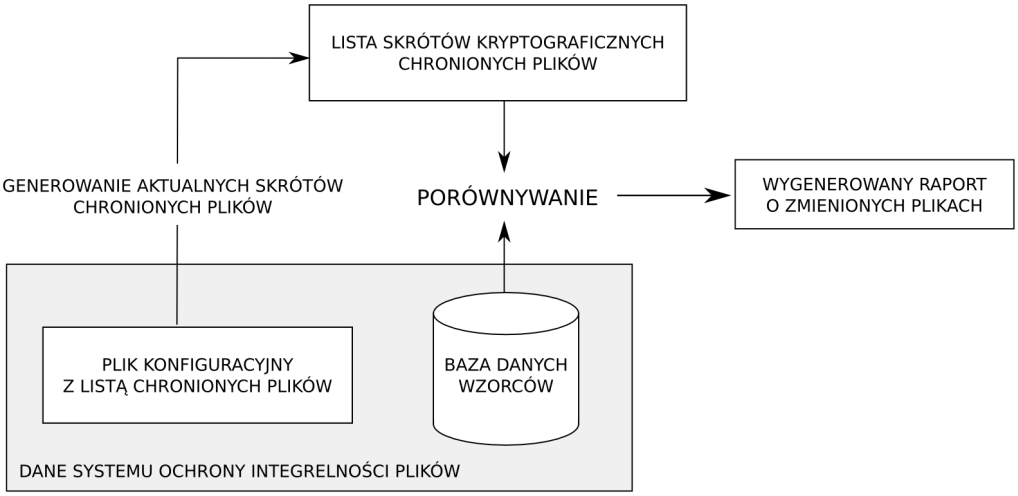
Ponieważ w dalszej części pracy zostanie przedstawiony własny oryginalny, system zapewniania bezpieczeństwa systemu operacyjnego oparty o kontrolę integralności systemów plików, aktualny stan wiedzy w tej dziedzinie zostanie szczegółowo przedstawiony. Uwzględnione zostaną zarówno programy działające w przestrzeni użytkownika, jak i programy zintegrowane z jądrem systemu operacyjnego. Zostaną przedstawione możliwości oraz ograniczenia w dotychczas stworzonych systemach ochrony integralności systemów plików. Przeprowadzona analiza stanie się podstawą do zaproponowania nowych, bardziej skutecznych metod zapewniania bezpieczeństwa systemów komputerowych.

3.1. Ochrona integralności plików w przestrzeni użytkownika

Systemy ochrony integralności plików mogą być zwykłą aplikacją uruchamianą jak każdy inny program komputerowy. Administrator systemu operacyjnego może uruchomić taki program ręcznie, lecz najczęściej programy kontroli uruchamiane są okresowo za pomocą odpowiednich programów systemowych, takich jak *Cron* w systemie Linux, czy *Harmonogram Zadań* w systemie Windows. Ręczne uruchamianie programu ochrony przez administratora systemu może zostać wykonane, jeśli z innych źródeł uzyska on informację o podejrzeniu włamania do zarządzanego systemu.

Przed rozpoczęciem korzystania z programu ochrony integralności plików koniecznie jest wybranie i zdefiniowanie grupy chronionych plików. Z uwagi na bardzo dużą liczbę plików, jakie może zawierać system operacyjny, nie jest możliwe monitorowanie zmian w nich wszystkich. Na podstawie zdefiniowanej listy plików program ochrony, wykorzystując funkcje kryptograficzne, generuje wzorcowe skróty i umieszcza je w bazie danych. W programach ochrony tego typu niezwykle istotnym zagadnieniem jest zapewnianie bezpieczeństwa samej bazy danych. Nie jest to zagadnienie trywialne, gdyż baza danych jest często przechowywana jako zwykły plik. Jeśli intruz będzie w stanie dokonać modyfikacji wzorców w bazie danych cały system ochrony integralności plików nie będzie godny zaufania i będzie nieskuteczny. Sytuacja taka może wręcz osłabić bezpieczeństwo systemu, gdyż administrator byłby przez taki system ochrony informowany, że nie doszło do żadnego ataku, gdy tymczasem mogłoby to nie być prawdą. Najczęściej stosowanym zabiegiem zapewniającymi bezpieczeństwo bazy danych jest zabezpieczanie kryptograficzne jej pliku za pomocą podpisu cyfrowego. Jednak w ten sposób blokowana jest jedynie możliwość niezauważonej podmiany lub modyfikacji pliku bazy danych. Zabezpieczenie podpisem cyfrowym nie chroni natomiast przed usunięciem pliku bazy danych z wzorcowymi skrótami.

Po dokonaniu wstępnej konfiguracji programu ochrony, dalsze jego działanie polega na okresowym sprawdzaniu integralności poprzez każdorazowe obliczenie skrótów kryptograficznych na podstawie aktualnej zawartości chronionych plików i porównaniu ich ze wzorcami zapisanymi w bazie danych. Jeżeli uzyskany skrót pliku jest różny od wzorca wszczynana jest procedura alarmowa. Na rysunku 3.1 został przedstawiony schemat działania programów ochrony integralności systemu plików działających w przestrzeni użytkownika [27].



Rysunek 3.1. Schemat działania programów ochrony integralności systemu plików

Należy zauważyć, że program działający w przestrzeni użytkownika sprawdza jedynie, czy chroniony plik został zmodyfikowany w przedziale czasu pomiędzy kolejnymi uruchomieniami programu. Niestety, w okresie pomiędzy kolejnymi kontrolami możliwa jest zarówno zmiana zawartości pliku chronionego, jak również uruchomienie zmienionego pliku przez autoryzowanego użytkownika systemu. W związku z tym, w rzeczywistości programy działające w przestrzeni użytkownika nie chronią systemu przed zmianami, a jedynie pozwalają na wykrycie, czy pliki chronione zostały zmodyfikowane przez intruza. Jeżeli system ochrony zasygnalizuje naruszenie integralności plików konieczne staje się przywrócenie poprawnej zawartości plików. W tym celu należy dokonać aktualizacji w systemie operacyjnym polegających na przywróceniu oryginalnej zawartości zmienionych plików z kopii zapasowych, a następnie ponownie uruchomić system operacyjny. Dzięki temu, że system ochrony integralności dostarcza informacji o tym, które pliki zostały zmienione, możliwe jest przywrócenie tylko ich zawartości, a nie całego systemu operacyjnego. Jest to znaczna zaleta zastosowania programów tego typu, gdyż czas potrzebny na ponowne uruchomienie systemu operacyjnego zostaje zasadniczo zmniejszony.

Programy działające w przestrzeni użytkownika są stosunkowo proste do zaprojektowania i zaimplementowania. Zasada, że programy tego typu nie monitorują systemów plików w sposób ciągły, a jedynie uruchamiane są okresowo ma pewne praktyczne uzasadnienie. Sam proces sprawdzania jest czasochłonny i mocno zwiększa wykorzystanie zasobów systemowych. Dlatego też proces ten nie może być uruchamiany zbyt często. Do wad programów działających w przestrzeni użytkownika należy również fakt, że pliki wchodzące w skład programu zapisane są w systemie jako zwykła aplikacja. W związku z tym istnieje stosunkowo prosta możliwość wyłączania lub wykasowania takiego systemu ochrony. Z tego względu zwiększenie bezpieczeństwa systemów operacyjnych dzięki wykorzystaniu systemów ochrony integralności systemów plików działających w przestrzeni użytkownika jest stosunkowo małe.

Pomimo pewnych wad opracowano wiele programów ochrony integralności działających w przestrzeni użytkownika. Pierwszym, używanym na szerszą skalę, był program o nazwie *Tripwire*. Program ten powstał w 1992 roku na amerykańskim Uniwersytecie Purdue. Program *Tripwire* jest pierwowzorem wszystkich innych programów tego typu.

Program *Tripwire* pozwala administratorowi systemu na monitorowanie zmian w systemie plików takich jak dodawanie, usuwanie i modyfikacja chronionych plików. Pozwala nie tylko na monitorowanie zmian w samych plikach, ale również, co jest istotne, w ich atrybutach, takich jak liczba dowiązań, czy prawa dostępu [66]. Zwłaszcza możliwość monitorowania zmian praw dostępu w grupie chronionych plików jest bardzo cenną funkcjonalnością z punktu widzenia bezpieczeństwa.

Do swojego działania program *Tripwire* wykorzystuje dwa pliki, z których jeden zawiera ustawienia konfiguracyjne, a drugi jest bazą danych z informacjami o chronionych plikach. W pliku konfiguracyjnym znajduje się lista katalogów i plików, które zostały przez administratora wybrane do monitorowania. Dodatkowo każdy zapis dotyczący konkretnego pliku zawiera maskę, która określa, jakie atrybuty pliku mogą zostać zmienione bez rozpoczynania procedury alarmowej. W ten sposób administrator może na przykład umożliwić zmianę praw dostępu do pliku. Baza danych programu *Tripwire* jest tworzona na podstawie pliku konfiguracyjnego. Dla każdego monitorowanego pliku zapisywane są w bazie danych szczegółowe informacje o nim, takie jak jego atrybuty i kryptograficzny wzorzec [67].

Podczas sprawdzania integralności monitorowanych plików program *Tripwire,* na podstawie danych zapisanych w pliku konfiguracyjnym, generuje zbiór skrótów zgodny z aktualną zawartością chronionych plików. Następnie dane te są porównywane z bazą wzorców, jaką stworzono w chwili instalacji systemu. Wynikiem takiego działania jest lista plików, które zostały dodane, usunięte lub zmienione od ostatniej kontroli integralności. Następnie dla danych zapisanych na liście sprawdzana jest maska z pliku konfiguracyjnego w celu określenia, czy i jaka procedura alarmowa ma zostać uruchomiona. Domyślnie generowany jest raport informujący administratora o stanie integralności systemu plików.

Pierwotnie program *Tripwire* był udostępniany publicznie i nieodpłatnie. W późniejszym okresie autorzy stworzyli wersję komercyjną. W 2001 roku program ten w wersji uproszczonej został udostępniony na zasadach Open Source. Jednak w związku z tym, że pełna wersja programu nie jest dostępna publicznie, powstało wiele projektów Open Source, w ramach których zostały stworzone programy mające podobną funkcjonalność co program *Tripwire*. Do najciekawszych z nich należy zaliczyć programy *AIDE*, *Veracity*, czy *integrit* [94]. Wymienione programy mają podobną funkcjonalność i niewiele różnią się między sobą z punktu widzenia swojego działania. Różnią się natomiast szczegółami implementacyjnymi, zwłaszcza szybkością działania czy możliwością instalacji na różnych systemach operacyjnych. Należy jednak podkreślić, iż ogólna zasada działania wszystkich programów ochrony integralności systemów plików działających w przestrzeni użytkownika jest podobna i oparta o mechanizmy zastosowane w programie *Tripwire*.

Pewną odmianą programów ochrony integralności plików działających w przestrzeni użytkownika są aplikacje uruchamiane na zdalnych komputerach (ang. *non-resident checkers*). Program taki jest uruchamiany na serwerze i kontroluje integralność plików przechowywanych na wielu komputerach podłączonych do sieci lokalnej. Sprawdzanie integralności plików możliwe jest dzięki zastosowaniu protokołów zdalnego dostępu do plików, takich jak NFS, CIFS, SSH, itp. Uruchamianie programów ochrony integralności na serwerze jest korzystne, gdyż pozwala na sprawdzanie integralności plików na dowolnej liczbie komputerów, a zarazem ułatwia ochronę bazy danych ze wzorcami plików. Dedykowane serwery zapewniające bezpieczeństwo systemów plików są na ogół wyposażone

w dodatkowe, rozszerzone mechanizmy bezpieczeństwa, które mają chronić przede wszystkim bazę danych wzorców. Najpopularniejsze programy tego typu to *Radmind*, *Osiris* oraz *SAMHAIN* [20, 24].

Systemy ochrony integralności systemów plików działające w przestrzeni użytkownika tylko częściowo zapewniają ochronę systemu operacyjnego. Pomimo małej skuteczności wynikającej z okresowego uruchamiania procedur sprawdzania plików programy tego typu są stosunkowo popularne. Natomiast największą wadą tych systemów jest fakt, że są to zwykłe programy komputerowe, zapisane w plikach, których lokalizacja jest jawna, a pliki mogą zostać zmodyfikowane lub usunięte w wyniku zaawansowanych działań intruza.

3.2. Systemy działające na poziomie jądra

W architekturze systemów operacyjnych występują dwa poziomy, na których mogą być uruchamiane programy. Jest to poziom użytkownika i poziom jądra systemu operacyjnego. Poziom użytkownika ma niższy priorytet niż poziom jądra i jest też gorzej zabezpieczony. Na tym poziomie użytkownik uruchamia aplikacje, czyli skompilowane lub interpretowane programy, wykorzystuje interfejsy, a także przygotowuje procesy do wykonania. Działanie użytkownika na tym poziomie są bardzo trudne do zabezpieczenia, gdyż system musi umożliwić działanie takich operacji, jak wprowadzanie danych, uruchamianie programów czy zarządzanie systemami plików. Z drugiej strony zabezpieczenia procesów działających na poziomie jądra są bardziej złożone i oferują większy poziom bezpieczeństwa. Przede wszystkim użytkownicy nie mogą ingerować w działanie procesów uruchomionych na tym poziomie. Oznacza to, że również intruz ma trudniejszy dostęp do procesów. Programy działające na poziomie jądra mają wyższe priorytety, co oznacza, że w procesie kolejkowania zadania zlecane przez jądro mają pierwszeństwo w stosunku do procesów użytkowników.

Poziom jądra z punktu widzenia ochrony integralności systemów plików umożliwia nie tylko detekcję zmian w plikach, ale również umożliwia podjęcie aktywnej ochrony systemu, np. poprzez zablokowanie możliwości uruchamiania procesów wykorzystujących zmienione pliki.

Bardzo ważną zaletą umieszczania mechanizmów bezpieczeństwa na poziomie jądra systemu operacyjnego jest eliminacja warstw pośrednich, w tym poziomu użytkownika. Należy zauważyć, że każdy moduł, z którym będzie współpracował program chroniący system komputerowy może generować błędy, czy też mieć niedopracowane komponenty, poprzez które intruz będzie mógł wyeliminować lub zmniejszyć skuteczność zabezpieczeń. Istnieje zasada, że każdy system jest tak niezawodny, jak najbardziej zawodny komponent, z którego się składa [48]. Dlatego umieszczenie dobrze zaprojektowanego i zaimplementowanego systemu ochrony plików w najniższej warstwie systemu operacyjnego, jakim jest poziom jądra, powinno z zasady znacznie podnosić bezpieczeństwo systemu. Zatem systemy bezpieczeństwa działające w przestrzeni jądra są bardziej skuteczne od systemów działających w przestrzeni użytkownika.

Sposób uruchamiania programu zabezpieczającego system komputerowy jest główną różnicą pomiędzy programami działającymi w przestrzeni użytkownika i w przestrzeni jądra. Programy w przestrzeni użytkownika są uruchamiane poprzez wydanie polecenia przez administratora lub okresowo przez odpowiedni mechanizm systemowy. Natomiast programy działające w przestrzeni jądra nie są tak naprawdę oddzielnymi aplikacjami. Stanowią jedynie

jedną z wielu części jądra systemu operacyjnego. Oznacza to, że są one uruchomione stale od momentu uruchomienia systemu operacyjnego i nie można ich wyłączyć. W zależności od rodzaju architektury jądra, mechanizm ochrony integralności plików może być zaimplementowany na różne sposoby. Dla jąder monolitycznych może zostać umieszczony bezpośrednio w kodzie jądra systemu operacyjnego lub jako jego moduł, natomiast dla mikrojądra jako serwer [101].

Ponieważ kod źródłowy jądra systemu operacyjnego Linux jest publicznie dostępny i może być dowolnie zmieniany, systemy ochrony integralności plików dla tego systemu operacyjnego będą przedmiotem szerszych rozważań. W systemie operacyjnym Linux będą również przeprowadzone badania eksperymentalne opracowywanych w ramach pracy mechanizmów bezpieczeństwa.

Programy działające w przestrzeni jądra systemu operacyjnego pozwalają na uruchamianie mechanizmów zabezpieczeń w chwili, gdy jakiś proces żąda dostępu do pliku. Jądro systemu operacyjnego zawsze w takiej sytuacji dokonuje pewnych czynności kontrolnych, między innymi sprawdza uprawnienia procesu. Jeśli system ochrony integralności jest częścią systemu operacyjnego, czynności kontrolne zostają rozszerzone o sprawdzenie, czy zawartość kontrolowanego pliku jest zgodna ze wzorcami zapisanymi w bazie danych.

Implementacja takiego mechanizmu ochrony plików jest stosunkowo prosta, gdyż wynika z podstawowej funkcjonalności jądra systemu operacyjnego związanej z uruchamianiem i otwieraniem plików dla procesów. W sytuacji, gdy weryfikacja skrótów kryptograficznych wykaże, że plik nie został zmodyfikowany, dostęp zostaje przydzielony do procesu, który chce z niego korzystać. Jeżeli natomiast plik nie przeszedł poprawnej weryfikacji, co może oznaczać, że został zmodyfikowany przez intruza, istnieje możliwość wykonania wielu dodatkowych procedur alarmowych. System ochrony może nie tylko wysłać powiadomienie do administratora, ale również zabronić dostępu do pliku, czy nawet podjąć inne bardziej restrykcyjne akcje. Do takich działań można zaliczyć natychmiastowe zablokowanie dostępu do pliku dla wszystkich procesów, a nawet wyłączenie komputera. Z uwagi na to, że istnieją możliwości aktywnej ochrony przed intruzem w systemach działających na poziomie jądra można je nazywać mechanizmami *ochrony* integralności plików. Natomiast w przypadku programów działających w przestrzeni użytkownika odpowiedniejsza byłaby nazwa mechanizmy *sprawdzania* integralności systemu plików.

3.2.1. Systemy wykorzystujące wywołania systemowe jądra

Integracja mechanizmów zabezpieczania systemu plików z jądrem systemu operacyjnego może zostać dokonana na kilka różnych sposobów. Jednym z nich jest wykorzystanie tzw. wywołań systemowych (ang. *System Calls*) istniejących w systemie operacyjnym. Wywołania systemowe są zbiorem funkcji, które udostępniają interfejs programistyczny do korzystania z niskopoziomowych mechanizmów jądra systemu operacyjnego.

W systemie operacyjnym Linux dostępnych jest kilkadziesiąt funkcji systemowych. Niektóre z nich pozwalają na zarządzanie zasobami systemowymi, w tym np. na kontrolę dostępu do plików zapisanych w systemie komputerowym. Również wszystkie odwołania do zasobów sprzętowych i plików z programów działających w przestrzeni użytkownika są obsługiwane przez jądro systemu operacyjnego właśnie z wykorzystaniem wywołań systemowych.

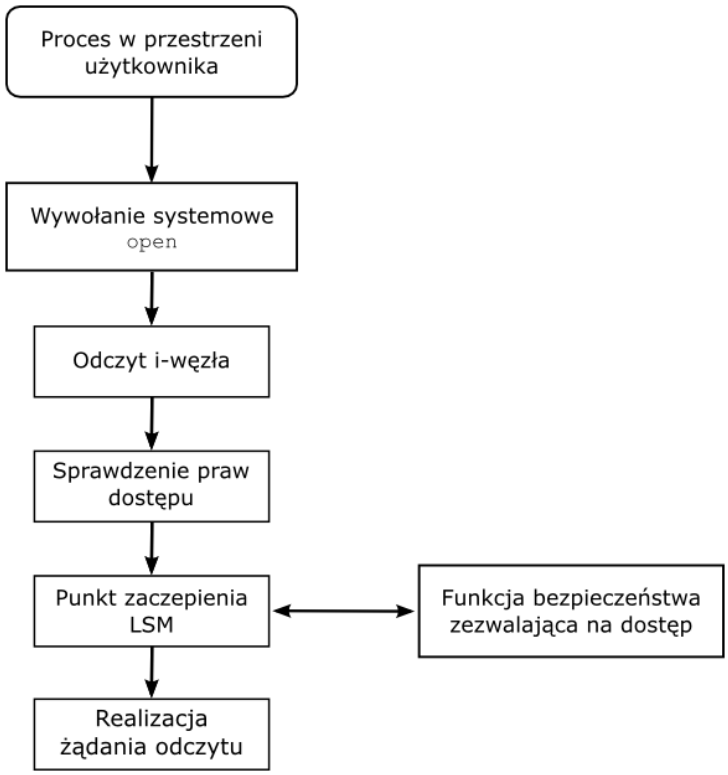
W systemie operacyjnym Linux, jeśli jakiś program działający w przestrzeni użytkownika wymaga dostępu do zasobów sprzętowych czy plików, wywołuje funkcję systemową, dzięki której może uzyskać dostęp m. in. do nośników danych, kart graficznych i innych urządzeń peryferyjnych. Ponieważ wywołania systemowe pełnią rolę pośredniczącą, rolę interfejsu pomiędzy uruchomionym przez użytkownika programem, a jądrem systemu operacyjnego, ich znaczenie w projektowaniu i implementowaniu mechanizmów zabezpieczania plików jest niezwykle ważne. W celu ułatwienia korzystania z wywołań systemowych przez programistów zostało stworzone tzw. API (ang. *Application Programming Interface*) systemu operacyjnego. Jest to zbiór funkcji, które umożliwiają wykonywanie wywołań systemowych z poziomu aplikacji. Należy podkreślić, że funkcje wchodzące w skład API nie muszą być tożsame z wywołaniami systemowymi. Funkcje API ułatwiają dostęp do określonych zasobów, a wywołania systemowe pozwalają na uruchamianie funkcji jądra za pośrednictwem przerwań programowych (ang. *software interrupt*) [15].

W celu zapewnienia kompatybilności na poziomie kodu źródłowego aplikacji pomiędzy wszystkimi systemami operacyjnymi z rodziny UNIX został stworzony standard ISO/IEC 9945, znany szerzej pod nazwą POSIX (ang. *Portable Operating System Interface*), którego najważniejszą częścią jest definicja funkcji API. Jednoznaczny zdefiniowany interfejs API pozwala na całkowite rozdzielenie szczegółów implementacyjnych poszczególnych składników jądra od programów działających w przestrzeni użytkownika. Dzięki standardowi POSIX, pomimo różnej architektury i implementacji np. systemów Linux i Solaris, czy AIX, program napisany na którykolwiek z tych systemów, będzie działał po skompilowaniu na każdym z pozostałych.

Pewnym rozszerzeniem i próbą standaryzacji mechanizmu zamiany wywołań systemowych wykorzystywanych w rozwiązaniach bezpieczeństwa systemów operacyjnych, jest szkielet (ang. *framework*) o nazwie *Linux Security Modules* (LSM). Podstawowym zadaniem LSM jest umożliwienie wprowadzenia rozszerzonych metod kontroli dostępu do zasobów, takich jak np. model obowiązkowej kontroli dostępu MAC. Ponadto framework LSM może być z powodzeniem wykorzystywany do podnoszenia bezpieczeństwa systemów operacyjnych w innych kierunkach. Szkielet LSM wprawdzie samodzielnie nie podnosi w żaden sposób bezpieczeństwa systemu operacyjnego, ale dostarcza „infrastruktury”, która ułatwia tworzenie różnych modułów zapewniania bezpieczeństwa [115].

Koncepcja frameworku Linux Security Modules została przedstawiona w 2001 roku przez Linusa Torvaldsa, twórcę systemu Linux. Inspiracją był opisany rozdziale 2.3 mechanizm *Security-Enhanced Linux* stworzony przez Narodową Agencję Bezpieczeństwa USA, który w sposób znaczący rozszerzał bezpieczeństwo systemu Linux. Torvalds postanowił stworzyć ogólny szkielet, dzięki któremu możliwe byłoby włączanie różnych mechanizmów bezpieczeństwa za pomocą dołączanych dynamicznie modułów jądra [127].

Implementacja szkieletu LSM polega na umieszczeniu tzw. punktów zaczepienia (ang. *hook*) w każdym miejscu, w którym programy działające w przestrzeni użytkownika mogą odwoływać się do zasobów systemowych. Sposób działania Linuksowych Modułów Bezpieczeństwa został schematycznie przedstawiony na rysunku 3.2. Standardowa procedura przydzielania dostępu do zasobów została rozszerzona o punkt zaczepienia LSM, oznaczony jako *LSM Hook*, w którym dochodzi do wywołania zarejestrowanej funkcji bezpieczeństwa, przed przydzieleniem dostępu do zasobu. Zwrócenie przez funkcję bezpieczeństwa kodu błędu powoduje zablokowanie dostępu do tego zasobu.



Rysunek 3.2. Zasada działania szkieletu Linux Security Modules

Szkielet LSM może być z powodzeniem wykorzystany do włączenia systemów ochrony integralności systemów plików do jądra systemu operacyjnego. Zastosowanie w tego sprawdzonego i przeznaczonego specjalnie do rozszerzania bezpieczeństwa szkieletu jest zgodne z założeniami twórców systemu Linux.

3.2.2. Mechanizmy wykorzystujące wieżowe systemy plików

Innym, równie skutecznym sposobem pozwalającym na włączenie mechanizmów ochrony integralności systemów plików do jądra systemu operacyjnego, jest wykorzystanie tzw. wieżowych systemów plików. Wieżowy system plików składa się z szeregu warstw, które są od siebie niezależne. Mechanizm ten pozwala na proste rozszerzanie funkcjonalności istniejących już systemów plików.

Budowa wieżowego systemu plików polega na nakładaniu na siebie kolejnych warstw, które nadpisują istniejące lub dodają nowe funkcje do istniejącego systemu plików leżącego w najniższej warstwie, zwanej fundamentem wieży [119]. Najpopularniejszym obecnie wieżowym systemem plików jest *UnionFS*, który został stworzonym na Uniwersytecie Stony Brook w USA w 2004 roku [45].

Dla klasycznego, jednolitego systemu plików wszelka zmiana funkcjonalności wymaga zmiany kodu źródłowego i ponownej instalacji jądra. Takie rozwiązanie ma sens jedynie wtedy, gdy tworzony jest całkowicie nowy system plików. Jest znanym zjawiskiem ciągła zmiana funkcjonalności jądra systemu operacyjnego, jak również ciągła modyfikacja systemów plików. Dlatego dla systemów operacyjnych, takich jak Linux, często udostępniane są nowe wersje jąder, których wdrożenie wymaga aktualizacji systemu operacyjnego.

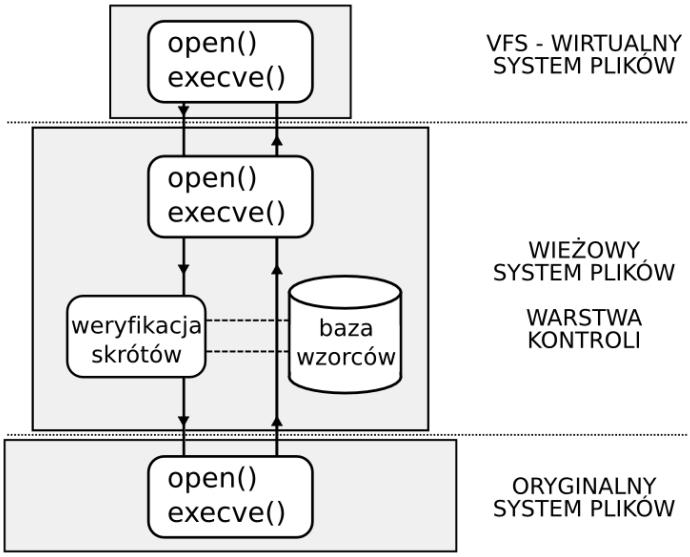
Wszystkie powszechnie stosowane systemy plików są ciągle rozwijane, dokładana jest nowa funkcjonalność, a także poprawiane są znalezione błędy. Przykładowo tylko w 2006 roku w jądrze systemu Linux dokonano 67 zmian w jego podstawowym systemie plików Ext3 [137].

Jeżeli zachodzi potrzeba jedynie rozszerzenia funkcjonalności systemu plików bardzo dobrym rozwiązaniem jest zastosowanie koncepcji wieżowych systemów plików. Rozwiązanie takie łączy w sobie łatwość tworzenia nowych funkcji w już istniejącym systemie plików oraz eliminuje konieczności ciągłej aktualizacji stworzonego oprogramowania w przypadku zmian w kodzie jądra systemu operacyjnego.

Ponieważ konstrukcja wieżowego systemu plików składa się ze zbioru niezależnie tworzonych warstw, każda nowa funkcjonalność może być zaimplementowana na osobnej warstwie [49]. Modyfikacja całego systemu plików sprowadza się w takiej sytuacji jedynie do domontowania nowej warstwy. Należy zauważyć, że nowa warstwa jest nadrzędna w stosunku do tych leżących poniżej. Oznacza to, że można nie tylko dodawać nowe funkcje, ale również modyfikować już istniejące, zaimplementowane na warstwach niższych. Modyfikacja może polegać nawet na wyłączaniu funkcji występujących na niżej położonych systemach plików.

Występuje tu pewien problem nazewnictwa. Pomimo, że przyjętym w informatyce tłumaczeniem angielskiego słowa *stack* jest stos w języku polskim, to w tym przypadku terminu *stackable filesystem* lepszym określeniem jest wieża. Informatyczny stos oznaczy zwykle ułożenie jednego elementu na drugim. Natomiast *stackable filesystem* pozwala na ułożenie jednej warstwy na innej, składającej się z więcej niż jednego elementu. Dlatego warto podkreślić tę funkcjonalność w słownictwie poprzez nazywanie takiego mechanizmu wieżowym systemem plików.

Wykorzystanie wieżowego systemu plików do zapewniania bezpieczeństwa i ochrony integralności systemów plików polega na stworzeniu specjalnie do tego celu przeznaczonej warstwy kontroli plików. Podczas montowania warstwa kontroli zostaje umieszczona na szczycie wieży nad kontrolowanym systemem plików. Schematyczna zasada działania takiego mechanizmu ochrony integralności plików zbudowanego jako wieżowy system plików została przedstawiona na rysunku 3.3.



Rysunek 3.3. Ochrona integralności plików z wykorzystaniem wieżowego systemu plików

W warstwie kontroli integralności konieczne jest umieszczenie dwóch funkcji systemowych odpowiedzialnych za dostęp do plików. Są to funkcje *open*, która jest wykorzystywana zawsze podczas otwierania pliku do czytania i pisania, oraz funkcja *execve* wywoływana każdorazowo przy otwieraniu pliku wykonywalnego, który ma zostać uruchomiony. Mechanizmy zaimplementowane w warstwie kontroli integralności, przed wykonaniem oryginalnych funkcji *open* i *execve* z przykrytego systemu plików, wykonują zaimplementowane dodatkowe operacje sprawdzenia integralności plików. Warstwa ta, wykorzystując bazę danych wzorców sprawdza, czy plik nie został zmodyfikowany. W momencie wykrycia nieprawidłowości, w zależności od możliwości konkretnych systemów ochrony plików, może zostać powiadomiony administrator o występującym zdarzeniu, a także może zostać zablokowany dostęp do pliku.

Zastosowanie wieżowego systemu plików, przy tworzeniu mechanizmu ochrony integralności plików, uwalnia projektanta od konieczności budowania całkowicie nowego systemu plików. Projektuje on i tworzy jedynie warstwę kontroli integralności. Pomimo znacznego ułatwienia, zadanie rozszerzenia możliwości systemu plików nadal pozostaje skomplikowane. Wymagane jest stworzenia niskopoziomowego kodu dla jądra systemu operacyjnego, co jest zawsze zadaniem skomplikowanym i wymagającym dużego doświadczenia praktycznego. Z tych względów opracowano kilka rozwiązań, które upraszczają tworzenie wieżowych systemów plików. Pierwszym rozwiązaniem, które zdobyło większą popularność, jest tzw. szablon o nazwie *Wrapfs*. Rozwiązanie to oferuje zestaw standardowych szablonów, które należy uzupełnić kodem źródłowym w odpowiednich, wskazanych miejscach. Tego typu szablony ukrywają wszelkie szczegóły implementacyjne charakterystyczne dla wieżowych systemów plików, ponieważ znaczna część kodu źródłowego jest już wypełniona. Pozwala to projektantowi na skoncentrowanie się jedynie na funkcjonalności realizowanej przez tworzone rozszerzenie [118]. Opracowano również specjalny język programowania o nazwie *FiST* (*File System Translator*), który w znaczący sposób ułatwia tworzenie wieżowych systemów plików. Jest to abstrakcyjny język wysokiego poziomu, który za pomocą dostarczanych narzędzi, może zostać przetłumaczony na język C. Stworzony w ten sposób kod modułu jest dodatkowo w pełni przenośny na wiele platform sprzętowych [120].

Wieżowe systemy plików mają liczne zalety, w szczególności pozwalają w łatwy sposób dokonywać modyfikacji systemów plików. Należy jednak podkreślić, że same w sobie nie tworzą funkcjonalności, a jedynie są platformą umożliwiającą włączenia rozszerzeń do jądra systemu operacyjnego Linux. Z wykorzystaniem wieżowych systemów plików powstało kilka rozwiązań zabezpieczania systemów operacyjnych, takich jak szyfrowany system plików *CryptFS* [117], system antywirusowy *Avfs* [84], czy system ochrony integralności systemów plików *I^3FS*, który zostanie opisany w dalszej części pracy.

3.3. Implementacja systemów ochrony integralności w jądrze

W licznych ośrodkach naukowych i działach badawczo-rozwojowych dużych przedsiębiorstw informatycznych prowadzone są badania nad stworzeniem mechanizmów zapewniających bezpieczeństwo systemów komputerowych. Jednym z ważniejszych kierunków badań jest zapewnienie bezpieczeństwa systemów plików. Jest to związane z faktem, że zadaniem praktycznie każdego ataku jest uzyskanie dostępu do danych zapisanych w systemach plików. Oczywiście cele ataków mogą być różne. Część intruzów usiłuje dokonać kradzieży poufnych

danych, takich jak hasła do rachunków bankowych, tajne plany rozwoju przedsiębiorstw czy inne cenne dane. Atak może mieć również na celu modyfikację danych, co ma miejsce podczas infekcji systemów wirusami komputerowymi, końmi trojańskimi czy podczas popularnych ostatnio nieautoryzowanych zmian stron internetowych. Często dochodzi również do ataków, których zadaniem jest zniszczenie plików zarówno użytkowych, jak i systemowych, co może prowadzić do dezorganizacji pracy zaatakowanej firmy czy utraty cennych danych. Co jakiś czas pojawiają się nowego rodzaju ataki na systemy plików. Jednym z nich są tzw. *ransomware*, czyli wirusy komputerowe, które po zainfekowaniu komputera szyfrują część danych zapisanych w systemach plików, a użytkownik może uzyskać klucz deszyfrujący po zapłaceniu pewnego rodzaju okupu [42, 77].

Uniemożliwienie intruzowi modyfikacji danych zapisanych w systemach plików pozwoliłoby na wyeliminowanie znacznej liczby ataków. Z tego powodu badania nad zapewnianiem bezpieczeństwa systemów plików wydają się szczególnie ważne.

Wszelkie próby implementacji mechanizmów ochrony integralności systemu plików na poziomie jądra systemu operacyjnego napotykają na swojej drodze podobne problemy. Oprócz wyboru metody włączenia ochrony na na najniższym poziomie systemu operacyjnego konieczne jest zapewnienie bezpieczeństwa i wydajności zarówno bazy danych jak i samemu systemowi ochrony.

3.3.1. Baza danych wzorców

Każde rozwiązanie zapewniające bezpieczeństwo systemu operacyjnego działające w oparciu o kryptograficzne skróty plików musi zawierać bazę danych, w której będą przechowywane obliczone wzorcowe sygnatury plików. Struktura takiej bazy, miejsce jej przechowywania, czy sposób dostępu do zgromadzonych w niej danych mogą być jednak różne. Baza wzorców może być zwykłym plikiem przechowywanym np. w głównym systemie plików. Takie rozwiązanie jest jednak mało bezpieczne. Jeżeli intruz uzyska możliwość modyfikacji plików w systemie, plik zawierający wzorce również może zostać dowolnie zmieniony lub nawet usunięty. W takiej sytuacji cały system ochrony integralności przestanie być skuteczny, a nawet może zostać całkowicie wyłączony. Zastosowanie dodatkowych mechanizmów ochrony bazy wzorców, takich jak np. cyfrowy podpis, eliminuje możliwość niezauważalnej zmiany pojedynczych wpisów w bazie. Nie daje to jednak gwarancji bezpieczeństwa całej bazy danych wzorców, ponieważ intruz może w dalszym ciągu uszkodzić plik zawierający wzorce lub po prostu go usunąć.

Bardziej bezpiecznym rozwiązaniem jest przechowywania bazy danych z wzorcami plików w przestrzeni jądra, jako część systemu operacyjnego. W przypadku Linuksa jądro jest kompilowane w momencie instalacji systemu operacyjnego na konkretnym sprzęcie komputerowym. Modyfikacja jądra nie może być dokonywana w trakcie pracy systemu operacyjnego. Jeśli zachodzi konieczność rozszerzenia funkcjonalności systemu lub jego aktualizacji do nowszej wersji, konieczna jest ponowna kompilacja jądra z plików źródłowych i restart komputera. Aktualnie większość dystrybucji przeznaczonych do użytku jako stacje robocze oferuje gotowe pakiety zawierające jądro systemu Linux, które nie wymaga konfiguracji i własnoręcznej kompilacji. W przypadku instalacji takiego pakietu konieczne jest jedynie ponowne uruchomienie komputera. W zastosowaniach serwerowych o znaczeniu krytycznym modyfikacje w najniższej warstwie systemu operacyjnego są przeprowadzane bardzo rzadko i z tego powodu najczęściej stosuje się własnoręcznie

konfigurowane i kompilowane pliki jąder. Zmiany mają miejsce jedynie po wykryciu poważnych błędów lub luk bezpieczeństwa. Z tych względów umieszczenie bazy wzorców jako części jądra systemu operacyjnego daje gwarancje, że nie będzie ona mogła być zmodyfikowana bez zatrzymania pracy całego systemu operacyjnego.

