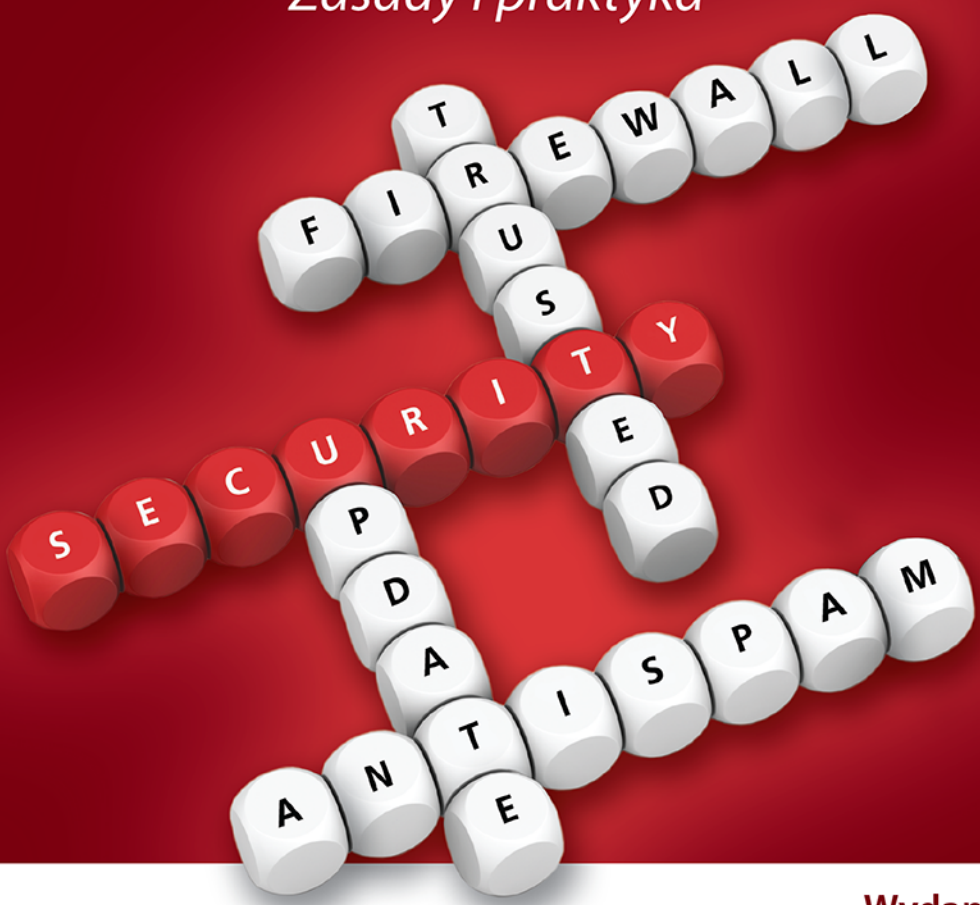
William Stallings • Lawrie Brown

BEZPIECZEŃSTWO SYSTEMÓW INFORMATYCZNYCH

Zasady i praktyka



TOM 2

Wydanie IV

Pearson

Helion

Tytuł oryginału: Computer Security: Principles and Practice (4th Edition)

Tłumaczenie: Radosław Meryk (rozdz. 14 – 27, dod. A – K), Zdzisław Płoski (wstęp)

ISBN: 978-83-8322-558-6

Authorized translation from the English language edition, entitled: COMPUTER SECURITY: PRINCIPLES AND PRACTICE, Fourth Edition; ISBN 0134794109; by William Stallings, and by Lawrie Brown, published by Pearson Education, Inc. Copyright © 2018, 2015, 2012, 2008 by Pearson Education, Inc.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/bes42v
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

SPIS TREŚCI

CZĘŚĆ V BEZPIECZEŃSTWO SIECI

ROZDZIAŁ 18

AUDYT BEZPIECZEŃSTWA

W TYM ROZDZIALE POZNASZ I ZROZUMIESZ:

- elementy składowe architektury audytu bezpieczeństwa;
- względne zalety różnych typów ścieżek z audytu bezpieczeństwa;
- kluczowe zagadnienia związane z implementacją funkcji logowania na potrzeby audytu bezpieczeństwa;
- proces analizy ścieżki audytu.

Audyt bezpieczeństwa to forma audytu, która koncentruje się na bezpieczeństwie zasobów informatycznych (zasobów IT) organizacji. Jest to kluczowy mechanizm komputerowego bezpieczeństwa. Audyt bezpieczeństwa może:

- Zapewnić o prawidłowym działaniu komputera pod względem bezpieczeństwa.
- Wygenerować dane, które mogą być wykorzystane w analizie ataku „po fakcie", bez względu na to, czy atak zakończył się powodzeniem, czy nie.
- Dostarczyć mechanizmów oceny niedociągnięć w usłudze bezpieczeństwa.
- Dostarczyć informacji, które można wykorzystać do zdefiniowania anormalnego zachowania.
- Utrzymać rejestr informacji przydatnych w śledztwach komputerowych.

Dwa kluczowe pojęcia to **audyt bezpieczeństwa** i **ścieżki audytu bezpieczeństwa**[1] zdefiniowane w tabeli 18.1.

Tabela 18.1. Terminologia związana z audytami bezpieczeństwa (RFC 4949)

Audyt bezpieczeństwa Niezależny przegląd i analiza zapisów i działań systemu w celu określenia adekwatności zabezpieczeń w systemie, zapewnienia zgodności z ustaloną polityką bezpieczeństwa i procedurami, wykrywania naruszeń w usługach bezpieczeństwa oraz zalecenia zmian wykonywanych w ramach działań zaradczych. Podstawowym celem audytu jest ustalenie odpowiedzialności za elementy systemowe, które inicjują zdarzenia i działania związane z bezpieczeństwem lub biorą w nich udział. Dlatego potrzebne są środki do generowania i rejestrowania śladu audytu bezpieczeństwa oraz do przeglądania i analizowania ścieżek audytu w celu wykrycia i zbadania ataków i naruszeń w zabezpieczeniach.
Ścieżka audytu bezpieczeństwa Chronologiczny zapis działań w systemie, wystarczający do odtworzenia i zbadania sekwencji zjawisk środowiskowych i działań lub czynności prowadzących do wykonania operacji, procedur lub zdarzeń w transakcji związanej z bezpieczeństwem — od zainicjowania do końcowych rezultatów.

[1] Standard NIST SP 800-12 (*An Introduction to Computer Security: The NIST Handbook* z października 1995) wskazuje, że niektórzy eksperci ds. bezpieczeństwa rozróżniają ślad audytu od logu audytu w następujący sposób: log jest zapisem zdarzeń, które zaszły w konkretnym pakiecie oprogramowania, a ścieżka audytu to cała historia zdarzenia, często z wykorzystaniem wielu logów. Jednak w społeczności specjalistów bezpieczeństwa nie stosuje się powszechnie tej definicji. W tej książce nie dokonujemy takiego rozróżnienia.

Proces generowania informacji audytu dostarcza danych, które mogą być przydatne w czasie rzeczywistym do wykrywania włamań; ten aspekt omówiono w rozdziale 8. W niniejszym rozdziale skupimy się na gromadzeniu, przechowywaniu i analizie danych związanych z bezpieczeństwem IT. Zaczniemy od ogólnego przeglądu zagadnień związanych z architekturą audytu bezpieczeństwa oraz pokazania, w jaki sposób odnosi się ona do działań towarzyszących wykrywaniu włamań. Następnie omówimy różne aspekty ścieżek audytu, znanych również jako logi audytu. Na koniec omówimy problematykę analizy danych z audytu.

18.1. ARCHITEKTURA AUDYTU BEZPIECZEŃSTWA

Dyskusję na temat audytu bezpieczeństwa rozpoczynamy od analizy elementów składających się na jego architekturę. Najpierw zbadamy model, który prezentuje audyt bezpieczeństwa w szerszym kontekście. Następnie przyjrzymy się funkcjonalnemu podziałowi audytu bezpieczeństwa.

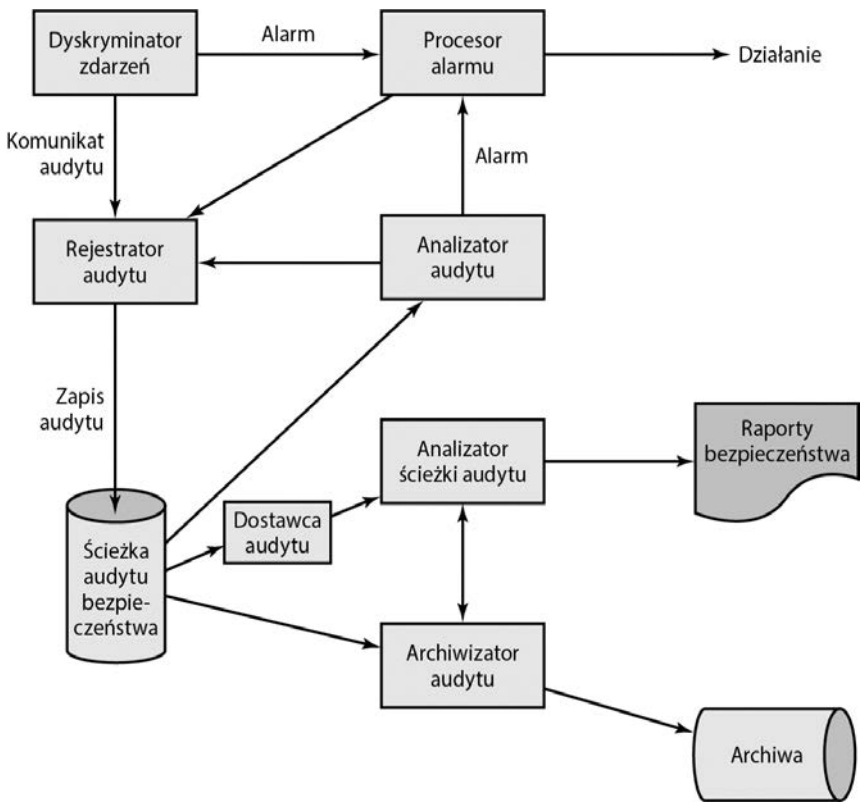
Model audytu bezpieczeństwa i alarmów

Rekomendacja X.816 ITU-T[2] definiuje model, w którym wskazano elementy funkcji audytu bezpieczeństwa i ich związek z alarmami bezpieczeństwa. Ten model przedstawiono na rysunku 18.1. Oto jego kluczowe elementy:

- **Dyskryminator zdarzeń** — wbudowana w oprogramowanie systemu logika, która monitoruje aktywność systemu i wykrywa związane z bezpieczeństwem zdarzenia skonfigurowane do wykrywania.
- **Rejestrator audytu** — dla każdego wykrytego zdarzenia dyskryminator zdarzenia przesyła informacje do rejestratora audytu. W modelu zaprezentowano tę komunikację w formie komunikatu. Audyt można również realizować poprzez rejestrację zdarzeń w obszarze pamięci współdzielonej.
- **Procesor alarmów** — niektóre zdarzenia wykryte przez dyskryminator zdarzeń są zdefiniowane jako alarmowe. W przypadku takich zdarzeń do procesora alarmów jest wysyłany alarm. Na tej podstawie procesor alarmów podejmuje określone działania. Te działania same w sobie są zdarzeniami, które można rejestrować, dlatego są przesyłane do rejestratora audytu.
- **Ścieżka audytu bezpieczeństwa** — rejestrator audytu tworzy sformatowany zapis każdego zdarzenia i rejestruje go w ścieżce audytu bezpieczeństwa.