W przypadku baz danych wykorzystywanych w systemach ochrony integralności plików, problem ochrony danych przed nieautoryzowanym dostępem i modyfikacją staje się kluczowy dla zapewnienia skutecznego działania całego systemu bezpieczeństwa. Przeszukiwanie bazy wzorców w mechanizmie ochrony integralności plików to dodatkowe, często czasochłonne zadanie jądra systemu operacyjnego. Ochrona znacznej liczby plików wymaga ciągłego przeglądania bazy w poszukiwaniu ich wzorców. Jeżeli baza zostanie zorganizowana w postaci sekwencyjnego pliku tekstowego, to czas dostępu do poszczególnych krotek będzie stosunkowo długi. W takiej sytuacji implementacja mechanizmu zapewniania bezpieczeństwa może znacząco wpływać na spadek wydajności działania całego systemu operacyjnego. Dlatego w celu zwiększenia prędkości wyszukiwania danych w bazie wzorców stosuje się różne rozwiązania przyspieszające wyszukiwanie danych, m. in. tworzy się strukturę pliku zawierającego wzorce w postaci B-drzewa. Przy włączaniu bazy danych ze wzorcami plików do jądra systemu Linux nie można skorzystać z wydajnych i powszechnie stosowanych w przestrzeni użytkownika Systemów Zarządzania Bazą Danych (ang. *Database Management System*), takich jak DB2, MySQL itp., z uwagi na duży rozmiar tych aplikacji i ich wymagania. W jądrze systemu Linux można natomiast zastosować specjalnie zmodyfikowaną odmianę bazy danych Berkley DB o nazwie KBDB (*In-Kernel Berkeley DB Databases*) [65]. W rozwiązaniu tym wykorzystano struktury B+drzew, funkcje mieszające oraz dedykowane systemy kolejkowe, co pozwoliło na uzyskanie wysokowydajnego i skalowalnego rozwiązania.

Systemy zapewniające bezpieczeństwo działające w oparciu o porównywanie skrótów kryptograficznych wymagają szybkiego dostępu do wzorcowych skrótów kontrolowanych plików, które są zapisane w bazie danych. Zgodnie z filozofią działania systemów operacyjnych z rodziny UNIX, plik składa się z dwóch części, z tzw. metadanych w postaci i-węzła, w którym zapisane są informacje o pliku i miejscu przechowywania jego fizycznej zawartości na nośniku danych. W systemie operacyjnym plik jest identyfikowany na dwa sposoby, albo w postaci ścieżki dostępu wraz z pełną nazwą pliku nadawaną przez użytkownika, albo w postaci numeru i-węzła przypisanego plikowi w chwili jego zapisu na nośniku przez system operacyjny. Istniejące systemy ochrony integralności plików wykorzystują obydwa sposoby do identyfikacji chronionych plików. Ponieważ to administrator musi stworzyć listę plików, które będą chronione, ich identyfikacja na podstawie pełnej nazwy znacznie ułatwia procedurę wyboru i zarządzanie taką listą. Natomiast wykorzystywanie numeru i-węzła do identyfikacji plików chronionych zmniejsza liczbę operacji, jakie musi wykonać system operacyjny w momencie żądania dostępu do pliku. Eliminowany jest etap translacji nazwy pliku na numer jego i-węzła. Takie rozwiązanie zapewnia szybsze działanie systemu ochrony integralności plików działającego na poziomie jądra systemu operacyjnego.

3.3.2. Zastosowanie buforów podręcznych dla skrótów kryptograficznych

Systemy bezpieczeństwa działające w oparciu o wzorce kryptograficzne wymagają obliczania skrótu podczas każdego żądania dostępu do wybranego przez proces pliku. Operacja obliczania skrótu kryptograficznego pliku jest czasochłonna i w rezultacie, przy ochronie dużej liczby plików, mechanizm zapewniania bezpieczeństwa, może spowodować znaczące spowolnienie działania całego systemu operacyjnego. Problem ten był szczegółowo analizowany na uniwersytecie UFRGS w Brazylii, gdzie przeprowadzono badania nad wydajnością systemów ochrony integralności plików. Jak stwierdzono, w zależności od rodzaju procesora, czas dostępu do pliku kiedy działa mechanizm ochrony integralności plików zwiększyć się może od półtora do dziesięciu razy w porównaniu do tego samego systemu z wyłączonymi zabezpieczeniami [99]. Czas potrzebny na uzyskanie dostępu zwiększa się zatem znacząco i w takiej sytuacji szybkość działania systemu operacyjnego zostaje zmniejszona w sposób nieakceptowalny.

W celu poprawy wydajności systemów ochrony integralności plików można wykorzystać mechanizm polegający zapamiętywaniu wyniku operacji porównywania skrótów kryptograficznych. Dzięki temu, przy kolejnej próbie dostępu do uprzednio zweryfikowanego pliku, nie będzie ponownie wykonywana czasochłonna funkcja kryptograficzna, ani operacja pobierania wzorca z bazy danych.

Takie rozwiązanie, polegające na dodatkowym przechowywaniu często wykorzystywanych danych, jest powszechnie stosowane w dziedzinie informatyki i nosi nazwę mechanizmu buforów podręcznych (ang. *cache buffer*). Buforem podręcznym nazywany jest zbiór dodatkowych danych wykorzystywanych do zwiększania wydajności programów. Najczęściej przechowywane są kopie danych już pobranych z systemu komputerowego, których ponowne pozyskanie lub obliczenie wartości na ich podstawie jest bardzo czasochłonne. W buforach podręcznych umieszczane są głównie często pobierane dane, dzięki czemu można uzyskiwać do nich szybki dostęp. Tego typu rozwiązania są wykorzystywane nie tylko w programach użytkowych, ale również w nowoczesnych procesorach, które są wyposażane w szybkie pamięci podręczne (ang. *cache memory*) wykorzystywane do ograniczania odwołań do wolniejszej pamięci operacyjnej [72]. Pamięć podręczna jest współcześnie stosowana w systemach wieloprocesorowych, rozproszonych czy wielowarstwowych w celu poprawy komunikacji pomiędzy poszczególnymi węzłami systemu. W takich przypadkach konieczne jest zastosowanie rozwiązań zapewniających spójność pamięci cache [50].

W przypadku systemów ochrony integralności plików wystarczy przechowywać informację o tym, czy plik został już pozytywnie zweryfikowany czy nie. Taka informacja zapisana w buforze podręcznym nazywana jest bitem weryfikacji, gdyż może przyjmować tylko dwie wartości, np. 1 dla pliku już zweryfikowanego i 0 dla niezweryfikowanego.

Zastosowanie nawet najprostszej pamięci podręcznej w systemach ochrony integralności plików przynosi wymierne korzyści, ponieważ znacznie zmniejsza liczbę operacji obliczania skrótów kryptograficznych plików. Na uniwersytecie UCLA w Stanach Zjednoczonych przeprowadzono badania dotyczące wykorzystania buforów podręcznych w systemach ochrony integralności plików. Jako przedmiot badań, który objęto ochroną, został użyty kod źródłowy systemu operacyjnego Linux, który składał się z 9000 plików o rozmiarze 130 MB. Według przeprowadzonych badań podczas procesu kompilacji jądra tylko dla niecałych 2% wszystkich operacji dostępu do plików konieczne było obliczenie wzorca. Pozostałe 98% operacji zostało zweryfikowanych na podstawie buforów podręcznych. Jest to związane z

faktem, że podczas kompilacji tak dużego programu, jakim jest jądro, część plików jest otwierana bardzo często. Jednak w przypadku operacji kompresji tego samego kodu źródłowego, bufor podręczny jest praktycznie niewykorzystywany. Na 9000 operacji otwarcia pliku tylko 15 było zweryfikowanych na podstawie buforów, ponieważ przy kompresji plików większość plików jest otwierana tylko jeden raz. Jednak nawet w takim przypadku zastosowanie buforów podręcznych przynosi duże korzyści, gdyż przy ponownym wykonaniu operacji na tym samym zbiorze plików, nie będzie konieczności wykonywana operacji obliczania skrótów kryptograficznych [14].

Przy zastosowaniu mechanizmu cache występują sytuacje w których trzeba dokonywać zmian danych zapisanych w buforze. Dotyczy to sytuacji, w której dochodzi do modyfikacji danych w plikach chronionych lub informacji zapisanych w i-węźle. W takim przypadku konieczne jest wyczyszczenie bitu weryfikacji, czyli ustawienie go na wartość niezweryfikowany. Również po aktualizacji systemu i wygenerowaniu nowej bazy wzorców, konieczne jest wyczyszczenia bitów weryfikacji dla wszystkich chronionych plików.

Sposób implementacji czyszczenia buforów po zmianie pliku zależy od ogólnego sposobu włączenia ochrony integralności plików w jądrze. W przypadku zastosowania wywołań systemowych można wykorzystać np. funkcję Wirtualnego Systemu Plików *mark_inode_dirty*, która jest wywoływana wtedy, gdy plik został zmieniony lub usunięty. Jeśli natomiast zastosowany został wieżowy system plików, operacja taka powinna zostać włączona w funkcji *write* [125].

Istotny jest również wybór miejsca przechowywania buforów podręcznych. Mogą one być przechowywane w bazie podobnej do tej z wzorcami plików. Przy zastosowaniu wieżowych systemów plików ciekawym rozwiązaniem jest przechowywania bitu weryfikacji bezpośrednio w strukturze i-węzła. Dzięki temu nie ma potrzeby tworzenia, przechowywania i zarządzania dodatkową strukturą. Dostęp do wszystkich pól i-węzła jest w każdej funkcji systemu plików, czyli również podczas wykonywania funkcji *open* czy *execve*.

3.3.3. Przegląd systemów ochrony integralności plików na poziomie jądra

Z licznych dotychczas prowadzonych projektów implementujących systemy ochrony integralności plików na poziomie jądra systemu operacyjnego na wyróżnienie zasługują programy o nazwie *SOFFIC* oraz *I³FS*. Pierwszy z nich, *SOFFIC*, był rozwijany na uniwersytecie UFRGS (*Universidade Federal do Rio Grande do Sul*) w Porto Alegre w Brazylii. *I³FS* został stworzony w Stanach Zjednoczonych na uniwersytecie Stony Brook w Nowym Jorku. Oba systemy zapewniają bezpieczeństwo chronionych plików w oparciu o bazę ich kryptograficznych wzorców.

System *SOFFIC* (ang. *Secure On-the-Fly File Integrity Checker*) przechwytuje wywołania systemowe do funkcji odpowiedzialnych za dostęp do plików poprzez modyfikację źródeł jądra systemu operacyjnego. Modyfikacja ta jest rozprowadzana w postaci tzw. łaty na jądro (ang. *kernel patch*). Lista chronionych plików wraz z bazą ich wzorców jest przechowywana w zwykłym systemie plików. Taki sposób przechowywania kluczowych dla działania systemu bezpieczeństwa danych nie jest zbyt bezpieczny, z tego względu pliki są dodatkowo zabezpieczone przed niepowołaną modyfikacją za pomocą pary asymetrycznych kluczy kryptograficznych. W celu przyspieszenia działania systemu wykorzystywane są bufory podręczne dla ostatnio zweryfikowanych plików (*RVFC*, ang. *Recently Verified Files Cache*),

które są przechowywane w przestrzeni jądra systemu operacyjnego. Dane w buforach są czyszczone podczas operacji modyfikacji chronionych plików [99]. System ten jest stosunkowo prosty, m. in. nie zawiera złożonych mechanizmów ochrony bazy danych wzorców. Z powodu skomplikowanej instalacji oraz wolnego rozwoju, system *SOFFIC* jest stosunkowo mało popularny i rzadko wykorzystywany.

Innym interesującym systemem zapewniania bezpieczeństwa systemów plików jest projekt *I³FS* (ang. *An In-Kernel Integrity Checker and Intrusion Detection File System*), stworzony w roku 2003 na uniwersytecie Stony Brook. System ten został zaprojektowany z wykorzystaniem wieżowego systemu plików. Z tego względu jego architektura jest bardziej przejrzysta, a instalacja łatwiejsza. Do implementacji został wykorzystany język FiST. System przechowuje dane w charakterystyczny dla siebie sposób, w czterech oddzielnych bazach danych, korzystających z silnika KBDB. W pierwszej bazie przechowywane są dane dotyczące polityki bezpieczeństwa względem poszczególnych plików. Druga baza zawiera sumy kontrolne plików podlegających ochronie. Pozostałe dwie bazy zawierają odpowiednio dane z sumami kontrolnymi metadanych oraz statystykami żądań dostępu do pliku. Wszystkie bazy danych są szyfrowane za pomocą algorytm AES (ang. *Advanced Encryption Standard*).

System *I³FS* posiada zaawansowany mechanizm buforów podręcznych. Przechowywane są w nich nie tylko wyniki poprzednich operacji weryfikacji sum kontrolnych, ale także sumy kontrolne stron pamięci z zapisanymi plikami. Rozwiązanie to pozwala na przechowywanie w pamięci podręcznej danych o wszystkich plikach o rozmiarze do 5MB [87]. Należy zauważyć, że plikami chronionymi są zwykle pliki systemowe, które nie przekraczają tej wielkości.

Z tych dwóch systemów projekt *I³FS* wydaje się dojrzalszy. Zastosowana w nim architektura oparta na wieżowych systemach plików pozwala na prostsze i bardziej przejrzyste zaprojektowanie systemu niż przy zastosowaniu modyfikacji wywołań systemowych.

Pomimo znaczących zalet, zarówno system *SOFFIC* jak i system *I³FS* posiadają tę samą wadę jak *Tripwire*, pierwszy system ochrony integralności działający w przestrzeni użytkownika. W przypadku ataku na jądro systemu operacyjnego i jego modyfikacji lub w przypadku zmian dokonanych w bazie danych wzorców, ochrona integralności systemu plików może zostać wyłączona. Systemy chronią co prawda bazy danych wzorców za pomocą algorytmów szyfrujących, lecz takie zabezpieczenie pozwala jedynie na zablokowanie możliwości modyfikacji danych w niej zapisanych, ale nie chroni przed fizycznym usunięciem plików baz danych.

Celem pracy doktorskiej było stworzenie skutecznego sposobu zabezpieczania systemów operacyjnych poprzez zapewnianie integralności systemów plików. W ramach badań został stworzony mechanizm, który rozszerza koncepcje systemów ochrony integralności plików o fizyczne zabezpieczenie kluczowych do działania tych systemów danych. Przy wykorzystaniu niemodyfikowalnych nośników danych, takich jak np. płyty CD-ROM stworzony system może chronić w sposób sprzętowy jądro systemu operacyjnego wraz z mechanizmem ochrony integralności plików oraz bazę danych wzorców. Z tego powodu opracowany mechanizm pozwala na całkowitą i zarazem skuteczną ochronę wybranych plików zapisanych w systemie komputerowym.

Rozdział 4.
Model zapewniania bezpieczeństwa systemu operacyjnego

Zapewnienie bezpieczeństwa systemów operacyjnych jest niezwykle trudnym i ważnym problemem informatyki. Nad tym zagadnieniem pracuje wiele ośrodków naukowych, jak również liczne firmy informatyczne, które opracowują, wytwarzają i wprowadzają na rynek różnego rodzaju systemy zabezpieczeń.

Na działający system operacyjny, zgodny ze standardem POSIX, składa się zbiór różnego typu plików oraz zbiór uruchomionych procesów [6]. Zapewnienie pełnego bezpieczeństwa systemu operacyjnego jest możliwe tylko wtedy, gdy ochroną zostanie objęta zarówno pamięć wraz działającymi procesami jak również dane zapisane w postaci plików.

Analizując dotychczasowe praktyki w dziedzinie zabezpieczania systemów komputerowych należy stwierdzić, że nie można całkowicie wyeliminować możliwości uzyskania przez intruza dostępu do systemu operacyjnego. Jednak ograniczanie ryzyka w informatyce sprowadza się nie tylko do eliminacji niepożądanych zdarzeń, ale również do redukcji negatywnych efektów wystąpienia takiego zdarzenia. Z tego powodu ochrona systemów komputerowych, oprócz działań prewencyjnych, obejmuje również wykrywanie i reagowanie na zagrożenia [16].

Celem niniejszej pracy jest stworzenie mechanizmu, który nawet w przypadku przejęcia przez intruza kontroli nad systemem operacyjnym, nie pozwoli na dokonanie uszkodzeń danych w nim zapisanych. Realizacja tego typu zabezpieczenia jest możliwa poprzez zapewnienie bezpieczeństwa systemowi plików, w którym zapisane są dane systemowe, konfiguracyjne, programy użytkowe i dane tworzone przez użytkowników. Niniejsza praca nie będzie zatem zajmować się ochroną procesów, dzięki którym intruz może uzyskać dostęp do systemu operacyjnego, jak również próbami eliminacji błędów w programach użytkowych.

Przed przystąpieniem do projektowania mechanizmów zabezpieczania systemów operacyjnych konieczne jest zdefiniowanie przedmiotu, celu i sposobu ochrony. W związku z tym należy przeprowadzić analizę, która da odpowiedzi na następujące pytania:

- Jaki składnik systemu będzie chroniony?
- Dlaczego należy go chronić?
- W jaki sposób ochrona będzie prowadzona?

Przedmiotem ochrony proponowanego mechanizmu zabezpieczeń jest system plików, który jest definiowany jako zorganizowany zbiór danych wraz z mechanizmami ich opisu, przeszukiwania, zarządzania i składowania na nośnikach fizycznych. Istnieje wiele systemów plików związanych z różnymi systemami operacyjnymi. Różnorodność tych systemów wynika z ich historycznego rozwoju, a także z różnic w ich planowanych zastosowaniach. Do najbardziej rozpowszechnionych systemów plików można zaliczyć FAT

(ang. *File Allocation Table*), NTFS (ang. *New Technology File System*), ext (ang. *Extended File System*) oraz HFS (ang. *Hierarchical File System*). Dane są przechowywane w formie plików i mogą to być zarówno zbiory danych systemowych i konfiguracyjnych oraz dane wygenerowane przez użytkowników. Pliki mogą być fizycznie zapisane na różnych nośnikach danych, takich jak dyski twarde, pamięci półprzewodnikowe Flash czy płyty CD-ROM lub DVD.

Cechą charakterystyczną systemów operacyjnych opartych o architekturę UNIX są liczne typy plików. Za pliki uważa się bowiem nie tylko pliki z danymi, ale również katalogi, a nawet urządzenia znakowe i blokowe. Oznacza to, że według takiego podejścia, za pomocą plików w systemie operacyjnym identyfikowane są m. in. klawiatura, monitor, drukarka czy dysk twardy. Każdy plik, zarówno zwykły zawierający dane, jak i specjalny np. identyfikujący urządzenia, posiada nazwę i jednoznacznie zdefiniowaną strukturę zawierającą tzw. metadane, takie jak wielkość, datę utworzenia, prawa dostępu, czy miejsce składowania danych na określonym nośniku.

System plików powinien być zabezpieczany przed atakami, nie dlatego, że jego modyfikacja może prowadzić do uzyskania dostępu do systemu operacyjnego, ale dlatego, że zmiany danych w zapisanych plikach mogą zmieniać konfigurację systemu i niszczyć ważne dane użytkowników. Można wyodrębnić trzy główne cele ataków intruza na system plików:

- zmiana danych, zarówno użytkowych, jak i systemowych,
- zniszczenie zapisanych danych i całego systemu plików,
- kradzież danych przechowywanych w plikach.

Wielkość strat wynikająca z kradzieży, zniszczenia czy modyfikacji danych jest bardzo trudna do oszacowania. Należy rozważyć, iż użytkownicy mogą w różny sposób oceniać wartość oraz istotność zgromadzonych przez siebie informacji. Bezsprzecznym jest fakt, iż uszkodzenie plików systemowych stwarza ogólne zagrożenie związane z niemożnością prawidłowego działania systemu. Tymczasem dane zgromadzone przez samych użytkowników mogą być oceniane w wymiernej sferze finansowej bądź też subiektywnie ujętej sferze sentymentalnej. Uszkodzenie danych subiektywnie istotnych dla użytkownika najczęściej nie ma wymiaru finansowego i nie wiąże się z zauważalnym stratami pieniężnymi, jednocześnie będąc dla zainteresowanego szczególnym obciążeniem wiążącym się z rozczarowaniem i niezadowoleniem z powstałej szkody. Na zupełnie odmiennej płaszczyźnie należy dokonywać analizy zniszczeń dokonywanych w sferze związanej z powstaniem wymiernych szkód finansowych. Tak jest na przykład w przypadku kradzieży pieniędzy, kiedy to modyfikacja danych zapisanych w systemach komputerowych może prowadzić do wymiernych strat, na przykład w przypadku włamania na serwery banków czy innych instytucji finansowych. Nawet tak prozaiczny atak, jak podmiana zawartości strony firmowej może prowadzić do spadku zaufania i w konsekwencji utraty klientów.

Istotnym zagrożeniem jest modyfikacja plików systemowych i konfiguracyjnych, która może prowadzić do uzyskania stałego dostępu do systemu operacyjnego. Dlatego jednym z kluczowych zagadnień zabezpieczania systemu operacyjnego jest zapewnienie ochrony dla plików systemowych i konfiguracyjnych. Taka ochrona ogranicza między innymi jeden z typów ataków zwany metodą tylnych drzwi (ang. *backdoor*). Instalacja tego typu programów, po udanym włamaniu do systemu, jest częstą praktyką, która pozwala intruzowi na stały dostęp do systemu poza kontrolą administratorów, bez konieczności przeprowadzania, często skomplikowanych, kolejnych ataków na sam system operacyjny.

Opracowany model mechanizmu zabezpieczania systemów operacyjnych rozszerza koncepcję systemów ochrony integralności o dwa innowacyjne elementy: wykorzystanie do zapisu krytycznych danych nośników ze sprzętową blokadą zapisu oraz automatyczne przywracanie zmodyfikowanych plików z kopii zapasowych.

Blokada zapisu danych na poziomie sprzętowym gwarantuje, że zmiana danych zapisanych na tych nośnikach będzie możliwa tylko i wyłącznie poprzez fizyczny dostęp do komputera i do urządzeń zapisujących dane. Dla przykładu dane systemowe zapisane na płycie CD-ROM nie mogą być w żaden sposób zmienione. Taka modyfikacja jest możliwa, ale tylko po przygotowaniu nowej płyty i jej podmianie w komputerze. Podobnie niektóre typu pamięci przenośnych USB posiadają przełącznik fizycznie blokujący zapis. Zmiana danych zapisanych na takim nośniku jest możliwa, ale tylko wtedy kiedy administrator znajduje się przy komputerze i w sposób fizyczny przełączy nośnik w tryb do zapisu. Takiej operacji nie można wykonać zdalnie, np. przez sieć komputerową. Istnieje również możliwość fizycznego zablokowania zapisu i modyfikacji danych zapisanych na dyskach twardych poprzez wykorzystanie specjalnie zmodyfikowanych złączy (np. *ICS ImageMasster DriveLock IDE*).

Automatyczne przywracania zawartości zmodyfikowanych plików z kopii przechowywanych na niemodyfikowalnych nośnikach danych zapewnia ciągłość pracy systemu operacyjnego i chroni użytkowników przed uruchamianiem złośliwego oprogramowania.

Podsumowując, podstawowe założenia przyjęte dla budowy modelu zabezpieczenia systemu operacyjnego to:

1. Chroniony jest system plików, a nie pamięć operacyjna i uruchomione procesy.
2. Kopie chronionych plików wraz z bazą danych wzorców kryptograficznych oraz systemem zabezpieczeń umieszczone są na niemodyfikowalnych nośnikach danych.
3. System zabezpieczeń działa z poziomu jądra systemu operacyjnego, którego jest integralną częścią.
4. W przypadku wykrycia zmiany w pliku, jego zawartość jest automatycznie przywracana z kopii zapasowej.

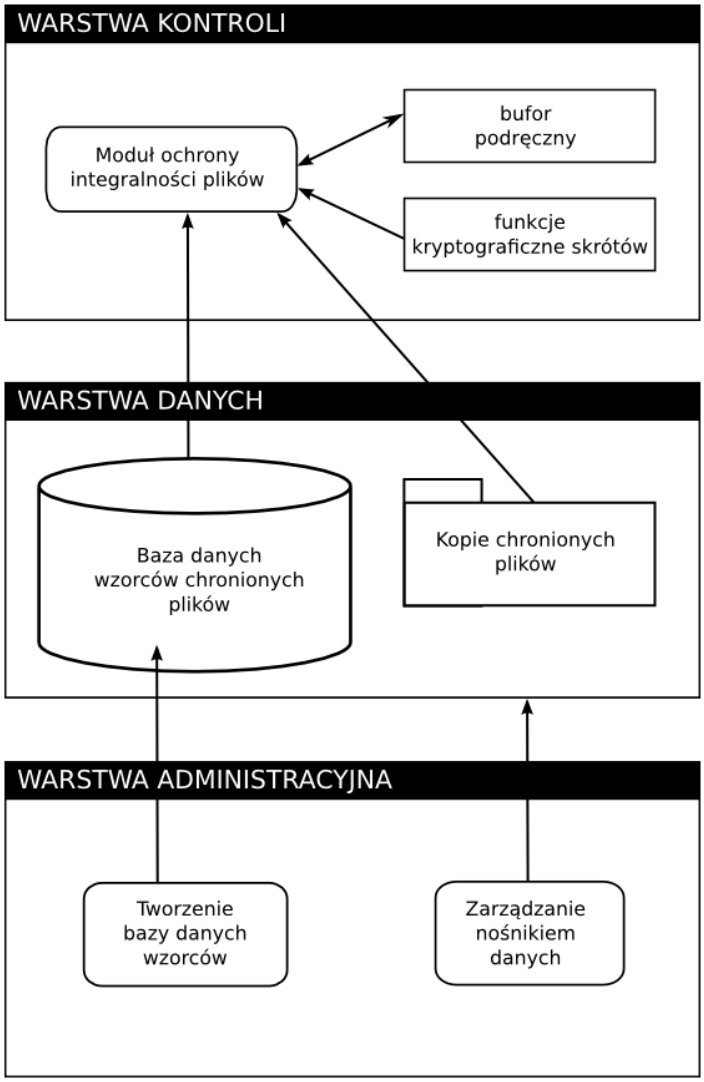
Opracowany w ramach niniejszej pracy mechanizm zabezpieczania systemów komputerowych uniemożliwia wykonywanie w sposób nieautoryzowany jakichkolwiek modyfikacji zawartości plików przeznaczonych do ochrony.

4.1. Model mechanizmu zabezpieczania systemu plików

W oparciu o przyjęte założenia mówiące, że chroniony jest system plików, wykorzystywane są niemodyfikowalne nośniki danych oraz system działa na poziomie jądra systemu operacyjnego, zaproponowano trójwarstwowy model mechanizmu zabezpieczania systemu plików, które przedstawiono na rysunku 4.1.

W skład pierwszej warstwy, zwanej warstwą kontroli, wchodzi moduł ochrony integralności systemu plików, który jest dołączony do jądra systemu operacyjnego. Podczas uruchamiania systemu operacyjnego jest wprowadzany do pamięci RAM. Moduł ten jest odpowiedzialny za sprawdzanie, czy nie doszło do próby zmiany danych w chronionych plikach. Po wykryciu modyfikacji moduł przywraca oryginalną zawartość chronionych plików z kopii zapasowej. Procedura kontroli jest każdorazowo uruchamiana w momencie żądania dostępu do pliku

przez dowolny proces. Moduł ochrony korzysta podczas obliczania skrótów ze zbioru funkcji kryptograficznych udostępnianych przez jądro. Częścią warstwy kontroli jest również bufor podręczny. Służy on do przechowywania rezultatów już wykonanych operacji kontroli dostępu do plików. Wykorzystanie buforów podręcznych przyspiesza działanie systemu ochrony, ponieważ eliminuje konieczność obliczania skrótów kryptograficznych dla plików uprzednio zweryfikowanych.



Rysunek 4.1. Powiązania pomiędzy komponentami mechanizmu zabezpieczeń

W drugiej warstwie modelu przechowywane są kluczowe dane przeznaczone dla mechanizmu zabezpieczeń. W skład tej warstwy wchodzi baza danych, w której zapisane są kryptograficzne wzorce chronionych plików. Na podstawie tych danych system ochrony jest w stanie stwierdzić, czy chroniony plik został zmodyfikowany. Warstwa danych zawiera również kopie chronionych plików. W przypadku wykrycia jakichkolwiek zmian w plikach chronionych, będzie możliwe przywrócenie ich oryginalnej zawartości. Należy podkreślić, że zgodnie z założeniami przyjętymi przy budowie tego modelu, baza danych wzorców kryptograficznych oraz kopie chronionych plików znajdują się na niemodyfikowalnych nośnikach danych.

Trzecią warstwę modelu zabezpieczeń stanowią narzędzia ułatwiające administratorom wdrożenie i utrzymanie systemu zabezpieczeń. Wśród opracowanych narzędzi znajdują się programy do tworzenia nowej bazy wzorców, do wyboru plików chronionych oraz programy pozwalające na kontrolowanie integralności systemu plików. Rezultaty działania modułu ochrony integralności plików są przekazywane administratorowi w postaci komunikatów alarmowych. Informacje o operacjach przywracania plików z kopii zapasowych, wykonanych przez system zabezpieczeń, mogą pozwolić administratorom na wykrycie i identyfikację zdarzenia polegającego na włamaniu intruza do systemu operacyjnego.

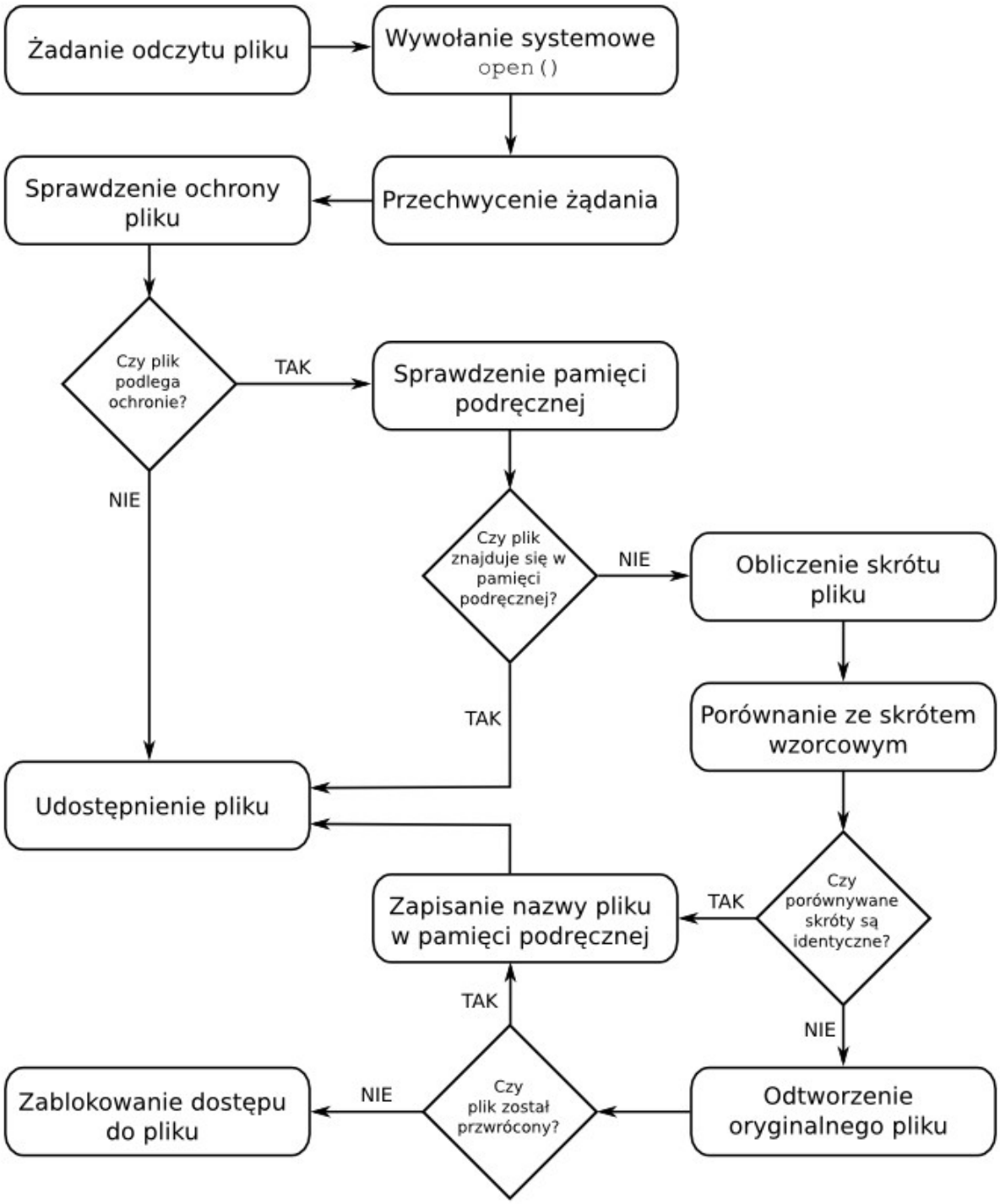
Przygotowanie danych dla systemu zabezpieczeń musi być wykonywane przy zachowaniu szczególnych środków ostrożności. Wskazane jest odłączenie systemu komputerowego od infrastruktury sieciowej i dokonanie sprawdzenia integralności całego systemu plików. Dopiero po wykonaniu tych czynności można wyłączyć system zabezpieczeń i wykonać aktualizację danych w systemie operacyjnym. Po przygotowaniu aktualizowanych danych dla systemu zabezpieczeń należy je zapisać na niemodyfikowalnych nośnikach danych. Zapisywanie danych na nośnikach niemodyfikowalnych wymaga fizycznego dostępu do komputera, a zatem nie może być wykonywana zdalnie np. poprzez sieć Internet. Takie rozwiązanie zapewnia skuteczność ochrony systemu operacyjnego.

W oparciu o zaproponowany model został zaprojektowany algorytm działania mechanizmu zabezpieczeń, który przedstawiono w postaci schematu blokowego na rysunku 4.2 [57]. Należy podkreślić, że przedstawiony algorytm działania systemu zabezpieczeń jest ogólny i może zostać zaimplementowany w dowolnym systemie operacyjnym.

Mechanizm zabezpieczania systemu plików jest uruchamiany przy każdorazowym żądaniu dostępu do chronionego pliku przez dowolny proces uruchomiony w systemie operacyjnym. Taka funkcjonalność może być osiągnięta na różne sposoby, zależne od specyfiki systemu operacyjnego, a także przyjętych założeń projektowych. Dla systemu operacyjnego Linux najpopularniejszymi metodami realizującymi tę funkcjonalność jest przechwytywanie wywołań systemowych lub zastosowanie wieżowych systemów plików.

W przypadku każdej operacji odczytu lub wykonania pliku przez proces, mechanizm zabezpieczania w pierwszej kolejności sprawdza, czy plik ten został włączony do grupy plików podlegającej ochronie. Wykorzystywane są w tym celu informacje zapisane w bazie danych wzorców. Jeśli w bazie nie ma rekordu dotyczącego sprawdzanego pliku, oznacza to, że nie podlega on ochronie i dostęp do pliku jest przyznawany procesowi. Jeżeli jednak plik, do którego żąda się dostępu, podlega ochronie, mechanizm zabezpieczeń sprawdza, czy zawartość pliku nie uległa modyfikacji od momentu uruchomienia systemu. Sprawdzanie polega na obliczeniu skrótu kryptograficznego dla aktualnej zawartości pliku i porównaniu rezultatu z wzorcową wartością zapisaną w bazie danych.

Pomimo zastosowania nowoczesnych algorytmów kryptograficznych operacja obliczania skrótów dla plików jest czasochłonna. Przy każdorazowym obliczaniu skrótów dla plików chronionych następuje spadek wydajności systemu operacyjnego. Zwiększenie wydajności systemu zabezpieczeń można uzyskać poprzez wyeliminowanie zbędnych operacji obliczania skrótów kryptograficznych. Jeśli przy pierwszym dostępie plik został zweryfikowany pozytywnie, to dopóki jego zawartość nie zostanie zmieniona, skrót kryptograficzny pozostaje ten sam i nie ma potrzeby jego ponownego obliczania. W związku z tym zaproponowano zastosowanie buforów podręcznych, w których przechowywane są informacje o poprawnie zweryfikowanych plikach.



Rysunek 4.2. Schemat blokowy algorytmu ochrony plików.

Zgodnie z algorytmem ochrony, po potwierdzeniu, że plik podlega ochronie, sprawdzane jest w buforze podręcznym, czy został on już poprawnie zweryfikowany. Jeżeli tak, to proces obliczania skrótu jest pomijany i dostęp do pliku jest przyznawany odpowiedniemu procesowi.

Jeśli w buforze podręcznym nie ma informacji, że plik był już sprawdzony, przeprowadzana jest pełna procedura kontroli. Obliczany jest skrót kryptograficzny dla pliku na podstawie jego aktualnej zawartości, a następnie otrzymany rezultat jest porównywany z wartością wzorcową zapisaną w bazie danych. Po poprawnej weryfikacji mechanizm zabezpieczeń zapisuje w buforze podręcznym informację o pozytywnym wyniku operacji kontroli danego pliku. W kolejnym kroku mechanizm zabezpieczeń zezwala na dostęp do pliku.