[2] Telecommunication Standardization Sector of the International Telecommunications Union. Informacje dotyczące tej i innych instytucji tworzących standardy można znaleźć w dodatku C.

- **Analizator audytu** — ścieżka audytu bezpieczeństwa jest dostępna dla analizatora audytu. Ten, na podstawie wzorca działania, może zdefiniować nowe, podlegające audytowi zdarzenie, które jest wysyłane do rejestratora audytu i może wygenerować alarm.
- **Archiwizator audytu** — moduł oprogramowania, który okresowo wyodrębnia rekordy ze ścieżki audytu w celu utworzenia stałego archiwum zdarzeń podlegających audytowi.
- **Archiwa** — archiwa audytu są trwałym magazynem zdarzeń związanych z bezpieczeństwem w tym systemie.
- **Dostawca audytu** — dostawca audytu jest aplikacją i (lub) interfejsem użytkownika do ścieżki audytu.
- **Analizator ścieżki audytu** — analizator ścieżki audytu to aplikacja lub użytkownik, który sprawdza ścieżkę audytu i archiwa audytu pod kątem historycznych trendów dla celów komputerowych śledztw oraz innych analiz.
- **Raporty bezpieczeństwa** — analizator ścieżki audytu przygotowuje czytelne dla człowieka raporty bezpieczeństwa.

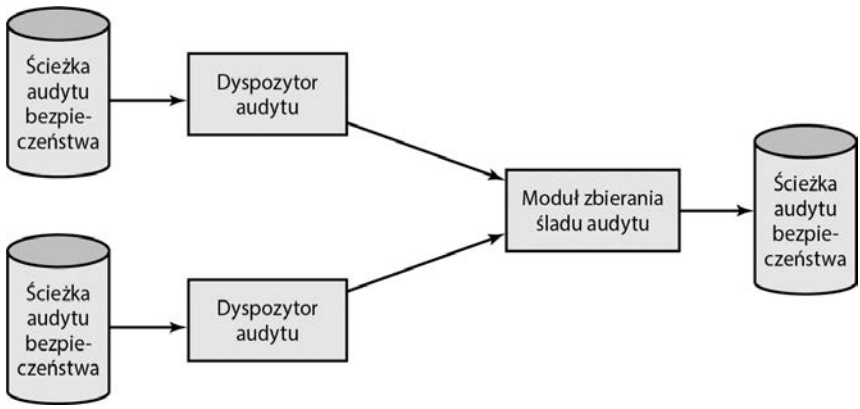


Rysunek 18.1. **Model audytu bezpieczeństwa i alarmów (X.816)**

Ten model ilustruje związek pomiędzy funkcjami audytu a funkcjami alarmowymi. Funkcja audytu tworzy rejestr zdefiniowanych przez administratora bezpieczeństwa zdarzeń związanych z bezpieczeństwem. Niektóre z tych zdarzeń mogą być rzeczywistymi naruszeniami bezpieczeństwa lub podejrzeniem naruszenia bezpieczeństwa. Takie zdarzenia są przekazywane za pośrednictwem alarmów do systemu wykrywania włamań lub zapory firewall.

Tak jak jest w przypadku wykrywania włamań, w systemach rozproszonych może być przydatna rozproszona funkcja audytu, w której tworzone jest scentralizowane repozytorium. Dla usługi rozproszonego audytu potrzebne są dwa dodatkowe elementy logiczne (patrz rysunek 18.2):

- **Zbieracz śladu audytu** (ang. *audit trail collector*) — moduł wchodzący w skład scentralizowanego systemu, który gromadzi zapisy ścieżki audytu z innych systemów i tworzy scaloną ścieżkę audytu.
- **Dyspozytor audytu** — moduł, który przesyła zapisy ścieżki audytu z lokalnego systemu do centralnego modułu zbierania ścieżek audytu.

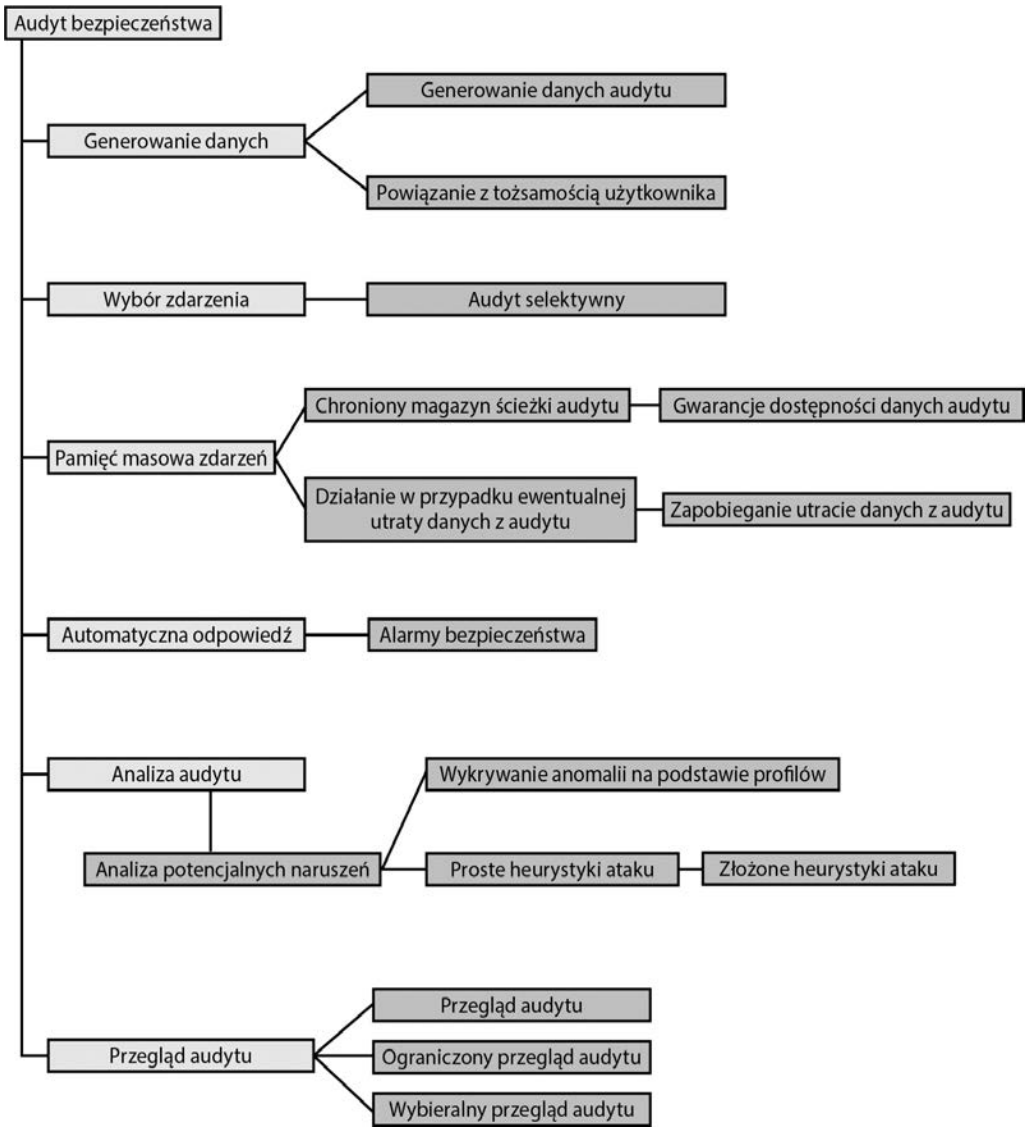


Rysunek 18.2. Model rozproszonego śladu audytu (X.816)

Funkcje audytu bezpieczeństwa

Warto przyjrzeć się kolejnemu podziałowi mechanizmu audytu bezpieczeństwa, opracowanemu w ramach specyfikacji Common Criteria [CCPS12a]. Podział audytu bezpieczeństwa na sześć głównych obszarów, z których każdy spełnia jedną lub więcej konkretnych funkcji, pokazano na rysunku 18.3:

- **Generowanie danych** — identyfikuje poziom rejestrowania, wylicza typy możliwych do zarejestrowania zdarzeń i identyfikuje minimalny zestaw dostarczanych informacji podlegających audytowi. Ta funkcja musi również rozwiązywać konflikt pomiędzy bezpieczeństwem a prywatnością i określać, dla których zdarzeń w wynikowym raporcie powinny się znaleźć dane tożsamości użytkownika powiązanego z działaniami.



Rysunek 18.3. **Dekompozycja klas audytu kontroli zgodnie ze specyfikacją Common Criteria**

- **Wybór zdarzeń** — uwzględnienie lub wykluczenie zdarzenia z zestawu zdarzeń podlegających rejestrowaniu. Umożliwia to skonfigurowanie systemu na różnych poziomach szczegółowości, tak by uniknąć generowania nieprzydatnych raportów.
- **Przechowywanie zdarzeń** — tworzenie i utrzymanie bezpiecznego śladu audytu. Funkcja przechowywania obejmuje mechanizmy zapewniające dostępność i zapobiegające utracie danych ze ścieżki audytu.
- **Automatyczna odpowiedź** — określa reakcje po wykryciu zdarzeń wskazujących na potencjalne naruszenia bezpieczeństwa.

- **Analiza audytu** — wykonywana za pośrednictwem zautomatyzowanych mechanizmów analizy aktywności w systemie i danych audytu w poszukiwaniu naruszeń bezpieczeństwa. Ten komponent identyfikuje zestaw zdarzeń, których wystąpienie lub skumulowane wystąpienie wskazuje na potencjalne naruszenie bezpieczeństwa. W przypadku takich zdarzeń przeprowadza się analizę w celu ustalenia, czy doszło do naruszenia bezpieczeństwa. W tej analizie wykorzystuje się mechanizmy wykrywanie anomalii i heurystykę ataków.
- **Przegląd audytu** — dostępny dla uprawnionych użytkowników, którzy wspomagają proces przeglądania danych z audytu. Komponent przeglądu audytu może zawierać wybieraną funkcję przeglądania, która dostarcza możliwości wykonywania operacji wyszukiwania z wykorzystaniem jednego bądź wielu kryteriów powiązanych relacjami logicznymi (tzn. `and`, `or`), sortowaniem danych audytu oraz filtrowaniem danych audytu przed ich przeglądaniem. Uprawnienia do przeglądu audytu mogą być ograniczone dla wybranych użytkowników.

Wymagania

Na podstawie zbioru funkcjonalności sugerowanych przez rysunki 18.1 i 18.3 możemy opracować zestaw wymagań dotyczących audytu bezpieczeństwa. Pierwszym wymaganiem jest **zdefiniowanie zdarzeń**. Administrator bezpieczeństwa powinien zdefiniować zestaw zdarzeń podlegających audytowi. Tym tematem zajmiemy się bardziej szczegółowo w następnym podrozdziale. Poniżej podajemy listę zaproponowaną w [CCPS12a]:

- wprowadzenie obiektów w części oprogramowania związanej z bezpieczeństwem do przestrzeni adresowej podmiotu;
- usuwanie obiektów;
- przydzielanie lub odwoływanie uprawnień lub możliwości dostępu;
- zmiany atrybutów podmiotu lub obiektu zabezpieczeń;
- kontrole zasad wykonywane przez oprogramowanie zabezpieczające w wyniku żądania podmiotu;
- korzystanie z praw dostępu w celu ominięcia sprawdzenia zasad;
- wykorzystanie funkcji identyfikacji i uwierzytelniania;
- czynności związane z bezpieczeństwem podjęte przez operatora i (lub) uprawnionego użytkownika (np. wyłączenie mechanizmu zabezpieczeń);
- importowanie (eksportowanie) danych z nośników wymiennych lub na nośniki wymienne (np. tworzenie wydruków, zapis na dyski magnetyczne bądź optyczne albo na przenośne urządzenia pamięci masowej USB).

Drugie wymaganie dotyczy dostępności w aplikacjach i oprogramowaniu systemowym odpowiednich haków pozwalających na **wykrywanie zdarzeń**. W celu zarejestrowania odpowiednich działań należy dodać do systemu oprogramowanie monitorujące. Potrzebny jest również mechanizm **nagrywania zdarzeń**, który uwzględnia potrzebę zapewnienia bezpiecznej pamięci trwałej odpornej na zmodyfikowanie lub usunięcie. W celu analizy zebranych danych, a także w celu badania trendów w danych i anomalii mogą być wykorzystywane **oprogramowanie, narzędzia i interfejsy do analizy zdarzeń i ścieżek audytu**.

Istnieje dodatkowe wymaganie dotyczące **bezpieczeństwa funkcji audytu**. Przed obchodzeniem lub naruszeniem muszą być chronione nie tylko ścieżka audytu, ale całe oprogramowanie do obsługi audytu i pośrednia pamięć. System audytu powinien mieć **minimalny wpływ na funkcjonalność**.

Wytyczne dotyczące wdrażania

Przydatny zestaw wytycznych dotyczących audytu systemów informatycznych zawiera standard ISO[3] 27002 (*Code of Practice for Information Security Management* z października 2013 r.):

1. Wymagania audytu dotyczące dostępu do systemów i danych powinny zostać uzgodnione z kierownictwem odpowiedniego szczebla zarządzania.
2. Zakres technicznych testów audytu powinien być uzgodniony i kontrolowany.
3. W testach kontrolnych powinien być wykorzystywany tylko dostęp do odczytu do oprogramowania i danych.
4. Dostęp do danych inny niż tylko do odczytu powinien być dozwolony tylko w przypadku kopii pojedynczych plików systemowych, które powinny zostać usunięte po zakończeniu audytu lub — jeżeli w wymaganiach dotyczących dokumentacji audytu istnieje obowiązek przechowywania takich plików — objęte odpowiednią ochroną.
5. Należy zidentyfikować i uzgodnić wymagania dotyczące specjalnego lub dodatkowego przetwarzania.
6. Testy kontrolne, które mogą wpływać na dostępność systemu, powinny być uruchamiane poza godzinami pracy.
7. W celu utworzenia ścieżki referencyjnej wszystkie dostępy powinny być monitorowane i rejestrowane.

[3] International Organization for Standardization. Więcej informacji na temat tej i innych instytucji tworzących standardy oraz listę dokumentów NIST i ISO można znaleźć w dodatku C.

18.2. ŚLAD AUDYTU BEZPIECZEŃSTWA

Ścieżki audytu zachowują rejestr aktywności w systemie. W tym podrozdziale zajmiemy się kwestiami dotyczącymi ścieżek audytu.

Co zbierać?

Wybór danych do zbierania zależy od wielu wymagań. Jednym z problemów jest ilość danych do zebrania, która zależy od zakresu obszarów zainteresowania i szczegółowości gromadzonych danych. Należy zastosować kompromis pomiędzy objętością a wydajnością. Im więcej gromadzonych danych, tym większe obniżenie wydajności systemu. Większe ilości danych mogą również niepotrzebnie obciążać różne algorytmy używane do badania i analizy danych. Ponadto istnienie dużej ilości danych tworzy pokusę generowania raportów bezpieczeństwa w nadmiernej liczbie lub objętości.

Z uwagi na powyższe ostrzeżenia pierwszym wymaganiem biznesowym w projektowaniu ścieżki audytu bezpieczeństwa jest wybór elementów danych do zebrania. Mogą to być:

- Zdarzenia związane z korzystaniem z oprogramowania do audytów (tzn. z wszystkich komponentów z rysunku 18.1).
- Zdarzenia związane z mechanizmami zabezpieczeń w systemie.
- Wszelkie zdarzenia zbierane w celu wykorzystania przez różne mechanizmy wykrywania zagrożeń i zapobiegania im. Obejmuje to elementy istotne dla działania mechanizmu wykrywania włamań oraz zapory firewall.
- Zdarzenia związane z zarządzaniem systemem i jego działaniem.
- Dostęp do systemu operacyjnego (np. za pośrednictwem wywołań systemowych podobnych do tych, które wyszczególniono w tabeli 8.2).
- Dostęp do wybranych aplikacji.
- Zdalny dostęp.

Jednym z przykładów zdarzeń do wyboru może być sugerowana lista zdarzeń podlegających audytowi pochodząca ze standardu X.816, którą zaprezentowano w tabeli 18.2. Standard wskazuje, że audytem powinny być objęte zarówno działania normalne, jak i nadzwyczajne; na przykład przedmiotem rejestrowania w ścieżce audytu może być każde żądanie połączenia, takie jak żądanie połączenia TCP, niezależnie od tego, czy było ono prawidłowe, czy nie, i niezależnie od tego, czy żądanie zostało zaakceptowane, czy nie. To jest bardzo istotna kwestia. Zbieranie danych do audytu wykracza poza potrzebę generowania alarmów bezpieczeństwa lub wprowadzania danych do modułu zapory firewall. Do identyfikowania normalnych i nienormalnych wzorców użycia, a zatem

Tabela 18.2. **Zdarzenia podlegające audytowi sugerowane w X.816**

Zdarzenia związane z bezpieczeństwem związane z konkretnym połączeniem • żądania połączenia; • potwierdzenia połączeń; • żądania rozłączenia; • potwierdzenia rozłączenia; • statystyki dotyczące połączenia. **Zdarzenia związane z bezpieczeństwem związane z korzystaniem z usług zabezpieczeń** • żądania usługi zabezpieczeń; • skorzystanie z mechanizmów zabezpieczeń; • alarmy bezpieczeństwa. **Zdarzenia związane z bezpieczeństwem dotyczące zarządzania** • operacje zarządzania; • powiadomienia związane z zarządzaniem. **Lista zdarzeń podlegających audytowi powinna zawierać przynajmniej** • odmowy dostępu; • uwierzytelnienia; • zmiany atrybutów; • tworzenie obiektów; • usunięcie obiektów; • modyfikowanie obiektów; • skorzystanie z uprawnień.	**W zakresie pojedynczych usług bezpieczeństwa ważne są następujące zdarzenia** • uwierzytelnianie: weryfikacja pomyślna; • uwierzytelnianie: weryfikacja niepomyślna; • kontrola dostępu: decyzja o przyznaniu dostępu; • kontrola dostępu: decyzja o odmowie dostępu; • niezaprzeczalność: niezaprzeczalne pochodzenie wiadomości; • niezaprzeczalność: niezaprzeczalne odebranie wiadomości; • niezaprzeczalność: nieudane odrzucenie zdarzenia; • niezaprzeczalność: pomyślne odrzucenie zdarzenia; • integralność: użycie osłony; • integralność: użycie bez osłony; • integralność: pomyślna weryfikacja poprawności; • integralność: niepomyślna weryfikacja poprawności; • poufność: skorzystanie z ukrycia; • poufność: wykorzystanie ujawnienia; • audyt: wybór zdarzenia do audytu; • audyt: anulowanie wyboru zdarzenia do audytu; • audyt: zmiana kryteriów wyboru zdarzeń do audytu.

spełniania roli danych wejściowych do analizy wykrywania włamań, mogą być używane dane reprezentujące zachowania niewyzwalające alarmów. Ponadto w przypadku ataku może być potrzebna analiza całej aktywności w systemie. To pozwala zdiagnozować atak i znaleźć właściwe środki zaradcze na przyszłość.

Kolejną przydatną listę zdarzeń podlegających audytowi (patrz tabela 18.3) zestawiono w dokumencie standardu ISO 27002. Podobnie jak jest w X.816, standard ISO wyszczególnia zarówno zdarzenia autoryzowane, jak i nieautoryzowane, a także zdarzenia wpływające na funkcje bezpieczeństwa systemu.

Ponieważ administrator bezpieczeństwa projektuje zasady zbierania danych audytu, przydatne jest podzielenie ścieżki audytu na kategorie w celu dokonania selekcji elementów danych do zebrania. Poniżej zaprezentujemy przydatne kategorie projektowania ścieżek audytu.

Tabela 18.3. Obszary monitorowania zasugerowane w standardzie ISO 27002

a)	identyfikatory użytkowników
b)	operacje systemowe
c)	daty, godziny i szczegóły kluczowych zdarzeń, na przykład logowania i wylogowania użytkowników
d)	tożsamość urządzeń lub, jeśli to możliwe, lokalizacja, oraz identyfikator systemowy
e)	zapisy udanych i odrzuconych prób dostępu do systemu
f)	zapisy udanych i odrzuconych prób dostępu do danych oraz innych zasobów
g)	zmiany w konfiguracji systemu
h)	korzystanie z uprawnień
i)	korzystanie z narzędzi i aplikacji systemowych
j)	dostęp do plików i rodzaj dostępu
k)	adresy i protokoły sieciowe
l)	alarmy wywoływane przez system kontroli dostępu
m)	aktywacje i dezaktywacje systemów ochrony, takich jak systemy antywirusowe i systemy wykrywania intruzów
n)	zapisy transakcji dokonanych przez użytkowników w aplikacjach

Ścieżki audytu na poziomie systemu

Zwykle są używane do monitorowania i optymalizacji wydajności systemu, ale mogą również spełniać funkcje audytu bezpieczeństwa. Niektóre aspekty polityki bezpieczeństwa egzekwuje sam system — na przykład dostęp do systemu. Ścieżka audytu na poziomie systemu powinna przechwytywać takie dane jak próby logowania, zarówno udane, jak i niepomyślne, używane urządzenia i wykonywane funkcje systemu operacyjnego. Dla audytu mogą być interesujące również inne funkcje na poziomie systemu — na przykład działanie systemu oraz wskaźniki wydajności sieci.

Przykład ścieżki audytu na poziomie systemu dla systemu operacyjnego UNIX zaprezentowano na rysunku 18.4a, pochodzącym ze standardu NIST SP 800-12 (*An Introduction to Computer Security: The NIST Handbook* z października 1995). Polecenie `shutdown` kończy wszystkie procesy i przełącza system do trybu pojedynczego użytkownika. Polecenie `su` tworzy powłokę UNIX.