Jeżeli obliczony skrót kryptograficzny nie jest identyczny ze skrótem wzorcowym przechowywanym w bazie danych, uruchamiana jest procedura ochronna. W pierwszym etapie system podejmuje próbę przywrócenia oryginalnej zawartości zmodyfikowanego pliku z kopii zapasowej. Oryginalna zawartość pliku jest kopiowana do systemu plików znajdującego się na dysku twardym komputera z niemodyfikowalnego nośnika danych. W ten sposób zmieniony przez intruza plik jest zastępowany przez plik poprawny. W przypadku udanej operacji przywracania poprawnej zawartości pliku chronionego, dalsze kroki algorytmu są takie same, jak przy pozytywnej weryfikacji pliku. Informacja o pozytywnej weryfikacji pliku jest również zapisywana w buforze podręcznym, a dostęp do pliku jest przyznawany procesowi.

Jeśli jednak, z różnych powodów, przywrócenie poprawnej zawartości z kopii zapasowej nie jest możliwe, mechanizm zabezpieczeń blokuje dostęp do żądanego pliku. Takie działanie algorytmu daje gwarancję, że żaden użytkownik systemu nie odczyta ani nie uruchomi pliku chronionego, który został zmodyfikowany w sposób nieautoryzowany. Taki plik nie będzie mógł być również użyty przez żaden proces działający w systemie operacyjnym.

Reakcja systemu zabezpieczeń po negatywnej weryfikacji zawartości pliku chronionego, może być różna w zależności od przyjętej polityki bezpieczeństwa. System zabezpieczeń może zapisywać informację o zaistniałej sytuacji w plikach logów systemowych, informować administratora za pomocą poczty elektronicznej, czy wiadomości SMS, zablokować dostęp z określonych adresów IP, a nawet całkowicie zamknąć system i wyłączyć komputer. Takie zdecydowane reakcje systemu zabezpieczeń mogą być dopuszczalne przez politykę bezpieczeństwa przy występowaniu z dużą częstotliwością niepoprawnych weryfikacji, świadczących o zmasowanym ataku na system operacyjny.

Działania mechanizmu zabezpieczeń systemu plików są przeźroczyste dla użytkowników systemu operacyjnego, co oznacza, że nie muszą oni wiedzieć o jego istnieniu. Użytkownik może nie zauważać, że plik, do którego jego proces się odwołał, został skontrolowany czy też, że jego zawartość została przywrócona z kopii zapasowej. Jedynie administrator ma dostęp do informacji o działaniach jakie podejmował system zabezpieczeń, gdyż są one każdorazowo zapisywane w plikach logów systemowych.

4.2. Zasady wyboru plików do ochrony

Polityka zapewniania bezpieczeństwa systemów operacyjnych jest prowadzona przez administratorów i zależy od wagi przechowywanych danych i wielkości potencjalnych strat wynikających z przeprowadzenia przez intruza skutecznego ataku. Z polityki bezpieczeństwa przyjętej przez administratora systemu wynika rodzaj i liczba plików przeznaczonych do ochrony. Zapewnienie ochrony dla wszystkich zapisanych plików jest praktycznie niemożliwe, z uwagi na nieunikniony spadek szybkości działania systemu operacyjnego.

Ponadto liczne pliki są bardzo często modyfikowane, np. pliki tymczasowe, czy pliki związane z uruchomionymi programami użytkowymi, w których zapisywane są bieżące rezultaty pracy.

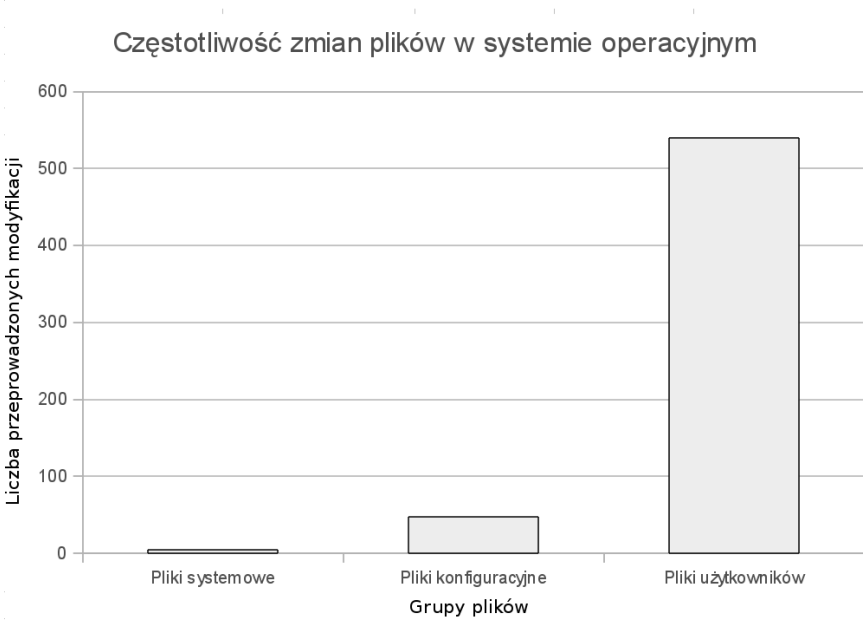
Zastosowanie nawet prostej klasyfikacji plików pozwala na opracowanie indywidualnej polityki bezpieczeństwa dla każdej z wyróżnionych grup. W związku z tym, na potrzeby opracowanego mechanizmu, podzielono pliki zapisane w systemie operacyjnym na trzy grupy:

- pliki systemowe – pliki programów tworzących system operacyjny,
- pliki konfiguracyjne – ustawienia programów systemowych i użytkowych,
- pliki użytkowników – dane wytwarzane przez użytkowników.

Podział ten został dokonany na podstawie częstości modyfikacji plików oraz ich znaczenia dla bezpieczeństwa systemu komputerowego. Należy podkreślić, że taka klasyfikacja jest naturalna i np. w systemach zgodnych ze standardem POSIX ma swoje odbicie w strukturze drzewa katalogów.

Do pierwszej grupy zaliczane są pliki jądra systemu operacyjnego, bibliotek i programów systemowych, a także wykonywalnych programów użytkowych. Pliki zawierające ustawienia programów systemowych takich jak serwery, usługi lub demony, a także globalną konfigurację środowiska roboczego, zostały zakwalifikowane do grupy plików konfiguracyjnych. W grupie umownie nazwanej plikami użytkowników, oprócz plików tworzonych w programach użytkowych, znajdują się również pliki tymczasowe programów, które mogą być zmieniane w dowolnej chwili przez działający program.

Pliki systemowe, w poprawnie zainstalowanym systemie operacyjnym zmieniają się niezwykle rzadko. Taka sytuacja ma miejsce głównie po wykryciu błędów w programach, konieczności instalacji nowych aplikacji lub aktualizacji ich do najnowszej wersji. Dane w plikach konfiguracyjnych mogą być modyfikowane częściej. Zależy to od wielkości systemu operacyjnego i doświadczenia administratorów. Jednak najczęściej zmienianymi plikami są dane użytkownika.



Rysunek 4.3. Częstotliwość zmian plików w systemie operacyjnym

Z drugiej strony pliki użytkowników zmieniają się bardzo często i dlatego nie powinny być zabezpieczane przez system operacyjny, lecz przez ich właścicieli. To ich zadaniem jest tworzenie regularnych kopii zapasowych z najistotniejszymi danymi.

Przeprowadzony został eksperyment, podczas którego zbadano liczbę dokonanych w ciągu 6 miesięcy modyfikacji plików w systemie operacyjnym Debian zainstalowanym na komputerze, który pełnił rolę serwera WWW. Na podstawie badania, z pewnym przybliżeniem można przyjąć, że częstotliwość zmian w plikach użytkownika jest o rząd wielkości większa niż w plikach konfiguracyjnych, która jest z kolei większa o rząd wielkości niż dla plików systemowych. Wyniki eksperymentu zostały zilustrowane na rysunku 4.3.

Przy zabezpieczaniu systemu operacyjnego Linux, należy uwzględnić jego standardowy podział plików i katalogów. Drzewo katalogów w większości dystrybucji systemu Linux wygląda tak samo lub bardzo podobnie. Standard definiujący ściśle określoną hierarchię drzewa katalogów i nazwy poszczególnych katalogów nosi nazwę *Linux Standard Base Filesystem Hierarchy Standard* (*LSB FHS*) [97]. Pliki są pogrupowane nie względem ich pochodzenia lecz według ich funkcji. Oznacza to, że pliki jednego tylko programu mogą być zapisane w różnych katalogach, np. pliki konfiguracyjne w katalogu /etc/, pliki bibliotek /usr/lib/, czy pliki wykonywalne w /usr/bin/. W tabeli 4.1 została przedstawiona standardowa struktura drzewa katalogów w systemie Linux wraz z przypisaniem poszczególnych katalogów do jednej z trzech, zdefiniowanych wcześniej grup.

Niektóre katalogi nie zostały przypisane do żadnej z trzech grup, gdyż są nieistotne z punktu widzenia ochrony przed modyfikacją. Są to katalogi, w których nie są przechowywane dane, lecz np. wirtualne pliki jądra (katalogi *sys*, *proc*), dynamicznie tworzone urządzenia (katalog *dev*), czy punkty montowania innych systemów plików (katalogi *media*, *mnt*). Również katalog przeznaczony do przechowywania plików tymczasowych (*tmp*), z uwagi na swoją naturę, nie podlega żadnej ochronie.

W celu zapewnienia bezpieczeństwa działania systemu operacyjnego dla każdej ze zdefiniowanych grup plików należy zastosować inną politykę ochrony. Zastosowane środki bezpieczeństwa zależą od częstotliwości zmian danych w plikach oraz przewidywanych rozmiarów potencjalnych strat, jakie mogą wyniknąć z powodu ich modyfikacji lub zniszczenia.

Z punktu widzenie ochrony całego systemu komputerowego kluczowym zagadnieniem jest zapewnienie bezpieczeństwa plikom systemowym. Modyfikacja zawartości plików systemowych przez intruza może doprowadzić do przejęcia kontroli nad systemem operacyjnym lub zainstalowania wrogich programów. W związku z tym, według zalecanego modelu bezpieczeństwa należy całkowicie zablokować możliwość modyfikacji plików systemowych, zwłaszcza w systemach przeznaczonych do zastosowań serwerowych. W historii informatyki znane są rozwiązania, w których sam system operacyjny zapisywany był w sposób niemodyfikowalny bezpośrednio w układzie scalonym, jak miało to miejsce np. w komputerach ośmiobitowych, takich jak np. Spectrum.

Część danych przydzielonych do grupy plików systemowych można również zakwalifikować jako część tzw. „godnej zaufania bazy obliczeniowej” (ang. *Trusted Computing Base*, *TCB*), która jest definiowana jako zbiór wszystkich mechanizmów, zarówno sprzętowych jak i programowych, odpowiedzialnych za bezpieczeństwo. W skład takiej bazy wchodzą wszystkie komponenty, których niepożądane działanie może doprowadzić do zagrożenia

bezpieczeństwa [2], czyli m.in. jądro systemu operacyjnego oraz pliki programów systemowych związanych z zapewnieniem poufności i integralności. W związku z tym, że jednym z wymagań dla TCB jest tzw. samo-ochrona (ang. *self-protecting*), zablokowanie możliwości modyfikacji plików wchodzących w skład bazy jest pożądane i całkowicie uzasadnione. Oczywiście zastosowanie opracowanego mechanizmu do ochrony kluczowych plików w systemie operacyjnym Linux nie spowoduje, że będzie on spełniał wszystkie postulaty *TCB*. Dla tego systemu operacyjnego powstało kilka systemów wykorzystujących opracowaną przez *Trusted Computing Group* platformę *TPM*, wśród których można wymienić UCLinux, TegLinux oraz Enforcer [73].

Tabela 4.1. Drzewo katalogów wg standardu LSB FHS

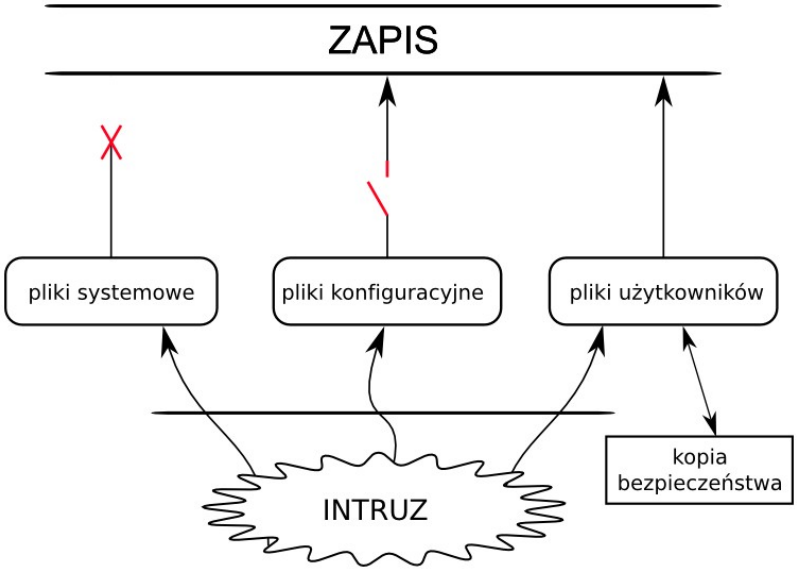
KATALOG	OPIS	GRUPA
bin	Podstawowe programy użytkowe	PLIKI SYSTEMOWE
boot	Pliki programu rozruchowego	PLIKI SYSTEMOWE
dev	Pliki urządzeń	
etc	Pliki konfiguracyjne programów	PLIKI KONFIGURACYJNE
home	Dane użytkowników	PLIKI UŻYTKOWNIKÓW
lib	Współdzielone biblioteki i moduły jądra	PLIKI SYSTEMOWE
media	Punkt montowania	
mnt	Tymczasowy punkt montowania	
opt	Dodatkowe oprogramowanie	PLIKI SYSTEMOWE
root	Pliki użytkownika root	PLIKI UŻYTKOWNIKÓW
proc	Wirtualny system plików jądra	
sbin	Podstawowe programy systemowe	PLIKI SYSTEMOWE
sys	Dane systemowe	
tmp	Pliki tymczasowe	
usr	Pliki programów użytkowych	PLIKI SYSTEMOWE
var	Dane tymczasowe aplikacji	PLIKI UŻYTKOWNIKÓW

Drugą grupę, którą należy rozpatrywać przy zapewnianiu bezpieczeństwa systemu operacyjnego stanowią pliki konfiguracyjne. Są one modyfikowane znacznie częściej od plików systemowych, a jednocześnie ich nieautoryzowana modyfikacja nie prowadzi do tak groźnych skutków, jak w przypadku modyfikacji plików systemowych. W związku z tym, w proponowanym modelu bezpieczeństwa należy uwzględnić możliwość okresowej zmiany danych zapisanych w tych plikach. Okresowa zmiana zawartości plików konfiguracyjnych wynika z prac administracyjnych, polegających na wprowadzaniu nowych ustawień programów, dodawaniu użytkowników do systemu czy zmiany ustawień sieciowych. Do administratorów serwerów należy wybór niektórych plików konfiguracyjnych, które będą podlegać całkowitej ochronie w zaprojektowanym systemie zabezpieczeń. Dla pozostałych plików konfiguracyjnych należy stworzyć mechanizm, który umożliwi ich zapisywanie na modyfikowalnych nośnikach danych z fizyczną blokadą zapisu. Zastosowanie takich urządzeń pozwala administratorom na zmianę plików konfiguracyjnych po odblokowaniu możliwości zapisu. Taka polityka wiąże się z dodatkowymi pracami administracyjnymi, ale jednocześnie zapewnia bezpieczeństwo systemu komputerowego.

Trzecią grupę, umownie nazywaną plikami użytkowymi, stanowią pliki z zapisanymi danymi wygenerowanymi przez użytkowników, pliki programów użytkowych oraz pliki tymczasowe aplikacji. Są one bardzo często modyfikowane i w związku z tym nie jest wskazany wybór tych plików do ochrony w zaproponowanym mechanizmie bezpieczeństwa. Pliki użytkowe powinny być regularnie archiwizowane, jak również powinny zostać stworzone odpowiednie procedury umożliwiające ich szybkie przywracanie po wykryciu włamania lub uszkodzenia. Zgodnie z regułami archiwizacji tego typu dane mogą być przechowywane zarówno w postaci kopii pełnych, przyrostowych jak i różnicowych.

Mechanizm zapewniający bezpieczeństwo systemów operacyjnych powinien działać niezależnie od liczby i typów plików wybranych do ochrony. Natomiast wybór liczby plików powinien być poprzedzony analizą wydajności działania systemu operacyjnego. Chodzi o to, żeby mechanizm zapewniający bezpieczeństwo w sposób znaczący nie zmniejszał szybkości działania systemu operacyjnego.

Na rysunku 4.4 przedstawiona została schematycznie proponowana polityka bezpieczeństwa.



Rysunek 4.4. Proponowana polityka zabezpieczania danych

W proponowanej polityce bezpieczeństwa pliki systemowe są niemodyfikowalne i intruz nie ma możliwości dokonania jakichkolwiek zmian danych w nich zapisanych. W przypadku plików konfiguracyjnych ich zmiana jest możliwa tylko w momencie prac administracyjnych. Należy podkreślić, że w tym czasie pliki są narażone na ataki i administratorzy powinni dodatkowo zabezpieczyć system, np. poprzez odłączenie sieci komputerowej. Pliki użytkowników są całkowicie modyfikowalne i przez to narażone na ataki intruzów. Dlatego polityka bezpieczeństwa dla tego typu plików sprowadza się do regularnego wykonywania kopii zapasowych [62].

Zastosowanie takiej polityki bezpieczeństwa zapewnia ochronę działającego systemu. W przypadku udanego ataku, intruz nie będzie mógł dokonać trwałych zniszczeń w systemie. Jedynie pliki użytkowników są narażone na modyfikację lub usunięcie. Jeśli jednak kopie zapasowe były wykonywane regularnie, przywrócenie tych danych nie będzie stanowić problemu.

4.3. Mechanizm przechowywania danych

Opracowanie mechanizmu ochrony integralności systemu plików wymaga rozwiązania dwóch problemów. Pierwszym jest stworzenie skutecznego mechanizmu kontrolującego, czy ważne pliki systemowe nie zostały zmodyfikowane. Drugi polega na zapewnieniu bezpieczeństwa danych samego systemu ochrony, które również mogą stać się celem ataku.

Podstawą działania systemów ochrony integralności plików jest wykorzystywanie skrótów kryptograficznych wynikających z zawartości chronionych plików [94]. Dla wszystkich plików wybranych do ochrony system zabezpieczeń wylicza wzorce. Według proponowanego mechanizmu zabezpieczenia, przy każdorazowym otwarciu lub uruchomieniu pliku chronionego sprawdzane jest, czy wzorcowy skrót kryptograficzny znajdujący się w bazie danych jest zgodny ze skrótem obliczonym na podstawie aktualnej zawartości pliku.

W celu wykonania niewykrywalnych dla systemu ochrony integralności zmian w pliku chronionym intruz może przeprowadzić dwa główne rodzaje ataków. Pierwszy polega na dostosowaniu zawartości pliku do jego wzorca kryptograficznego zapisanego w bazie danych, a drugi na zmianie wartości jego skrótu w bazie danych wzorców. Pierwszy typ ataku, zwany w języku angielskim *preimage atack*, jest teoretycznie możliwy do przeprowadzenia, a wynika z własności kryptograficznych funkcji skrótu, które dla argumentu o dowolnej długości tworzą wynik w postaci ciągu znaków o stałej liczbie znaków. Oznacza to, że dla różnej zawartości plików ich skrót kryptograficzny może być taki sam. Taka sytuacja nosi nazwę kolizji. Jednak zastosowanie nowoczesnych funkcji kryptograficznych o dużej długości klucza, praktycznie uniemożliwia wygenerowanie takiej kolizji w rozsądnym czasie obliczeniowym. Dlatego bardziej prawdopodobny zatem jest atak na słabo zabezpieczoną bazę wzorców.

W proponowanym modelu bezpieczeństwa wprowadzono zasadę, że baza danych wzorców chronionych plików przechowywana jest na niemodyfikowalnym nośniku danych. Jest to bardzo istotne założenie z punktu widzenia polityki bezpieczeństwa i z punktu widzenia skuteczności działania systemu ochrony integralności. Zabezpieczanie w sposób sprzętowy przed niepowołaną modyfikacją jest sprawą kluczową, ponieważ gwarantuje skuteczność działania systemu ochrony integralności.

W proponowanym modelu bezpieczeństwa chroni się na poziomie sprzętowym za pomocą niemodyfikowalnych nośników danych, trzy krytyczne składniki systemu ochrony integralności plików, bazę danych wzorców, kopie plików chronionych oraz jądro systemu operacyjnego. Dopiero zapewnienie bezpieczeństwa tych wszystkich składników gwarantuje pełną ochronę systemu operacyjnego. Zapewnienie bezpieczeństwa bazy danych wzorców plików chronionych oraz ich kopii jest podstawą działania proponowanego systemu bezpieczeństwa. Natomiast zapewnienie bezpieczeństwa jądra systemu operacyjnego, poprzez umieszczenie go na niemodyfikowalnym nośniku, uniemożliwia jego modyfikację.

Zastosowanie niemodyfikowalnych nośników danych, takich jak np. pamięci przenośne USB z blokadą zapisu, w prosty i tani, a zarazem absolutnie efektywny sposób rozwiązuje problem bezpiecznego przechowywania danych potrzebnych do pracy systemu ochrony. Proponowany sposób ochrony krytycznych danych jest rozwiązaniem nowym, które nie zostało do tej pory zastosowane w żadnym systemie ochrony integralności plików. Zapewnienie bezpieczeństwa danych wykorzystywanych do pracy systemów ochrony plików, a także wszystkich innych systemów bezpieczeństwa jest niezwykle istotne przy ich projektowaniu.

Wykorzystywane dotychczas inne sposoby ochrony danych, jak np. ukrywanie bazy lub zabezpieczanie ich za pomocą rozwiązań kryptograficznych nie zawsze są skuteczne. Pierwszy sposób jest przykładem strategii zapewniania bezpieczeństwa przez utajnianie (ang. *security by obscurity*), która powszechnie jest uważana za bardzo słabe zabezpieczenie [83]. Wykorzystywanie podpisu cyfrowego do zabezpieczania bazy danych wzorców pozwala jedynie na stwierdzenie, czy baza uległa modyfikacji. Jednak w dalszym ciągu plik bazy danych jest niezabezpieczony i może zostać uszkodzony, czy wręcz usunięty, co w praktyce oznacza wyłączenie mechanizmu bezpieczeństwa. Dodatkowo trzeba chronić przed modyfikacją klucz publiczny, który musi być przechowywany bezpośrednio w module zabezpieczającym system. W przypadku wykorzystania niemodyfikowalnego nośnika danych do przechowywania bazy wzorców, eliminowana jest konieczność stosowania dodatkowych zabezpieczeń, gdyż intruz nie ma fizycznej możliwości dokonania jakichkolwiek zmian.

Oczywistą wadą przechowywania bazy danych wzorców i kopii chronionych plików na nośnikach niemodyfikowalnych, jest konieczność przygotowywania nowej, niemodyfikowalnej wersji bazy danych za każdym razem, gdy zostanie zmieniony choćby jeden, pojedynczy plik. Ponadto proces ten powinien być wykonywany tylko w tzw. trybie bezpiecznym. Oznacza to, że musi zostać sprawdzona integralność systemu plików oraz wskazane jest odłączenie komputera od sieci w celu zminimalizowania ryzyka włamania w trakcie tworzenia nowej bazy danych. Takie wymaganie wiąże się ze zwiększeniem nakładu pracy koniecznej do zapewnienia bezpieczeństwa systemu operacyjnego. Jednak, jak wykazano, zmiany w plikach systemowych dojrzałych systemów operacyjnych dokonywane są stosunkowo rzadko. Przykładowo według danych zespołu pracującego nad dystrybucją Debian, jednej z największych dystrybucji systemu Linux, w ciągu pierwszego półrocza 2007 roku pojawiły się tylko 84 poprawki bezpieczeństwa na ogólną liczbę prawie 20.000 pakietów. Uwzględniając fakt, że standardowa instalacja systemu Linux składa się z około 2.000 pakietów można przyjąć, że statystycznie w ciągu roku konieczne jest uwzględnienie kilkunastu poprawek bezpieczeństwa. Ten dodatkowy nakład pracy może zostać zmniejszony poprzez przygotowanie procedur i narzędzi dla administratorów systemów, które będą ułatwiać i częściowo automatyzować proces zmiany bazy danych wzorców i kopii chronionych plików przechowywanych na niemodyfikowalnych nośnikach danych.

W taki sam sposób, jak baza danych wzorców i kopie chronionych plików, może być zabezpieczone jądro systemu operacyjnego ze zintegrowanym modułem bezpieczeństwa. Jest to trzeci z kluczowych składników proponowanego mechanizmu ochrony integralności plików, który może być przechowywany na niemodyfikowalnych nośnikach danych. Na podstawie przeprowadzonych rozważań można postawić tezę, że:

> **Wykorzystanie niemodyfikowalnych nośników do przechowywania kluczowych danych dla działania systemu ochrony zwiększa jego odporność na ataki**.

Przy projektowaniu mechanizmów zabezpieczania systemów komputerowych niezwykle istotną, a zarazem często pomijaną kwestią jest ochrona danych samego systemu bezpieczeństwa. Dostępne obecnie systemy stosują liczne mechanizmy programowe w celu ochrony swoich danych. Jednak żaden z nich nie daje całkowitej gwarancji bezpieczeństwa danych. Proponowane wykorzystanie niemodyfikowalnych, na poziomie sprzętowym, nośników danych całkowicie uniemożliwia intruzowi modyfikację jakichkolwiek zapisanych na nim danych. Rozwiązanie to wydaje się być skuteczne, efektywne i jednocześnie tanie oraz proste do wdrożenia.

Rozdział 5.
Projekt systemu ochrony integralności plików ICAR

Zaproponowany w ramach pracy doktorskiej model bezpieczeństwa systemów operacyjnych, oparty o ochronę integralności plików, zakłada wykorzystanie niemodyfikowalnych nośników danych. W oparciu o ten model można zrealizować mechanizmy zabezpieczeń, które mogą być wykorzystywane w większości nowoczesnych systemów operacyjnych. Jednak z uwagi na fakt, że kod źródłowy programów stanowiących jądro systemu operacyjnego Linux jest dostępny publicznie, w dalszej części pracy zostanie przedstawiony projekt mechanizmu zabezpieczeń przeznaczony do wdrożenia w tym właśnie systemie. Licencja GPL (ang. *GNU Public License*), na której udostępniany jest Linux, pozwala na wprowadzanie dowolnych zmian w kodzie źródłowym systemu. Główna zasada tej licencji tzw. *copyleft* gwarantuje, że każdy nowy element wprowadzony do programu na zawsze pozostanie dostępny na licencji GPL. Oznacza to, że nie ma możliwości zamknięcia źródeł programów, czyli nikt nie będzie mógł utajnić własnych poprawek do kodu. Dzięki tej licencji programiści z całego świata, a wśród nich naukowcy z wielu ośrodków badawczych, mogą być pewni, że ich rozszerzenia jądra systemu Linux nie zostaną wykorzystane w zamkniętych rozwiązaniach komercyjnych bez ich zgody. Udostępnienie kodu swoich programów na licencji *Open Source* jest również korzystne dla twórców programów, gdyż szeroka grupa użytkowników może testować wprowadzone rozwiązania, informować o znalezionych błędach, a nawet udostępniać własne poprawki i rozszerzenia.

Mechanizm zapewniania bezpieczeństwa systemu operacyjnego, zaprojektowany w ramach niniejszej pracy doktorskiej, rozszerza funkcjonalność mechanizmów ochrony integralności systemów plików o automatyczne przywracanie zmienionych przez intruza danych. System został nazwany *ICAR*, co jest akronimem od angielskich słów *Integrity Checking and Restoring*, czyli sprawdzanie integralności i przywracanie plików.

Kluczowym założeniem systemu *ICAR* jest wykorzystanie niemodyfikowalnych nośników do przechowywania ważnych dla działania systemu zabezpieczeń danych. W związku z doświadczeniem autora pracy doktorskiej w tworzeniu dystrybucji LiveCD, czyli systemu operacyjnego Linux uruchamianego bezpośrednio z niemodyfikowalnych płyt CD-ROM, podjęto decyzję, że wdrożenie systemu *ICAR* odbędzie się z wykorzystaniem tego typu dystrybucji. Zastosowanie technologii LiveCD pozwala na proste i bezpieczne uruchamianie systemu operacyjnego, zawierającego mechanizmy zapewniania bezpieczeństwa, a także w sposób sprzętowy zabezpiecza wszystkie pliki niezbędne dla poprawnego działania systemu zabezpieczeń.

5.1. Systemy operacyjne wykorzystujące technologię LiveCD

Raz zapisane optyczne płyty, takie jak CD-ROM lub DVD są ze swojej natury niemodyfikowalnymi nośnikami danych. Uruchomienie systemu operacyjnego z takiej płyty jest całkowicie bezpieczne i odporne na próby zniszczenia plików systemowych. Oczywiście po uzyskaniu kontroli nad systemem intruz może zamontować dyski twarde znajdujące się w komputerze i dowolnie zmieniać lub niszczyć dane na nich zapisane.

Technologia pozwalająca na uruchamianie systemów operacyjnych z płyt CD-ROM nosi nazwę LiveCD. Systemy uruchamiane w ten sposób nie muszą być zainstalowane na dysku twardym komputera, jak ma to miejsce w przypadku tradycyjnie uruchamianych systemów operacyjnych. W szczególnym przypadku system komputerowy może w ogóle nie posiadać dysków twardych. W takiej sytuacji do pracy wykorzystywane są pliki zapisane na płycie CD-ROM i w pamięć operacyjnej RAM komputera.

Pierwszym systemem operacyjnym uruchamianym w ten sposób była stworzona w 1990 modyfikacja systemu AmigaOS przeznaczona dla komputera *Amiga CDTV*. Pierwszą dystrybucją LiveCD systemu operacyjnego Linux, przeznaczoną dla komputerów PC, był wydany w 1995 roku system *Yggdrasil Linux*. Obecnie najpopularniejszą dystrybucją tego typu jest stworzona w 2000 roku niemiecka dystrybucja o nazwie *Knoppix* [70].

Dane zapisywane na płycie CD-ROM są zorganizowane w system plików zgodnie ze standardem ISO 9660 [109]. Standard ten w pierwotnej formie nie przewidywał jednak możliwości uruchamiania systemu operacyjnego bezpośrednio z nośników optycznych. Dlatego w 1995 roku naukowcy z ośrodka badawczo-rozwojowego IBM w Boca Raton w USA wraz pracownikami Phoenix Technologies, ówcześnie wiodącego producenta programów BIOS dla płyt głównych komputerów PC, opracowali rozszerzenie dla standardu ISO 9660 o nazwie El-Torito [128].

W założeniu, rozszerzenie El-Torito miało umożliwić uruchamianie systemów operacyjnych z płyt CD-ROM przy możliwie niewielkiej modyfikacji programów BIOS zainstalowanych na płytach głównych. Zaprojektowany mechanizm działa na zasadzie emulacji dyskietki lub dysku twardego. Na płycie CD-ROM, z której ma zostać uruchomiony system operacyjny, umieszczany jest plik będący obrazem, czyli dokładną kopią bit po bicie dyskietki startowej lub dysku twardego. W momencie uruchomienia komputera system BIOS znajduje plik obrazu i przetwarza dane w nim zapisane w taki sam sposób, jak gdyby pochodziły z dyskietki lub dysku twardego. W związku z tym, że rozszerzenie El-Torito zachowuje standardowy sposób uruchamiania systemu komputerowego, przy tworzeniu płyt LiveCD należy wziąć pod uwagę następujące ograniczenia:

- maksymalny rozmiar pliku obrazu dyskietki wynosi 1,44 MB,
- przy emulacji dyskietki, istniejący napęd dyskietek A: dostępny jest jako B:,
- przy emulacji dysku twardego, zainstalowany dysk twardy nie jest w ogóle dostępny.

Dodatkowo w specyfikacji El-Torito umożliwiono uruchamianie programów rozruchowych (ang. *bootloaders*) w tzw. trybie nieemulacyjnym (ang. *no emulation mode*). Jeśli na płycie CD-ROM zamiast pliku obrazu zostanie zapisany niskopoziomowy program ładujący, to system BIOS po włączeniu komputera uruchomi taki program, który dalej będzie odpowiedzialny za start systemu operacyjnego.

Najbardziej popularnym programem ładującym wykorzystywanym do uruchamiania systemów operacyjnych z płyt CD-ROM jest program *ISOLINUX*. Pozwala on na uruchamianie systemu Linux, a także dowolnych systemów działających w trybie rzeczywistym. Bardziej zaawansowanym programem ładującym, który również może być uruchamiany z płyt CD-ROM, jest program *GRUB*.

W celu umieszczenia większej ilości danych na pojedynczej płycie CD-ROM czy DVD, dystrybucje LiveCD stosują do przechowywania danych skompresowane systemy plików (ang. *compressed file systems*). Najczęściej używane są *cramfs*, *cloop* oraz *squashfs*. Wszystkie są systemami plików tylko do odczytu, a ich współczynnik kompresji wynosi około 2,5:1, co oznacza, że na standardowej płycie CD-ROM o pojemności 700MB można zapisać około 1,7 GB danych. Dostępną pojemność danych należy brać pod uwagę przy projektowaniu dystrybucji systemu Linux typu LiveCD.

Wprawdzie płyty LiveCD gwarantują pełne bezpieczeństwo systemu operacyjnego poprzez zablokowanie możliwości trwałej modyfikacji plików, jednak jest to jednocześnie pewnym ograniczeniem dla użytkowników. Jest to zarówno ograniczenie sprzętowe – płyta CD-ROM jest nośnikiem tylko do odczytu, jak również programowe, gdyż stosowane skompresowane systemy plików również nie umożliwiają modyfikacji i tworzenia nowych danych. W celu umożliwienia zapisywania danych w głównym systemie plików, dystrybucje LiveCD wykorzystują wieżowe systemy plików. Pozwalają one na rozszerzenie systemu plików tylko do odczytu o warstwę pozwalającą na zapis zmienionych danych w ulotnej pamięci RAM lub na partycji dysku twardego. Warstwa taka może zawierać dane często modyfikowane. Pierwszy powszechnie stosowany tego typu systemem plików, który jest do dziś używany, nazywa się *UnionFS*. Dzięki zastosowaniu wieżowych systemów plików użytkownik może podczas bieżącej pracy zmieniać również pliki systemowe. Jednak podczas ponownego uruchomienia systemu z tej samej płyty wszelkie zmiany zostaną utracone, zarówno te wprowadzone przez użytkowników, jak i intruzów.

5.2. Dystrybucja cdlinux.pl

System Linux jest rozpowszechniany w formie tzw. dystrybucji. W jej skład wchodzi jądro systemu operacyjnego Linux, narzędzia systemowe GNU oraz zbiór programów użytkowych. Dostępnych jest wiele dystrybucji, część z nich jest ogólnego przeznaczenia, inne zaś przystosowane do specjalnych zadań. W celu wykazania skuteczności modelu zapewniania bezpieczeństwa systemów plików, który został przedstawiony w rozdziale 3, wybrano do badań polską dystrybucję o nazwie *cdlinux.pl*. Autor rozprawy doktorskiej jest jej głównym twórcą [59].

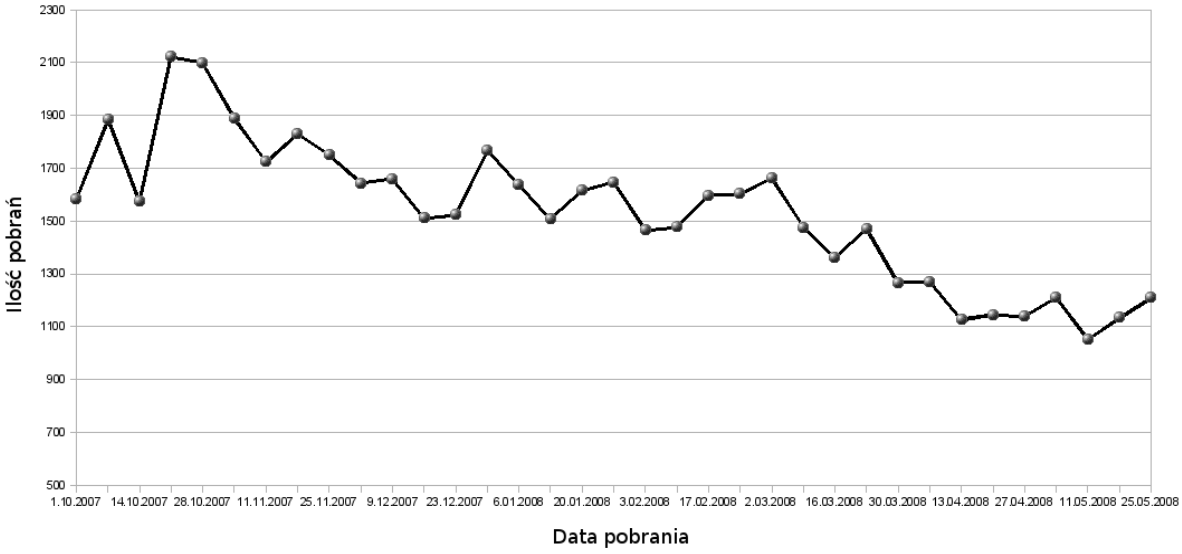
Dystrybucja *cdlinux.pl* powstała w 2002 roku w ramach pracy magisterskiej autora, pod kierownictwem dr hab. inż Jerzego Kaczmarka. Od tego czasu dystrybucja jest intensywnie rozwijana. Do końca 2007 roku okazało się jej 11 pełnych wersji. Dodatkowo każde wydanie było poprzedzone kilkoma wersjami testowymi. W czerwcu 2007 roku ukazała się stabilna wersja dystrybucji *cdlinux.pl* oznaczona numerem 1.0. Przez rok czasu z oficjalnej strony projektu *http://cdlinux.pl* została pobrana około 30 tysięcy razy.