Ścieżki audytu na poziomie aplikacji

Mogą służyć do wykrywania naruszeń bezpieczeństwa w aplikacji lub do wykrywania luk w interakcji aplikacji z systemem. W przypadku aplikacji o kluczowym znaczeniu lub takich, które dotyczą wrażliwych danych, ścieżka audytu na poziomie aplikacji może zapewniać pożądany poziom szczegółowości pozwalający na ocenę zagrożeń i skutków dla bezpieczeństwa. Na przykład w przypadku aplikacji e-mail ścieżka audytu może rejestrować nadawcę i odbiorcę, rozmiar wiadomości i typy załączników. Ścieżka audytu dla interakcji z bazą danych za pomocą zapytań SQL (ang. *Structured Query Language*) może rejestrować użytkownika, typ transakcji, a nawet poszczególne tabele, wiersze, kolumny lub elementy danych, do których uzyskano dostęp.

```
Jan 27 17:14:04 host1 login: ROOT LOGIN console
Jan 27 17:15:04 host1 shutdown: reboot by root
Jan 27 17:18:38 host1 login: ROOT LOGIN console
Jan 27 17:19:37 host1 login: rebooted by root
Jan 28 09:46:53 host1 su: 'su root' succeeded for user1 on /dev/ttyp0
Jan 28 09:47:35 host1 shutdown: reboot by user1
Jan 28 09:53:24 host1 su: 'su root' succeeded for user1 on /dev/ttyp1
Feb 12 08:53:22 host1 su: 'su root' succeeded for user1 on /dev/ttyp1
Feb 17 08:57:50 host1 date: set by user1
Feb 17 13:22:52 host1 su: 'su root' succeeded for user1 on /dev/ttyp0
```

(a) Przykładowy plik logu systemowego zawierający komunikaty uwierzytelniania

```
Apr 9 11:20:22 host1 AA06370: from=<user2@host2>, size=3355, class=0
Apr 9 11:20:22 host1 AA06370: to=<user1@host1>, delay=00:00:02,stat=Sent
Apr 9 11:59:51 host1 AA06436: from=<user4@host3>, size=1424, class=0
Apr 9 11:59:52 host1 AA06436: to=<user1@host1>, delay=00:00:02, stat=Sent
Apr 9 12:43:52 host1 AA06441: from=<user2@host2>, size=2077, class=0
Apr 9 12:43:53 host1 AA06441: to=<user1@host1>, delay=00:00:01, stat=Sent
```

(b) Zapis audytu na poziomie aplikacji dla systemu dostarczania poczty

```
rcp user1 ttyp0 0.02 secs Fri Apr 8 16:02
ls user1 ttyp0 0.14 secs Fri Apr 8 16:01
clear user1 ttyp0 0.05 secs Fri Apr 8 16:01
rpcinfo user1 ttyp0 0.20 secs Fri Apr 8 16:01
nroff user2 ttyp2 0.75 secs Fri Apr 8 16:00
sh user2 ttyp2 0.02 secs Fri Apr 8 16:00
mv user2 ttyp2 0.02 secs Fri Apr 8 16:00
sh user2 ttyp2 0.03 secs Fri Apr 8 16:00
col user2 ttyp2 0.09 secs Fri Apr 8 16:00
man user2 ttyp2 0.14 secs Fri Apr 8 15:57
```

(c) Log pokazujący chronologiczną listę poleceń wykonywanych przez użytkowników

Rysunek 18.4. **Przykłady ścieżek audytu**

Na rysunku 18.4b zaprezentowano przykład ścieżki audytu na poziomie aplikacji dla systemu dostarczania poczty.

ŚCIEŻKI AUDYTU NA POZIOMIE UŻYTKOWNIKA

Zawierają rejestr aktywności pojedynczych użytkowników na przestrzeni czasu. Mogą być wykorzystywane do pociągania użytkowników do odpowiedzialności za ich działania. Takie ścieżki audytu są również użyteczne jako dane wejściowe do programów analitycznych, które potrafią zestawić zachowania normalne z zachowaniami odbiegającymi od normy.

Ścieżka audytu na poziomie użytkownika może służyć do rejestrowania interakcji użytkownika z systemem — na przykład wydawanych poleceń, prób identyfikacji i uwierzytelniania, a także plików i zasobów, do których użytkownik uzyskał dostęp. W ścieżce audytu mogą również być przechowywane informacje o aplikacjach wykorzystywanych przez użytkownika.

Przykład ścieżki audytu na poziomie użytkownika w systemie UNIX zaprezentowano na rysunku 18.4c.

Ścieżki audytu dostępu fizycznego

Mogą być generowane przez sprzęt zarządzający fizycznym dostępem. Następnie uzyskane dane są przesyłane do centralnego hosta w celu dalszego przechowywania i analizy. Przykładami są systemy kart kluczy i systemy alarmowe. W standardzie NIST SP 800-12 zaprezentowano następujące przykłady danych, które mogą być przedmiotem zainteresowania audytów dostępu fizycznego:

- Data i godzina, gdy próbowano uzyskać dostęp lub go uzyskano; ponadto powinny być zarejestrowane dane bramy lub drzwi, przez które próbowano uzyskać dostęp, oraz personalia osoby (lub identyfikator użytkownika) podejmującej próbę uzyskania dostępu do bramy lub drzwi.
- Nieudane próby dostępu powinny być monitorowane i logowane w niekomputerowych ścieżkach audytu, podobnie jak w przypadku ścieżek audytu systemów komputerowych. Jeśli ktoś próbuje uzyskać dostęp w niedozwolonych godzinach, należy o tym fakcie poinformować kierownictwo.
- W rejestrowanych danych powinny się także znaleźć próby dodawania, modyfikowania lub usuwania fizycznych uprawnień dostępu (np. przyznanie nowemu pracownikowi dostępu do budynku lub przyznanie przeniesionego pracownikowi dostępu do nowego biura [i, co oczywiste, usunięcie jego starych uprawnień dostępu, stosownie do przypadku]).
- Podobnie jak jest w przypadku ścieżek audytu systemów i aplikacji, mechanizmy audytu niekomputerowych funkcji mogą być zaimplementowane tak, aby wysyłały komunikaty do personelu ochrony, wskazując na udane lub nieudane próby zdobycia dostępu do kontrolowanych przestrzeni. Aby nie obniżać czujności strażników lub osób monitorujących, zdarzenia dostępu nie powinny powodować wysyłania komunikatów na ekran ich monitorów. Osoby monitorujące powinny być informowane wyłącznie o sytuacjach wyjątkowych, na przykład o nieudanych próbach dostępu.

Ochrona danych z audytu

W dokumencie RFC 2196 (*Site Security Handbook* z 1997 roku) wymieniono trzy alternatywne sposoby przechowywania zapisów z audytu:

- plik na hoście do odczytu i zapisu;
- urządzenie do jednokrotnego zapisu i wielokrotnego odczytu (np. CD-ROM lub DVD-ROM);
- urządzenie wyłącznie do zapisu (np. drukarka).

Rejestrowanie w systemie plików jest stosunkowo łatwe do skonfigurowania i wymaga najmniej zasobów. Dostęp do zapisów jest natychmiastowy, co może się przydać jako środek przeciwdziałania trwającemu atakowi. Jednak stosowanie takiego podejścia stwarza wiele zagrożeń. Jeśli napastnik uzyska uprzywilejowany dostęp do systemu, ścieżka audytu może zostać zmodyfikowana lub usunięta.

Nagrywanie na płyty DVD-ROM lub stosowanie podobnej metody przechowywania jest znacznie bezpieczniejsze, ale mniej wygodne. Konieczne jest ciągłe uzupełnianie nagrywalnych nośników. Dostęp do danych może być opóźniony.

Logi drukowane dostarczają papierowej wersji ścieżki audytu, ale są niepraktyczne dla celów przechowywania szczegółowych danych audytu w rozbudowanych systemach lub systemach sieciowych. Dokument RFC 2196 sugeruje, że papierowy log może się przydać, gdy stały, dostępny natychmiast dziennik jest wymagany, nawet w przypadku awarii systemu.

Ochrona ścieżki audytu obejmuje zarówno integralność, jak i poufność. Szczególnie ważna jest integralność, ponieważ intruz może próbować usunąć dowody włamania poprzez modyfikację ścieżki audytu. W przypadku logów w systemie plików prawdopodobnie najlepszym sposobem zapewnienia integralności są podpisy cyfrowe. Integralność automatycznie zapewniają urządzenia jednokrotnego zapisu — na przykład płyty DVD-ROM lub papier. Kolejnym środkiem zapewniającym integralność są silne mechanizmy kontroli dostępu.

Poufność jest ważna w przypadku, gdy ścieżka audytu zawiera informacje użytkowników, które są poufne i nie powinny być ujawniane wszystkim użytkownikom, na przykład informacje o zmianach wynagrodzenia lub grupy wynagrodzenia. Do ochrony tego rodzaju informacji można wykorzystać mechanizmy silnej kontroli dostępu. Skutecznym środkiem jest szyfrowanie symetryczne (np. z wykorzystaniem algorytmu AES [ang. *Advanced Encryption Standard*] lub potrójnego algorytmu DES [ang. *Data Encryption Standard*]). Klucz tajny musi być chroniony i dostępny tylko dla oprogramowania audytowego oraz pomocniczego oprogramowania do analizy audytu.

Należy zwrócić uwagę, że mechanizmy integralności i poufności chronią dane ścieżki audytu nie tylko w pamięci lokalnej, ale również podczas przesyłania do centralnego repozytorium.

18.3. IMPLEMENTACJA FUNKCJI LOGOWANIA

Podstawę do stworzenia mechanizmu audytu bezpieczeństwa stanowi początkowe pobranie danych audytu. Wymaga to, aby oprogramowanie zawierało haki lub punkty przechwytu, które wyzwalają gromadzenie i zapisywanie danych w odpowiedzi na wybrane wcześniej zdarzenia. Taka funkcja gromadzenia lub rejestrowania zależy od rodzaju oprogramowania i może być różna w zależności od wykorzystywanego systemu operacyjnego i aplikacji. W tym podrozdziale przyjrzymy się sposobom podejścia do implementacji funkcji logowania dla ścieżek audytu na poziomie systemu i na poziomie użytkownika oraz ścieżek audytu na poziomie aplikacji.

Logowanie na poziomie systemu

Znaczna część zadań logowania na poziomie systemu może być zaimplementowana przy użyciu istniejących urządzeń, które są częścią systemu operacyjnego. W tym punkcie pokrótce przyjrzymy się mechanizmowi logowania w systemie operacyjnym Windows, a następnie opiszemy mechanizm *syslog* w systemach operacyjnych z rodziny UNIX.

Dziennik zdarzeń systemu Windows

Zdarzenie w dzienniku zdarzeń systemu Windows to jednostka opisująca jakieś interesujące zdarzenia w systemie komputerowym. Zdarzenia zawierają liczbowy kod identyfikacyjny, zestaw atrybutów (zadanie, kod operacyjny, poziom szczegółowości, wersja i słowa kluczowe) oraz opcjonalne dane dostarczone przez użytkownika. W systemie Windows dostępne są trzy typy logów zdarzeń:

- **Log zdarzeń systemowych** — używany przez aplikacje działające z tożsamością kont usług systemowych (zainstalowanych usług systemowy), sterowniki, komponenty lub aplikacje, które posługują się zdarzeniami związanymi z kondycją systemu komputerowego.
- **Log zdarzeń aplikacji** — zdarzenia dla wszystkich aplikacji na poziomie użytkownika. Ten dziennik nie jest zabezpieczony i jest otwarty dla wszystkich aplikacji. Aplikacje, które logują obszerny zbiór informacji, powinny definiować dziennik specyficzny dla aplikacji.
- **Log zdarzeń bezpieczeństwa** — log audytu systemu Windows. Ten log zdarzeń służy wyłącznie do lokalnego użytku mechanizmu *Windows Local Security Authority*. W danych audytu mogą pojawiać się zdarzenia użytkownika tylko wtedy, gdy są obsługiwane przez wskazaną aplikację.

W przypadku wszystkich logów zdarzeń lub ścieżek audytu informacje o zdarzeniach mogą być przechowywane w formacie XML. Listę informacji przechowywanych dla każdego zdarzenia zamieszczono w tabeli 18.4. Na rysunku 18.5 pokazano przykład danych eksportowanych z logu zdarzeń systemu Windows.

Tabela 18.4. Elementy schematu zdarzeń systemu Windows

Wartości właściwości zdarzenia zawierające dane binarne	Diagnostyczne pole `LevelName` preprocesora WPP systemu Windows wykorzystywane w zdarzeniach debugowania w kanałach debugowania
Dane binarne dostarczane przez log zdarzeń systemu Windows	Poziom renderowany dla zdarzenia
Kanał, na którym zostanie opublikowane renderowane zdarzenie	Stopień istotności zdarzenia
Złożone dane dla parametru dostarczanego przez dostawcę zdarzenia	Diagnostyczne pole `FormattedString` preprocesora WPP systemu Windows wykorzystywane w zdarzeniach debugowania w kanałach debugowania

Tabela 18.4. **Elementy schematu zdarzeń systemu Windows — *ciąg dalszy***

Diagnostyczne pole `ComponentName` preprocesora WPP używane w zdarzeniach debugowania	Komunikat renderowany dla zdarzenia
Komputer, na którym miało miejsce zdarzenie	Opkod, który zostanie wyrenderowany dla zdarzenia
Dwie 128-bitowe wartości, które można wykorzystać do wyszukiwania powiązanych zdarzeń	Działanie lub moment podczas działania, które było wykonywane w aplikacji w czasie, gdy zaistniało zdarzenie
Nazwa elementu danych zdarzenia, który spowodował błąd podczas przetwarzania danych zdarzenia	Elementy, które definiują zdarzenia instrumentacji
Dane składające się na jedną część złożonego typu danych dostarczoną przez dostawcę zdarzenia	Informacje o dostawcy, który opublikował zdarzenie
Dane dla parametru dostarczonego przez dostawcę zdarzenia	Wydawca zdarzenia, który opublikował renderowane zdarzenie
Wartości właściwości zdarzeń preprocesora WPP systemu Windows	Informacje, które będą renderowane dla zdarzenia
Kod błędu, który został zgłoszony, gdy wystąpił błąd przetwarzania danych zdarzenia	Identyfikator bezpieczeństwa użytkownika
Strukturalna informacja opisująca jakieś interesujące zdarzenie w systemie	Pole `SequenceNum` preprocesora WPP systemu Windows używane w zdarzeniach debugowania w kanałach debugowania
Numer identyfikacyjny zdarzenia	Pole `SubComponentName` preprocesora WPP systemu Windows używane w zdarzeniach debugowania w kanałach debugowania
Informacje o procesie i wątku, w którym wystąpiło zdarzenie	Informacje automatycznie wypełniane przez system po wystąpieniu zdarzenia lub w momencie jego zapisywania do pliku logu
Binarne dane zdarzenia, które spowodowało błąd podczas przetwarzania danych związanych ze zdarzeniem	Zadanie, które zostanie wyrenderowane dla zdarzenia
Informacje o procesie i wątku, w których wystąpiło zdarzenie	Zadanie z wartością symboliczną
Pole `FileLine` preprocesora WPP systemu Windows używane w zdarzeniach debugowania w kanałach debugowania	Informacje o czasie zdarzenia
Pole `FlagsName` preprocesora WPP systemu Windows używane w zdarzeniach debugowania w kanałach debugowania	Zdefiniowana przez dostawcę część, która może składać się z dowolnej prawidłowej zawartości XML komunikującej informacje o zdarzeniu
Pole `KernelTime` preprocesora WPP systemu Windows używane w zdarzeniach debugowania w kanałach debugowania	Pole `UserTime` preprocesora WPP systemu Windows używane w zdarzeniach debugowania w kanałach debugowania
Słowa kluczowe, które zostaną wyrenderowane dla zdarzenia	Wersja zdarzenia
Słowa kluczowe używane przez zdarzenie	

```
Event Type:       Success Audit
Event Source:     Security
Event Category:   (1)
Event ID:         517
Date:             3/6/2006
Time:             2:56:40 PM
User:             NT AUTHORITY[[backslash]]SYSTEM
Computer:         KENT
Description:      The audit log was cleared
Primary User Name:     SYSTEM          Primary Domain:       NT AUTHORITY
Primary Logon ID:      (0x0,0x3F7)     Client User Name:     userk
Client Domain:         KENT            Client Logon ID:      (0x0,0x28BFD)
```

Rysunek 18.5. **Przykład wpisu w logu systemu Windows**

System Windows pozwala swym użytkownikom rejestrować dane audytu w dziewięciu różnych kategoriach:

- **Zdarzenia logowania do kont** — działania związane z uwierzytelnianiem użytkownika z punktu widzenia systemu, który zweryfikował próbę. Przykłady: przyznano autoryzację; żądanie żetonu uwierzytelniającego nie powiodło się; zmapowano konto do logowania; nie można zmapować konta logowania. Pojedyncze działania w tej kategorii nie dostarczają zbyt wielu danych, ale duża liczba niepowodzeń może wskazywać na trwające skanowanie portów, ataki siłowe na indywidualne konta lub rozprzestrzenianie się automatycznych eksploitów.
- **Zarządzanie kontami** — działania administracyjne związana z tworzeniem, zarządzaniem i usuwaniem indywidualnych kont i grup użytkowników. Przykłady: utworzono konto użytkownika; próba zmiany hasła; usunięto konto użytkownika; dodano członka globalnej grupy bezpieczeństwa; zmieniono zasady domeny.
- **Dostęp do usług katalogowych** — dostęp na poziomie użytkownika do dowolnego obiektu usługi Active Directory, dla którego zdefiniowano systemową listę kontroli dostępu (SACL). Lista SACL zawiera zbiór użytkowników i grup użytkowników, dla których wymagany jest szczegółowy audyt.
- **Zdarzenia logowania** — działania związane z uwierzytelnianiem użytkownika, zarówno na komputerze lokalnym, jak i przez sieć, z poziomu systemu, z którego zainicjowano działanie. Przykłady: pomyślne logowanie użytkownika; błąd logowania, nieznana nazwa użytkownika lub złe hasło; błąd logowania ze względu na wyłączenie konta; błąd logowania z powodu utraty ważności konta, użytkownik nieuprawniony do logowania się na tym komputerze; wylogowanie użytkownika; błąd logowania, konto zablokowane.
- **Dostęp do obiektów** — dostęp na poziomie użytkownika do systemu plików i obiektów rejestru, które mają zdefiniowane systemowe listy kontroli dostępu (SACL). Zapewnia stosunkowo łatwy oraz zintegrowany z systemem operacyjnym sposób śledzenia dostępu do odczytu, jak również zmian we wrażliwych plikach. Przykłady: obiekt otwarty; obiekt usunięty.

- **Zmiany zasad** — zmiany administracyjne w zasadach dostępu, konfiguracji audytu i innych ustawieniach na poziomie systemu. Przykłady: przypisano uprawnienia użytkownika; nowa zaufana domena; zmodyfikowano politykę audytu.
- **Korzystanie z uprawnień** — w systemie Windows istnieje pojęcie uprawnień użytkownika — szczegółowego zezwolenia na wykonywanie określonego zadania. Jeśli włączymy audyt korzystania z uprawnień, rejestrowane będą wszystkie przypadki dostępu użytkowników do określonych funkcji systemowych (tworzenie obiektów, debugowanie kodu wykonywalnego lub tworzenie kopii zapasowej systemu). Przykłady: dodano określone uprawnienia do tokenu dostępowego użytkownika (podczas logowania); użytkownik próbował wykonać uprzywilejowane działania usługi systemowej.
- **Śledzenie procesu** — generuje szczegółowe informacje audytowe w momencie rozpoczynania i zakańczania procesów, aktywowania programów lub pośredniego dostępu do obiektów. Przykłady: utworzono nowy proces; proces zakończył się; dane podlegające audytowi zostały zabezpieczone; dla danych podlegających audytowi usunięto zabezpieczenia; użytkownik próbował zainstalować usługę.
- **Zdarzenia systemowe** — rejestruje informacje o zdarzeniach, które wpływają na dostępność i integralność systemu, w tym komunikaty podczas startu systemu oraz komunikat o zamknięciu systemu. Przykłady: system się uruchamia; system zamyka się; wyczerpanie się zasobów w podsystemie logowania; niektóre zapisy audyty zostały utracone; wyczyszczono log audytu.

Syslog

Syslog to uniwersalny mechanizm logowania dostępny we wszystkich wersjach systemów UNIX i Linux. Składa się z następujących elementów:

- **`syslog()`** — interfejs API, do którego odwołuje się kilka standardowych narzędzi systemowych i który jest dostępny dla aplikacji.
- **`logger`** — uniksowe polecenie służące do dodawania pojedynczych zapisów do loga systemowego.
- ***/etc/syslog.conf*** — plik konfiguracyjny służący do kontroli logowania i routowania zdarzeń logu systemowego.
- **`syslogd`** — demon systemowy służący do odbierania i routowania zdarzeń logu systemowego za pomocą wywołań `syslog()` i `logger`.

W różnych implementacjach Uniksa dostępne są różne warianty mechanizmu `syslog`. Nie istnieje jednolity format logów obowiązujący dla wszystkich systemów. Mechanizm `syslog` dla systemu Linux przeanalizowano w rozdziale 25. W niniejszym rozdziale zaprezentujemy krótki przegląd niektórych funkcji związanych z mechanizmem `syslog` i przyjrzymy się protokołowi `syslog`.

Podstawową usługą mechanizmu syslog w Uniksie jest możliwość przechwytywania wskazanych zdarzeń, mechanizm pamięci masowej oraz protokół przesyłania komunikatów z innych maszyn do centralnego komputera, spełniający rolę serwera syslog. Oprócz tych podstawowych funkcji dostępne są również inne usługi, często jako pakiety zewnętrzne, a w niektórych przypadkach jako wbudowane moduły. Standard NIST SP 800-92 (*Guide to Computer Security Log Management*, z września 2006 r.) wyszczególnia następujące najczęstsze funkcje dodatkowe:

- **Rozbudowane filtrowanie.** Podstawowe implementacje mechanizmu syslog pozwalają na specyficzną obsługę wiadomości tylko w zależności od ich obiektu i priorytetu; nie pozwalają na dokładniejsze filtrowanie. Niektóre współczesne implementacje syslog oferują bardziej rozbudowane funkcje filtrowania, takie jak spersonalizowana obsługa komunikatów w zależności od hosta lub programu, który wygenerował komunikat, albo wyrażenia regularnego pasującego do zawartości wiadomości. Niektóre implementacje umożliwiają również zastosowanie wielu filtrów do pojedynczego komunikatu, co zapewnia bardziej złożone filtrowanie.
- **Analiza logów.** Początkowo serwery syslog nie wykonywały żadnej analizy danych logów. Po prostu tworzyły framework danych logowania do rejestrowania i transmisji. Do analizy danych syslog administratorzy mogą wykorzystywać oddzielne programy dodatkowe. Niektóre implementacje mechanizmu syslog mają wbudowane ograniczone funkcjonalności analizy logów, takie jak zdolność korelowania wielu wpisów w logu.
- **Reakcja na zdarzenie.** Niektóre implementacje syslog mogą inicjować działania w odpowiedzi na wystąpienie określonych zdarzeń. Przykłady działań obejmują wysyłanie pułapek SNMP, ostrzeganie administratorów za pośrednictwem stron lub wiadomości e-mail oraz uruchomienie oddzielnego programu lub skryptu. Możliwe jest również utworzenie nowego komunikatu syslog, który wskazuje wykrycie określonego zdarzenia.
- **Alternatywne formaty komunikatów.** Niektóre implementacje syslog mogą akceptować dane w formatach innych niż syslog, na przykład jako pułapki SNMP. Może się to przydać do uzyskania danych zdarzeń bezpieczeństwa z hostów, które nie obsługują mechanizmu syslog i nie można tego zmienić.
- **Szyfrowanie plików logów.** Niektóre implementacje syslog można skonfigurować tak, aby automatycznie szyfrowały pliki logów poddawane rotacji, co chroni ich poufność. Można to również osiągnąć za pomocą programów do szyfrowania dostępnych w systemie operacyjnym lub programów zewnętrznych.
- **Przechowywanie logów w bazie danych.** W niektórych implementacjach wpisy w logach mogą być przechowywane zarówno w tradycyjnych plikach syslog, jak i w bazie danych. Przechowywanie wpisów logów w formacie bazy danych może być bardzo pomocne do późniejszej analizy.