Rysunek 5.1. Logo dystrybucji cdlinux.pl

Dystrybucja *cdlinux.pl* była również kilkukrotnie dołączana, na płytach CD lub DVD, do czasopism komputerowych, takich jak *Linux+*, *Linux Magazine*, czy *PC World Komputer*. Każde z tych wydań miało nakład od kilkunastu do kilkudziesięciu tysięcy egzemplarzy.

W 2007 roku została wydana książka „*Szkoła systemu Linux*", której autor rozprawy doktorskiej jest współautorem. Została do niej dołączona specjalnie przygotowana wersja dystrybucji *cdlinux.pl*.

Dystrybucja *cdlinux.pl* cieszy się niesłabnącym zainteresowaniem. Na rysunku 5.2 został przedstawiony wykres odwiedzin stron WWW związanych z projektem. W ciągu tygodnia strona jest odwiedzana średnio przez półtora tysiąca unikalnych internautów.



Rysunek 5.2. Statystyka odwiedzin strony http://cdlinux.pl/

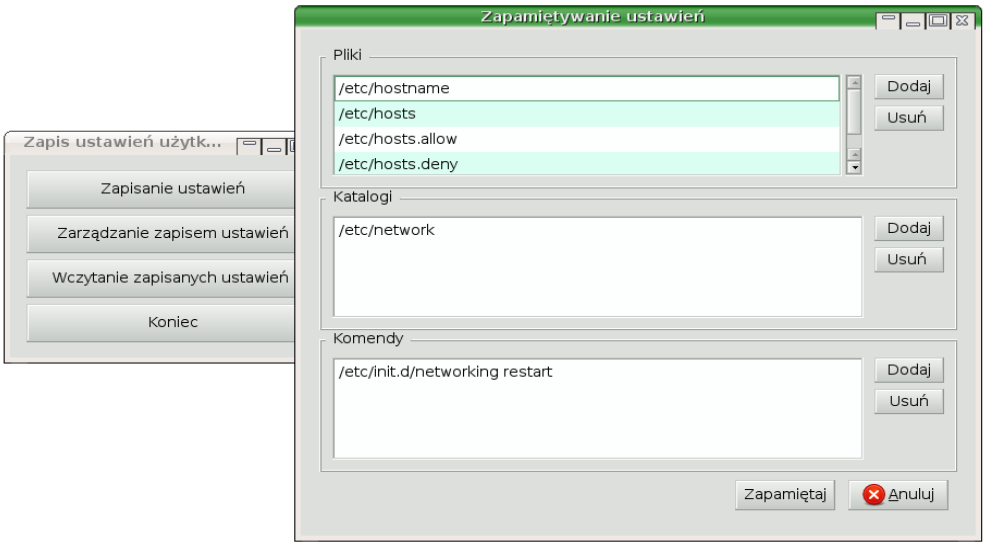
Użytkownicy dystrybucji mogą uzyskać pomoc na forum internetowym, na którym zarejestrowanych jest ponad 900 osób. Od początku powstania forum zostało na nim zadanych około 2000 pytań i udzielonych prawie 10.000 odpowiedzi. Jest to olbrzymia baza wiedzy stworzona wraz z użytkownikami zgromadzonymi wokół projektu.

Cdlinux.pl tworzony jest z pakietów przygotowanych dla dystrybucji *Debian* i jest z nią w pełni zgodny. Oznacza to, że używając dystrybucji *cdlinux.pl* można liczyć nie tylko na pomoc osób z nią związanych, ale również na wsparcie społeczności zgromadzonej wokół największej dystrybucji systemu Linux, jaką jest *Debian*. Dystrybucja bazowa została wybrana głównie z uwagi na to, że *Debian* jest największą, niekomercyjną dystrybucją systemu Linux. Jej główne założenia zostały spisane w dokumencie

„*Wytyczne Debiana dotyczące Wolnego Oprogramowania*" (DFSG, ang. *Debian Free Software Guidelines*). Twórcy Debiana gwarantują, że dystrybucja ta na zawsze pozostanie produktem Open Source [135]. Dzięki temu wszystkie dystrybucje bazujące na *Debiana*, w tym *cdlinux.pl*, będą mogły bez ograniczeń czerpać z bogatego repozytorium projektu.

Dystrybucja *cdlinux.pl* początkowo była projektowana jako dystrybucja LiveCD, czyli przeznaczona do pracy bezpośrednio z płyty CD-ROM. Przy takiej pracy system nie potrzebuje do poprawnego działania dysku twardego. Wszystkie dane są przechowywane w postaci skompresowanych obrazów na płycie CD-ROM lub w pamięci RAM. Dzięki wykorzystaniu wieżowego system plików o nazwie UnionFS, w pamięci operacyjnej zapisywane są tylko te pliki, które zostały zmodyfikowane w trakcie pracy systemu. Pozwala to na wydajną pracę bezpośrednio z płyty CD-ROM nawet na starszych komputerach.

W ramach prac badawczych, *cdlinux.pl* jako jedna z pierwszych dystrybucji LiveCD została wyposażona w program umożliwiający szybką instalację systemu na dysku twardym. Model instalacji pozwalający użytkownikowi na zapoznanie się z systemem Linux dzięki mechanizmowi LiveCD, a następnie na przeniesienie działającego w pamięci RAM systemu na dysk twardy, okazał się bardzo dobrym rozwiązaniem. Kilka lat po wdrożeniu w dystrybucji *cdlinux.pl* ten model został również zastosowany w jednej z najpopularniejszych dystrybucji systemu Linux o nazwie Ubuntu.



Rysunek 5.3. Okno programu gUserCopy

Dla dystrybucji *cdlinux.pl* zostało zaprojektowanych wiele mechanizmów i zaimplementowano wiele programów, z których warto wymienić program do zapamiętywania ustawień konfiguracyjnych oraz mechanizm uruchamiania dystrybucji z przenośnych pamięci USB. Mechanizmy te rozszerzają w pewnym stopniu bezpieczeństwo systemu, gdyż zapisanie ustawień konfiguracyjnych na pamięciach USB z blokowaną fizycznie możliwością zapisu gwarantuje ochronę tych danych. Z tych względów został wykonany program o nazwie *gUserCopy*, który umożliwia wybór plików stanowiących zbiór danych konfiguracyjnych. W chwili uruchomienia systemu komputerowego z płyty LiveCD, dane konfiguracyjne zostają automatycznie przywrócone i system operacyjny zostaje uruchomiony z zapamiętanymi ustawieniami konfiguracyjnymi. Tego typu rozwiązanie pozwala na personalizację systemu operacyjnego przy uruchamianiu go z płyt LiveCD. Przykładowe okna programu do wyboru

plików konfiguracyjnych i ich zapisu zostały pokazane na rysunku 5.3. Program *gUserCopy* pozwala na zapisywanie domyślnych zbiorów danych, takich jak ustawienia trybu graficznego, interfejsów sieciowych czy popularnych programów, a dodatkowo umożliwia tworzenie własnych zestawów danych do zapisu.

W ramach poprawy użyteczności i bezpieczeństwa dystrybucji *cdlinux.pl* wykonano mechanizm, który pozwala na instalację dystrybucji *cdlinux.pl* na przenośnej pamięci USB. Obecne pojemności pamięci USB są na tyle duże, że umożliwiają zapis zarówno całego systemu operacyjnego, jak również programów użytkowych. Wykonana wersja tego rozwiązania pozwala nie tylko na uruchamianie systemu z pamięci Flash USB, ale również umożliwia przechowywanie dowolnych danych bezpośrednio w pamięci USB, a nie w ulotnej pamięci RAM. Dzięki temu, użytkownik może posiadać przenośny system operacyjny skonfigurowany według swoich potrzeb, który może być uruchamiany na dowolnych komputerach. Tego typu system będzie również zachowywał wszystkie zmodyfikowane ustawienia i dane. Takie podejście minimalizuje znaczenie sprzętu w systemach komputerowych, gdyż umożliwia w prosty sposób uruchamianie tego samego systemu operacyjnego, z tym samym zestawem programów użytkowych na dowolnym dostępnym komputerze.

Dystrybucja *cdlinux.pl*, podobnie jak większość dystrybucji LiveCD, zapewnia bezpieczeństwo plików systemowych, gdyż nie ma możliwości ich trwałej zmiany, a po każdym ponownym uruchomieniu komputera przywracane są oryginalne pliki systemowe. Również wykonane oprogramowanie, które pozwala na zapis danych konfiguracyjnych na nośnikach niemodyfikowalnych, ich wybór, a także możliwość uruchamiania systemu z pamięci przenośnych USB, zwiększa bezpieczeństwo zarówno systemu operacyjnego, jak również części danych konfiguracyjnych i danych użytkowników.

Jest to jednak mechanizm bardzo prosty, gdyż sprowadza się do eliminacji wszelkich modyfikacji wykonywanych w trakcie pracy i uruchamiania komputera z pierwotnymi plikami systemowymi i ustawieniami. Przy wyłączeniu komputera następuje czyszczenie wprowadzonych zmian. Jest to prosty zabieg, który nie pozwala na przywracanie niektórych zmodyfikowanych plików, ale dla prostych zastosowań, jak np. korzystanie z komputera jedynie do przeglądania sieci Internet takie działania są dość skuteczne. Jednak w przypadku systemów serwerowych takie rozwiązanie jest mało przydatne, choćby z uwagi na fakt, że serwery są bardzo rzadko wyłączane. Dlatego tego typu mechanizmy bezpieczeństwa są zbyt proste do zabezpieczania ważnych systemów komputerowych.

5.3. Analiza jakości dystrybucji cdlinux.pl

Projektując i wytwarzając oprogramowanie przeznaczone dla szerokiej grupy użytkowników należy brać pod uwagę jakość takiego produktu. W przypadku systemów operacyjnych zagadnienie to jest o wiele bardziej złożone, ponieważ są one wykorzystywane przez wszystkich użytkowników komputerów. Można ich generalnie podzielić na użytkowników indywidualnych, wykorzystujących komputery do pracy zawodowej lub rozrywki oraz na administratorów systemów komputerowych, których zadaniem jest utrzymywania i zapewnianie bezpieczeństwa komputerom firmowym czy sieciom lokalnym. Wprawdzie oprogramowanie systemowe przeznaczone dla komputerów typu *desktop* i dla serwerów nie różnią się zasadniczo, to jednak sposoby jego wykorzystania są różne.

Zapewnienie wysokiej jakości jest obecnie podstawowym zadaniem stojącym przed projektantami oprogramowania. Wysoka jakość wytwarzanego oprogramowania przekłada się na zadowolenie użytkowników, a co za tym idzie na popularność oferowanego programu. Problem jakości dotyczy zarówno produktów komercyjnych, jak i oprogramowania Open Source, które jest wykorzystywane coraz powszechniej. W związku z potrzebą zapewnienia jakości produktów informatycznych konieczne jest mierzenie jakości wytwarzanego oprogramowania.

Z tych względów dokonano analizy jakości i potrzeb użytkowników dystrybucji cdlinux.pl. Jakość oprogramowania jest definiowana jako te charakterystyki produktu, które zarówno w sposób jawny, jak i niejawny wpływają na zadowolenie użytkownika [54]. Do najczęściej rozpatrywanych charakterystyk jakości należą funkcjonalność, niezawodność, użyteczność, efektywność, łatwość zarządzania oraz przenośność.

Pomiar jakości oprogramowania jest ciągłym procesem, który polega na definiowaniu, zbieraniu i analizowaniu danych dotyczących gotowego produktu jak również procesu jego wytworzenia. Wykonanie oceny jakości może prowadzić do polepszenia kontroli nad procesem powstawania i działania produktu informatycznego, co powinno przełożyć się na poprawę jego jakości.

Zarówno wybór charakterystyk do oceny jakości oprogramowania, jak i sama ocena produktu jest zadaniem trudnym, a wynik może być różny w zależności od osób przeprowadzających badania. Jest to spowodowane trudnością wyrażenia jakości oprogramowania w jednostkach mierzalnych.

Istnieją dwa podejścia do oceny jakości produktu informatycznego. Pierwsze tzw. top-down, polega na ocenie procesu wytwórczego produktu informatycznego. Najważniejszym modelem tego typu jest CMM (ang. *Capability Maturity Model for Software*), który bada dojrzałość organizacji i określa sposoby wprowadzania ulepszeń. Model CMM odnosi się do procesu wytwórczego, a nie bezpośrednio do produktu informatycznego.

Odmienne podejście reprezentują tzw. modele *bottom-up*, takie jak GQM (ang. *Goal, Question, Metric*) czy QIP (ang. *Quality Improvement Paradigm*). Modele te pozwalają na poprawę kontroli procesu wytwórczego w wybranym kontekście.

5.3.1. Pomiar ukierunkowany na cel

Analiza jakości produktów informatycznych, podobnie jak innych wytwarzanych produktów, jest o tyle trudna, że można zdefiniować bardzo dużą liczbę różnych atrybutów jakości. W zależności od ich wyboru, ocena jakości może być bardzo różna. Przy dużej liczbie rozpatrywanych atrybutów jakości występują problemy z ich mierzeniem i ustalaniem kryteriów akceptacji. Stąd ocena jakości produktu może być niejednoznaczna i różna w zależności od intencji oceniającego.

Jedną z najczęściej stosowanych metod oceny jakości produktów informatycznych jest metoda GQM. Została ona opracowana w latach osiemdziesiątych ubiegłego wieku przez Basiliego i Rombacha [9]. Jest to metoda pomiaru jakości, której istotą jest jasne zdefiniowanie celu takiego pomiaru. Podejście to zakłada, że proces zbierania i analizowania danych jest tylko środkiem do zrealizowania jawnie postawionych celów, w odróżnienia od podejścia *bottom-up*, w którym dopiero analiza danych wyznacza cele poprawy [31].

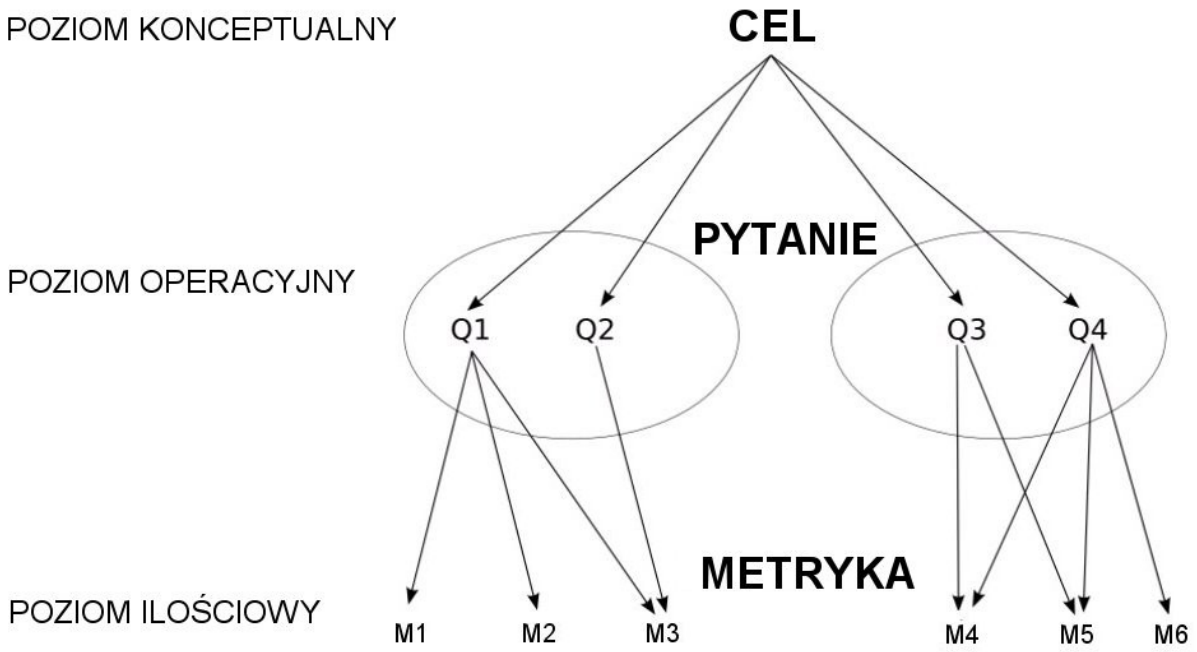
Paradygmat GQM odwzorowuje praktyczne podejście do problemu pomiaru jakości oprogramowania. Jest elastycznym narzędziem, które pozwala skoncentrować się na celu, jaki ma zostać osiągnięty. Ważną zaletą metody GQM jest możliwość dowolnego zdefiniowania zarówno celów pomiaru, jak i mierzonych metryk w zależności od potrzeb organizacji lub użytkowników oprogramowania. Pomimo, że podejście takie pociąga za sobą konieczność dokładnego opracowania celów i metryk, a nie wykorzystania gotowych, predefiniowanych metryk, w rezultacie daje jednak dokładniejsze wyniki i pozwala na lepszą identyfikację problemów.

Model pomiaru wg metody GQM podzielony jest na trzy poziomy:

- **konceptualny** – definiowany jest cel pomiaru (ang. *goal*), jaki ma zostać osiągnięty dla produktu,
- **operacyjny** – układane są pytania (ang. *question*), które pomagają zrozumieć, w jaki sposób można osiągnąć postawione cele,
- **ilościowy** – dobierane są metryki (ang. *metric*), które pozwalają w sposób liczbowy odpowiedzieć na pytania.

Badanie metodą GQM podzielone jest na cztery fazy. W pierwszej, tzw. fazie planowania, wybierany jest projekt, określany jest sposób pomiarów i termin ich realizacji. W fazie definiowania określany jest jawnie cel badania, dobierane są pytania oraz metryki. Trzecia faza polega na przeprowadzeniu oceny produktu na podstawie zdefiniowanych pytań i metryk. Ostatnia faza, tzw. faza interpretacji, polega na wyciągnięciu wniosków z zebranych danych poprzez identyfikację najważniejszych problemów w osiągnięciu postawionego celu. W tej fazie określane są również środki, jakie w przyszłości pozwolą na wyeliminowanie występujących problemów [105].

Na rysunku 5.4 przedstawiono model zależności pomiędzy poszczególnymi poziomami w metodzie GQM.



Rysunek 5.4. Schemat modelu GQM

W ramach badań autora zastosowano metodę GQM do oceny jakości dystrybucji *cdlinux.pl* w celu uzyskania obiektywnych danych pomiarowych niektórych metryk definiujących jakość oraz w celu wyboru mechanizmów zapewniania bezpieczeństwa systemów komputerowych, które zostaną zaakceptowane przez potencjalnych użytkowników. Główny problem, który należało rozważyć, polegał na tym, czy wprowadzone mechanizmy bezpieczeństwa nie zmniejszą wydajności systemu, nie spowodują zwiększenia złożoności jego obsługi i czy zostaną zaakceptowane przez użytkowników.

Badanie jakości dystrybucji *cdlinux.pl* zostało wykonane wśród jej użytkowników. Został zdefiniowany cel poprawy produktu, którym było zwiększenie popularności dystrybucji poprzez polepszenie jej jakości [60].

Jako metodę badania przyjęto przeprowadzenie analizy danych pochodzących z ankiety opublikowanej w sieci Internet. Wykorzystanie sieci pozwoliło na szybki dostęp do szerokiego grona użytkowników. Ankietę dostarczono respondentom w formie strony internetowej, z której wyniki, po wypełnieniu przez użytkowników, były bezpośrednio zapisywane w bazie danych.

W związku z faktem, że ankieta została przeprowadzona wśród użytkowników dystrybucji, którzy mogli nie mieć dotychczas kontaktu z badaniem jakości oraz w związku ze specyfiką badania za pomocą strony internetowej, zdecydowano się na zastosowanie ujednoliconego systemu ocen. Użyto tzw. ocen szkolnych, czyli użytkownik mógł przypisywać każdej metryce wartość od 1 - 5, gdzie 1 oznaczało wartość minimalną, a 5 maksymalną.

5.3.3. Definicje celu, pytań i metryk dla metody GQM

W celu zdefiniowania prostego i zrozumiałego celu pomiaru posłużono się szablonem zaproponowanym przez Basiliego [9]. Pomaga on zdefiniować cel badania, kontekst oraz punkt widzenia dzięki zastosowaniu odpowiedniej struktury. W tabeli 5.1 został przedstawiony cel badania jakości dystrybucji *cdlinux.pl*.

Tabela 5.1. Cel badania jakości dystrybucji cdlinux.pl

Słowo szablonu	Pytanie szablonu	Cel badania
Analizować	(co?)	**Analizować dystrybucję cdlinux.pl**
z zamiarem	(dlaczego?)	**z zamiarem poprawy jakości**
w odniesieniu do	(jaki atrybut?)	**w odniesieniu do użyteczności dystrybucji**
z punktu widzenia	(czyjego?)	**z punktu widzenia użytkowników**
w środowisku	(w jakim kontekście?)	**w środowisku polskich użytkowników**

Jako cel pomiaru dystrybucji *cdlinux.pl* metodą GQM została wybrana poprawa jakości dystrybucji pod względem użyteczności. Użyteczność systemów operacyjnych typu LiveCD, które z założenia są przeznaczone dla początkujących użytkowników, jest najważniejszym wymaganiem dla twórców dystrybucji. Wysoka użyteczność systemu oraz zainstalowanego oprogramowania przekłada się bezpośrednio na popularność dystrybucji. Dodatkowym założeniem jest uwzględnienie specyfiki polskiego środowiska. Dystrybucja *cdlinux.pl* jest przeznaczoną dla polskich użytkowników. W związku z tym należy uwzględnić zarówno

dostępność polskiego języka w aplikacjach i programach konfiguracyjnych, jak również możliwości konfiguracji i użytkowania popularnego w Polsce sprzętu peryferyjnego. Do tak postawionego celu zostało zidentyfikowanych pięć kluczowych obszarów użyteczności:

- wydajność działania dystrybucji,
- łatwość konfiguracji,
- łatwość użytkowania,
- jakość wsparcia technicznego,
- jakość polonizacji programów w dystrybucji.

Wydajność działania dystrybucji *cdlinux.pl* jest kluczowym elementem użyteczności dla dystrybucji LiveCD, z uwagi na wolniejszy odczyt danych z płyt CD w porównaniu do danych zapisanych na dyskach twardych. Z punktu widzenia użytkowników długi czas uruchamiania dystrybucji, czy poszczególnych programów powoduje małą użyteczność dystrybucji w codziennej pracy.

Drugi obszar użyteczności dotyczy nakładu pracy potrzebnego do konfiguracji dystrybucji na komputerze. W związku z przeznaczeniem dystrybucji dla początkujących użytkowników konieczne jest, aby konfiguracja komputera niezbędna do jego poprawnego działania była przeprowadzona w sposób możliwie prosty. Zbadanie łatwości użytkowania dystrybucji pozwoliło na ocenę, czy obsługa dystrybucji *cdlinux.pl* przez początkujących użytkowników nie sprawia im problemów.

Wsparcie techniczne jest bardzo ważnym kryterium dla użytkowników, gdy natkną się na problem, którego sami nie będą w stanie rozwiązać. Dobre i przyjazne wsparcie techniczne przekłada się na zadowolenie użytkownika, co jest równoznaczne z dobrą oceną jakości dystrybucji.

Ostatni badany obszar użyteczności odnosi się do dostosowania dystrybucji do środowiska polskich użytkowników, zarówno w kontekście lokalizacji programów, jak i wsparcia dla specyficznego sprzętu stosowanego w Polsce.

Do każdego ze zdefiniowanych obszarów użyteczności zostało sformułowane pytanie i zbiór metryk, przedstawiony poniżej.

Pytanie 1. Jaka jest wydajność dystrybucji?

1. **Szybkość uruchomienia dystrybucji.**
2. **Szybkość uruchomienia programu Mozilla Firefox.**
3. **Szybkość uruchomienia programu OpenOffice.org Writer.**
4. **Szybkość instalacji dystrybucji na dysku twardym.**

Pytanie 2. Jaki nakład pracy jest potrzebny do konfiguracji dystrybucji w środowisku?

1. **Pracochłonność skonfigurowania sieci.**
2. **Pracochłonność skonfigurowania karty graficznej.**
3. **Pracochłonność zapamiętania i przywrócenia konfiguracji.**

Pytanie 3. Jaki nakład pracy jest potrzebny do wykonania podstawowych czynności?

1. **Pracochłonność zamontowania dysków.**
2. **Pracochłonność zmiany konfiguracji sieci.**
3. **Pracochłonność zapisania danych na dyskach.**
4. **Pracochłonność instalacji na dysku twardym.**
5. **Pracochłonność stworzenie własnej wersji dystrybucji.**

Pytanie 4. Jaka jest dostępność wsparcia technicznego?

1. **Szybkość odpowiedzi na zgłoszony problem.**
2. **Liczba dostępnych kanałów wsparcia technicznego.**

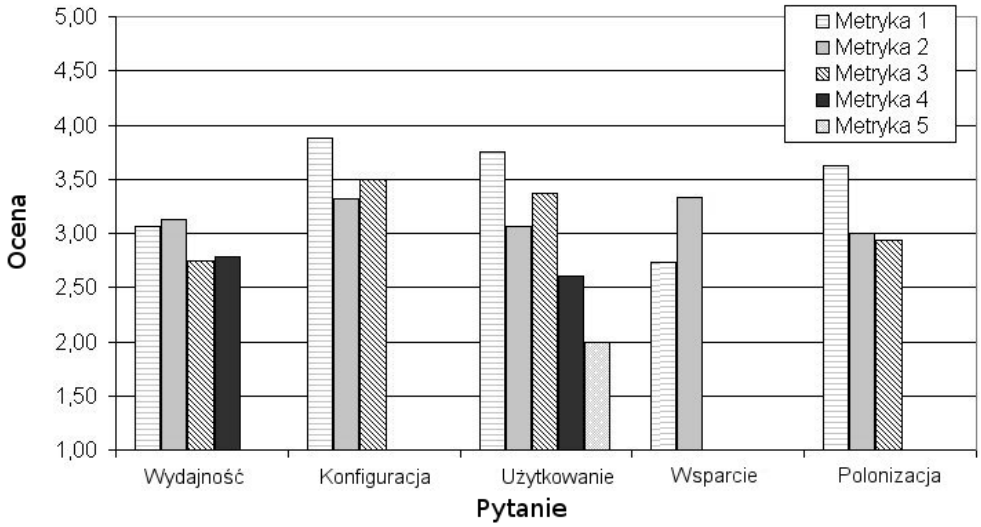
Pytanie 5. Jaki jest stopień lokalizacji dystrybucji dla Polski?

1. **Liczba spolonizowanych programów.**
2. **Ilość dostępnej dokumentacji w języku polskim.**
3. **Liczba wspieranych urządzeń peryferyjnych specyficznych dla Polski.**

5.3.4. Uzyskane dane pomiarowe metryk jakości

Po zdefiniowaniu pytań, z których wynikają atrybuty jakości, ankieta została umieszczona na stronie internetowej. Po wypełnieniu i wysłaniu ankiety przez użytkowników dane zostały przekazane do programu napisanego w języku PHP, który zapisywał je w bazie PostgreSQL. Ankieta została przeprowadzona we wrześniu 2006 roku. Udział w badaniu wzięło 87 użytkowników dystrybucji *cdlinux.pl*.

Zebrane dane przedstawiono na rysunku 5.5 w postaci wykresu, który ilustruje średnią wartość metryk dla każdego pytania. Uzyskane wyniki ankiety pokazują, że użytkownicy najlepiej ocenili prostotę konfiguracji dystrybucji w środowisku użytkownika. Wsparcie dla podstawowych operacji wykonywanych podczas użytkowania systemu zostało ocenione pozytywnie, natomiast czynności zaawansowane, takie jak instalacja na dysku twardym i tworzenie własnej wersji dystrybucji LiveCD, zostały uznane za trudne. Użytkownicy dobrze ocenili stopień polonizacji programów wchodzących w skład dystrybucji.



Rysunek 5.5. Wyniki pomiarów jakości dystrybucji cdlinux.pl

Przeprowadzona metodą GQM ocena jakości dystrybucji *cdlinux.pl* pozwoliła na identyfikację głównych problemów użytkowników dystrybucji. Z rezultatów przeprowadzonego badania wynika, że głównym obszarem użyteczności dystrybucji, który wymagał poprawy jest szybkość działania dystrybucji oraz wsparcie techniczne. W dalszej perspektywie należałoby również poprawić programy wspomagające wykonywanie zaawansowanych czynności przez użytkowników, takich jak instalacja na dysku twardym oraz tworzenie własnej wersji dystrybucji LiveCD.

Ponieważ z badań wynikało, że użytkownicy nisko oceniali wsparcie techniczne oferowane dla dystrybucji, w 2007 roku została napisana i wydana książka pod tytułem „*Szkoła systemu Linux*". Opisano w niej sposób obsługi systemu operacyjnego Linux na bazie dystrybucji *cdlinux.pl*. Książka liczy 320 stron i dotychczas rozeszła się w ponad tysięcznym nakładzie.

Z przeprowadzonych badań wynika, że użytkownicy dystrybucji bardzo sobie cenią łatwość użytkowania, którą zresztą oceniają wysoko. Wszystkie rozwiązania rozszerzające funkcjonalność, a zwłaszcza mechanizmy zapewniania bezpieczeństwa systemu komputerowego powinny być tak wykonane, aby nie zmniejszały łatwości użytkowania, co mogłoby prowadzić do zmniejszenia popularności dystrybucji *cdlinux.pl*.

5.4. Analiza problemów implementacyjnych zabezpieczeń

Opracowany w ramach pracy doktorskiej model systemu bezpieczeństwa, opisany w rozdziale 4 rozszerza funkcjonalność systemu operacyjnego o ochronę integralności systemu plików wraz z mechanizmem automatycznego przywracania zawartości zmienionych plików. Model ten stał się podstawą do stworzenia, dla systemu operacyjnego Linux, mechanizmu ochrony o nazwie ICAR (ang. *Integrity Checking And Restoring)*. W wykonanym systemie połączono koncepcję systemu ochrony integralności plików z mechanizmem pozwalającym na uruchamianie systemów operacyjnych bezpośrednio z niemodyfikowalnych nośników danych.

Przed przystąpieniem do implementacji mechanizmu bezpieczeństwa ICAR konieczne było przeprowadzenie analizy problemów jakie pojawiają się przy rozszerzaniu funkcjonalności jądra systemu operacyjnego Linux. Należy podkreślić, że model mechanizmu zabezpieczania jest ogólny i może być zaimplementowany w dowolnym systemie operacyjnym. Wykorzystano system operacyjny Linux, ponieważ licencja GPL, na której jest on wydawany, daje szerokie możliwości dokonywania zmian nawet bezpośrednio w kodzie źródłowym jądra.

5.4.1. Włączenia modułu zabezpieczenia do jądra systemu operacyjnego

Kluczowym problemem implementacyjnym, przy rozszerzaniu funkcjonalności systemu operacyjnego Linux o mechanizm zapewniania bezpieczeństwa z rozszerzoną ochroną integralności plików, jest sposób włączania modułu zabezpieczeń do dowolnej wersji jądra systemu Linux. Należy przeanalizować poprawność identyfikacji zdarzenia polegającego na nieuprawnionej zmianie systemu plików, a także przeanalizować sposób reakcji na takie zdarzenie. Po wykryciu nieautoryzowanej zmiany, moduł ochrony powinien podjąć działania obronne, którymi mogą być:

- przywrócenie oryginalnej wersji pliku z niemodyfikowalnej kopii,
- zablokowanie dostępu do zmodyfikowanego pliku,
- powiadomienie administratora systemu o zaistniałej sytuacji,
- awaryjne zamknięcie systemu operacyjnego.

Przywracanie oryginalnej zawartości pliku z niemodyfikowalnej kopii jest czynnością, która pozwala użytkownikom systemu operacyjnego na ciągłą pracę bez względu na operacje wykonywane przez intruza. Dlatego taka reakcja na atak intruza jest najbardziej pożądana i uzasadniona.

Blokowanie dostępu do całego systemu operacyjnego, np. przy powtarzających się atakach, jest działaniem ryzykownym, ponieważ może prowadzić do strat wynikających z braku dostępu do systemu przez użytkowników. Z tego powodu takie działanie jest domyślnie nieaktywne, a jego włączenie może nastąpić podczas instalacji systemu ICAR.

W prototypowej implementacji systemu ICAR administrator zawsze jest powiadamiany o wykryciu zmian w chronionych plikach poprzez odpowiedni wpis w plikach logów systemowych. Jednak podczas dalszego rozwoju systemu konieczne będzie umożliwienie wykorzystania bardziej zaawansowanych metod informowania administratorów. Szczególnie jest to istotne, gdy system może blokować dostęp do systemu czy wyłączać komputer. W takiej sytuacji pożądane byłoby wykorzystanie do powiadamiana takich narzędzi jak poczta elektroniczna, SMS, czy praktycznie nie wykorzystywany w Polsce, ale ciągle popularny na świecie Pager.

Kolejnym problemem wymagającym szczegółowej analizy jest sposób, w jaki system zapewniania bezpieczeństwa będzie dołączany do jądra systemu operacyjnego. Ta decyzja musi zostać podjęta na wczesnym etapie projektowania systemu ochrony. Dostępnych jest wiele sposobów włączania rozszerzeń bezpieczeństwa do jądra systemu operacyjnego. Powszechnie stosowane są dwa rozwiązania. Pierwszy polega na stworzeniu wieżowego systemu plików, np. z wykorzystaniem języku FiST [120]. Drugi zakłada modyfikację wywołań systemowych, np. poprzez wykorzystanie szkieletu linuksowych modułów bezpieczeństwa LSM [127]. Opis obu rozwiązań został przedstawiony w rozdziale 3.

W tabeli 5.2 przedstawiono porównanie podstawowych cech charakterystycznych dla mechanizmu wykorzystującego wieżowy system plików i mechanizmu wykorzystującego modyfikację wywołań systemowych.

Tabela 5.2. Porównanie sposobów dołączania mechanizmów zabezpieczeń

	Wieżowy system plików	**Modyfikacja wywołań systemowych**
Sposób uruchomienia	zamontowanie systemu plików	załadowanie modułu
Sposób wyłączenia	odmontowanie systemu	odłączenie modułu
Konieczne zmiany	Implementacja warstwy systemu plików ze zmienionymi funkcjami odczytu plików	Implementacja i podmiana funkcji odczytu plików
Ułatwienia	język *FiST*	*Linux Security Module*
Język programowania	abstrakcyjny język wysokiego poziomu	język C
Nakład pracy	porównywalny	
Prędkość działania	porównywalna	
Przenośność	duża	

Z przeprowadzonej analizy wynika, że implementacja mechanizmu ochrony integralności systemu plików z wykorzystaniem koncepcji wieżowych systemów plików jak i modyfikacji wywołań systemowych, będzie wymagała podobnego nakładu pracy. W obu przypadkach konieczne jest stworzenie nowych procedur, które będą uruchamiane zamiast standardowych funkcji dostępu do plików. Jednak zastosowanie mechanizmów ułatwiających tworzenie rozszerzeń, takich jak język FiST dla wieżowych systemów plików i Linuksowe Moduły Bezpieczeństwa dla wywołań systemowych, znacznie upraszcza sam proces implementacji. Również prędkość działania i przenośność obu implementacji jest podobna. Kluczowa dla bezpieczeństwa działania systemu ICAR jest kwestia uniemożliwienia jego wyłączenia podczas pracy systemu operacyjnego. Zarówno implementacja za pomocą wieżowych systemów plików, jak i modułów LSM gwarantuje, że system bezpieczeństwa nie będzie mógł zostać wyłączony.

Wykorzystanie szkieletu *Linux Security Modules* do włączenia systemu ICAR do jądra systemu operacyjnego jest rozwiązaniem naturalnym i jednocześnie zgodnym z ogólną koncepcją bezpieczeństwa w Linuksie. Dlatego też do zintegrowania systemu ICAR, z jądrem systemu operacyjnego, został wybrany mechanizm Linuksowych Modułów Bezpieczeństwa.

Szkielet Linuksowych Modułów Bezpieczeństwa udostępnia szereg funkcji, które są wywoływane przy każdym odwołaniu do zasobów sprzętowych. Domyślnie funkcje te są puste. Tworzenie modułu bezpieczeństwa polega na nadpisaniu funkcji LSM własnym kodem, realizującym np. operację kontroli pliku. Szkielet LSM jest inicjalizowany podczas uruchamiania jądra systemu operacyjnego, a stworzone funkcje ładowane za pomocą funkcji *register_security* [115].

Wśród wielu dostępnych funkcji LSM, które pozwalają na zablokowanie dowolnej operacji na poziomie jądra, dostępne są również funkcje pozwalające na kontrolę dostępu do plików. Właśnie te funkcje zostały nadpisane podczas implementacji systemu ICAR. Dzięki temu, przy każdej operacji dostępu do pliku, działanie procesu jest przerywane i sterowanie jest przekazywane do funkcji systemu ICAR. Zgodnie z opracowanym algorytmem, przedstawionym w rozdziale 4.1, po zweryfikowaniu poprawności pliku i ewentualnym przywróceniu jego oryginalnej zawartości z kopii, zwracana jest do systemu operacyjnego odpowiedź, czy dostęp do pliku ma zostać przyznany czy też zablokowany.

Przeprowadzony eksperyment, polegający na implementacji prostego prototypu systemu ICAR dowiódł, że szkielet *Linuksowych Modułów Bezpieczeństwa* może być z powodzeniem wykorzystany do implementacji modułu realizującego model zabezpieczeń zaproponowany w niniejszej pracy.

5.4.2. Baza danych

W ramach analizy dalszych problemów implementacyjnych dla proponowanego mechanizmu zabezpieczeń rozpatrzono problem wydajności systemu operacyjnego z włączonym systemem ochrony integralności plików. Jednym z ważnych czynników wpływających na wydajność całego systemu operacyjnego z system ICAR jest prędkość odczytu rekordów z bazy danych. Dlatego niezwykle istotne jest dobranie takiego rozwiązania, które pozwoli na uzyskanie dostępu do rekordów w jak najkrótszym czasie. Wczesny prototyp systemu ICAR wykorzystywał własną bazę opartą na tablicach haszujących. Jednak wydajność takiego rozwiązania okazała się niewystarczająca.