- **Ograniczanie tempa zapisu.** Niektóre implementacje pozwalają ograniczać liczbę komunikatów syslog lub połączeń TCP z określonego źródła w podanym przedziale czasu. Jest to przydatne w zapobieganiu atakom DoS przeciwko serwerowi syslog, co mogłoby prowadzić do utraty komunikatów syslog z innych źródeł. Ponieważ technika ta służy do celowego pomijania wiadomości ze źródła, które przytłacza serwer syslog, to podczas niekorzystnego zdarzenia, które generuje niezwykle dużą liczbę komunikatów, może prowadzić do utraty niektórych danych logu.

Protokół syslog zapewnia transport umożliwiający wysyłanie w sieciach IP komunikatów z powiadomieniami o zdarzeniach do mechanizmów zbierania komunikatów o zdarzeniach — znanych również jako serwery syslog. W ramach systemu możemy przeglądać proces przechwytywania i rejestrowania zdarzeń w kontekście różnych aplikacji i obiektów systemowych wysyłających komunikaty do demona *syslogd* w celu ich zapisania w logu systemowym. Ponieważ każdy proces, aplikacja i implementacja systemu operacyjnego Unix może stosować inną konwencję formatowania dla rejestrowanych zdarzeń, protokół syslog udostępnia tylko bardzo ogólny format komunikatu do transmisji między systemami. Popularną wersję protokołu syslog bazującą na protokole TCP/IP pierwotnie opracowano dla dystrybucji BSD stworzonej na Uniwersytecie Kalifornijskim w Berkeley) systemu Unix. Wersję tę udokumentowano w dokumencie RFC 3164 (*The BSD Syslog Protocol* z 2001 roku). Następnie organizacja IETF opublikowała RFC 5424 (*The Syslog Protocol*, 2009), który ma być standardem internetowym i który różni się pewnymi szczegółami od wersji BSD. W dalszej części tego rozdziału opiszemy wersję BSD.

Komunikaty w formacie syslog dystrybucji BSD składają się z trzech części:

- **PRI** — składa się z kodu reprezentującego opisane później wartości komunikatu: funkcje (ang. *facilities*) i istotność (ang. *severity*).
- **Nagłówek** — zawiera znacznik czasu i informacje o nazwie hosta lub adresie IP urządzenia.
- **Msg** — składa się z dwóch pól: pole `TAG` to nazwa programu lub procesu, który wygenerował komunikat; `CONTENT` zawiera szczegóły wiadomości. Część `Msg` tradycyjnie była dowolnym komunikatem złożonym z drukowalnych znaków, który dostarcza pewnych, szczegółowych informacji o zdarzeniu.

Kilka przykładów komunikatów *syslog*, z których wyłączono fragment PRI, pokazano na rysunku 18.6.

Wszystkie wiadomości wysyłane do demona `syslogd` zawierają informacje o funkcji i istotności (patrz tabela 18.5). Funkcja identyfikuje aplikację lub komponent systemu, który wygenerował komunikat.

```
Mar 1 06:25:43 server1 sshd[23170]: Accepted publickey for server2 from
172.30.128.115 port 21011 ssh2
Mar 1 07:16:42 server1 sshd[9326]: Accepted password for murugiah from
10.20.30.108 port 1070 ssh2
Mar 1 07:16:53 server1 sshd[22938]: reverse mapping checking getaddrinfo
for ip10.165.nist.gov failed - POSSIBLE BREAKIN ATTEMPT!
Mar 1 07:26:28 server1 sshd[22572]: Accepted publickey for server2 from
172.30.128.115 port 30606 ssh2
Mar 1 07:28:33 server1 su: BAD SU kkent to root on /dev/ttyp2
Mar 1 07:28:41 server1 su: kkent to root on /dev/ttyp2
```

Rysunek 18.6. **Przykłady komunikatów syslog**

Tabela 18.5. **Funkcje i poziomy istotności uniksowego mechanizmu Syslog**

(a) Funkcje Syslog

Funkcja	Opis komunikatu (wygenerowany przez)
`kern`	jądro systemu
`user`	proces użytkownika
`mail`	system e-mail
`daemon`	demon systemowy, na przykład `ftpd`
`auth`	programy autoryzacyjne `login`, `su` i `getty`
`Syslogd`	komunikaty generowane wewnętrznie przez demona syslogd
`lpr`	system drukowania
`news`	system News UseNet
`uucp`	podsystem UUCP
`clock`	demon zegara
`ftp`	demon FTP
`ntp`	podsystem NTP
`log audit`	zarezerwowany do użytku systemowego
`log alert`	zarezerwowany do użytku systemowego
`local0` – `local7`	do 8 zdefiniowanych lokalnie kategorii

(b) poziomy istotności Syslog

Istotność	Opis
`emerg`	Najbardziej istotne komunikaty, na przykład natychmiastowe zamknięcie systemu
`alert`	Sytuacje w systemie wymagające natychmiastowej uwagi
`crit`	Sytuacje o znaczeniu krytycznym, takie jak awaria sprzętu lub oprogramowania
`err`	Inne błędy systemowe; łatwe do naprawy
`warning`	Komunikaty ostrzegawcze
`notice`	Sytuacja nadzwyczajna, która zasługuje na zbadanie; znaczące zdarzenie, które zwykle jest częścią normalnej, codziennej pracy
`info`	Komunikaty informacyjne
`debug`	Komunikaty do celów diagnostycznych

Istotność lub poziom komunikatu wskazuje względną ważność komunikatu. Można jej użyć do podstawowego filtrowania.

Logowanie na poziomie aplikacji

Aplikacje, szczególnie te wymagające określonego poziomu uprawnień, stwarzają problemy bezpieczeństwa, które mogą nie zostać przechwycone przez dane audytu na poziomie systemu lub użytkownika. Słabe punkty aplikacji stanowią duży procent luk w zabezpieczeniach zgłaszanych na listach mailingowych. Rodzajem luki, który może być wykorzystany przez złośliwe oprogramowanie, jest brak dynamicznej kontroli danych wprowadzanych przez użytkownika, co umożliwia przepełnienie bufora (patrz rozdział 10). Przyczyną innych luk w zabezpieczeniach mogą być także błędy w logice aplikacji. Na przykład aplikacja z uprzywilejowanymi uprawnieniami może odczytywać i drukować określony plik. Błąd w aplikacji może pozwolić napastnikowi na skorzystanie z nieoczekiwanej interakcji ze środowiskiem powłoki, aby zmusić aplikację do odczytu i wydrukowania innego pliku, co skutkuje naruszeniem bezpieczeństwa.

Audyt na poziomie systemu nie zapewnia szczegółowości pozwalającej na wychwycenie błędu w logice aplikacji. Ponadto systemy wykrywania intruzów szukają sygnatur ataków lub nietypowych zachowań, które w przypadku ataków bazujących na błędach w logice aplikacji nie pojawią się. Zarówno z punktu widzenia możliwości wykrycia, jak i audytu może być konieczne szczegółowe zbadanie zachowania aplikacji, wykraczające poza jej dostęp do usług systemowych i systemów plików. Informacje potrzebne do wykrycia ataków na poziomie aplikacji mogą być niedostępne lub zbyt trudne do wydobycia z niskopoziomowych informacji zawartych w śladach wywołań systemowych oraz w rekordach audytu wygenerowanych przez system operacyjny.

W dalszej części tego podrozdziału zbadamy dwa podejścia do zbierania danych audytu z aplikacji: biblioteki interpozycyjne (ang. *interposable libraries*) i dynamiczne przepisywanie binarne (ang. *dynamic binary rewriting*).

Biblioteki interpozycyjne

Technika opisana w [KUPE99] i [KUPE04] zapewnia audyt na poziomie aplikacji poprzez tworzenie nowych procedur, które przechwytują wywołania funkcji bibliotek współdzielonych w celu oprzyrządowania działania. Interpozycja umożliwia generowanie danych audytu bez konieczności ponownej kompilacji bibliotek systemowych lub aplikacji będącej przedmiotem audytu. Dzięki temu dane audytu mogą być generowane bez modyfikacji systemowych bibliotek współdzielonych lub konieczności dostępu do kodu źródłowego pliku wykonywalnego, w którym ma być wykonana interpozycja. Takie podejście może być zastosowane w dowolnym systemie UNIX lub Linux oraz w niektórych innych systemach operacyjnych.

Technika wykorzystuje mechanizm dynamicznych bibliotek w systemie UNIX. Przed przeanalizowaniem tej techniki zaprezentujemy krótkie wprowadzenie w tematykę bibliotek współdzielonych.

BIBLIOTEKI WSPÓŁDZIELONE

System operacyjny zawiera setki funkcji bibliotecznych języka C zapisanych w archiwach bibliotek. Każda biblioteka składa się ze zbioru zmiennych i funkcji, które są skompilowane i połączone ze sobą. Funkcja łączenia rozwiązuje wszystkie odwołania do pamięci w celu sięgania do danych i kodu programu w ramach biblioteki i generuje adresy logiczne lub adresy względne. Na żądanie, podczas kompilacji, funkcja może być połączona z programem wykonywalnym. Jeśli funkcja nie jest częścią kodu programu, to konsolidator przeszukuje listę bibliotek i łączy pożądany obiekt z docelowym plikiem wykonywalnym. Podczas ładowania programu do pamięci wirtualnej ładowana jest oddzielna kopia dołączanej funkcji bibliotecznej. Ten schemat nazywa się **statycznym łączeniem bibliotek**.

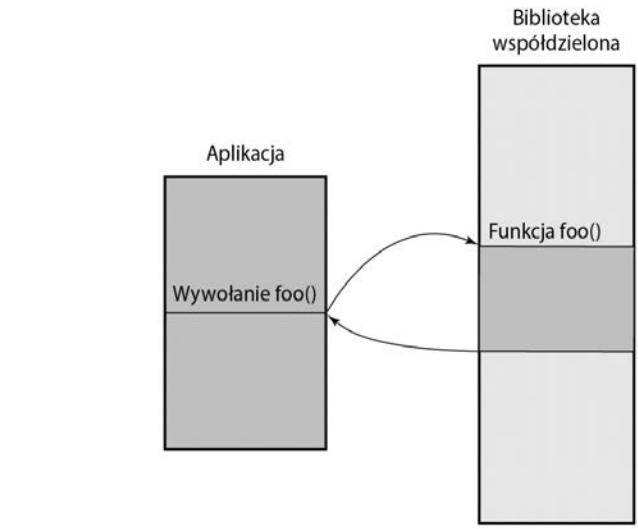
Bardziej elastyczny schemat, wprowadzony po raz pierwszy w wersji 3 systemu UNIX System V, polega na wykorzystaniu **statycznie połączonych bibliotek współdzielonych**. Podobnie jak w przypadku bibliotek łączonych statycznie, obiekt współdzielony jest dołączany do docelowego pliku wykonywalnego przez konsolidator. Jednak każdemu obiektowi w dołączanej statycznie bibliotece współdzielonej jest przypisywany stały adres wirtualny. Podczas tworzenia pliku wykonywalnego konsolidator łączy obiekty zewnętrzne z ich definicjami w bibliotece, podstawiając ich adresy wirtualne. Dzięki temu istnieje tylko jedna kopia każdej funkcji bibliotecznej. Ponadto funkcja może być modyfikowana bez zmiany jej adresu wirtualnego. W celu wprowadzenia modyfikacji skompilowany musi być tylko modyfikowany obiekt, natomiast nie muszą być kompilowane programy wykonywalne, które się do niego odwołują. Jednak ogólnie rzecz biorąc, modyfikacje mogą być drobne. Zmiany muszą być wprowadzane tak, aby adres początkowy oraz adresy zmiennych, stałych lub etykiet programów w kodzie nie zostały zmienione.

W UNIX System V Release 4 wprowadzono koncepcję **bibliotek współdzielonych dołączanych dynamicznie**. W przypadku bibliotek dołączanych dynamicznie połączenie z bibliotekami współdzielonymi jest opóźniane do czasu ładowania programu. W tym momencie żądana zawartość biblioteki jest mapowana na wirtualną przestrzeń adresową procesu. W związku z tym, jeśli do biblioteki zostaną wprowadzone zmiany przed załadowaniem programu, nie będzie to miało wpływu na żaden z programów korzystających z biblioteki.

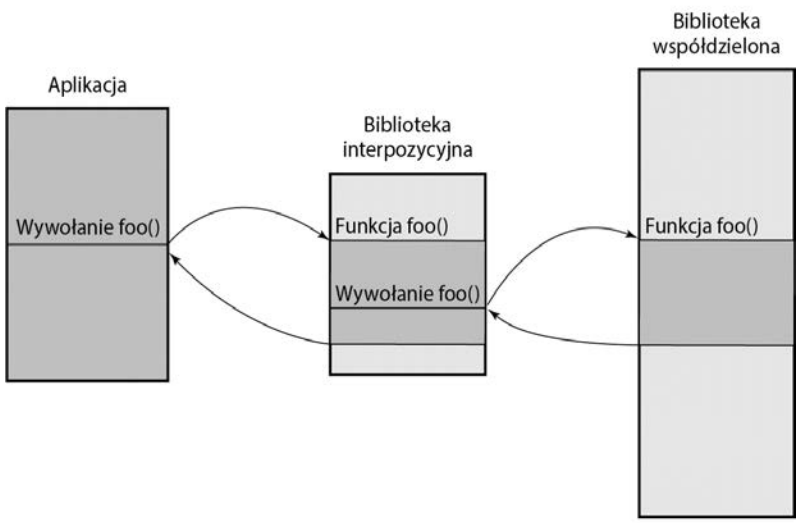
W przypadku bibliotek współdzielonych zarówno dołączanych statycznie, jak i dynamicznie, współdzielone strony pamięci muszą być oznaczone jako tylko do odczytu. Jeśli program wykonuje aktualizację pamięci na współdzielonej stronie, system używa schematu kopiowania przy zapisie: System przydziela procesowi kopię strony, która może być modyfikowana bez wpływu na innych użytkowników strony.

UŻYWANIE BIBLIOTEK INTERPOZYCYJNYCH

Normalny tryb działania, gdy program wywołuje procedurę w bibliotekach współdzielonych dołączanych dynamicznie, pokazano na rysunku 18.7a. W czasie ładowania programu odwołanie do procedury `foo` w programie jest zastępowane adresem pamięci wirtualnej początku procedury `foo` w bibliotece współdzielonej.



(a) Normalna technika wywoływania funkcji z biblioteki



(b) Wywołanie biblioteki z interpozycją

Rysunek 18.7. **Korzystanie z bibliotek interpozycyjnych**

W przypadku stosowania bibliotek interpozycyjnych tworzona jest specjalna wersja biblioteki, zatem podczas ładowania program łączy się z biblioteką interpozycyjną zamiast z biblioteką współdzieloną. Dla każdej funkcji w bibliotece współdzielonej, dla której ma być wywołany audyt, występuje funkcja o takiej samej nazwie w bibliotece interpozycyjnej. Jeśli nie ma żądanej funkcji w bibliotece interpozycyjnej, program ładujący kontynuuje wyszukiwanie w bibliotece współdzielonej i łączy się bezpośrednio z funkcją docelową.

Moduł interpozycyjny może wykonać dowolną funkcję związaną z audytem, na przykład zarejestrować fakt wywołania, zapisać przekazane i zwrócone parametry, adres powrotu w programie wywołującym, i tak dalej. Zazwyczaj rzeczywistą współdzieloną funkcję wywołuje moduł interpozycyjny (patrz rysunek 18.7b). Dzięki temu zachowanie aplikacji nie ulega zmianie, a jedynie zostaje „oprzyrządowane".

Technika ta pozwala na przechwytywanie pewnych wywołań funkcji i przechowywanie stanu pomiędzy takimi wywołaniami bez konieczności ponownej kompilacji programu wywołującego lub obiektów współdzielonych.

Przykład interpozycyjnej funkcji bibliotecznej napisanej w C (patrz rysunek 18.8) zaprezentowano w [KUPE99]. Funkcję można opisać w następujący sposób:

1. Na początku każdej funkcji interpozycyjnej umieszczane jest wywołanie `AUDIT_CALL_START` (wiersz 8). Dzięki temu do każdej funkcji można łatwo wstawić dowolny kod inicjalizacji.
2. Makro `AUDIT_LOOKUP_COMMAND` (wiersz 10 na rysunku 18.8a, szczegóły na rysunku 18.8b) wykonuje wyszukiwanie wskaźnika do następnej definicji funkcji w bibliotekach współdzielonych za pomocą polecenia `dlsym(3x)`. Specjalna flaga `RTLD_NEXT` (patrz rysunek 18.8b, wiersz 2.) wskazuje, że zostanie zwrócona następna referencja ze ścieżki przeszukiwania bibliotek używanej przez moduł ładujący fazy wykonania. Jeśli zostanie znaleziona referencja, wskaźnik funkcji jest zapisywany w `fptr`. W przeciwnym razie do programu wywołującego zostaje zwrócona wartość błędu.
3. Wiersz 12 zawiera polecenia, które zostaną uruchomione przed wywołaniem funkcji.
4. W tym przypadku funkcja interpozycyjna wykonuje pierwotne wywołanie funkcji i zwraca wartość użytkownikowi (wiersz 14). Inne możliwe działania obejmują badanie, rejestrowanie lub przekształcanie argumentów; zapobieganie rzeczywistej realizacji wywołania funkcji bibliotecznej oraz zbadanie, zarejestrowanie lub przekształcenie wartości zwracanej.
5. Przed zwróceniem wyniku można wstawić dodatkowy kod (wiersz 16), ale w tym przykładzie nie wstawiono żadnego kodu.

```
1 /*************************************
2 * Rejestracja wykorzystania określonych funkcji *
3 *************************************/
4 char *strcpy(char *dst, const char *src) {
5     char *(*fptr)(char *,const char *); /* wskaźnik do rzeczywistej funkcji */
6     char *retval; /* wartość zwracana z wywołania */
7
8     AUDIT_CALL_START;
9
10    AUDIT_LOOKUP_COMMAND(char *(*)(char *,const char *),"strcpy",fptr,NULL);
11
12    AUDIT_USAGE_WARNING("strcpy");
13
14    retval=((*fptr)(dst,src));
15
16     return(retval);
17 }
```

(a) Definicja funkcji (elementy zapisane samymi wielkimi literami oznaczają makra zdefiniowane w innym miejscu)

```
1 #define AUDIT_LOOKUP_COMMAND(t,n,p,e)
2     p=(t)dlsym(RTLD_NEXT,n);
3     if (p==NULL) {
4         perror("wyszukiwanie polecenia");
5         syslog(LOG_INFO,"nie znaleziono %s w bibliotece: %m",n);
6         return(e);
7 }
```

(b) Makro używane w funkcji

Rysunek 18.8. **Przykład funkcji w bibliotece interpozycyjnej**

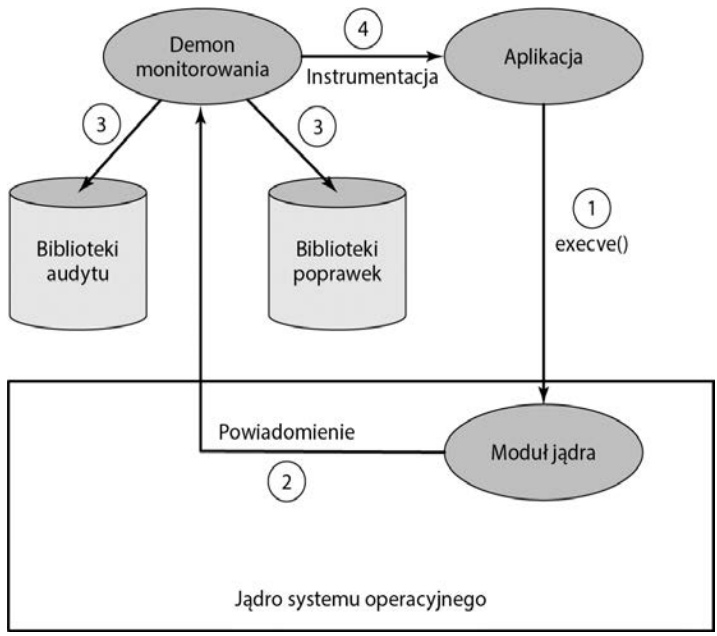
Dynamiczne przepisywanie binarne

Technika interpozycji została zaprojektowana do pracy z bibliotekami współdzielonymi dołączanymi dynamicznie. Nie można jej użyć do przechwytywania wywołań funkcji w programach łączonych statycznie, jeśli wszystkie programy w systemie w momencie wprowadzenia biblioteki nie zostaną powtórnie skonsolidowane. [ZHOU04] opisuje technikę, zwaną dynamicznym przepisywaniem binarnym, której można użyć zarówno dla programów łączonych statycznie, jak i dynamicznie.

Dynamiczne przepisywanie binarne to technika realizowana po kompilacji, która bezpośrednio modyfikuje binarny kod plików wykonywalnych. Zmiany są wprowadzane w czasie ładowania, a modyfikacji podlega tylko obraz programu w pamięci, a nie binarny plik programu w pamięci trwałej. Podobnie jak w technice interpozycji, dynamiczne przepisywanie binarne nie wymaga ponownej kompilacji pliku binarnego aplikacji. Wybór modułu realizującego audyt jest opóźniany do czasu wywołania aplikacji, co pozwala na elastyczny wybór konfiguracji audytu.