Z tego powodu zdecydowano się na zastosowanie silnika bazy danych, opracowanego na uniwersytecie Stony Brook przez zespół pod kierownictwem profesora Ereza Zadoka, o nazwie Kernel Berkeley Database (KBDB). Jest to implementacja bazy danych Berkeley w jądrze systemu operacyjnego Linux. Baza Berkeley pozwala na szybkie wyszukiwanie krotek w oparciu o tablice haszujące, b-drzewa oraz indeksowanie [64]. Zastosowanie tego rozwiązania pozwala na szybki dostęp do danych, przy niskich nakładach koniecznych do dostosowania silnika bazy danych na potrzeby systemu ICAR.

Oprócz wyboru silnika bazy danych ważnym zagadnieniem jest sposób indeksowania plików w bazie danych. W systemie ICAR chronione pliki są identyfikowane za pomocą nazwy oraz pełnej ścieżki dostępu. Alternatywnym rozwiązaniem jest przechowywanie tylko numerów i-węzłów plików chronionych. Jednak zaletą zastosowanego rozwiązania jest łatwość znajdowania kopii pliku zapisanej na płycie LiveCD. Natomiast wadą takiego rozwiązania jest znacznie większy rozmiar pól w bazie danych przeznaczonych na przechowywanie identyfikatora pliku.

Wadą sposobu identyfikowania pliku za pomocą i-węzła jest natomiast konieczność kontrolowania każdej operacji zmiany nazw plików. Wynika to z faktu, że atakujący może zmienić nazwę pliku kontrolowanego na inną, a następnie spreparowanemu plikowi nadać nazwę oryginalną. W takiej sytuacji zmiana nazwy pliku nie pociąga za sobą zmiany numeru i-węzła, a podmieniony plik nie będzie chroniony przez system. Dlatego rozwiązanie polegające jedynie na identyfikowaniu pliku poprzez i-węzeł należy odrzucić z uwagi na możliwość prostego ominięcia tego typu zabezpieczenia.

Podjęto zatem decyzję, że w bazie danych zostało zastosowane identyfikowanie pliku za pomocą jego nazwy i pełnej ścieżki dostępu, ponieważ jest to odporne na możliwość eliminacji systemu zabezpieczeń.

5.4.3. Implementacja systemu ICAR

Wykorzystując szkielet LSM zaimplementowany został, w języku C, system ICAR zgodny z zaproponowanym modelem bezpieczeństwa. Kod źródłowy systemu ICAR składa się 3500 wierszy kodu, z czego najważniejsza część odpowiedzialna za ochronę plików i obsługę buforów podręcznych liczy ponad 1000 wierszy. System ICAR zawiera również programy użytkowe pozwalające na stworzenie bazy danych i przygotowanie zbioru informacji o plikach przeznaczonych do ochrony. W celu ułatwienia tworzenia obrazu płyty LiveCD czy przygotowywania pamięci przenośnej USB opracowano skrypt automatyzujący czynności niezbędne do stworzenia systemu operacyjnego zawierającego mechanizm bezpieczeństwa ICAR

Kod źródłowy modułu bezpieczeństwa ICAR wraz z zestawem programów narzędziowych został umieszczony na stronie *http://www.cdlinux.pl/icar*. Jest on udostępniony na licencji GPL i może stać się przedmiotem badań innych zainteresowanych, jak również może być powszechnie wykorzystywany w systemach komputerowych.

5.5. Wykorzystanie modelu bezpieczeństwa

Mechanizm ICAR został przystosowany do wykorzystania w systemach operacyjnych, uruchamianych z płyt LiveCD, z wykorzystaniem nośników optycznych, takich jak CD lub DVD, a także pamięci przenośnych USB. ICAR może być również wykorzystany do zabezpieczania systemów operacyjnych zainstalowanych trwale na dysku twardym komputera.

5.5.1. Mechanizm ICAR w systemach LiveCD

W przypadku wykorzystywania technologii LiveCD do uruchamiania systemu operacyjnego Linux, moduł bezpieczeństwa wraz z jądrem jest zapisany na nośniku niemodyfikowalnym, zabezpieczonym w sposób sprzętowy przed zmianami. Podczas pracy systemu pliki są odczytywane z płyty LiveCD, a wszelkie zmiany zapisywane są w ulotnej pamięci RAM. Po zakończeniu pracy i wyłączeniu komputera wszystkie wprowadzone modyfikacje są tracone. W związku z tym, pod względem bezpieczeństwa konieczna jest jedynie ochrona plików przed nieuprawnioną modyfikacją podczas pracy systemu operacyjnego.

Koncepcja wykorzystania modułu ochrony integralności jądra ICAR w systemach LiveCD zakłada, że na niemodyfikowalnej płycie LiveCD znajduje się jądro systemu operacyjnego z modułem bezpieczeństwa, baza danych wzorców i kopie chronionych plików.

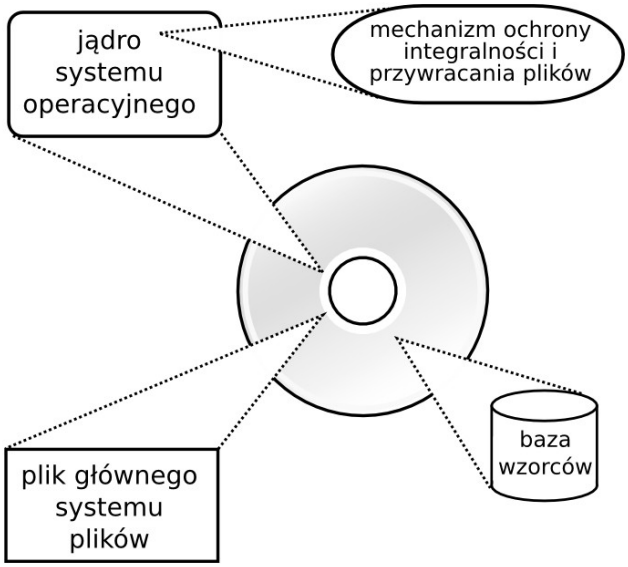
Wykorzystanie niemodyfikowalnej płyty CD-ROM do załadowania jądra systemu operacyjnego uniemożliwia intruzowi dokonanie jakichkolwiek zmian w jego kodzie. Najprostszy, a zarazem najskuteczniejszy atak na istniejące dotychczas systemy ochrony integralności plików działające w najniższej warstwie systemu operacyjnego polega na podmienieniu pliku jądra na taki, w którym ochrona jest wyłączona. Zabezpieczenie sprzętowe, jakim jest przechowywanie pliku jądra na niemodyfikowalnej płycie CD-ROM, uniemożliwia zdalne dokonanie tego typu ataku. Może on być przeprowadzony jedynie poprzez fizyczny dostęp do komputera i podmianę płyty CD-ROM w czytniku. Taki sposób uruchamiania systemu operacyjnego gwarantuje bezpieczeństwo w zakresie ochrony jego jądra.

Ładowanie jądra systemu operacyjnego z płyty CD-ROM nie oznacza konieczności uruchamiania całego systemu w ten sposób. Wszystkie pozostałe pliki systemu operacyjnego mogą być zapisane na dysku twardym. Po załadowania jedynie jądra systemu operacyjnego do pamięci RAM z płyty CD, dalszy proces startu może się odbywać w tradycyjny sposób z wykorzystaniem danych zapisanych na dysku twardym. Można wyodrębnić trzy metody uruchamiania systemu operacyjnego:

1. Tylko jądro jest uruchamiane z płyty CD-ROM, a pozostałe pliki są zapisane na dyskach twardych komputera.
2. Cały system operacyjny wraz z plikami programów użytkowych jest uruchamiany z płyty LiveCD, jedynie dane wytworzone przez użytkowników przechowywane są na dyskach twardych.
3. Wszystkie pliki są ładowane z płyty LiveCD, a wybrane pliki konfiguracyjne i dane użytkowników są przywracane za pomocą opracowanego programu,

Wybranie odpowiedniej polityki bezpieczeństwa jest związane ze sposobem wykorzystania systemu komputerowego przez jego użytkowników oraz administratorów.

Na rysunku 5.6 została zilustrowana zawartość płyty LiveCD dla systemu operacyjnego, w którym zastosowano zaprojektowany system ochrony plików. Z płyty pobierany jest plik jądra systemu operacyjnego wraz ze zintegrowanym modułem bezpieczeństwa ICAR, który jest odpowiedzialny za ochronę wybranych plików. Oprócz bazy wzorców na płycie są zapisane kopie chronionych plików. W przypadku zastosowania mechanizmu ładowania plików systemowych lub wszystkich plików z płyty CD nie ma konieczności tworzenia oddzielnej kopii chronionych plików. System może korzystać z plików, które zostały wykorzystane do uruchomienia systemu operacyjnego.



Rysunek 5.6. Zawartość płyty LiveCD systemu ochrony integralności plików

Jeśli wystąpi konieczność zmiany plików konfiguracyjnych, aktualizacji bądź instalacji nowych programów, niezbędne jest stworzenie nowej wersji płyty LiveCD. Z tych względów do tworzenia nowych wersji obrazów płyt LiveCD został przygotowany zestaw specjalnych skryptów, które ułatwiają administratorom wykonanie tej czynności. Tworzenie nowej płyty LiveCD polega na wprowadzeniu wszelkich niezbędnych zmian w plikach zapisanych w pamięci RAM. Następnie za pomocą skryptu administrator wskazuje, które pliki mają zostać objęte ochroną. Na podstawie zebranych danych skrypt tworzy nowy obraz płyty, który zawiera jądro systemu operacyjnego z modułem bezpieczeństwa, bazę danych skrótów chronionych plików, a także ich kopie bezpieczeństwa. Tak przygotowany obraz płyty, zapisany w standardzie ISO9660 może być nagrany na płytę CD-ROM i będzie stanowił kompletną, uruchamianą płytę LiveCD.

Dzięki wykonanemu w ramach badań autora oprogramowaniu, można przeprowadzić w stosunkowo prosty sposób tworzenie nowych płyt LiveCD zawierających zmodyfikowane dane, zarówno systemowe, jak i konfiguracyjne. Proces ten odbywa się w następujących krokach:

1. Zmiana danych konfiguracyjnych lub instalacja nowego oprogramowania.
2. Stworzenie na podstawie zmienionych plików nowej płyty LiveCD.
3. Ponowne uruchomienie komputera z nowo przygotowanej płyty LiveCD.

Praca ze standardową wersją dystrybucji cdlinux.pl, bez włączonego systemu ICAR, jest bezpieczna w tym sensie, że intruz nie ma możliwości dokonania jakichkolwiek trwałych zmian w systemie plików. Podczas pracy z płytą, dzięki zastosowaniu wieżowego systemu plików *UnionFS*, użytkownik, a także intruz może dokonywać modyfikacji zawartości plików. Jednak po ponownym uruchomieniu komputera pliki systemowe są wgrywane na nowo z niemodyfikowalnego nośnika danych. Tego typu sposób działania jest przeznaczony do wykorzystania na tzw. komputerach biurkowych (ang. *workstations*) i pozwala na zabezpieczenie samego systemu operacyjnego. Natomiast zadaniem użytkownika jest dbanie o bezpieczeństwo danych wytwarzanych w programach użytkowych, np. poprzez regularne tworzenie kopii zapasowych.

Rozszerzenie bezpieczeństwa systemu operacyjnego uruchamianego z płyty LiveCD poprzez włączenie modułu ICAR polega na ochronie plików również podczas pracy systemu. Takie rozwiązanie może być przydatne w sytuacjach szczególnie wrażliwych na atak intruzów. Przy wdrażaniu systemu ICAR należy mieć na uwadze, że ochronie nie podlegają pliki użytkowników zapisane na dyskach twardych, które powinny być chronione na przykład poprzez kopie zapasowe.

5.5.2. Wykorzystanie mechanizmu ICAR na dyskach twardych

Uruchamianie systemów operacyjnych z płyt typu LiveCD jest zdecydowanie mniej elastyczne i wydajne niż w przypadku systemów zainstalowanych na dyskach twardych komputerów. Wykorzystywanie płyt LiveCD jest dobrym rozwiązaniem przy wykonywaniu prostych czynności, takich jak praca w sieci, czy przy testowaniu nowych rozwiązań. Jednak nie nadaje się do zastosowania w bardziej skomplikowanych sytuacjach, w których konieczne jest wykorzystanie zaawansowanego oprogramowania użytkowego, czy w przypadku serwerów produkcyjnych. Z tego powodu konieczne stało się dostosowanie zaprojektowanego mechanizmu bezpieczeństwa do pracy z systemami operacyjnymi zainstalowanymi na dyskach twardych.

Stworzenie skutecznego mechanizmu bezpieczeństwa dla systemów operacyjnych zapisanych na dyskach twardych jest zagadnieniem niezwykle złożonym, ponieważ w przypadku uzyskania przez intruza uprawnień użytkownika uprzywilejowanego *root*, ma on duże możliwości ingerencji w działanie systemu operacyjnego. Z tych też względów niezbędne jest połączenie zapisu danych na dyskach twardych z zapisem na niemodyfikowalnych nośnikach, ponieważ należy założyć, jak już niejednokrotnie podkreślano w niniejszej pracy, że intruz zawsze będzie w stanie uzyskać kontrolę nad systemem. Uniemożliwienie intruzom zmiany plików systemowych jest niezwykle skutecznym rozwiązaniem podnoszącym bezpieczeństwo systemu operacyjnego. Istnieje kilka metod blokowania zapisu zmian w plikach przechowywanych na dyskach. Metody te różnią się między sobą elastycznością, wydajnością, a także łatwością ich wdrożenia. Można wyróżnić metody czysto programowe, czysto sprzętowe, a także pośrednie polegające na połączeniu metod programowych i sprzętowych.

Metoda programowego blokowania zmian w plikach w systemie Linux polega na uniemożliwieniu zapisu plików na poziomie Wirtualnego Systemu Plików (VFS). Podczas montowania w drzewie katalogów systemu plików można wskazać, że ma być on podłączony w trybie tylko do odczytu (ang. *read-only mode*). W ten sposób zablokowana zostaje jakakolwiek możliwość tworzenia i modyfikacji plików. Jednak użytkownik z uprawnieniami

administracyjnymi może w każdej chwili zmienić opcję montowania (ang. *remount*) i przełączyć zamontowany system plików w tryb do odczytu i zapisu (ang. *read-write mode*). W praktyce oznacza to, że intruz w momencie przejęcia konta administratora może wyłączyć takie programowe zabezpieczenie przed zapisem. W związku z tym metoda ta nie gwarantuje całkowitego bezpieczeństwa chronionych danych i nie nadaje się do zabezpieczania najważniejszych plików systemowych. Metoda programowa blokowania zapisu ma jednak jedną, niepodważalną zaletę, jaką jest łatwość modyfikacji plików chronionych w ten sposób. Administrator po przełączeniu systemu w tryb umożliwiający zapis może dokonać niezbędnych modyfikacji, a następnie wrócić do trybu tylko do odczytu. Podsumowując, blokowanie możliwości zapisu plików chronionych poprzez montowanie systemu plików w trybie tylko do odczytu może być częściowo przydatne do ochrony danych konfiguracyjnych, szczególnie w systemach, w których są one często modyfikowane.

Druga metoda blokowania opiera się tylko i wyłącznie na zabezpieczeniach sprzętowych, tzn. wykorzystuje nośniki danych, które na poziomie fizycznym nie mają możliwości zapisywania danych. Oprócz nośników optycznych takich jak płyty CD/DVD, można wykorzystywać dyski twarde podłączane poprzez specjalnie zmodyfikowane interfejsy IDE lub SATA. Do sprzętowego blokowania zapisu można również wykorzystać podłączane przez złącze USB pamięci przenośne Flash EEPROM wyposażone w przełącznik fizycznie blokujący zapis danych. Zastosowanie metody sprzętowej do zabezpieczenia danych systemowych i konfiguracyjnych zapewnia pełną ochronę plików chronionych przed wrogim działaniem włamywacza. Nawet po uzyskaniu pełnej kontroli nad systemem, intruz nie ma możliwości zdalnego wyłączenia blokady zapisu, gdyż nośniki danych fizycznie na to nie pozwalają. Wadą zastosowania takiego rozwiązania jest mała elastyczność zmiany zawartości plików. Zmiana jakichkolwiek danych chronionych w ten sposób wymaga dużego nakładu pracy. Wiąże się to z koniecznością czasowego wyłączenia systemu, co dla części systemów serwerowych może stanowić duży problem. Pewnym rozwiązaniem zwiększającym elastyczność modyfikacji danych jest zastosowanie pamięci USB z fizycznym przełącznikiem blokującym zapis. Wadą takiego rozwiązania jest jednak stosunkowo niska prędkość odczytu danych – ponad dwukrotnie mniejsza niż oferowana przez powszechnie stosowane dyski twarde.

Wady obydwu metod, programowej i sprzętowej, są na tyle duże, że w praktyce uniemożliwiają wdrożenie ich w systemach produkcyjnych, z wyjątkiem wąskiej grupy zastosowań specjalistycznych. Przykładem mogą być np. urządzenia sieciowe, takie jak routery, których pliki systemowe nie muszą być aktualizowane, a bezpieczeństwo konfiguracji nie jest zagadnieniem krytycznym. W urządzeniach tego typu można zastosować do przechowywania plików systemowych nośniki tylko do odczytu, a do danych konfiguracyjnych z programową blokadą zapisu.

Możliwe jest opracowanie pewnego rozwiązania pośredniego, które połączy w sobie zalety przedstawionych wyżej rozwiązań, a jednocześnie będzie pozbawione ich wad. Takie rozwiązanie zostało stworzone w ramach prac badawczych nad systemem ICAR. Opracowany system łączy w sobie bezpieczeństwo danych przechowywanych na nośnikach z fizycznie zablokowaną możliwością zapisu z nieskomplikowanym i stosunkowo mało czasochłonnym mechanizmem wprowadzania zmian w plikach. Zaproponowane rozwiązanie polega na wykorzystaniu systemu ochrony integralności i przywracania plików ICAR.

W zależności od specyfiki chronionego systemu operacyjnego system ICAR może być wykorzystywany do chronienia wszystkich lub tylko wybranych plików systemowych i konfiguracyjnych.

Główną zaletą zastosowania w systemie ICAR zapisu na dyskach twardych komputera w stosunku do wykorzystywania tylko pamięci przenośnej USB lub płyt CD-ROM jest znacznie większa szybkość odczytu danych. Na płycie CD przechowywana jest tylko baza danych wzorców oraz kopie plików. Płyta ta jest wykorzystywana tylko podczas startu systemu operacyjnego, w którym wczytywane jest jądro wraz z modułami oraz baza danych wzorców. Podczas późniejszego działania systemu operacyjnego ICAR korzysta z danych zapisanych na płycie tylko wtedy, kiedy wykryje zmianę w chronionym pliku. Takie rozwiązanie podnosi bezpieczeństwo systemu operacyjnego, nie powodując jednocześnie zbyt dużego spadku wydajności i elastyczności jego działania.

5.5.3. Instalacja systemu ICAR

Wdrażanie nowych zabezpieczeń do działającego systemu komputerowego jest zawsze operacją skomplikowaną, która często wymaga zmiany istniejącej polityki bezpieczeństwa stosowanej w przedsiębiorstwie bądź instytucji. W celu uproszczenia procesu integracji systemu ochrony ICAR z dowolną dystrybucją systemu Linux stworzono szereg narzędzi, które automatyzują większość skomplikowanych czynności niezbędnych do wdrożenia zabezpieczeń. Wykorzystując opracowane narzędzia, zadanie administratora systemu operacyjnego ogranicza się do podjęcia decyzji, które pliki zostaną włączone do ochrony oraz kompilacji jądra systemu operacyjnego z przygotowanych kodów źródłowych.

Głównym programem ułatwiającym wdrażanie systemu ICAR jest skrypt napisany w języku Perl o nazwie *icarctl*. Jego zadanie polega na przygotowaniu danych niezbędnych do poprawnego działania systemu zabezpieczeń. Na podstawie stworzonej przez administratora systemu listy plików przeznaczonych do ochrony, program tworzy bazę danych zawierającą skróty kryptograficzne każdego chronionego pliku. Tworzona jest również kopia zapasowa każdego chronionego pliku, tak aby była możliwość przywrócenia zawartości pliku po wykryciu nieautoryzowanej zmiany. Na rysunku 5.7 pokazany został zrzut ekranu z komunikatami, jakie wyświetla program *icarctl* w trakcie tworzenia bazy danych i wypełniania jej skrótami kryptograficznymi.

W chwili uruchamiania programu *icarctl* należy podać katalog, do którego zostaną zapisane pliki bazy danych ze skrótami kryptograficznymi oraz kopie chronionych plików. Do poprawnego i w pełni bezpiecznego działania systemu ICAR, konieczne jest zapisanie tych danych na nośniku zabezpieczonym na poziomie fizycznym przed zapisem. W zależności od wybranej technologii administrator może nagrać pliki na płyty CD-ROM, przenieść je na odpowiednie urządzenie USB, czy zapisać na innym nośniku danych. Wskazane jest, aby tego typu czynności wykonywać po gruntownym sprawdzeniu zabezpieczanego systemu pod kątem obecności wrogiego oprogramowania (np. wirusów, czy koni trojańskich), a także przy wyłączonych interfejsach sieciowych, aby nie doszło do ataku intruza w trakcie procesu inicjalizacji zabezpieczeń.

```
# ./icarctl -d ../icar.db -s /mnt/backup$a +a /bin/
w-a--- 3bf1e392d6bf70529e7d0a1d3fc44edff6111966e3be6ba502afae4b87fd6339 /bin/tar /mnt/backup/bin/tar
w-a--- 6110f9d382cce49a6c74facbb4ed5f00ee8cfac42072914376bba908ce881459 /bin/bzexe /mnt/backup/bin/bzexe
w-a--- 354a56950a6c510eb6ece30c2727a7ee737dd379c82d2d66b7f04496fb5458b3 /bin/chmod /mnt/backup/bin/chmod
w-a--- 1223235f2aed85c831896b18975d90caca8a356044250a40e985b1edb6b672cb /bin/sync /mnt/backup/bin/sync
w-a--- 6dad6e431080eb70f6d2759f088f66d738b6766bf93706b9751233a4b42769bf /bin/zforce /mnt/backup/bin/zforce
w-a--- 455c2dcf37247f21b592a893350734e3b9ac3459398ce5b24ee5b405f9fd3dd3 /bin/zcat /mnt/backup/bin/zcat
w-a--- 45bd59e481a8f363478bcafdd5c12627627f22e9d48b9f8a9c4d1e93d059ab98 /bin/more /mnt/backup/bin/more
w-a--- d3a243c1b733bc6bc25943a4aca11e3e95423401d1801ac2661920df12bb2d33 /bin/true /mnt/backup/bin/true
w-a--- 9751d3be223c521d33f822ff284ac4b0dd47aab43c4c10626118ea12ee9b3498 /bin/sleep /mnt/backup/bin/sleep
w-a--- 1f8910d898dc95a35a2aaf8e4638320370e114a855fb16eb33635c285336b25a /bin/su /mnt/backup/bin/su
w-a--- 3ad206b0aca861483633bed5e05d8870754505025c90e0e14dba63daf6aa4b3d /bin/bunzip2 /mnt/backup/bin/bunzip2
w-a--- 375dd63f3f51ee122ecbe481760d385b1d0792cbb2860398436797ed93e4be5e /bin/nano /mnt/backup/bin/nano
w-a--- 65adc879b2d5f154fc17ec9adcb76d9a0d1bdc03b624948be1374d866c1c4d33 /bin/chgrp /mnt/backup/bin/chgrp
w-a--- 471983e38c6b9a7760ec78f616b3cf0252a8c128dac1a1a3a6d54dee7aa50e04 /bin/zless /mnt/backup/bin/zless
w-a--- b987705d40503e42c747bc4bd9d2639c99af360a261dd542f999db3447198456 /bin/bzip2recover /mnt/backup/bin/bzip2recover
w-a--- ed6e3848522f865b095bcb8178109dc9490084827b3a42f51600e7a2fd7de08a /bin/kill /mnt/backup/bin/kill
w-a--- ee4e61382a38184c6f0183366d543dc7a88c8bf536e7f97e6184c87ed38e61d8 /bin/lessecho /mnt/backup/bin/lessecho
w-a--- 56ae71cc45676ab08f5edfd2a5cdbad8ab2101dad8d79e4aab8a898f0af73872 /bin/zcmp /mnt/backup/bin/zcmp
w-a--- f54343f0c9a11011e3d2498357fd823a711b235c5dea3604d42407ec00fd2a02 /bin/ls /mnt/backup/bin/ls
w-a--- ac9eb28c29c76bfe7858a4f0c7fa92ff4307ebfe66dcbbe548ebaf31043820d9 /bin/domainname /mnt/backup/bin/domainname
w-a--- ac9eb28c29c76bfe7858a4f0c7fa92ff4307ebfe66dcbbe548ebaf31043820d9 /bin/ypdomainname /mnt/backup/bin/ypdomainname
w-a--- 804a0ef6a9ea3d7801730b2b1dc95fa9aa03e7c9a5ff612dcf2e671583283b9f /bin/umount /mnt/backup/bin/umount
w-a--- 2ae38460385590 38f20d20142438572986a64d2b15abd8ea3ed3da13a7a1ee68 /bin/loadkeys /mnt/backup/bin/loadkeys
w-a--- be031b61a535a8fc17687e4d303b3106438e2ef1bb7c80bab3ed6a367873cb83 /bin/dmesg /mnt/backup/bin/dmesg
w-a--- ac9eb28c29c76bfe7858a4f0c7fa92ff4307ebfe66dcbbe548ebaf31043820d9 /bin/nisdomainname /mnt/backup/bin/nisdomainna
w-a--- 6331e7f55e62d3b0dec669cf6eda1d77d726c54ac947f3f62b5315e5a932fe87 /bin/ps /mnt/backup/bin/ps
```

Rysunek 5.7. Komunikaty programu *icarctl*

Wykorzystanie systemu zabezpieczeń ICAR wymaga dołączenia przygotowanego modułu do jądra systemu operacyjnego i jego kompilację. W związku z tym, że system ICAR jest rozpowszechniany w formie kodów źródłowych, jego instalacja jest możliwa na każdej dystrybucji systemu operacyjnego Linux, która jest wyposażona w odpowiednią wersję jądra. Prototypowe źródła systemu ICAR zostały przystosowane dla integracji z jądrem systemu operacyjnego Linux w wersji 2.6.30. Do kompilacji modułu ICAR niezbędny jest cały kod źródłowy jądra.

Proces instalacji modułu ICAR składa się z trzech kroków. Na początku należy na oryginalny kod źródłowy jądra Linux (ang. *vanilla kernel source code*) nałożyć łatę (ang. *patch*) z kodem źródłowym systemu ICAR. Drugi krok polega na włączeniu mechanizmu ICAR w konfiguracji jądra. Należy również wskazać miejsce przechowywania bazy danych wzorców oraz kopii chronionych plików. Informacja ta, podczas kompilacji, zostaje na sztywno zapisana w pliku binarnym modułu ICAR. Celowo zrezygnowano z możliwości dynamicznej zmiany tych kluczowych informacji podczas pracy systemu operacyjnego. Dzięki takiemu zabiegowi, intruz nawet w momencie całkowitego przejęcia kontroli nad systemem operacyjnym nie będzie w stanie przeprowadzić ataku na system ICAR polegającego np. na wskazaniu innych, spreparowanych kopii plików chronionych czy bazy danych wzorców. Zmiana konfiguracji jądra może zostać wykonana za pomocą standardowych narzędzi systemu Linux, np. polecenia *make menuconfig*. Ostatnim krokiem jest skompilowanie kodu źródłowego jądra do postaci binarnej i jego instalacji w programie rozruchowym.

Tak przygotowany system ochrony ICAR jest gotowy do pracy i zostanie aktywowany przy ponownym uruchomieniu systemu operacyjnego. Od tego momentu każda operacja otwarcia lub uruchomienia pliku będzie monitorowana i po wykryciu zmiany w pliku chronionym jego zawartość zostanie przywrócona z kopii zapasowych.

Jedną z podstawowych zalet wykorzystanego przy implementacji mechanizmu ICAR szkieletu *Linux Security Modules* jest blokada możliwości wyłączenia modułu ochrony. Oznacza to, że przy ciągłej pracy systemu operacyjnego intruz nie ma możliwości wyłączenia mechanizmu ICAR. Jedyną możliwością obejścia zabezpieczeń jest uruchomienie systemu operacyjnego za pomocą spreparowanego pliku jądra. Na taki atak narażone są wszystkie,

dotychczas stworzone mechanizmy ochrony integralności plików. Dlatego w celu wyeliminowania możliwości podmiany pliku jądra proponuje się, aby w systemach serwerowych o dużym znaczeniu, wykorzystywać do uruchamiania systemu operacyjnego technologię LiveCD.

Oczywiście nie ma potrzeby, aby cały system operacyjny był uruchamiany z niemodyfikowalnego nośnika. Wystarczy, że plik jądra i pliki jego modułów będą zabezpieczone przed zapisem na poziomie fizycznym. Taki mieszany sposób uruchamiania systemu operacyjnego, gdzie jądro i jego moduły są uruchamiane z nośnika niemodyfikowalnego, a wszystkie pozostałe pliki z dysku twardego, zapewnia bezpieczeństwo systemu ICAR zachowując jednocześnie elastyczność i szybkość działania normalnie zainstalowanego systemu operacyjnego na dysku twardym.

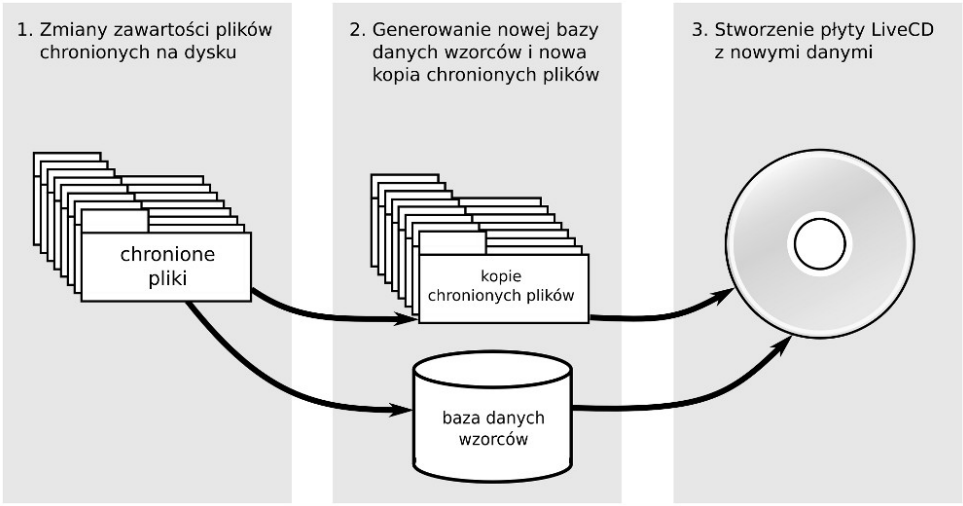
Jednym z głównych założeń podjętych przy projektowaniu systemu ICAR była transparentność jego działania. Użytkownik systemu nie musi być świadomy jego obecności. Oznacza to, że z punktu widzenia użytkownika wszystkie pliki w systemie operacyjnym mogą być modyfikowane. Podczas zmiany zawartości chronionego pliku nie są generowane żadne komunikaty czy błędy o braku możliwości zapisu. Wszelkie informacje o pliku, jego metadane, takie jak np. data ostatniej modyfikacji czy rozmiar pliku zmienia się zgodnie z dokonanymi modyfikacjami. System zabezpieczeń ICAR dopiero w momencie próby odczytu zmodyfikowanego pliku przeprowadza operację sprawdzenia integralności pliku i w przypadku wykrycia zmiany, przywraca oryginalną zawartość pliku z kopii przechowywanej na bezpiecznym nośniku. Dotyczy to tylko tych plików, które zostały wybrane do ochrony, a nie wszystkich plików w systemie.

Celem zastosowania takiego opóźnionego wykrywania zmian w plikach w stosunku do natychmiastowego blokowania możliwości zapisu pliku jest próba ukrycia przed intruzem obecności systemu zabezpieczeń ICAR. Włamywacz podczas zamiany pliku systemowego na wrogi program (np. backdoor) nie będzie świadomy, że jego działanie zakończyło się niepowodzeniem. W przeciwnym razie mógłby podjąć działania destrukcyjne, takie jak np. usunięcie plików użytkowników. Dzięki zastosowaniu systemu ICAR administrator zostanie poinformowany o wykryciu zmiany w pliku chronionym i będzie miał czas na wyeliminowanie błędów, dzięki którym intruz uzyskał dostęp do systemu operacyjnego przed jego ponownym atakiem.

Jedna z ważniejszych czynności, jaką musi okresowo wykonywać administrator zarządzający systemem operacyjnym z włączonym systemem zabezpieczeń ICAR jest związana z aktualizacją plików systemowych. W przypadku konieczności instalacji nowych wersji programów systemowych, których pliki podlegają ochronie czy zmianie chronionych ustawień konfiguracyjnych, konieczne jest stworzenie nowej wersji bazy danych skrótów kryptograficznych i kopii chronionych plików.

Na rysunku 5.8 przedstawiono trzy etapy zmiany zawartości plików chronionych oraz związanej z tym aktualizacji danych mechanizmu bezpieczeństwa ICAR. W pierwszym kroku administrator dokonuje aktualizacji chronionych plików na dysku twardym komputera przy wyłączonym systemie zabezpieczeń ICAR. Następnie na podstawie listy chronionych plików generowana jest nowa baza wzorców oraz tworzone są kopie chronionych plików. Ostatni krok polega na zapisaniu, na niemodyfikowalnym nośniku danych, zaktualizowanej bazy danych, kopii plików oraz przekompilowanego jądra systemu operacyjnego z modułem zabezpieczającym.

Istotną zaletą proponowanego rozwiązania jest krótki czas przerwy w działaniu systemu operacyjnego. Przy zastosowaniu systemu ICAR konieczne jest wykonanie tylko jednego restartu systemu operacyjnego, związanego ze zmianą niemodyfikowalnego nośnika danych. Tego typu działania można wykonywać np. w nocy, kiedy większość użytkowników nie pracuje. Pomimo faktu, że modyfikacja chronionych za pomocą systemu ICAR danych nie jest zadaniem trywialnym, to z wykorzystaniem przygotowanych narzędzi nie jest czynnością czasochłonną.



Rysunek 5.8. Procedura aktualizacji danych systemu ICAR

Wdrożenie mechanizmu zabezpieczeń ICAR w systemie operacyjnym Linux powinno zwiększyć bezpieczeństwo systemów komputerowych. Intruz nawet po uzyskaniu dostępu do konta administratora z pełnymi uprawnieniami systemowymi nie będzie w stanie trwale zmienić zawartość chronionych plików, gdyż będą one automatycznie przywracane z kopii zapasowej w momencie próby odczytu ich zawartości. Operacja trwałej zmiany chronionych danych jest możliwa tylko po uzyskaniu fizycznego dostępu do komputera i zamianie niemodyfikowalnego nośnika. Na podstawie przedstawionych w tym rozdziale badań można więc postawić tezę, że:

> **Automatyczne przywracanie poprawnej zawartości plików chronionych w momencie wykrycia nieautoryzowanej modyfikacji zapewnia poprawność działania systemu operacyjnego.**

System ICAR uniemożliwia intruzowi zdalną instalację wrogich programów, takich jak konie trojańskie czy programy typu *backdoor*. Należy stwierdzić z całą stanowczością, że zastosowanie niemodyfikowalnych nośników zapewnia bezpieczeństwo przechowywanych danych. Jednocześnie, nawet przy wykorzystaniu wolniejszych nośników, takich jak płyty CD-ROM, wydajność działania systemu operacyjnego zabezpieczonego mechanizmem ICAR nie zmniejsza się w sposób znaczący. Dzięki swoim zaletom zaprojektowany mechanizm bezpieczeństwa nadaje się do powszechnego zastosowania, nawet w systemach produkcyjnych o dużym znaczeniu strategicznym.

Rozdział 6.
Ocena mechanizmu bezpieczeństwa ICAR

Opracowany system ochrony integralności plików ICAR został zaimplementowany w postaci modułu jądra systemu operacyjnego Linux. Przeprowadzone testy wykazały, że zachowuje się zgodnie z zaprojektowanym algorytmem. Jednak w celu udowodnienia przydatności proponowanego rozwiązania konieczne było przeprowadzenie oceny z uwzględnieniem różnych aspektów.

Na początku ocenie poddana została wydajność. Każde rozszerzenie bezpieczeństwa wpływa na czas przetwarzania zadań w systemie operacyjnym. Jest to związane w wykonywaniem dodatkowych operacji, takich jak np. kontrola praw dostępu do plików. Jednym z warunków jakie musi spełniać system ochrony, aby mógł być powszechnie stosowany, jest niewielki i akceptowalny wzrost czasu wykonywania zadań. W przypadku, gdy taki wzrost będzie zbyt duży zyski wynikające z ochrony będą mniejsze niż straty spowodowane gorszą wydajnością. Nie można jednak arbitralnie określić akceptowalnych wartości gdyż będą one różne w zależności od stosowanego systemu oraz jego przeznaczenia. Dla systemów czasu rzeczywistego najmniejsze opóźnienia w wykonywaniu zadań mogę dyskwalifikować mechanizm ochrony. Natomiast w przypadku serwerów obliczeniowych, gdzie czas trwania zadań jest liczony w godzinach, a nawet dniach, wzrost czasu wykonywania o kilka minut nie będzie zauważalny. Mechanizm bezpieczeństwa ICAR zwiększa czas dostępu do plików, który w nowoczesnych systemach komputerowych jest nie dłuższy niż kilka sekund dla bardzo dużych plików. Przy przeprowadzaniu testów przyjęto, że wzrost czasu wykonywania operacji dla plików chronionych nie może być większy niż 50%.

W ramach procesu weryfikacji systemu ICAR została przeprowadzona również ocena wiarygodności opracowanego mechanizmu. Do oceny wiarygodności została wybrana metoda *Trust Case*. Pozwala ona na wydanie decyzji o wiarygodności systemów informatycznych w ściśle zdefiniowanym kontekście. Na podstawie wielowymiarowej oceny zbioru twierdzeń dotyczących poszczególnych aspektów badanego systemu można określić poziom wiarygodności całego produktu [38].

W niniejszym rozdziale została również opisana przydatność opracowanego mechanizmu pod kątem oceny bezpieczeństwa systemu operacyjnego zgodnie z normą PN-ISO/IEC 15408, powszechnie znaną jako *Common Criteria*. Mechanizm ICAR nie jest oddzielnie działającym systemem, lecz częścią systemu operacyjnego. W związku z tym analizie zostały poddane korzyści dla bezpieczeństwa systemu operacyjnego wynikające z zastosowania opracowanego mechanizmu, w wyniku której zidentyfikowano dodatkowe wymagania funkcjonalne, nie zdefiniowane w dokumencie *Profil Ochrony Systemów Operacyjnych*.

Rozdział został zakończony dyskusją dotyczącą perspektyw rozwoju systemu ICAR. Dalsze badania będą prowadzone w kierunku wykorzystania technik wirtualizacji do zwiększenia elastyczności konfiguracji systemu oraz jego bezpieczeństwa.

6.1. Testy wydajnościowe systemu ICAR

W celu oceny praktycznej przydatności mechanizmu ICAR zostały przeprowadzone badania jego wydajności, które miały określić, o ile zwiększa się czas wykonania operacji na grupie plików chronionych. Zostały przeprowadzone dwa typy eksperymentów. Pierwszy polegał na pomiarze czasów obliczania sum kontrolnych dla zbioru różnej wielkości plików. W drugim eksperymencie mierzono natomiast czasy wykonywania typowych operacji przetwarzania plików chronionych, takich jak zmiana wielkości plików graficznych, kompilacji programu użytkowego czy konwersja plików wideo. W czasie przeprowadzania testów na komputerze testowym nie były uruchomione żadne dodatkowe aplikacje. Testy przeprowadzono na komputerze o następujących parametrach: procesor Intel Core2 Quad Q6600 2.40GHz, 3GB pamięci RAM, dysk twardy Samsung HD322IJ 320GB *72*00rpm, jądro systemu operacyjnego Linux w wersji 2.6.30.

Pomiary wydajności systemu zabezpieczeń były przeprowadzane w trzech wariantach:

1. Bez systemu ochrony ICAR (NOICAR),
2. Z systemem ICAR wraz z mechanizmem buforów podręcznych (CACHE),
3. Z systemem ICAR bez buforów podręcznych (NOCACHE).

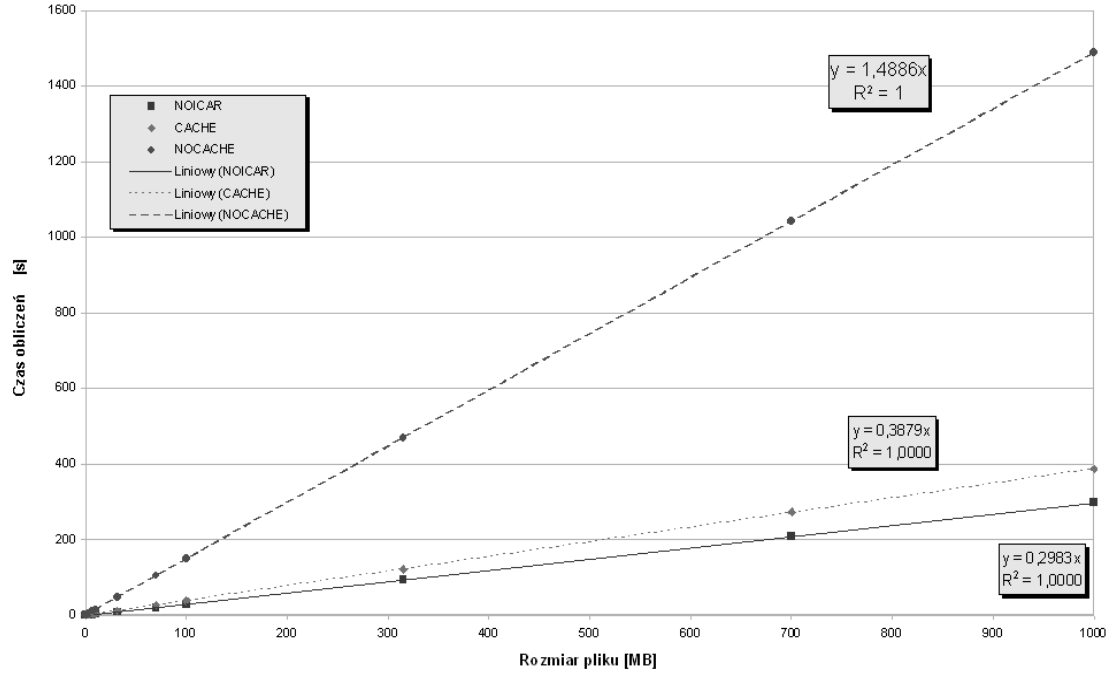
Skróty poszczególnych wariantów podane w nawiasach będą wykorzystane na rysunkach obrazujących rezultaty pomiarów.

W pierwszym typie eksperymentu, w celu oceny wpływu wielkości pliku podlegającego ochronie na czas uzyskania do niego dostępu, przygotowano pliki zawierające dane binarne o rozmiarach od 100kB do 1GB. Dla każdego pliku, w celu uzyskania mierzalnych wielkości, dokonywano 100 operacji liczenia sum kontrolnych. Taki pomiar dokonywano dziesięciokrotnie, rejestrując czasy wykonania. Analizując rozrzut uzyskanych wyników można stwierdzić, że ich powtarzalność jest duża, a średniokwadratowe odchylenie czasów wykonania nie przekracza 0,01%. Oznacza to, że wyniki są powtarzalne, o ile system komputerowy nie jest obciążony innymi procesami związanymi z działaniem aplikacji.

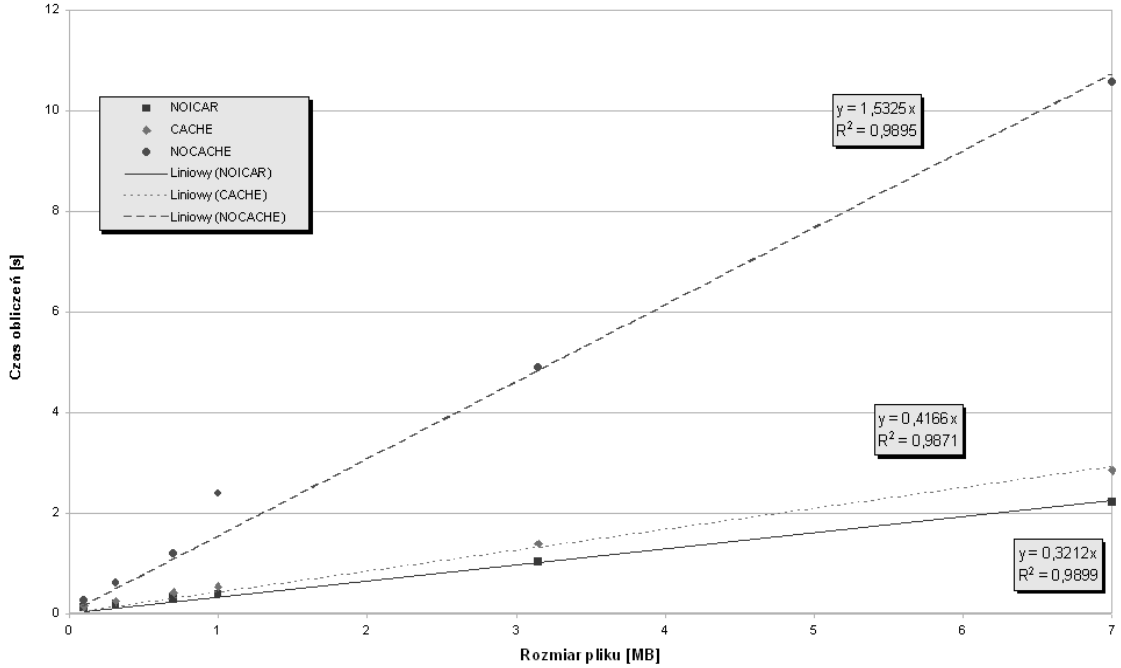
Tabela 6.1. Pomiary czasu liczenia sum kontrolnych

ROZMIAR PLIKU [MB]	NOICAR [s]	CACHE [s]	NOCACHE [s]
0,100	0,13	0,15	0,26
0,315	0,20	0,25	0,61
0,700	0,31	0,42	1,19
1,000	0,39	0,54	2,39
3,150	1,03	1,39	4,88
7,000	2,21	2,84	10,56
10,000	3,11	3,99	15,01
31,500	9,55	12,32	47,30
70,000	21,00	27,16	104,89
100,000	29,96	38,72	149,09
315,000	94,27	122,06	469,85
700,000	208,82	273,02	1042,37
1000,000	298,18	386,90	1487,99

Wyniki pomiarów operacji obliczania sum kontrolnych dla zdefiniowanych przypadków testowych przedstawiono w tabeli 6.1. Na podstawie uzyskanych wyników została przeprowadzona analiza regresyjna, której celem było stwierdzenie czy zależność czasu dostępu od wielkości plików jest liniowa. Ponieważ rozmiary plików są w dużym przedziale zmienności, przeanalizowano uzyskane wyniki w dwóch zakresach wielkości, do 7MB oraz do 1GB. Uzyskane rezultaty przedstawiono na rysunkach 6.1 i 6.2.



Rysunek 6.1. Czas wykonania operacji na pliku w zależności od jego rozmiaru (do 1GB)



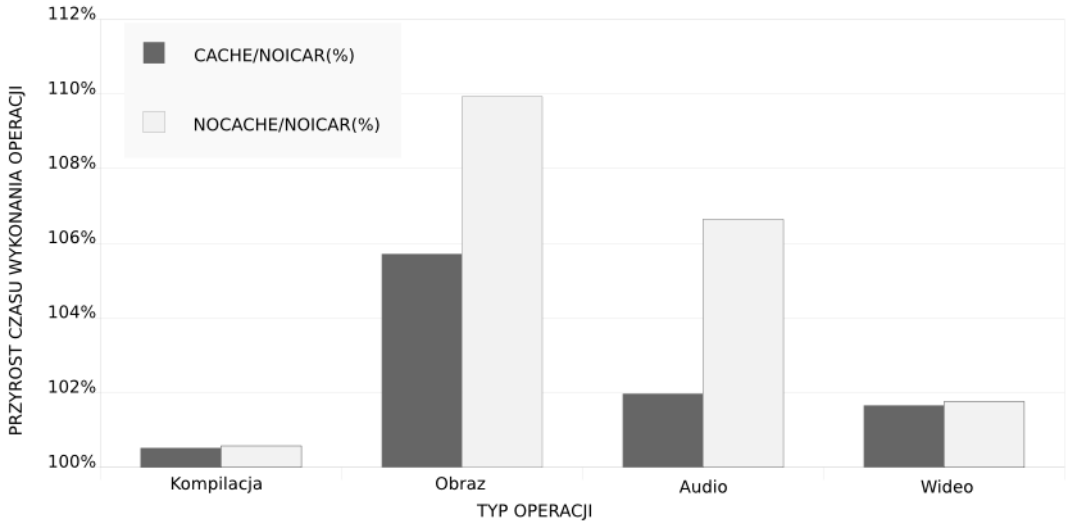
Rysunek 6.2. Czas wykonania operacji na pliku w zależności od jego rozmiaru (do 7MB)

Na podstawie uzyskanych pomiarów przedstawionych na rysunku 6.1 można stwierdzić, że zależność pomiędzy czasem obliczania sumy kryptograficznej a wielkością plików jest liniowa. Dotyczy to zarówno systemu operacyjnego bez mechanizmu ICAR, jak również w przypadku, gdy mechanizm ten jest uruchomiony. Przeprowadzone obliczenia korelacji liniowej Pearsona (ang. *Pearson product-moment correlation coefficient*) dla trzech zdefiniowanych wariantów dały wynik równy 1,0, co oznacza całkowitą korelację. Jednakże szczegółowa analiza wyników uzyskanych dla plików o małych rozmiarach, które zostały przedstawione na rysunku 6.2, wskazuje na pewne odchylenia od zależności liniowej, ponieważ współczynnik korelacji liniowej Pearsona wynosi 0,98. Należy podkreślić, że rezultat taki uzyskuje się również przy pomiarach sum kryptograficznych, gdy system operacyjny działa bez mechanizmu zabezpieczeń. Oznacza to, że zjawisko to związane jest z działaniem systemu operacyjnego, a nie obecnością mechanizmu zabezpieczeń ICAR. Pewien wpływ może mieć także dokładność pomiarów krótkich czasów obliczeń sum kontrolnych. Pomimo niewielkich odchyleń korelacji liniowej Pearsona od wartości 1,0 dla małych plików, należy stwierdzić, że zależność pomiędzy rozmiarem pliku a czasem obliczenia sumy kontrolnej jest liniowa, zarówno bez systemu zabezpieczeń, jak również, gdy jest on aktywny. Wynika z tego, że dołączenie do jądra systemu operacyjnego modułu ICAR nie zmienia złożoności obliczeniowej mechanizmów dostępu do plików dla systemu operacyjnego.

Na rysunku 6.1 przedstawiono wyniki pomiarów czasów obliczeń sum kontrolnych dla systemu ICAR bez aktywnego mechanizmu buforowania (NOCACHE), jak również z działającym mechanizmem buforowania (CACHE). Porównując uzyskane wyniki pomiarów czasu dostępu do plików przez system operacyjny oraz czasy dostępu do tych plików, kiedy dokonana zostanie dodatkowa operacja weryfikacji integralności pliku przez system ICAR, można zauważyć, że przy uruchomionym mechanizmie buforów podręcznych czas wykonania operacji wzrasta nie więcej niż o 30%. W przypadku braku mechanizmu buforów podręcznych czas operacji wzrasta nawet do 400%. Zmniejszenie czasu wykonywania operacji przy zastosowaniu mechanizmu buforów podręcznych wynika z faktu, że weryfikacja integralności pliku polegająca na obliczeniu skrótu kryptograficznego SHA-256, wykonywana jest tylko raz w testowej próbie stu operacji, a nie każdorazowo, jak ma to miejsce w przypadku systemu ICAR bez mechanizmu buforów podręcznych. Uzyskane wyniki dowodzą przydatności zastosowania buforów podręcznych w systemie zapewniania bezpieczeństwa ICAR.

Drugi typ eksperymentu polegał na kontroli czasu wykonania operacji przetwarzania plików przy włączonym i wyłączonym systemie ICAR. Przeprowadzono badania dla przykładowych operacji, takich jak kompilacja programów użytkowych, zmiana rozmiarów zdjęć cyfrowych, konwersja plików audio oraz kompresja plików wideo. Badania miały na celu określenie wpływu typu operacji przetwarzania plików na efektywność systemu ICAR. W tego typu badaniach trudno zdefiniować typowe warunki testowe, dlatego uzyskane rezultaty są jedynie pewnym oszacowaniem wydajności systemu ICAR z punktu widzenia użytkownika.

Eksperyment składał się z kompilacji programu użytkowego, zmiany wielkości plików graficznych oraz konwersji plików audio i wideo. Do kompilacji wybrano program napisany w języku C, którego kod źródłowy składał się z ponad 12 tysięcy plików o łącznym rozmiarze 332 MB. Przetwarzanie 20 plików graficznych o średnim rozmiarze 4MB polegało na zmianie ich rozdzielczości. Zbiór 20 plików muzycznych o średnim rozmiarze 5MB został przekonwertowany z formatu MP3 do formatu Ogg Vorbis. Natomiast jeden plik wideo o rozmiarze 1,4GB został poddany kompresji.



Rysunek 6.3. Wzrost czasu wykonywania wybranych operacji na plikach

Na rysunku 6.3 przedstawiono procentowy wzrost czasu przeprowadzenia operacji przetwarzania plików przy działającym mechanizmie ICAR w stosunku do systemu operacyjnego bez tego mechanizmu. Słupek lewy dotyczy działania systemu ICAR z włączonym mechanizmem buforów podręcznych (CACHE/NOICAR), a prawy bez tego mechanizmu (NOCACHE/NOICAR). Uzyskane wyniki wskazują na przydatność mechanizmu buforowania, zwłaszcza przy operacjach, w których dokonuje się wielokrotnego otwierania tego samego pliku w trakcie jego przetwarzania. Można to zauważyć porównując rezultaty pomiarów przetwarzania zdjęć i filmów. Pomimo, że plik wideo ma znacznie większy rozmiar niż pliki graficzne, to zysk z wykorzystania buforów podręcznych jest minimalny. Wynika to z faktu, że program do konwersji plików graficzny otwiera taki plik wielokrotnie, natomiast program do kompresji plików wideo otwiera taki plik tylko jeden raz. Zwiększenie wydajności wynikające z zastosowania buforów podręcznych zależy zarówno od typu przetwarzanych danych, jak i od sposobu implementacji mechanizmu dostępu do plików w programach użytkowych.

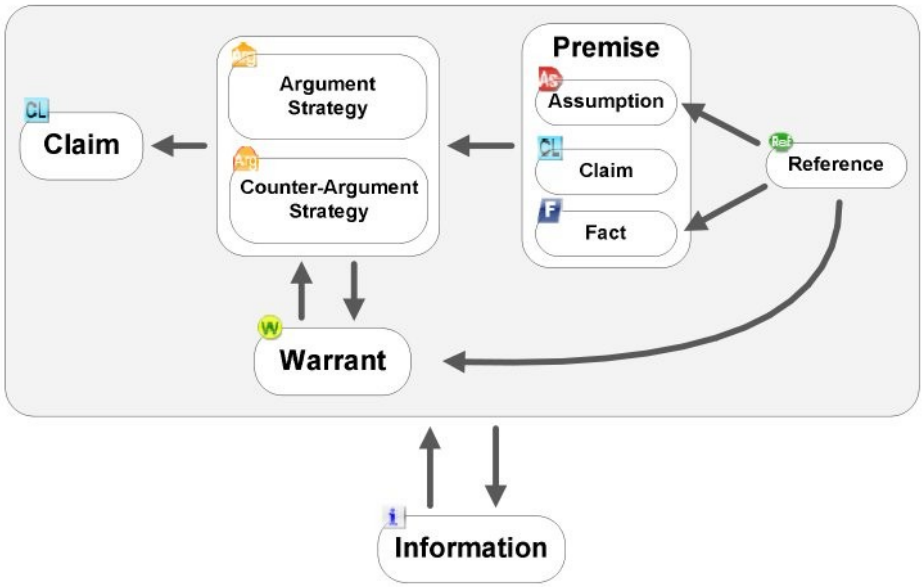
Z przeprowadzonych pomiarów wynika, że przyrost czasu dokonywania typowych operacji przetwarzania plików z działającym mechanizmem zabezpieczeń systemu plików ICAR nie przekracza 10%. Oznacza to, że istnienie i działanie mechanizmu ICAR będzie praktycznie niezauważalne dla użytkownika systemu komputerowego.

Na podstawie przeprowadzonych testów, porównując czasy dostępu do plików chronionych systemem ICAR z włączonym i wyłączonym mechanizmem buforów podręcznych można postawić tezę, że:

> **Zastosowanie funkcji skrótów kryptograficznych wraz z mechanizmem buforów podręcznych do zabezpieczania kluczowych danych nie zmniejsza znacząco wydajności systemu operacyjnego.**

6.2. Ocena wiarygodności systemu ICAR

Opracowany model zabezpieczania systemów plików i wykonany w ramach pracy doktorskiej system ICAR pozwala na zabezpieczanie kluczowych plików systemowych, poprzez kontrolę modyfikacji ich zawartości. W celu udowodnienia wiarygodności stworzonego systemu przeprowadzono ocenę wiarygodności metodą *Trust Case* z wykorzystaniem frameworku *Trust-IT*, który został opracowany przez grupę *Information Assurance Group* (*IAG*), działająca na Politechnice Gdańskiej. Framework *Trust-IT* ułatwia wykorzystywanie metody Trust Case do przeprowadzania ocen wiarygodności systemów informatycznych [37].



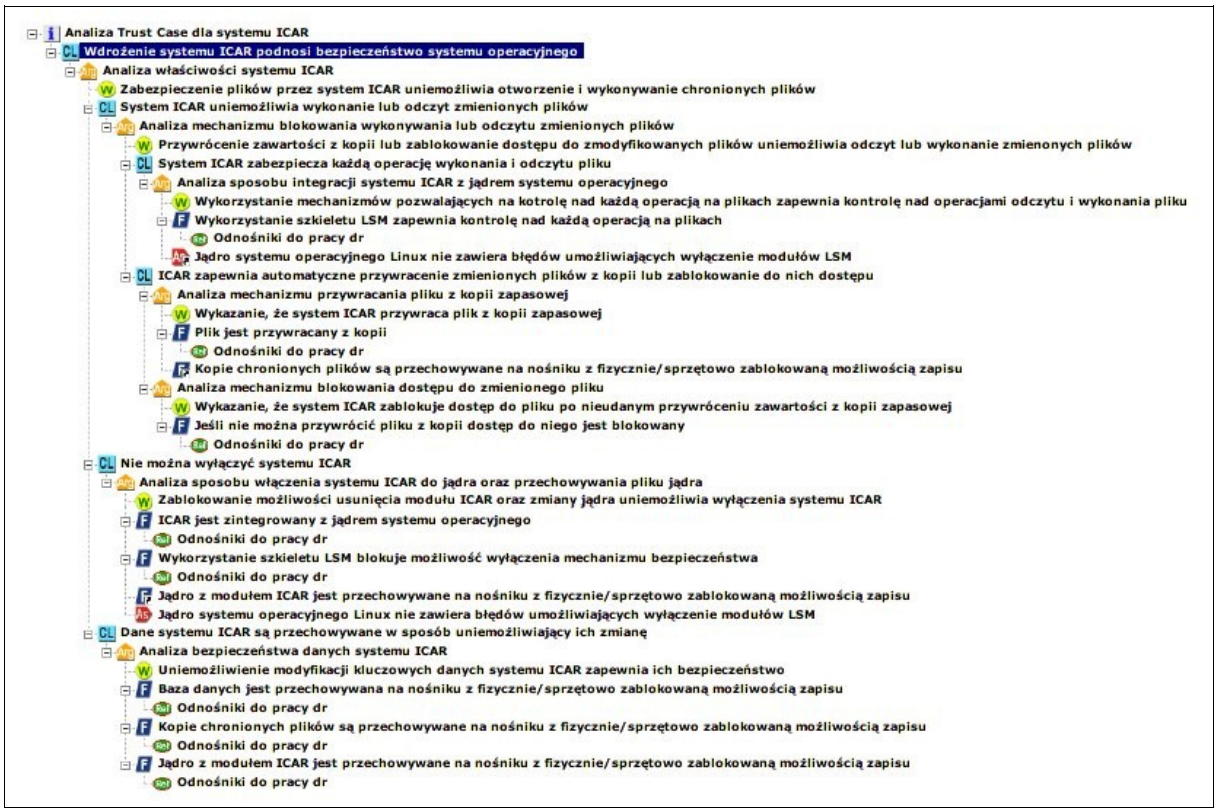
Rysunek 6.4. Model argumentacji Trust-IT [35]

Na rysunku 6.4 przedstawiono schematyczny model argumentacji Trust-IT, który bazuje na powszechnie uznanym i wykorzystywanym modelu Toulumina. Ocena wiarygodności jest tworzona poprzez wybranie dokładnie sprecyzowanego zbioru twierdzeń (ang. *claim*) dotyczących ocenianego systemu, a następnie zebraniu przesłanek (ang. *premise*) na ich poparcie. Relacja, jaka zachodzi pomiędzy zbiorem przesłanek a twierdzeniem jest reprezentowana poprzez strategię argumentacji (ang. *argument strategy*). Zgodnie z modelem Toulumina, każda strategia argumentacji musi posiadać uzasadnienie (ang. *warrant*), które wyjaśnia, w jaki sposób przesłanki udowodniają twierdzenia [35].

Przesłankami stanowiącymi uzasadnienie twierdzeń mogą być fakty, założenia oraz inne twierdzenia. Faktem (ang. *fact*) jest zweryfikowane twierdzenie, którego prawdziwość nie budzi wątpliwości. Założenie (ang. *assumption*) jest stwierdzeniem, które przyjmowane jest jako prawda, jednak w odróżnieniu od faktu nie musi posiadać materiałów dowodowych. Przesłanką może być również inne twierdzenie (ang. *claim*), które będzie następnie udowadniane za pomocą kolejnych przesłanek [36]. Dzięki takiej konstrukcji możliwe jest stworzenie struktury drzewiastej dowolnej głębokości, w korzeniu której znajduje się twierdzenie do udowodnienia.

Celem oceny systemu ICAR metodą Trust Case było zbadanie wiarygodności następującego twierdzenia: **Wdrożenie systemu ICAR podnosi bezpieczeństwo systemu operacyjnego**. Do przeprowadzenia oceny został wykorzystany program *TCTEditor,* stworzony przez grupę

IAG. Na rysunku 6.5, wygenerowanym w programie *TCTEditor*, została przedstawiona struktura drzewiasta, w której na początku znajduje się twierdzenie, a następnie poszczególne elementy udowadniające twierdzenie.



Rysunek 6.5. Ocena wiarygodności systemu ICAR w programie *TCTEditor*

Udowodnienie twierdzenia mówiącego, że oceniany system podnosi bezpieczeństwo, zostało przeprowadzone w oparciu o analizę właściwości systemu ICAR. Podstawowym działaniem systemu ICAR jest zablokowanie możliwości wykonywania i odczytu zmodyfikowanych plików chronionych. W związku z tym pierwsze twierdzenie mówi, że: **System ICAR uniemożliwia wykonywanie i odczyt zmienionych plików**. W celu dokonania dokładnej oceny wiarygodności tego twierdzenia argumentacja została przeprowadzone poprzez analizę mechanizmu blokowania wykonywania operacji odczytu i uruchamiania plików chronionych. Dodatkowo postawione zostały kolejne dwa twierdzenia mówiące o tym, że system ICAR zabezpiecza każdą operację odczytu i uruchamiania plików oraz, że system zapewnia automatyczne przywrócenie zawartości zmodyfikowanego pliku z kopii zapasowej lub w przypadku gdy okaże się to niewykonywalne, zablokuje do niego dostęp. Do tych twierdzeń zostały przypisane fakty i założenia, które zostaną przedstawione szczegółowo w dalszej części rozdziału, przy opisie poszczególnych przesłanek.

Główne założenie posiada jeszcze dwie przesłanki, które dotyczą bezpieczeństwa działania samego systemu. Pierwsze twierdzenie zakłada brak możliwości wyłączenia systemu ICAR. Drugie natomiast dotyczy ochrony danych kluczowych do działania systemu zabezpieczeń poprzez uniemożliwienie ich modyfikacji.

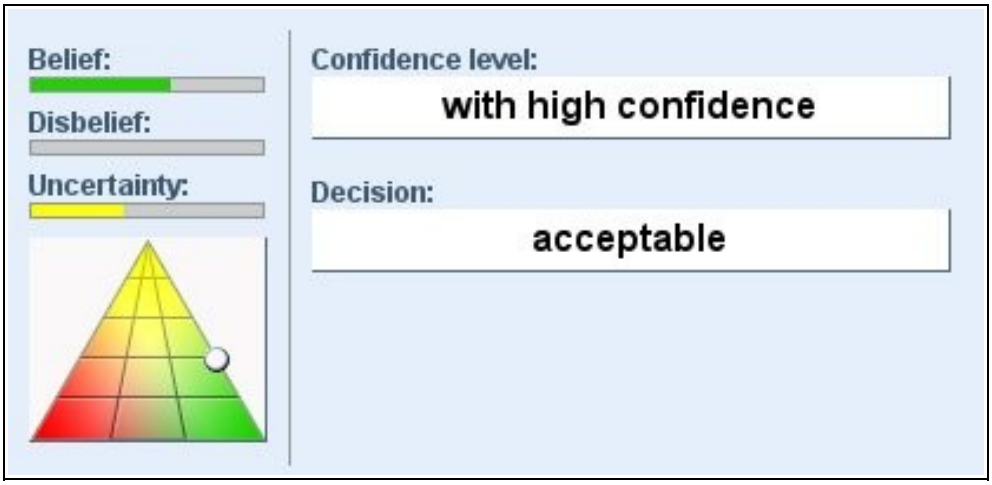
Tak skonstruowany model może zostać poddany ocenie. Program *TCTEditor* umożliwia przydzielenie każdemu faktowi, założeniu i uzasadnieniu dwuwymiarowej oceny. W pierwszym kroku oceniający określa poziom swojej pewności (ang. *confidence level*) dotyczący oceny, przydzielając jedną z czterech wartości:

- akceptowalna (ang. *acceptable*),
- tolerowana (ang. *tolerable*),
- podważalna (ang. *opposable*),
- nieakceptowalna (ang. *rejectable*).

W drugim kroku wydawana jest decyzja (ang. *decision*) co do wiarygodności przesłanki, również w skali opisowej. Dostępnych jest sześć wartości:

- na pewno (ang. *for sure*),
- z bardzo dużą pewnością (ang. *with very high confidence*),
- z dużą pewnością (ang. *with high confidence*),
- z małą pewnością (ang. *with low confidence*),
- z bardzo małą pewnością (ang. *with very low confidence*),
- brak pewności (ang. *lack of confidence*).

Kombinacja tych wartości pozwala na przydzielenie ocenie wiarygodności jednej z dwudziestu czterech wartości opisowych, takich jak np. *ocena akceptowalna z bardzo dużą pewnością*. Ocena poszczególnych węzłów drzewa argumentacji została wykonana w programie *TCTEditor*, w którym jest ilustrowana na trójkącie opinii Josanga (ang. *Josang's opinion triangle*) [25]. Przykładowa ocena z programu została pokazana na rysunku 6.6.



Rysunek 6.6. Ocena wiarygodności węzła w programie TCTEditor

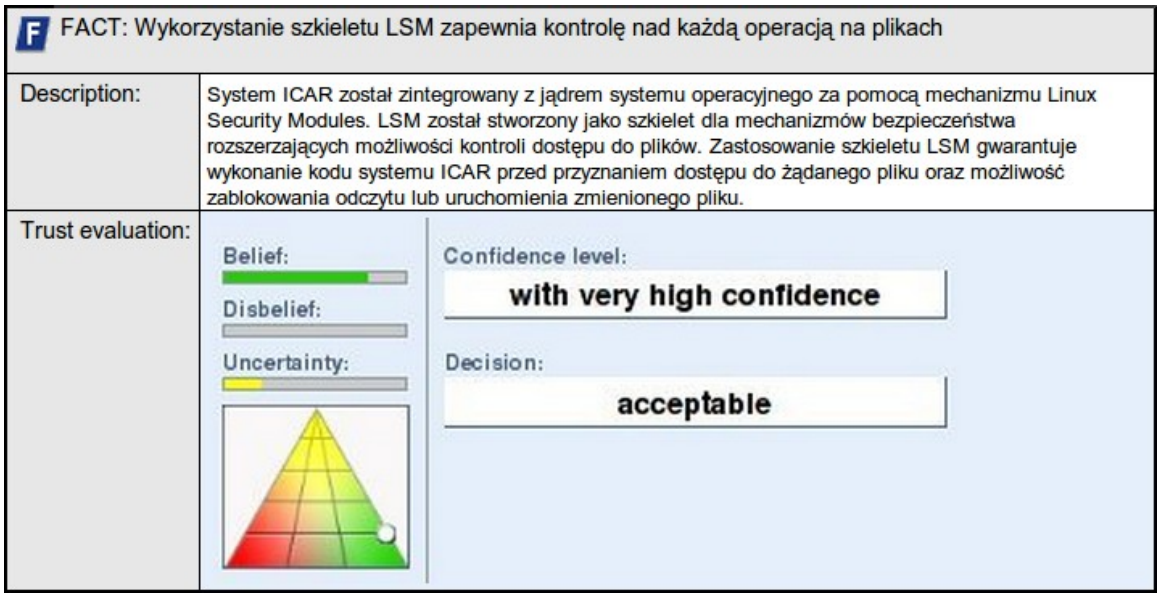
Takie przedstawienie oceny pokazuje w sposób obrazowy, że przy mniejszym poziomie zaufania, decyzja co do wiarygodności ma mniejsze znaczenie, gdyż ocena wiarygodności jest niepewna (ang. *uncertainty*).

Wykorzystując metodę Trust Case zostaną opisane fakty i założenia uzasadniające twierdzenia z drzewa argumentacji wraz ze szczegółową ich oceną. Następnie zostanie pokazane, jak oceny wydane na liściach drzewa argumentacji propagują do jego korzenia.

W ramach przeprowadzonej argumentacji dokonano oceny 5 twierdzeń, które zostały przedstawione poniżej.

Twierdzenie 1: System ICAR zabezpiecza każdą operację wykonania i odczytu pliku

Zbiór przesłanek, które dowodzą to twierdzenie składa się z jednego faktu i jednego założenia. Fakt mówi, że: *Wykorzystanie szkieletu LSM w implementacji systemu ICAR zapewnia kontrolę nad każdą operacją na plikach*. Mechanizm LSM został szczegółowo opisany w rozdziałach 3.2.1 i 5.4.1. Wyłączając sytuacje wynikające z błędnej implementacji jądra systemu Linux fakt ten jest prawdziwy. W związku z tym otrzymał ocenę: *with very high confidence acceptable*. Tabela dla tej przesłanki wraz z trójkątem Josanga, wygenerowana w programie *TCTEditor* została pokazana na rysunku 6.7.

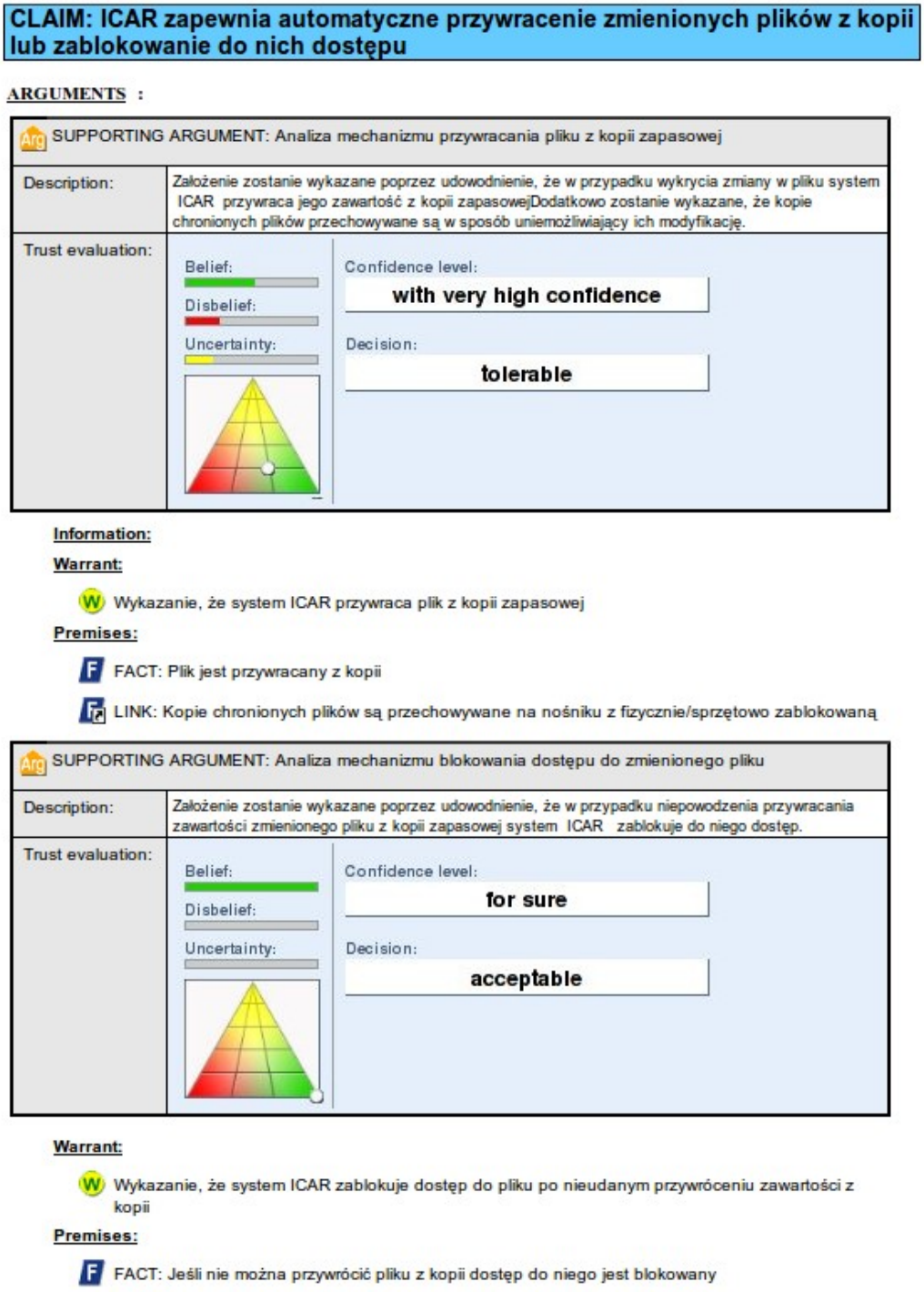


Rysunek 6.7. Ocena faktu w programie TCTEdior

Druga przesłanka, która jest założeniem brzmi następująco: *Jądro systemu operacyjnego Linux nie zawiera błędów umożliwiających wyłączenie modułów LSM*. W związku z tym, że z modułu LSM korzysta wiele mechanizmów bezpieczeństwa jest mało prawdopodobne, że jego implementacja będzie zawierała tak poważne błędy, które umożliwią wyłączenie mechanizmów bezpieczeństwa. Jeśli jednak błędy takie zostaną znalezione, to są szybko usuwane. Podobnie jak w poprzednim przypadku przesłankę oceniono jako: *with very high confidence acceptable.*

Twierdzenie 2: ICAR zapewnia automatyczne przywracanie zawartości zmienionych plików chronionych z kopii lub zablokowanie do nich dostępu

Dla tego twierdzenia zostały stworzone dwie oddzielne argumentacje, pokazane na rysunku 6.8. Pierwsza dotyczyła sytuacji typowej, gdy po wykryciu zmiany w pliku jego zawartość jest przywracana z kopii zapasowej. Jako przesłanki zostały przedstawione dwa fakty.

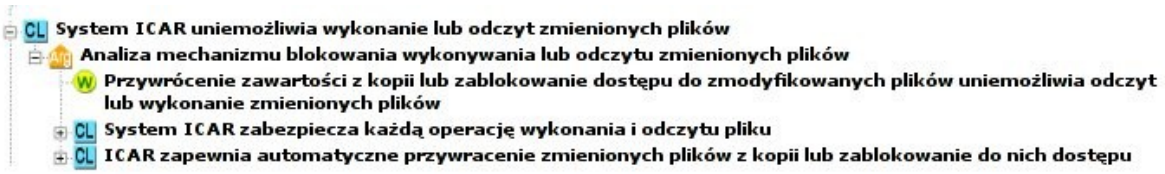


Rysunek 6.8. Dwie argumentacje do jednego twierdzenia

Pierwszy wynika bezpośrednio z funkcjonalności modułu ICAR i głosi, że dla każdego chronionego pliku, w którym wykryto zmianę, zostanie przeprowadzone automatyczne przywrócenie zawartości z kopii. Drugi fakt jest związany z zabezpieczeniem kopii chronionych plików. W związku z tym, że wszelkie kluczowe dane systemu bezpieczeństwa, w tym kopie plików, przechowywane są na nośnikach z fizycznie zablokowaną możliwością zapisu, fakt ten został oceniony jako *for sure acceptable*. Jednak przywrócenie zawartości

kopii plików, w pewnych przypadkach może nie zostać przeprowadzone z powodzeniem. Może mieć to związek na przykład z uszkodzeniem nośnika danych czy błędnie przeprowadzoną operacją tworzenia kopii. Dlatego też fakt *Plik jest przywracany z kopii* został oceniony jako *with very high confidence tolerable*. Nie oznacza to jednak, że jeśli nie można przywrócić kopii, system umożliwi odczyt lub uruchomienie pliku. W takim przypadku system automatycznie zablokuje dostęp do zmodyfikowanego pliku. Tej sytuacji dotyczy druga argumentacja *Analiza mechanizmu blokowania dostępu do zmienionego pliku*, na poparcie której został przedstawiony fakt, głoszący, że *Jeśli nie można przywrócić pliku z kopii, dostęp do niego jest blokowany*. Jest to związane z działaniem systemu ICAR i został oceniony jako *for sure acceptable*.

Dwa przedstawione powyżej twierdzenia są przesłankami do udowodnienia kolejnego twierdzenia. Zależność pomiędzy nimi została pokazana na zrzucie ekranu z programy TCTEditor na rysunku 6.9.



Rysunek 6.9. Zależności pomiędzy twierdzeniami

Twierdzenie 3: System ICAR uniemożliwia wykonanie lub odczyt zmienionych plików

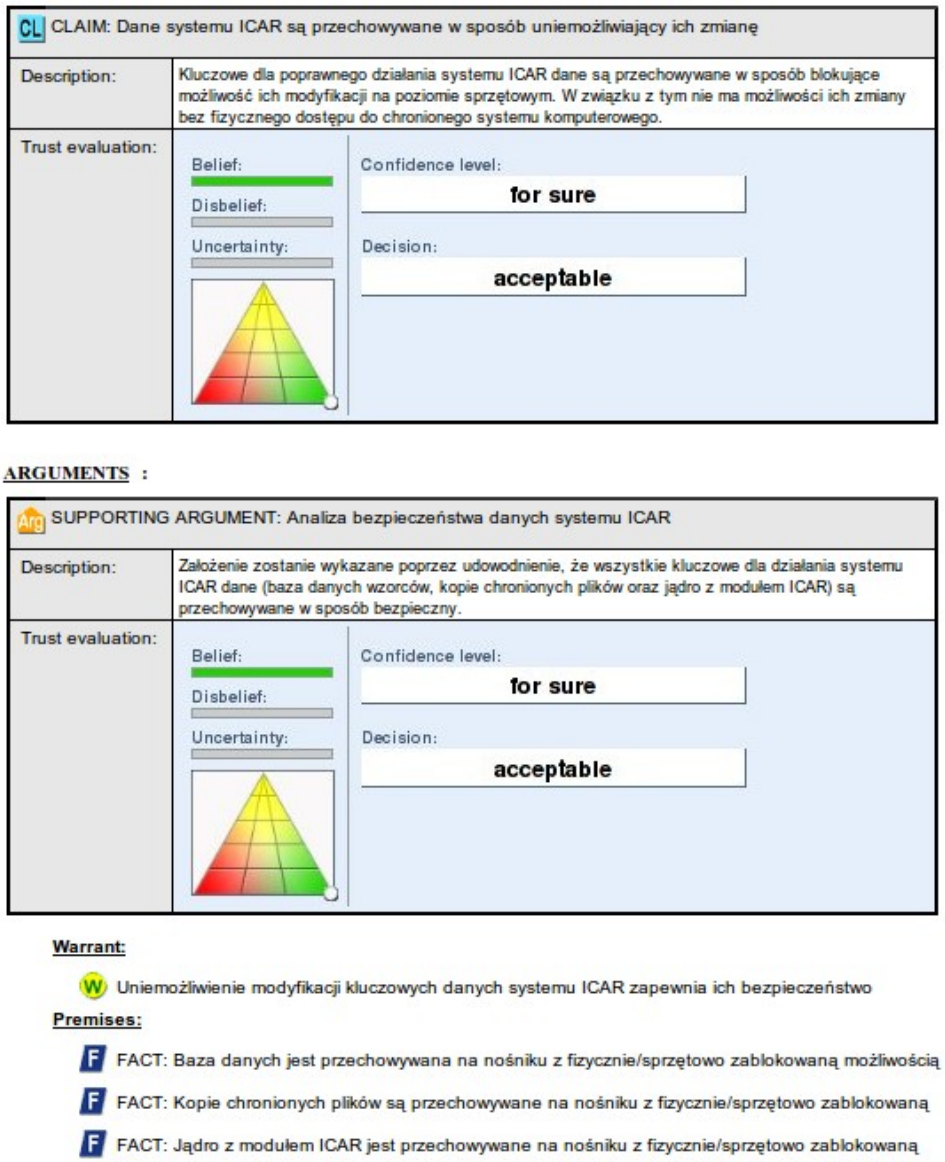
Jako argumentacja została przyjęta *Analiza mechanizmu blokowania wykonywania lub odczytu zmienionych plików*. Na udowodnienie uzasadnienia *Przywrócenie zawartości z kopii lub zablokowanie dostępu do zmodyfikowanych plików uniemożliwia odczyt lub wykonanie zmienionych plików*, zostały przedstawione dwa opisane wyżej fakty.

Twierdzenie 4: Nie można wyłączyć systemu ICAR

Udowodnienie tego twierdzenia zostało przeprowadzone poprzez analizę sposobu włączenia systemu ICAR do jądra systemu operacyjnego. Na poparcie zostały przedstawione cztery przesłanki. Fakt *ICAR jest zintegrowany z jądrem systemu operacyjnego* wynika z wybranego sposobu integracji systemu z jądrem, został oceniony jako *for sure acceptable*. Kolejny fakt, *Wykorzystanie szkieletu LSM blokuje możliwość wyłączenia mechanizmu bezpieczeństwa*, wynika bezpośrednio z założeń projektowych szkieletu LSM i jest zgodny z jego podstawową zasadą bezpieczeństwa, w związku z czym również został oceniony jako *for sure acceptable*. Ostatni fakt, *Jądro z modułem ICAR jest przechowywane na nośniku z fizycznie/sprzętowo zablokowaną możliwością zapisu*, wynika ze sposobu przechowywania danych w systemie ICAR i został oceniony jako *for sure acceptable*. Ostatnią przesłanką jest założenie głoszące, że *Jądro systemu operacyjnego Linux nie zawiera błędów umożliwiających wyłączenie modułów LSM*. Można z dużą dozą prawdopodobieństwa zaakceptować to założenie, jednak w związku z tym, że zawsze istnieje ryzyko wystąpienia błędów w każdym programie komputerowym, zostało ono ocenione jako *with very high confidence acceptable*.

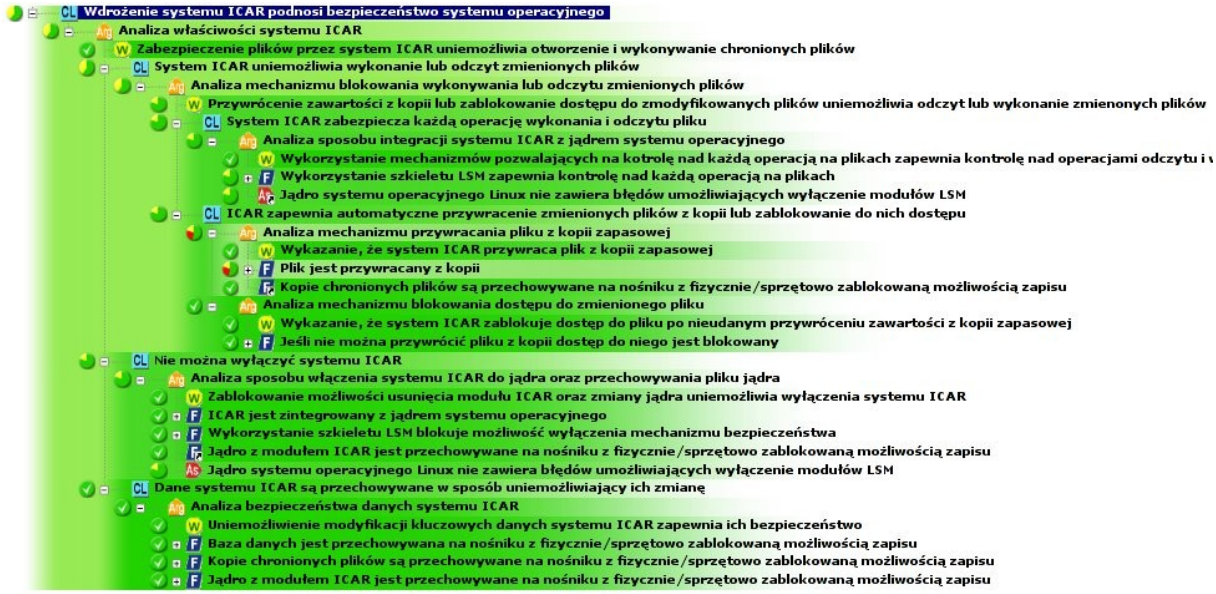
Twierdzenie 5: Dane systemu ICAR są przechowywane w sposób uniemożliwiający ich zmianę

Ostatnia przesłanka dotycząca głównego twierdzenia badania jest związana z bezpieczeństwem przechowywania danych w systemie ICAR. Analiza sposobu, w jaki kluczowe dane są przechowywane, pozwala na ocenę wiarygodności postawionego twierdzenia. Jako przesłanki zostały przedstawione trzy, podobne do siebie fakty: *Baza danych jest przechowywana na nośniku z fizycznie/sprzętowo zablokowaną możliwością zapisu*, *Kopie chronionych plików są przechowywane na nośniku z fizycznie/sprzętowo zablokowaną możliwością zapisu* oraz *Jądro z modułem ICAR jest przechowywane na nośniku z fizycznie/sprzętowo zablokowaną możliwością zapisu*. Wszystkie, zgodnie z założeniami projektowymi systemu bezpieczeństwa ICAR są prawdziwe i zostały ocenione jako *for sure acceptable*. Zrzut ekranu z programu *TCTEditor* dla tego twierdzenia został przedstawiony na rysunku 6.10.



Rysunek 6.10. Dowodzenie twierdzenia w programie TCTEditor

Program *TCTEditor,* po dokonaniu oceny poszczególnych przesłanek znajdujących się w liściach drzewa argumentacji, automatycznie generuje ocenę twierdzeń nadrzędnych, w tym twierdzenia głównego *Wdrożenie systemu ICAR podnosi bezpieczeństwo systemu operacyjnego*. Na rysunku 6.11 przedstawione zostało drzewo argumentacji z zaznaczonymi ocenami każdego węzła. Ikona oznacza ocenę *for sure acceptable*, ikona *with very high confidence acceptable*, ikona *with high confidence acceptable*, natomiast *with very high confidence tolerable*.



Rysunek 6.11. Ocena poszczególnych węzłów w programie TCTEditor

Jak widać na powyższym rysunku, wiarygodność głównego twierdzenia *Wdrożenie systemu ICAR podnosi bezpieczeństwo systemu operacyjnego* zostało ocenione jako z dużym prawdopodobieństwem akceptowalne. Największy negatywny wpływ na ocenę końcową miały czynniki niezależne od zaprojektowanego mechanizmu bezpieczeństwa lecz od wiarygodności bezpieczeństwa systemu operacyjnego Linux. Poprawa oceny założenia *Jądro systemu operacyjnego Linux nie zawiera błędów umożliwiających wyłączenie modułów LSM* oraz faktu *Wykorzystanie szkieletu LSM zapewnia kontrolę nad każdą operacją na plikach*, wpłynęłoby na zmianę oceny wiarygodności głównego twierdzenia na *akceptowalne z bardzo dużym prawdopodobieństwem*. Jednak z uwagi na możliwość wystąpienia błędów, nawet w tak kluczowych mechanizmach systemów operacyjnego nie zdecydowano się na zmianę ocen przesłanek.

Badanie wiarygodności systemu ICAR metodą *Trust Case* pozwoliło na usystematyzowaną ocenę stworzonego w ramach pracy systemu. Pozytywnie została zweryfikowana teza, że:

> **Skutecznym sposobem zapewniania bezpieczeństwa systemów komputerowych jest kontrola poprawności danych zapisanych w kluczowych plikach, z wykorzystaniem skrótów kryptograficznych.**

Jako główne czynniki osłabiające ocenę wiarygodności zostały zidentyfikowane czynniki zewnętrzne dotyczące bezpieczeństwa samego systemu operacyjnego Linux i modułów LSM, a nie wynikające bezpośrednio z opracowanego mechanizmu bezpieczeństwa.

6.3. Norma PN-ISO/IEC 15408

Projektowanie i wykonywanie produktów informatycznych o dużym znaczeniu powinno spełniać przyjęte normy międzynarodowe, między innymi dotyczące bezpieczeństwa. Norma PN-ISO/IEC 15408 została opracowana w celu określenia wspólnych kryteriów bezpieczeństwa (ang. *common criteria*) [52]. Został w niej zdefiniowany szereg zasad pozwalających na dokonanie specyfikacji oraz analizy wymogów bezpieczeństwa produktów informatycznych, a także ustalenia dla nich odpowiednich poziomów zaufania. Zakres normy PN-ISO/IEC 15408 pokrywa poufność, integralność oraz dostępność informacji [13].

Zaprezentowane w normie podejście polega na analizie systemu i wyborze spośród zdefiniowanych celów bezpieczeństwa (ang. *security objectives*) wymagań pokrywających cały zakres produktów IT. Norma może być używana zarówno przez użytkowników końcowych, programistów oraz ekspertów oceniających bezpieczeństwo produktów informatycznych. Na podstawie przeprowadzonej oceny użytkownicy mogę zdecydować, czy oferowany produkt informatyczny spełnia ich oczekiwania pod względem bezpieczeństwa. Tworzenie produktów informatycznych zgodnie z normą ISO 15408 pozwala na wczesne wyspecyfikowanie wymagań bezpieczeństwa i wpisanie ich w proces wytwarzania od samego początku, w odróżnieniu od podejścia ad-hoc do problemów bezpieczeństwa. Dla ekspertów oceniających bezpieczeństwo norma dostarcza szablon pozwalający na dokonanie usystematyzowanej i spójnej oceny [28].

Norma ta powstała pod koniec lat dziewięćdziesiątych jako połączenie wcześniejszych standardów bezpieczeństwa, takich jak:

- *Trusted Computer System Evaluation Criteria* (*TCSEC*) - standard opracowany przez Departament Obrony USA, określający podstawowe wymagania dla oceny skuteczności systemów bezpieczeństwa. TCSEC został opracowany na potrzeby służb wojskowych USA, głównie w celu zagwarantowania poufności informacji. Standard ten był powszechnie znany pod nazwą *Pomarańczowa Księga* (ang. *Orange Book*).
- *Information Technology Security Evaluation Criteria* (*ITSEC*) - europejski standard opublikowany w 1990 roku. Zgodnie z nim produkt jest oceniany w kontekście wymogów operacyjnych i zagrożeń jakie mogą wystąpić.
- *Canadian Trusted Computer Evaluation Criteria* (*CTCPEC*) - standard opracowany na początku lat dziewięćdziesiątych w Kanadzie na bazie TCSEC i ITSEC. Oceniane są m.in. poufność, integralność, dostępność i odpowiedzialność.

Z uwagi na swoje cele oraz genezę norma nosi nazwę *Common Criteria for Information Security Evaluation*, w skrócie *CC* (*Wspólne kryteria do oceny zabezpieczeń informatycznych*). Norma składa się z trzech niezależnych części:

- PN-ISO/IEC 15408-1 (*CC Part 1*) - definiuje główne założenia i cele oceny bezpieczeństwa produktów informatycznych oraz prezentuje ogólny model oceny.
- PN-ISO/IEC 15408-2 (*CC Part 2*) - określa zbiór komponentów funkcjonalnych (ang. *functional components*) pozwalających na zdefiniowanie funkcjonalnych wymagań bezpieczeństwa (ang. *Security Functional Requirements*, *SFR*).

- PN-ISO/IEC 15408-3 (*CC Part 3*) - określa zbiór komponentów uzasadniających zaufanie (ang. *assurance components*), które pozwalają na zdefiniowanie wymagań uzasadniających zaufanie do zabezpieczeń (ang. *Security Assurance Requirements*, *SAR*) oraz prezentuje skalę oceny wiarygodności EAL (ang. *Evaluation Assurance Levels*).

Zbiory komponentów funkcjonalnych oraz komponentów uzasadniających zaufanie są przedstawione w formie ustrukturalizowanych katalogów podzielonych na klasy (ang. *classes*), rodziny (ang. *families*) oraz komponenty (ang. *components*) [12].

W nomenklaturze normy produkt informatyczny podlegający ocenie nosi nazwę *przedmiot oceny* (ang. *Target of Evaluation*, *TOE*). Przedmiotem oceny mogą być zarówno aplikacje, systemy operacyjne, ale również sieci komputerowe czy inteligentne urządzenia elektroniczne.

Zgodnie z metodologią *Common Criteria* funkcjonalne wymagania bezpieczeństwa i wymagania uzasadniające bezpieczeństwo produktów IT mogą być przedstawione w dwóch formach: jako *Profil Ochrony* (ang. *Protection Profile*, *PP*) oraz *Zadania Zabezpieczeń* (ang. *Security Target*, *ST*).

Profil Ochrony pozwala na tworzenie zbiorów wymogów bezpieczeństwa dla całych kategorii produktów informatycznych, takich jak np. programy antywirusowe (*Protection Profile Anti-Virus Applications for Workstations in Basic Robustness Environments*) [129], systemy baz danych (*Protection Profile Database Management Systems for Basic Robustness Environments*) [130], czy systemy operacyjne (*Operating System Protection Profile*) [126]. Opracowany dokument jest z założenia wielokrotnego użycia i może zostać wykorzystany do specyfikacji i identyfikacji wymagań bezpieczeństwa dla konkretnych produktów IT.

Dokument *Zadania Zabezpieczeń* (*ST*) jest natomiast opracowywany dla konkretnego produktu IT. Wymagania bezpieczeństwa mogą zostać wyspecyfikowane na podstawie profilu ochrony stworzonego dla odpowiedniej grupy produktów. Dokument *ST* zawiera specyfikację przedmiotu oceny (*TOE*), wraz z jego wymogami bezpieczeństwa i ich uzasadnieniem. Dokument ten powinien zawierać opis implementacji wymagań bezpieczeństwa w postaci funkcji zabezpieczających TOE (ang. *TOE Security Functions*, *TSF*), do których zaliczane są wszystkie mechanizmy powiązane bezpośrednio lub pośrednio z bezpieczeństwem.

Mechanizm ICAR jest rozszerzeniem systemu operacyjnego i nie może być wg normy traktowany jako oddzielny produkt informatyczny. Dlatego przeprowadzono analizę korzyści wynikających z jego wdrożenia dla systemu operacyjnego pod kątem wymagań funkcjonalnych zdefiniowanych w normie PN-ISO/IEC 15408.

Jako dokument referencyjny został wybrany Profil Ochrony Systemów Operacyjnych (*Operating System Protection Profile, OSPP*) [126] opracowany przez niemiecki Federalny Urząd Bezpieczeństwa Informacji (niem. *Bundesamt für Sicherheit in der Informationstechnik*). Obecnie dostępny jest w wersji 2.0, która została opublikowana w czerwcu 2010 roku. Profil ochrony dotyczy systemów operacyjnych ogólnego przeznaczenia (ang. *general-purpose operating systems*) działających w dobrze zarządzanym środowisku. Przeznaczony jest głównie do przeprowadzania oceny systemów serwerowych. Możliwe jest również wykorzystanie do oceny stacji roboczych działających w przedsiębiorstwach lub organizacjach rządowych.

Można więc przyjąć, że wymagania bezpieczeństwa systemów operacyjnych zawarte w OSPP są wystarczające. W rzeczywistości większość współczesnych systemów operacyjnych, a zwłaszcza tych przeznaczonych dla zastosowań serwerowych, spełnia te wymagania, wliczając w to dystrybucje systemu Linux.

Zastosowanie systemu bezpieczeństwa ICAR pozwala na większą ochronę systemów operacyjnych niż jest to wymagane przez dokument OSPP. Oznacza to, że system operacyjny Linux z włączonym system bezpieczeństwa ICAR nie tylko spełnia wymagania bezpieczeństwa według profilu ochrony *OSPP*, ale dodatkowo zwiększa zakres ochrony. Norma PN-ISO/IEC 15408 pozwala na zdefiniowanie dodatkowych wymagań bezpieczeństwa. Z przeprowadzonej analizy możliwości systemu bezpieczeństwa ICAR wynika, że jest możliwe zdefiniowanie dwóch dodatkowych wymagań bezpieczeństwa, niezawartych w Profilu Ochrony Systemów Operacyjnych. Spełnienie tych wymagań może przyczynić się do większego bezpieczeństwa systemów operacyjnych. Pierwsze wymaganie, określane w *Common Criteria* jako FDP_SDI (ang. *User data protection, Stored data inegrity*) dotyczy ochrony integralności przechowywanych danych. Natomiast drugie, FPT_RCV (ang. *Protection of the TSF, Trusted recovery*) dotyczy zabezpieczania i przywracania danych systemów ochrony.

FDP_SDI

Wymaganie funkcjonalne FDP_SDI należy do klasy FDP (*User Protection Data*), która odpowiada za specyfikację wymagań oraz politykę bezpieczeństwa dla TOE dotyczących ochrony danych użytkowych. W profilu ochrony OSPP znajdują się już komponenty należące do tej klasy i są to m.in. rodziny:

- *Access control policy* (*FDP_ACC*) – opisujące politykę kontroli dostępu do zasobów,
- *Access control functions* (*FDP_ACF*) – opisujące wymagania dla funkcji implementujących politykę dostępu do zasobów,
- *Information flow control policy* (*FDP_IFC*) – opisujące politykę kontroli przepływu informacji,
- *Information flow control functions* (*FDP_IFF*) – opisujące wymagania dla funkcji implementujących politykę kontroli przepływu informacji,
- *Import from outside TSF control* (*FDP_ITC*) – opisujące mechanizm pobierania danych spoza zakresu ochrony TSF.

Zastosowanie systemu ICAR pozwala na rozszerzenie ochrony o wymaganie dotyczące integralności przechowywanych danych, oznaczone skrótem *FDP_SDI* (ang. *Stored Data Integrity*). Ochrona jaką ICAR obejmuje wybrane pliki nie może być zakwalifikowana do rodziny *FDP_ACF* gdyż dostęp do modyfikacji przechowywanych danych jest zblokowany dla wszystkich użytkowników, więc nie ma miejsca na „kontrolę dostępu”. Możliwość modyfikacji danych posiadają wyłącznie użytkownicy z uprawnieniami administracyjnymi, posiadający fizyczny dostęp do komputera.

Dodatkowo komponent oznaczony jako *FDP_SDI.2* pozwala nie tylko na kontrolę integralności danych, ale również na podjecie odpowiedniej akcji w przypadku wykrycia modyfikacji. W przypadku wykorzystania systemu *ICAR* akcją taką będzie przywrócenie danych z niemodyfikowalnego nośnika.

FPT_RCV

Drugie ze zidentyfikowanych wymagań, oznaczone skrótem FPT_RCV, należy do klasy FPT (*Protection of the TSF*), która zawiera rodziny wymagań funkcjonalnych związanych z ochroną mechanizmów bezpieczeństwa *TSF* oraz danych potrzebnych do prawidłowego działania zabezpieczeń. W profilu ochrony systemów operacyjnych (*OSPP*) nie zostały uwzględnione żadne komponenty z tej klasy. Rodzina komponentów *FTP_RCV* (*Trusted recovery*) specyfikuje wymagania pozwalające na zabezpieczenie *TOE* przed niepowołanymi modyfikacjami mechanizmów ochrony.

Opisane w rozdziale 4.5 metody wykorzystania systemu ICAR do ochrony systemu operacyjnego zakładają przechowywanie wszelkich danych systemu zabezpieczeń, w tym jądra systemu operacyjnego, na nośnikach niemodyfikowalnych. W ten sposób zablokowana jest, na poziomie sprzętowym, możliwość wyłączenia systemu ICAR, a także zmiana danych potrzebnych do jego prawidłowego działania. Następnie za pomocą systemu ICAR zabezpieczone mogę zostać pozostałe składniki systemu operacyjnego odpowiedzialne za bezpieczeństwo. Mechanizm automatycznego przywracania zmienionych danych przez system ICAR spełnia wymagania opisane w komponencie *FTP_RCV.2*.

Wykorzystanie mechanizmu ICAR w zabezpieczaniu systemu operacyjnego pozwala na rozszerzenie bezpieczeństwa ponad to, które jest wymagane przez *Profil Ochrony Systemów Operacyjnych OSPP*. Ochrona integralności przechowywanych danych oraz mechanizmów ochrony pozwala na bardziej niezawodne działanie systemów operacyjnych i z tych względów zwiększa niezawodność ich działania. W przyszłości należy rozważyć zasadność dołączenia przedstawionych komponentów do profilu ochrony *OSPP*.

6.4. Dalsze możliwości rozwoju mechanizmu bezpieczeństwa ICAR

Zaprezentowany w niniejszej rozprawie system ICAR zapewnia bezpieczeństwo systemu operacyjnego poprzez kontrolę zmian w chronionych plikach. Może on zostać rozszerzony na kilka sposobów, począwszy od zastosowania zaawansowanych nośników danych z blokadą zapisu (takich jak np. specjalne złącza IDE/SATA dla dysków twardych), poprzez rozproszone przechowywanie danych w sieci lokalnej, a skończywszy na zapisie systemu operacyjnego na stałe w pamięci komputera, czy też wykorzystaniu technik wirtualizacji. Dla autora rozprawy doktorskiej szczególnie interesujące, z naukowego punktu widzenia, wydaje się ostatnie rozwiązanie. W niniejszym rozdziale zostaną przedstawione możliwości, zalety i wady wykorzystania wirtualizacji w dalszym rozwoju systemu ICAR.

Wirtualizacja jest obecnie uważana za jeden z najważniejszych kierunków rozwoju mechanizmów zabezpieczeń systemów operacyjnych. Umożliwia ona jednoczesne uruchomienie wielu systemów operacyjnych na jednym komputerze. W wielu ośrodkach badawczych prowadzone są prace, które mają na celu wykorzystanie korzyści płynących z zastosowania technik wirtualizacji do podnoszenia bezpieczeństwa systemów komputerowych. Dostępne obecnie mechanizmy bezpieczeństwa działające w oparciu o zastosowanie maszyn wirtualnych zostały opisane w rozdziale 2.3.3.

Pomimo tego, że technika wirtualizacji zaczęła być powszechnie wykorzystywana dopiero na początku XXI wieku, sama koncepcja stworzenia maszyny wirtualnej nie jest nowa. Pierwsze badania były prowadzone przez firmę IBM w latach sześćdziesiątych XX wieku. Ich celem

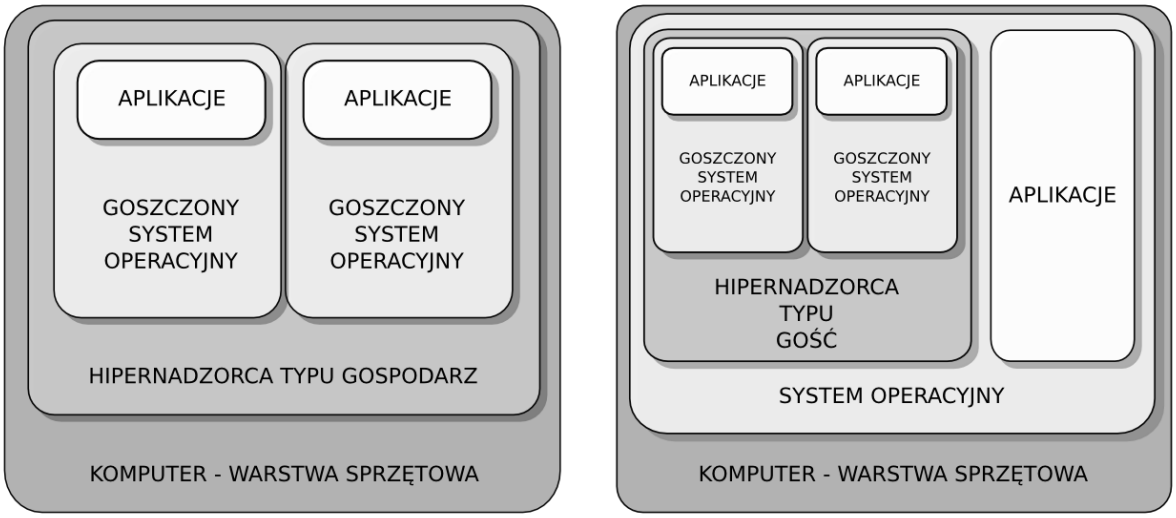
było umożliwienie wykorzystywania komputerów typu mainframe do wykonywania wielu zadań jednocześnie. Pierwszym systemem, który zapewniał całkowite odseparowanie wirtualnych maszyn był wykonany w 1972 roku system operacyjny VM/370 przeznaczony do pracy na komputerach IBM System/370. System ten oferował praktycznie wszystkie funkcje dostępne we współczesnych programach maszyn wirtualnych [3]. W 1974 roku powstały dwa kluczowe artykuły, napisane przez naukowców z uniwersytetu kalifornijskiego UCLA, dające naukowe podstawy do rozwoju wirtualizacji [43, 89]. Z uwagi na wysokie koszty sprzętu komputerowego o wydajności pozwalającej na efektywne obsługiwanie wirtualizacji, do końca XX wieku była ona praktycznie używana tylko w specjalistycznych zastosowaniach. W związku ze znacznym zwiększeniem mocy obliczeniowej komputerów, obecnie istnieją możliwości wykorzystywania maszyn wirtualnych.

W ostatnich latach prowadzone są liczne badania naukowe nad możliwością wykorzystania wirtualizacji do podnoszenia bezpieczeństwa systemów komputerowych. Koncentrują się one m.in. na włączaniu systemów ochrony na poziomie tak zwanego monitora maszyny wirtualnej (ang. *Virtual Machine Monitor, VMM*). Na rysunku 6.13 została zilustrowana zasada działania hipernadzorcy przy uruchomionych dwóch maszynach wirtualnych.

Rysunek 6.13. Architektura maszyny wirtualnej

Wyróżniane są dwa typy hipernadzorców: gospodarz (ang. *native virtual machine monitor*) i gość (ang. *hosted virtual machine monitor*). Hipernadzorca typu gospodarz jest prostym systemem operacyjnym uruchamianym bezpośrednio na komputerze. Przykładami tego typu produktów jest m.in. system Xen, VMware ESX Server, czy Microsoft Hyper-V Server. Hipernadzorca typu gość jest natomiast programem komputerowym uruchamianym wewnątrz zainstalowanego już systemu operacyjnego. Do tej grupy programów należą m.in. VMware Workstation, VirtualBox, QEMU, czy Microsoft VirtualPC. Różnica w architekturze obu typów hipernadzorców została przedstawiona na rysunku 6.14.

Modyfikacja kodu źródłowego monitora maszyny wirtualnej daje autorom systemów zabezpieczeń bardzo duże możliwości. Z punktu widzenia ochrony systemów operacyjnych pracujących w maszynie wirtualnej jest to równoznaczne z modyfikacją sprzętu komputerowego, co daje duże możliwości budowy systemów zabezpieczeń.



Rysunek 6.14. Porównanie architektury hipernadzorców typu gospodarz i gość

Zainstalowane na poziomie hipernadzorcy programy działają całkowicie niezależnie od procesów w maszynie wirtualnej. Takie odseparowanie mechanizmu ochrony od systemu operacyjnego daje szereg korzyści. Przede wszystkim z poziomu systemu operacyjnego nie ma możliwości dostępu do warstwy hipernadzorcy, a przez to nawet po przejęciu kontroli nad maszyną wirtualną intruz nie będzie mógł wyeliminować systemu ochrony. Kolejną zaletą wirtualizacji jest uniezależnienie implementacji mechanizmów zabezpieczeń od typów systemów operacyjnych. Dzięki temu błędy w kodzie źródłowym systemu operacyjnego nie wpływają na skuteczność systemu bezpieczeństwa. Maszyna wirtualna może być również zatrzymana w ściśle określonym stanie, a nawet stan taki może być dowolnie modyfikowany, co w przypadku rzeczywistych komputerów jest praktycznie niemożliwe.

Wykorzystanie wirtualizacji jest obarczone pewnymi ograniczeniami. Przede wszystkim należy brać pod uwagę wydajność takiego rozwiązania. Konieczność emulacji całego środowiska sprzętowego przez program komputerowy w sposób oczywisty przekłada się na spadek prędkości działania systemów operacyjnych. Jednak przy zastosowaniu nowoczesnych procesorów, z których część wspiera wirtualizację w sposób sprzętowy, zmniejszenie prędkości może być niezauważalne. Innym wyzwaniem stojącym przed twórcami systemów bezpieczeństwa działających na poziomie monitora maszyny wirtualnej jest konieczność zapewnienia odpowiedniej interpretacji informacji otrzymywanych z systemu operacyjnego. Przykładowo, po wysłaniu przez jądro żądania dostępu do pliku, maszyna wirtualna otrzymuje tylko informację o numerze bloku na dysku, a nie o numerze i-węzła czy nazwie pliku. Chcąc zapewnić skuteczną kontrolę danych na poziomie systemu plików, konieczna jest translacja odwołań sprzętowych na poziom organizacji plików [19].

Na poziom hipernadzorcy mogą zostać przeniesione systemy detekcji włamań (*IDS*), w tym także systemy ochrony integralności plików. Systemy detekcji włamań można podzielić na działające bezpośrednio na komputerze *HIDS* (ang. *Host Intrusion Detection System*) oraz działające w sieci lokalnej *NIDS* (ang. *Network Intrusion Detection System*). Wadą systemów *HIDS* jest słabe odseparowanie od chronionego systemu, gdyż w przypadku udanego włamania intruz ma możliwość wyłączenia zabezpieczenia. Natomiast systemy *NIDS* działające w sieci lokalnej mają ograniczony dostęp do zasobów chronionego komputera, co powoduje, że nie mogą w pełni monitorować stanu systemu operacyjnego. Rozwiązaniem

przedstawionych problemów jest wirtualizacja, czyli instalacja systemu *IDS* w monitorze maszyny wirtualnej. Dzięki takiemu rozwiązaniu system *IDS* jest skutecznie odizolowany od chronionego systemu operacyjnego i dlatego przejęcie kontroli przez intruza nad maszyną wirtualną nie prowadzi do przejęcia kontroli nad hipernadzorcą. Ponadto system zabezpieczeń ma dostęp do wszelkich operacji wykonywanych pomiędzy jądrem systemu operacyjnego a warstwą sprzętu, dzięki czemu ma możliwość efektywnej ochrony zasobów [40].

Dotychczas zostało stworzonych kilka skutecznych systemów ochrony danych wykorzystujących technologię wirtualizacji. Jednym z bardziej zaawansowanych jest system *ReVirt*. System powstał w ramach prac grupy badawczej *CoVirt*, działającej na uniwersytecie Michigan, która zajmuje się wykorzystywaniem hipernadzorców do zapewniania bezpieczeństwa systemów komputerowych System *ReVirt* umożliwia ochronę logów systemowych, czyli plików z zapisaną historią wykonywanych operacji oraz informacji pochodzących od działającego systemu operacyjnego. Dane z logów systemowych są niezwykle istotne przy analizie wrogich działań przeprowadzanych przez intruza i są w pierwszej kolejności niszczone po przeprowadzeniu udanego włamania do systemu operacyjnego. Przy zastosowaniu systemu *ReVirt* program zapisuje logi systemowe na poziomie monitora maszyny wirtualnej. Dzięki temu intruz nie jest w stanie usunąć zgromadzonych danych oraz, co równie ważne, nie może wyłączyć systemu *ReVirt*. Takie działanie tego systemu daje administratorom możliwość analizy informacji o wszelkich działaniach intruza podczas jego aktywności [29].

Innym ciekawym rozwiązaniem wykorzystującym wirtualizację jest system plików *SVFS* (ang. *Secure Virtual File System*), który pozwala na zabezpieczenie kluczowych plików przed niepowołaną modyfikacją. Do zapewnienia ochrony danych wykorzystywane są dwie maszyny wirtualne, jedna przeznaczona jest do przechowywania chronionych plików, a na drugiej zainstalowany jest system operacyjny. Na poziomie monitora maszyny wirtualnej zaimplementowany został mechanizm umożliwiający transparentne korzystanie z chronionych plików przez system operacyjny [123].

Z innych systemów ochrony danych z wykorzystaniem wirtualizacji, należy wspomnieć o przeznaczonym do ochrony danych w systemach rozproszonych programie *HyperSpector* [71] czy systemie *Livewire* [40].

Główny problem w wykorzystywaniu mechanizmu ochrony integralności ICAR jest związany ze zmianą zawartości plików chronionych, niezbędną w niektórych okolicznościach. W takiej sytuacji konieczne jest ponowne obliczenie wartości skrótów kryptograficznych oraz zapis zmodyfikowanych danych na nośnikach niemodyfikowalnych. Prowadzi to do konieczności wykonania nowych płyt CD-ROM lub odblokowania i zapisu danych w pamięciach USB.

Problem ten można rozwiązać wykorzystując wirtualizację i hipernadzorcę, który umożliwia dokonywanie zmian poniżej warstwy jądra systemu operacyjnego, ponieważ system operacyjny traktuje hipernadzorcę jako sprzęt komputerowy. Z punktu widzenia systemu operacyjnego dane są dostępne w trybie tylko do odczytu, jednak z poziomu monitora maszyny wirtualnej mogą być bez problemu modyfikowane. Takie rozwiązanie zwiększa elastyczność modyfikacji krytycznych danych.

Analiza możliwości implementacji systemu ICAR z wykorzystaniem maszyn wirtualnych, pozwoliła na opracowanie dwóch modeli integracji. W pierwszym przypadku rozszerzenie polega na modyfikacji obecnej koncepcji działania, w której moduł bezpieczeństwa ICAR

zostaje dołączony do jądra systemu operacyjnego. Jedynie dane potrzebne do działania systemu bezpieczeństwa, takie jak baza danych skrótów kryptograficznych czy kopie chronionych plików, przechowywane są na zewnątrz systemu operacyjnego. Dostęp do tych danych jest możliwy poprzez mechanizmy zaimplementowane w warstwie hipernadzorcy. Do przechowywania i zarządzania kluczowymi danymi dla działania systemu zabezpieczeń można zastosować dodatkową maszynę wirtualną, do której dostęp jest kontrolowany przez hipernadzorcę. Takie rozwiązanie wiąże się ze stworzeniem stosunkowo prostego rozszerzenia programu monitora maszyny wirtualnej.

Bardziej zaawansowane rozwiązanie polega na całkowitym przeniesienia systemu ICAR na poziom hipernadzorcy. W takim przypadku jądro systemu operacyjnego nie ulega modyfikacji, natomiast moduł ochrony znajduje się wewnątrz monitora maszyny wirtualnej. Zaimplementowany w ten sposób mechanizm jest traktowany przez system operacyjny jako rozszerzenie warstwy sprzętowej.

Zaletą takiego rozwiązania jest całkowite uniezależnienie systemu ochrony integralności plików od typu systemu operacyjnego. Pozwala to na wykorzystanie systemu ICAR do ochrony danych w dowolnych systemach operacyjnych, takich jak Linux, Windows, MacOS i innych. Implementacja takiego rozwiązania jest jednak bardziej skomplikowana, gdyż jak wcześniej wspomniano, monitor maszyny wirtualnej działa na poziomie logiki sprzętowej. Oznacza to, że wszelkie informacje dotyczące danych pochodzące z systemu operacyjnego gościa otrzymywane są w formie żądań konkretnych bloków czy sektorów dyskowych. Chcąc chronić pliki, konieczne jest przetłumaczenie otrzymanych informacji z powrotem na logikę systemu plików. Niezbędne jest określenie numerów i-węzłów, nazw plików i ich lokalizacji w drzewie katalogów. Jest to zadanie złożone, które może zmniejszać wydajność systemu operacyjnego, ale z uwagi na swoją uniwersalność i możliwość wykorzystania do ochrony dowolnych typów systemów operacyjnych ma duże perspektywy rozwoju i powszechnego zastosowania. W tym kierunku będą prowadzone dalsze prace badawcze autora.

Rozdział 7.
Podsumowanie

Niniejsza praca stanowi jedną z wielu prób rozwiązania niezwykle istotnego problemu współczesnej informatyki, jakim jest zapewnianie bezpieczeństwa systemów komputerowych. Zagadnienie to jest szczególnie istotne z uwagi na zwiększający się poziom zagrożeń atakami, zwłaszcza, że prowadzą one między innymi do poważnych strat materialnych, kradzieży czy utraty zgromadzonych danych.

W pracy skoncentrowano się na zapewnianiu bezpieczeństwa systemu plików, który stanowi główny cel ataków intruzów. Zapewnienie stuprocentowego bezpieczeństwa systemu komputerowego wymaga również opracowania mechanizmów do ochrony procesów i danych umieszczanych w pamięci operacyjnej komputera. Zagadnienia zabezpieczania systemów komputerowych jest przedmiotem licznych prac badawczych w ośrodkach naukowych i rządowych na całym świecie

W proponowanym przez autora mechanizmie ochrony kluczowych plików uwzględniono charakter wybranej przez administratora systemu polityki bezpieczeństwa, zastosowano niemodyfikowalne nośniki do zapisu istotnych danych oraz wykorzystano mechanizm automatycznego przywracania poprawnej treści plików. Wydaje się, że stworzono warunki do niemal całkowitego zabezpieczenia plików, które zostały przeznaczone do ochrony. Można przypuszczać, że proponowany mechanizm zapewni całkowite bezpieczeństwo plików przeznaczonych do ochrony, jednakże dopiero jego powszechne wykorzystywanie w praktyce może zweryfikować ten wniosek w przyszłości.

W niniejszej pracy zaproponowano mechanizm bezpieczeństwa, który ingeruje w sposób przydzielania procesom dostępu do plików, dzięki czemu istnieje możliwość weryfikacji, czy otwierane pliki są zgodne z ich pierwotną zawartością. Zastosowanie niemodyfikowalnych nośników danych do zapisu bazy danych wzorców kryptograficznych uniemożliwia ich modyfikację, a dołączenie mechanizmu zabezpieczeń jako modułu jądra systemu operacyjnego z wykorzystaniem szkieletu LSM blokuje możliwość jego wyłączenia.

Do oryginalnych osiągnięć pracy należy zaliczyć:

1. Opracowanie modelu zapewnienia bezpieczeństwa systemu operacyjnego, w którym wykorzystano skróty kryptograficzne wybranych w ramach polityki bezpieczeństwa plików, mechanizm porównywania i buforowanie nazw plików uprzednio zweryfikowanych oraz mechanizm przywracania poprawnej zawartości plików.
2. Wykonanie systemu ICAR (*Integrity Checking And Restoring*), w którym zaimplementowano opracowany model zapewniania bezpieczeństwa. Jak wykazano w wykonanych testach wydajność systemu jest akceptowalna, a zależność czasu dostępu do pliku od jego rozmiaru pozostaje liniowa.

3. Ocenę wiarygodności zaproponowanego mechanizmu przeprowadzoną metodą Trust Case. Za pomocą tej metody udowodniono twierdzenie, że wdrożenie systemu ICAR podnosi bezpieczeństwo systemu operacyjnego.

Wśród znanych systemów ochrony integralności opracowany mechanizm wyróżnia się dwoma innowacyjnymi rozwiązaniami:

1. Umożliwia automatyczne przywracanie zawartości plików zmodyfikowanych w sposób nieuprawniony z kopii zapasowych.
2. Wykorzystuje niemodyfikowalne nośniki danych do przechowywania kluczowych danych dla prawidłowego działania systemu zabezpieczającego.

Dotychczasowe rozwiązania posiadały nieskuteczne mechanizmy zabezpieczania samego systemu ochrony przed jego wyeliminowaniem. Stworzony mechanizm został zintegrowany z jądrem systemu operacyjnego z wykorzystaniem szkieletu LSM, co uniemożliwia jego wyłączenie podczas pracy systemu komputerowego. Przechowywanie jądra systemu operacyjnego wraz z mechanizmem ICAR oraz bazą danych wzorców kryptograficznych i kopii chronionych plików na nośnikach danych z fizyczną blokadą zapisu całkowicie eliminuje możliwość modyfikacji danych przez intruza.

Działanie systemu zabezpieczeń jest przezroczyste dla użytkowników systemu operacyjnego, co oznacza, że nie muszą wiedzieć, że kontrolowana jest poprawność zawartości każdego pliku przedzielanego do procesu. Zastosowanie buforów podręcznych zwiększa wydajność mechanizmu zabezpieczeń, ponieważ eliminuje konieczność każdorazowego obliczania funkcji skrótu kryptograficznego. Przeprowadzone testy pozwalają stwierdzić, że działanie mechanizmu ICAR nie będzie skutkowało zauważalnym spadkiem wydajności chronionego systemu operacyjnego.

Wykonany system jest skuteczny, efektywny i jednocześnie tani oraz prosty do wdrożenia. Kod źródłowy systemu został udostępniony na licencji GPL na stronie projektu *cdlinux.pl*, pod adresem *http://cdlinux.pl/icar*. Zaprojektowany mechanizm został zaimplementowany w systemie Linux, ale może zostać również wdrożony w innych systemach operacyjnych.

Dzięki swoim zaletom zaprojektowany mechanizm bezpieczeństwa nadaje się do powszechnego zastosowania, nawet w systemach produkcyjnych o dużym znaczeniu strategicznym.

W dalszych pracach badawczych planowane jest zwiększenie elastyczności i skuteczności mechanizmu ICAR poprzez zaimplementowanie go na poziomie monitora maszyny wirtualnej, czyli poniżej poziomu jądra systemu operacyjnego. Kierunek badań wynika z faktu, że należy przypuszczać, iż w niedalekiej przyszłości dojdzie zmiany architektury systemów komputerowych. System operacyjny i mechanizmy, które go zabezpieczają przed atakami będą całkowicie odseparowane od aplikacji i danych użytkowników. Takie rozwiązanie pozwoli na powszechną realizację słusznej idei *Trusted Computing Base*.

Niniejsza praca stanowi przyczynek do opracowania mechanizmów, które zapewnią całkowite bezpieczeństwo komputerów, a tym samym wyeliminują straty, jakie powstają w wyniku ataków i upadków systemów informatycznych.

Bibliografia

[1] Amor-Iglesias J.J., Gonzalez-Barahona J.M., Robles-Martinez G., Herraiz-Tabernero I.: Measuring libre software using debian 3.1 (sarge) as a case study: Preliminary results, CEPIS promotes, 2005

[2] Anderson R.J.: Security Engineering: A guide to building dependable distributed systems, Wiley Publishing, 2001

[3] Arce I.: Ghost in the virtual machine, IEEE Security & Privacy, vol. 5, no. 4, pp. 68—71, 2007

[4] Axelsson S.: The Base-Rate Fallacy and its Implications for the Difficulty of Intrusion Detection, ACM Transactions on Information and System Security (TISSEC), vol. 3, no. 3, 2000

[5] Bace R., Mell P.: Intrusion Detection Systems, National Institute of Standards and Technology, SP 800-31, 2001

[6] Bach M., Budowa systemu operacyjnego UNIX, WNT, Warszawa, 1995

[7] Barber R.: Hackers Profiled — Who Are They and What Are Their Motivations?, Computer Fraud & Security, vol. 2001, no. 1, pp. 14—17, Elsevier, 2001

[8] Barford P., Yegneswaran V.: An Inside Look at Botnets, Special Workshop on Malware Detection, Advances in Information Security, 2006

[9] Basili R.V., Caldiera G., Rombach H.D.: The Goal Question Metric Approach, Encyclopedia of Software Engineering, vol. 1, 1994

[10] Bauer, M.: Paranoid penguin: an introduction to Novell AppArmor, Linux Journal, no. 148, 2006

[11] Bessani A.N., Sousa P., Correia M., Neves N.F., Verissimo P.: The CRUTIAL way of critical infrastructure protection, IEEE Security and Privacy, vol. 6. no. 6, pp. 44–51, 2008

[12] Białas A.: Informatyczne produkty sprzętowe, oprogramowanie oraz systemy o zadanym poziomie uzasadnionego zaufania, Materiały konferencyjne EMTECH'2009 – Zasilanie, informatyka techniczna i automatyka w przemyśle wydobywczym, 2009

[13] Bishop M.: The art and science of computer security, Addison-Wesley Longman Publishing Co., 2002

[14] Borchardt M., Maziero C., Jamhour E.: An Architecture for On-the-fly File Integrity Checking, Lecture notes in computer science, pp. 117—126, Springer, 2003

[15] Bovet D.P., Cesati M.: Understanding the Linux Kernel, O'Reilly, 2005

[16] Cavusoglu H., Mishra B., Raghunathan, S.: The value of intrusion detection systems in information technology security architecture, Information Systems Research, vol. 16, no. 1, pp. 28—46, 2005

[17] Chapman D.B.: Network (In) Security Through IP Packet Filtering, Proceedings of the Third UNIX Security Symposium, 1992

[18] Chapman D.B., Zwicky E.D.: Building Internet Firewalls, O'Reilly, 2002

[19] Chen P.M., Noble B.D.: When virtual is better than real, Proceedings of the 2001 Workshop on Hot Topics in Operating Systems (HotOS), pp. 133—138, 2001

[20] Chuvakin A.: Ups and Downs of UNIX/Linux Host-Based Security Solutions,;login: The Magazine of USENIX and SAGE, vol. 28, no. 2, 2003

[21] Cormen T.H., Leiserson C.E., Rivest R.L.: Wprowadzenie do algorytmów, WNT, Warszawa, 2001

[22] Courses E., Surveys T.: A Storm (Worm) Is Brewing, Computer, IEEE Computer Society, vol. 41, no. 2, 2008

[23] Cowan C., Wagle P., Pu C., Beattie S., Walpole J.: Buffer overflows: attacks and defenses for the vulnerability of the decade, DARPA Information Survivability Conference and Expo, 2000

[24] Craig W., McNeal P.: Radmind: The Integration of Filesystem Integrity Checking with File System Management, Proceedings of the 17th USENIX Large Installation System Administration Conference (LISA 2003), 2003

[25] Cyra Ł., Górski J.: Expert assessment of arguments: a method and its experimental evaluation, Computer safety, reliability, and security, 27th International Conference SAFECOMP 2008, pp. 291—304, Springer, 2008

[26] Davis W.S.: Operating systems: a systematic view, Addison-Wesley Publishing Company, Reading 1983

[27] Debar H., Dacier M., Wespi A.: Towards a taxonomy of intrusion detection systems, Computer Networks, vol. 31, no. 8, pp. 805—822, 1999

[28] Diallo M.H., Romero-Mariona J., Sim S.E., Richardson, D.J.: A comparative evaluation of three approaches to specifying security requirements, Proceedings of the 12th Working Conference on Requirements Engineering: Foundation for Software Quality, Luxembourg, 2006

[29] Dunlap G.W., King S.T., Cinar S., Basrai M.A., Chen P.M.: ReVirt: Enabling intrusion analysis through virtual-machine logging and replay, ACM SIGOPS Operating Systems Review, vol. 36, pp. 211—224, 2002

[30] Eastlake D., Jones P.: US Secure Hash Algorithm 1 (SHA1), Network Working Group, RFC 3174, 2001

[31] Fenton N.E.: Software metrics – A rigorous Approach, Chapman & Hall, 1993

[32] Ferguson N., Schneier B.: Kryptografia w praktyce, Helion, Gliwice, 2004

[33] Ferraiolo D., Cugini J., Kuhn D.R.: Role-based access control (RBAC): Features and motivations, Proceedings of 11th Annual Computer Security Application Conference, 1995

[34] Ford R.: The future of virus detection, Information Security Technical Report, vol. 9, no. 2, pp 19—26, Elsevier, 2004

[35] Górski J., Cyra Ł., Jarzębowicz A., Miler J.: Representing and appraising Toulmin model arguments in trust cases, The 8th International Workshop on Computational Models of Natural Argument (CMNA 8), pp. 26—30, 2008

[36] Górski J., Jarzębowicz A., Leszczyna R., Miler J., Olszewski M.: Trust Case: justifying trust in an IT solution, Reliability Engineering and System Safety, vol. 89, pp. 33—47, 2005

[37] Górski J.: Trust-IT - a framework for trust cases, DSN 2007 : 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pp. 204—209, 2007

[38] Górski J.: Trust Case - a case for trustworthiness of IT infrastructures, Cyberspace Security and Defense: Research Issues, NATO Science Series II: Mathematics, Physics and Chemistry, vol. 196, pp. 125—142, Springer-Verlag, 2005

[39] Garcia-Teodoro P., Diaz-Verdejo J., Macia-Fernandez G., Vazquez E.: Anomaly-based network intrusion detection: Techniques, systems and challenges, Computers & Security, vol. 28, no. 1-2. pp. 18–28, 2009

[40] Garfinkel T., Rosenblum M.: A Virtual Machine Introspection Based Architecture for Intrusion Detection, Proceedings of the Network and Distributed Systems Security Symposium, pp. 253-285, 2003

[41] Gasser M.: Building a Secure Computer System, Van Nostrand Reinhold, 1988

[42] Gazet A.: Comparative analysis of various ransomware virii, Journal in Computer Virology, vol. 6, no. 1, pp. 70—90, Springer, 2010

[43] Goldberg R.: Survey of virtual machine research, IEEE Computer Magazine, vol. 7, no. 6, pp 34—45, 1974

[44] Goncalves M.: Firewalls Complete, McGraw Hill, New York, 1998

[45] Gupta P., Krishnan H., Wright C.,Zubair M., Dave M., Zadok E.: Versatility and Unix Semantics in a Fan-Out Unification File System, ACM Transactions on Storage (TOS), vol. 2, no. 1, 2006

[46] Halderman J., Schoen S., Heninger N., Clarkson W., Paul W., Calandrino J., Feldman A., Appelbaum J., Felten E., Foundation. E.: Lest We Remember: Cold Boot Attacks on Encryption Keys, Proceedings of the 17th conference on USENIX Security Symposium, pp. 45—60, 2008

[47] Hanke M., Hauser F.: On the effects of stock spam e-mails, Journal of Financial Markets, vol. 11, no. 1, pp. 57–83, Elsevier, 2008

[48] Hedbom H., Lindskog S., Jonsson E.: Risks and dangers of security extensions, Proceedings of Security and Control of IT in Society-II (IFIP SCITS-II), pp. 231–248, Bratislava, 2001

[49] Heidemann J.S., Popek G.J.: File-system development with stackable layers, ACM Transactions on Computer Systems (TOCS), vol. 12, no. 1, pp. 58—89, 1994

[50] Heidemann J., Popek G.: Performance of Cache Coherence in Stackable Filing, Proceedings of the fifteenth ACM symposium on Operating systems principles, pp. 127—141, 1995

[51] Ianelli N., Hackworth A.: Botnets as a Vehicle for Online Crime, The International Journal of Forensic Computer Science, vol. 2, no. 1, 2007

[52] Information technology – Security techniques – Evaluation criteria for IT security, ISO/IEC 15408:2005 (Common Criteria v3.0). 2005

[53] Ingham K., Forrest S.: A history and survey of network firewalls., Tech. Rep. 2002-37, University of New Mexico, 2002

[54] ISO/IES 9126, Software engineering — Product quality — Quality model, 2001

[55] Jóźwiak I., Laskowski W., Szleszyński A.: Safety of Corporation Electronic Mail Servers, Journal of Konbin, vol. 5, no 2, pp. 71—83, 2008

[56] Kabiri P., Ghorbani A.: Research on intrusion detection and response: A survey, International Journal of Network Security, vol. 1, no. 2, 2005

[57] Kaczmarek J., Wróbel M.: Modern approaches to file system integrity checking, Proceeedings of the 1st International Conference on Information Technology, pp. 403—406, Gdańsk, 2008

[58] Kaczmarek J., Wróbel M.: Nowoczesne mechanizmy ochrony integralności systemów plików, Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej, nr 26, s. 65—68, 2009

[59] Kaczmarek J., Wróbel M.: Obszary zastosowań dystrybucji CDLINUX.PL, Zeszyty Naukowe Wydziału ETI Politechniki Gdańskiej. Technologie Informacyjne, nr 3, s. 221—226, 2004

[60] Kaczmarek J., Wróbel M.: Ocena jakości dystrybucji systemu operacyjnego LINUX typu LIVECD metodą GQM, Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej, nr 22, s. 93—98, 2006

[61] Kaczmarek J., Wróbel M.: Przegląd mechanizmów zabezpieczania systemu operacyjnego, Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej, nr 23, s. 57—60, 2007

[62] Kaczmarek J., Wróbel M.: Zagadnienia bezpieczeństwa systemów operacyjnych, Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej, nr 21, s. 131—136, 2005

[63] Kamel I.: A schema for protecting the integrity of databases, Computers & Security, vol. 28, no. 7, pp. 698–709, 2009

[64] Kashyap A., Dave J., Zubair M., Wright C., Zadok E.: Using Berkeley Database in the Linux kernel, Stony Brook University, 2004

[65] Kashyap A.: File system extensibility and reliability using an in-kernel database, Technical Report, State University of New York, Stony Brook, 2004

[66] Kim G., Spafford E.: Experiences with Tripwire: Using Integrity Checkers for Intrusion Detection, Purdue Technical Report CSD-TR-94-012, 1994

[67] Kim G., Spafford E., The Design and Implementation of Tripwire: A File System Integrity Checker, Proceedings of the 2nd ACM Conference on Computer and Communications Security, pp. 18–29, 1994

[68] Kirda E., Jovanovic N., Kruegel C., Vigna G.: Client-side cross-site scripting protection, Computers & Security, vol. 28, no. 7, pp. 592–604, 2009

[69] Klein G.: Operating system verification - an overview, Sadhana, vol. 34, no. 1, pp. 27–69, Springer, 2009

[70] Knopper K.: Building a self-contained autoconfiguring linux system on an ISO9660 filesystem, In Proceedings of the 4th Annual Linux Showcase and Conference, 2000

[71] Kourai K., Chiba S.: HyperSpector: virtual distributed monitoring environments for secure intrusion detection, Proceedings of the 1st ACM/USENIX International Conference on Virtual Execution Environments, pp. 197—207, 2005

[72] Kroft D.: Lockup-free instruction fetch/prefetch cache organization, Proceedings of the 8th annual symposium on Computer Architecture, 1981

[73] Kyle D. Brustoloni J.C.: Uclinux: a linux security module for trusted-computing-based usage controls enforcement, Proceedings of the 2007 ACM workshop on Scalable trusted computing, pp. 63—70, 2007

[74] Leckie C., Zhou C.V., Karunasekera S.: A survey of coordinated attacks and collaborative intrusion detection, Computers & Security, vol. 29, no. 1, pp. 124–140, 2009

[75] Liska A.: The Practice Of Network Security. Deployment Strategies For Production Environments, Prentice Hall PTR, 2002

[76] Li Y., Wang J.L., Tian Z.H., Lu T.B., Young C.: Building lightweight intrusion detection system using wrapper-based feature selection mechanisms, Computers & Security, vol. 28, no. 6, pp. 466–475, Elsevier, 2009

[77] Luo X., Liao Q.: Ransomware: A New Cyber Hijacking Threat to, Handbook of Research on Information Security and Assurance, 2008

[78] Maraia V.: The Build Master: Microsoft's Software Configuration Management Best Practices, Addison-Wesley, USA, 2006

[79] Marcus J., Ranum A.: Network Frewall, Proceedings of the World Conference on System Administration and Security, Waszyngton, pp. 153–163, 1992

[80] Mazieres D., Kaminsky M., Kaashoek M.F., Witchel E.: Separating key management from file system security, ACM SIGOPS Operating Systems Review, vol. 33, no. 5, 1999

[81] McHugh J.: Intrusion and Intrusion Detection, International Journal of Information Security, vol. 1, no. 1, pp. 14—35, Springer, 2001

[82] Menezes A.J., Oorschot P.C. van, Vanstone S.A.: Kryptografia Stosowana, WNT, Warszawa, 2005

[83] Mercuri R., Neumann P.: Security by Obscurity, Communications of the ACM, vol. 46, no. 11, 2003

[84] Miretskiy Y., Das A., Wright C.P., Zadok E.: Avfs: an on-access anti-virus file system, Proceedings of the 13th conference on USENIX Security Symposium, vol. 13, pp. 73—88, 2004

[85] Mirkovic J.: A taxonomy of DDoS attack and DDoS defense mechanisms, ACM SIGCOMM Computer Communication Review, vol. 34, no. 2, 2004

[86] Motara Y.M., Irwin B.V.W.: File Integrity Checkers: State of the Art and Best Practices, Information Security South Africa (ISSA), Johannesburg, 2005

[87] Patil S., Kashyap A., Sivathanu G., Zadok E.: I3FS: An In-Kernel Integrity Checker and Intrusion Detection File System, Proceedings of the 18th USENIX conference on System administration, 2004

[88] Piegdon D., Pimenidis L.: Hacking in physically adressable memory, Procceedings of the 4th International Conference on Detection of Intrusions & Malware, and Vulnerability Assessment, 2007

[89] Popek G., Goldberg R.: Formal requirements for virtualizable third generation architectures, Communications of the ACM, vol. 17, no. 7, pp. 412–421, 1974

[90] Ports D.R.K., Garfinkel T.: Towards application security on untrusted operating systems, Proceedings of the 3rd conference on Hot topics in security, 2008

[91] Provos N.: A Virtual Honeypot Framework, Proceedings of the 13th USENIX Security Symposium, vol. 132, 2004

[92] Rivest R.: The MD5 Message-Digest Algorithm, RFC1321, 1992

[93] Robling-Denning D.E.: Kryptografia i ochrona danych, WNT, Warszawa, 1993

[94] Rocke A., DeMara R.: Mitigation of network tampering using dynamic dispatch of mobile agents, Computers & Security, vol. 23, no. 1, pp. 31—42, Elsevier, 2004

[95] Roeckl C., Director C.M.: Stateful Inspection Firewalls, Juniper Networks White Paper, 2004

[96] Rosenblum D.: What anyone can know: The privacy risks of social networking sites, IEEE Security & Privacy, vol. 5, no. 3, pp. 40–49, 2007

[97] Russell R., Quinlan D., Yeoh C.: Filesystem Hierarchy Standard, Technical Report, Filesystem Hierarchy Standard Group, 2004

[98] Satoh T., Haga M., Kurosawa K.: Towards secure and fast hash functions, IEICE Transactions on Fundamentals Electronics Communications And Computer Sciences, vol. 82, pp. 55—62, 1999

[99] Serafim V., Weber R.: Restraining and repairing file system damage through file integrity control, Computers & Security, vol. 23, no. 1, Elsevier 2004

[100] Shaw A.C., Projektowanie logiczne systemów operacyjnych, WNT, Warszawa, 1980

[101] Silberschatz A., Peterson J.L., Galvin P.B., Podstawy systemów operacyjnych, WNT, Warszawa, 2006

[102] Skoudis E., Zeltser L.: Malware: Fighting Malicious Code, Prentice Hall, NJ, USA, 2003

[103] Smalley S.: Configuring the SELinux Policy, NAI Labs Report #02-007, 2002

[104] Smith R.G. , Holmes M.N., Kaufmann P.: Nigerian Advance Fee Fraud, Trends and Issues in Crime and Criminal Justice, vol. 121, 1999

[105] Solingen R. van, Berghout E.: The Goal/Question/Metric Method – A practical Guide for Quality Improvement of Software Development, McGraw-Hill, 1999

[106] Spitzner L.: Honeypots: catching the insider threat, Proceedings of the 19th Computer Security Applications Conference, 2003

[107] Spitzner L.: Honeypots: Tracking Hackers, Addison-Wesley Professional, 2003

[108] Stallings W.: Systemy operacyjne: struktury i zasady budowy, Wydawnictwo Naukowe PWN, Warszawa 2006

[109] Summers C.: Introduction to ISO 9660, what it is, how it is implemented, and how it has been extended, Disc Manufacturing, 1993

[110] Taber M.: Maximum Security: A Hacker's Guide to Protecting Your Internet Site and Network, Sams Publishing, 1998

[111] Wack J.P., Carnahan L.J.: Keeping your site comfortably secure: An introduction to internet firewalls, US Dept. of Commerce, National Institute of Standards and Technology, 1994

[112] Wang X., Yin Y. L., Yu H.: Finding collisions in the full SHA-1. Advances in Cryptology – EURO CRYPT 2005, Springer-Verlag, 2005

[113] Wang X., Yu H.: How to break MD5 and Other Hash Functions, Advances in Cryptology – EURO CRYPT 2005, pp. 19—35, Springer-Verlag, 2005

[114] Willems C., Holz T., Freiling F.: Toward automated dynamic malware analysis using cwsandbox, IEEE Security & Privacy, vol. 5, no. 2, pp. 32–39, 2007

[115] Wright C., Cowan C., Morris J., Smalley S., Kroah-Hartman G.: Linux security modules: general security support for the linux kernel, Proceedings of the 11th USENIX Security Symposium, vol. 2, 2002

[116] Yegneswaran V., Barford P., Ullrich J.: Internet intrusions: global characteristics and prevalence, Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, ACM New York, 2003

[117] Zadok E., Badulescu I., Shender A.: Cryptfs: A stackable vnode level encryption filesystem, Technical Report CUCS-021-98, Computer Science Department, Columbia University, 1998

[118] Zadok E., Badulescu I., Shender. A.: Extending File System Using Stackable Templates, Proceedings of the Annual USENIX Technical Conference, 1999

[119] Zadok E., Iyer R., Joukov N., Sivathanu G., Wright C.: On Incremental File System Development, ACM Transactions on Storage (TOS), vol. 2, no. 2, 2006

[120] Zadok E., Nieh. J.: FiST: A Language for Stackable File System, Proceedings of the Annual USENIX Technical Conference, pp. 55—70, 2000

[121] Zalenski R.: Firewall technologies, IEEE Potentials, vol. 21, no. 1, pp. 24–29, New York, 2002

[122] Zhang X., Li C., Zheng, W.: Intrusion Prevention System Design, The Fourth International Conference on Computer and Information Technology, 2004

[123] Zhao X., Borders K., Prakash A.: Towards protecting sensitive files in a compromised system, Third IEEE International Security in Storage Workshop, 2005

Źródła internetowe

[124] Boileau A.: Hit by a Bus: Physical Access Attacks with Firewire, http://security-assessment.com/files/presentations/ab_firewire_rux2k6-final.pdf, 2006, z dnia 11.01.2010

[125] Goldt S., van der Meer S., Burkett S., Welsh M.: The Linux Programmer's Guide, Linux Documentation Project, http://unix.co.kr/page/lpg/lpg-0.4.pdf, 1995, z dnia 11.01.2010

[126] Operating System Protection Profile Version 2.0, Bundesamt fur Sicherheit in der Informationstehnik, 2010, https://www.bsi.bund.de/cln_183/DE/Themen/ZertifizierungundAkkreditierung/ZertifizierungnachCCundITSEC/SchutzprofileProtectionProfiles/schutzprofileprotectionprofiles_node.html

[127] Smalley S., Fraser T., Vance C.: Linux Security Modules: General Security Hooks for Linux, http://tali.admingilde.org/linux-docbook/lsm.pdf, z dnia 11.01.2010

[128] Stevens C., Merkin S.: "El Torito" Bootable CD-ROM Format Specification, 1995, http://bochs.sourceforge.net/techspec/el-torito.pdf.gz, z dnia 11.01.2010

[129] U.S. Government Approved Protection Profile - U.S. Government Protection Profile Anti-Virus Applications for Workstations in Basic Robustness Environments Version 1.2, 2007, http://www.niap-ccevs.org/pp/

[130] U.S. Government Protection Profile Database Management Systems for Basic Robustness Environments, Version 1.2, 2007, http://www.niap-ccevs.org/pp/

[131] http://media.mbank.pl/PressOffice/PressRelease.92918.po, z dnia 11.01.2010

[132] https://www.sklepy24.pl/zakupy/raport_e-handel_2009, z dnia 05.05.2010

[133] http://www.alert24.pl/alert24/1,84880,4841501.html, z dnia 11.01.2010

[134] http://www.cert.org/advisories/CA-2001-09.html , z dnia 11.01.2010

[135] http://www.debian.org/social_contract, z dnia 11.01.2010

[136] http://www.kb.cert.org/vuls/id/945216, z dnia 11.01.2010

[137] http://www.kernel.org/pub/linux/kernel/v2.6, z dnia 14.11.2008

[138] http://www.securityfocus.com/bid/27704/discuss, z dnia 11.01.2010

[139] http://www.shadowserver.org/wiki/pmwiki.php/Shadowserver/Mission, z dnia 11.01.2010

Wykaz skrótów

API	Application Programming Interface, interfejs udostępniany przez program komputerowy w celu umożliwienia komunikacji z innym aplikacjami
CC	Common Criteria for Information Security Evaluation, przyjęty w normie PN-ISO/IEC 15408, zbiór kryteriów bezpieczeństwa dla produktów informatycznych
CERT	Computer Emergency Response Team, grupy ekspertów zajmujące się bezpieczeństwem systemów komputerowych
CMM	Capability Maturity Model for Software, model oceny jakości produktów informatycznych badający dojrzałość organizacji wytwarzającej
DAC	Discretionary Access Control, swobodna kontrola dostępu do zasobów, która pozwala właścicielowi na udostępnianie własnych zasobów
DDoS	Distributed Denial-of-Service, atak polegający na wysyłaniu w krótkim czasie licznych zapytań do serwisu internetowego przez rozproszone geograficznie komputery
EXT	Extended File System, rodzina systemów plików stworzonych dla systemu Linux
FAT	File Allocation Table, rodzina systemów plików ogólnego przeznaczenia
FiST	File System Translator, język programowania ułatwiający tworzenie wieżowych systemów plików
GPL	GNU Public License, licencja do rozprowadzania produktów informatycznych zgdonie z zasadami wolnego oprogramowania
GQM	Goal, Question, Metric, metoda oceny jakości, której istotą jest jasne zdefiniowanie celu pomiaru
HFS	Hierarchical File System, rodzina systemów plików stworzonych dla systemu MacOS
HIDS	Host Intrusion Detection System, systemy wykrywania włamań zainstalowane na chronionym komputerze
IDS	Intrusion Detection System, system wykrywania włamań
IPS	Intrusion Prevention System, rozszerzenie systemów IDS o możliwość podejmowania działań prewencyjnych
ISO-OSI	ISO Open Systems Interconnection, standard opisujący strukturę połączeń sieciowych
LSB FHS	Linux Standard Base Filesystem Hierarchy Standard, standard definiujący strukturę katalogów w systemie Linux
LSM	Linux Security Modules, szkielet pozwalający na włączanie mechanizmów bezpieczeństwa do jądra systemu operacyjnego Linux
MAC	Mandatory Access Control, modelem obowiązkowej kontroli dostępu

MD5	Message-Digest algorithm 5, prosta funkcja skrótu kryptograficznego
MLS	Multi-Level Security, wielopoziomowy model bezpieczeństwa, powszechnie stosowany w systemach wojskowych
NIDS	Network Intrusion Detection System, systemy wykrywania włamań działające w lokalnej sieci komputerowej
NTFS	New Technology File System, system plików stworzony dla systemów Windows NT
OSPP	Operating System Protection Profile, Profil Ochrony Systemów Operacyjnych, dokument opisujący na podstawie normy PN-ISO/IEC 15408 wymagania bezpieczeństwa dla systemów operacyjnycjh
POSIX	Portable Operating System Interface, standard opisujący API dla systemów operacyjnych bazujących na systemie UNIX
PP	Protection Profile, Profil Ochrony, dokument normy PN-ISO/IEC 15408 zawierający zbiorów wymogów bezpieczeństwa dla całych kategorii produktów informatycznych
RBAC	Role-Based Access Control, kontrola dostępu oparta na rolach, pozwala na zarządzanie dostępem do zasobów np. w modelu MAC
SHA	Secure Hash Algorithm, rodzina funkcji skrótów kryptograficznych
ST	Security Target, Zadania Zabezpieczeń, dokument oceny bezpieczeństwa zgodnie z normą PN-ISO/IEC 15408 produktu informtycznego
TCB	Trusted Computing Base, zbiór wszystkich mechanizmów, zarówno sprzętowych jak i programowych, odpowiedzialnych za bezpieczeństwo systemu komputerowego
TOE	Target of Evaluation, w nomenklaturze normy PN-ISO 15408 produkt informatyczny podlegający ocenie
TSF	TOE Security Functions, funkcje zabezpieczające TOE, do których zaliczane są wszystkie mechanizmy powiązane bezpośrednio lub pośrednio z bezpieczeństwem
VFS	Virtual File System, Wirtualny System Plików, abstrakcyjna warstwa w jądrze systemu operacyjnego pozwalająca współdziałanie różnych systemów plików