Technika ta została zaimplementowana w systemie Linux za pomocą dwóch modułów: ładowalnego modułu jądra i demona monitorowania. Linux ma strukturę kolekcji modułów, z których wiele można na żądanie ładować i usuwać z pamięci. Te stosunkowo niezależne bloki są określane jako **moduły ładowalne** [GOYE99]. Zasadniczo moduł jest plikiem obiektowym, którego kod może być połączony z jądrem i odłączony w czasie wykonywania programu. Zazwyczaj moduł implementuje określoną funkcję, na przykład system plików, sterownik urządzenia lub inną własność górnej warstwy jądra. Moduł nie działa jako osobny proces lub wątek, chociaż — jeśli jest to konieczne — może tworzyć wątki jądra do różnych celów. Zamiast tego moduł jest wykonywany w trybie jądra w imieniu bieżącego procesu.

Strukturę tego podejścia pokazano na rysunku 18.9. Moduł jądra dzięki przechwyceniu wywołania systemowego `execve()` zapewnia niemożliwe do obejścia oprzyrządowanie. Funkcja `execve()` ładuje nowy plik wykonywalny do nowej przestrzeni adresowej procesu i rozpoczyna jego wykonywanie. Dzięki przechwyceniu tego wywołania moduł jądra zatrzymuje aplikację przed wykonaniem jej pierwszej instrukcji i może wstawić procedury audytu do aplikacji przed jej uruchomieniem.



Rysunek 18.9. **Środowisko wykonawcze dla audytu aplikacji**

Faktyczne oprzyrządowanie aplikacji jest realizowane przez demona monitorowania, który jest uprzywilejowanym procesem przestrzeni użytkownika. Demon zarządza dwoma repozytoriami: repozytorium poprawek i repozytorium audytu. Repozytorium poprawek zawiera kod do instrumentacji monitorowanych aplikacji. Repozytorium audytu zawiera kod inspekcji, który należy wstawić do aplikacji. Kod w obydwu repozytoriach

ma postać dynamicznych bibliotek. Dzięki użyciu dynamicznych bibliotek można zaktualizować kod w bibliotekach w czasie, gdy działa demon. Ponadto w tym samym czasie może istnieć wiele wersji bibliotek.

Sekwencja zdarzeń jest następująca:

1. Funkcja systemowa `execve()` wywołuje monitorowaną aplikację.
2. Moduł jądra przechwytuje połączenie, zatrzymuje aplikację i ustawia proces nadrzędny na demona monitorowania. Następnie moduł jądra wysyła powiadomienie do działającego w przestrzeni użytkownika demona, którego uruchomiła monitorowana aplikacja.
3. Demon monitorowania lokalizuje funkcje biblioteki poprawek i audytu odpowiednie dla tej aplikacji. Demon ładuje funkcje biblioteczne inspekcji do przestrzeni adresowej aplikacji i w określonych punktach kodu aplikacji wstawia wywołania funkcji inspekcji.
4. Po oprzyrządowaniu aplikacji demon umożliwia aplikacji rozpoczęcie działania.

W celu uproszczenia procesu tworzenia kodu audytu i poprawek opracowano specjalny język. Ogólnie rzecz biorąc, poprawki można wstawiać do procedury biblioteki współdzielonej w dowolnym punkcie wywołania funkcji. Poprawka może wywoływać procedury audytu, a także wywoływać procedurę biblioteki współdzielonej, w sposób logicznie podobny do opisanej wcześniej techniki interpozycji.

18.4. ANALIZA ŚLADU AUDYTU BEZPIECZEŃSTWA

Programy i procedury analizy ścieżek audytu znacznie się różnią w zależności od konfiguracji systemu, obszarów największego zainteresowania, dostępnego oprogramowania, polityki bezpieczeństwa organizacji oraz wzorców zachowań użytkowników i intruzów. W tym podrozdziale zamieszczono pewne spostrzeżenia dotyczące analizy ścieżek audytu.

Przygotowania

Aby przeprowadzić użyteczną analizę audytu, analityk lub administrator bezpieczeństwa musi zrozumieć dostępne informacje i sposób ich wykorzystywania. Standard NIST SP 800-92 oferuje kilka użytecznych porad na ten temat, które podsumujemy w niniejszym podrozdziale.

Wpisy w logu

Administrator bezpieczeństwa (albo inna osoba przeglądająca i analizująca logi) musi zrozumieć kontekst poszczególnych wpisów. Istotne informacje mogą znajdować się w innych pozycjach w tym samym logu, wpisach w innych logach oraz w źródłach innych niż log — na przykład wpisach w systemie zarządzania konfiguracją. Administrator

powinien rozumieć zagrożenia wynikające z niewiarygodnych wpisów, na przykład z pakietu zabezpieczeń, o którym wiadomo, że podczas poszukiwania szkodliwej aktywności często generuje fałszywe alarmy.

Większość formatów plików audytu zawiera mieszankę prostego języka z tajemniczymi komunikatami lub kodami, które mają znaczenie dla dostawcy oprogramowania, ale niekoniecznie dla administratora. Administrator musi dołożyć starań, aby jak najdokładniej odszyfrować informacje zawarte we wpisach w logu. W niektórych przypadkach oprogramowanie do analizy logów realizuje zadanie redukcji danych, co zmniejsza obciążenia administratora. Mimo to administrator powinien dobrze rozumieć znaczenie surowych danych dostarczanych do oprogramowania do analizy i przeglądu, aby móc ocenić użyteczność tych pakietów.

Najskuteczniejszym sposobem na zdobycie solidnego zrozumienia danych w logach jest ich regularne przeglądanie i analizowanie (np. codziennie). Celem jest uzyskanie zrozumienia punktu odniesienia typowych wpisów w logu, stanowiących zdecydowaną większość wpisów.

Zrozumienie kontekstu

Aby przeprowadzić skuteczne przeglądy i analizę, administratorzy powinni dobrze rozumieć każdą z następujących kwestii. Potrzebną wiedzę powinni zdobyć dzięki uczestnictwu w szkoleniach lub z osobistych doświadczeń:

- Obowiązujące w organizacji zasady dotyczące dozwolonego użycia. Dzięki temu administratorzy potrafią rozpoznać naruszenia zasad.
- Oprogramowanie zabezpieczające wykorzystywane na hostach, w tym typy zdarzeń związanych z bezpieczeństwem, które może wykryć każdy program, oraz ogólny profil wykrywania każdego programu (np. znane fałszywe alarmy).
- Systemy operacyjne i główne aplikacje (np. serwery e-mail, WWW) zainstalowana na hostach, w szczególności możliwości i cechy każdego systemu operacyjnego i głównej aplikacji dotyczące bezpieczeństwa i logowania.
- Charakterystyka typowych technik ataku, szczególnie sposób rejestracji użycia tych technik w systemie.
- Oprogramowanie potrzebne do przeprowadzenia analizy, takie jak przeglądarki logów, skrypty redukujące logi oraz narzędzia obsługi zapytań do bazy danych.

Dobór czasu

Ścieżki audytu mogą być wykorzystywane na wiele sposobów. Rodzaj analizy zależy, przynajmniej częściowo, od tego, kiedy ma być ona przeprowadzona. Możliwe są następujące opcje:

- **Przegląd ścieżki audytu po wystąpieniu zdarzenia**. Ten typ przeglądu jest wywoływany po zaobserwowaniu zdarzenia, na przykład znanego problemu z systemem lub aplikacją, znanego naruszenia polityki bezpieczeństwa przez użytkownika, albo jakiegoś niewyjaśnionego problemu z systemem lub użytkownikiem. W wyniku przeglądu można zebrać informacje, aby rozwinąć wiedzę na temat zdarzenia, zdiagnozować przyczynę lub problem oraz zaproponować działania naprawcze i przyszłe środki zaradcze. Ten rodzaj przeglądu koncentruje się na zapisach śladu audytu, które są istotne dla konkretnego zdarzenia.
- **Okresowy przegląd danych ścieżki audytu**. Ten rodzaj przeglądu obejmuje analizę wszystkich danych zgromadzonych w śladzie lub wskazanych podzbiorów. Spełnia wiele możliwych celów. Przykłady celów to szukanie zdarzeń lub wzorców, które sugerują problem z bezpieczeństwem, opracowanie profilu normalnego zachowania oraz wyszukiwanie anormalnych zachowań oraz tworzenie profili dla poszczególnych użytkowników w celu utrzymania stałego zapisu zachowań z podziałem na użytkowników.
- **Analiza audytu w czasie rzeczywistym**. Narzędzia do analizy audytu mogą być również używane w czasie rzeczywistym lub w czasie zbliżonym do rzeczywistego. Analiza w czasie rzeczywistym jest częścią mechanizmów wykrywania włamań.

Przegląd audytu

Oprócz analizy danych ze ścieżki audytu za pomocą narzędzi do redukcji i analizy danych można przeprowadzać przeglądy audytu. Funkcja przeglądu audytu umożliwia administratorowi odczytanie informacji z wybranych zapisów audytu. W specyfikacji Common Criteria [CCPS12a] określono mechanizm, który umożliwia wybór danych audytu przed zapisaniem lub po zapisaniu i selektywny przegląd następujących elementów:

- działania jednego lub większej liczby użytkowników (np. identyfikacja, uwierzytelnianie, wejścia do systemu i działania związane z kontrolą dostępu);
- działania wykonywane na określonym obiekcie lub zasobie systemowym;
- wszystkie podlegające kontroli wyjątki lub określony ich zestaw;
- działania związane z określonym systemem lub atrybutem bezpieczeństwa.

Przegląd audytu może skupiać się na zapisach pasujących do określonych atrybutów, takich jak użytkownik lub grupa użytkowników, okno czasowe, typ zapisu i tak dalej.

Jednym z automatycznych narzędzi, które mogą być przydatne w przeglądzie kontroli, są programy do ustalania priorytetów zapisów audytu na podstawie danych wprowadzonych przez administratora. Zapisy można uszeregować według priorytetów na podstawie kombinacji czynników. Oto przykłady:

- typ wpisu (np. kod komunikatu 103, klasa komunikatu `CRITICAL`);
- nowość typu wpisu (tzn. czy ten typ wpisu pojawił się w logach wcześniej?);

- źródło logu;
- źródłowy lub docelowy adres IP (np. adres źródłowy na czarnej liście; adres docelowy kluczowego systemu; poprzednie zdarzenia związane z konkretnym adresem IP);
- pora dnia lub dzień tygodnia (np. określone działanie może być dopuszczalne w określonych godzinach, ale niedozwolone w innych);
- częstotliwość wpisu (np. x razy w ciągu y sekund).

Tego rodzaju przegląd audytu może spełniać wiele możliwych celów. Przegląd audytu może pozwolić administratorowi na zapoznanie się z bieżącym działaniem systemu, profilami użytkowników i aplikacji w systemie, częstotliwością ataków oraz innymi zdarzeniami związanymi z użytkowaniem systemu i jego bezpieczeństwem. Przegląd audytu może być wykorzystany do uzyskania zrozumienia po incydencie z atakiem i reakcji systemu, co może prowadzić do zmian w oprogramowaniu i procedurach.

Podejścia do analizy danych

Spektrum podejść i algorytmów stosowanych do analizy danych z audytu jest o wiele za szerokie, by można je było dokładnie opisać w tej książce. Zamiast tego na bazie dyskusji w [SING04] zaprezentujemy informacje na temat niektórych najważniejszych podejść.

Podstawowe alarmowanie

Najprostszą formą analizy jest informacja wysłana przez oprogramowanie o tym, że miało miejsce określone zdarzenie. Jeżeli informacja ta jest przekazywana w czasie rzeczywistym, może służyć jako część mechanizmu wykrywania włamań. W przypadku zdarzeń, które nie osiągają poziomu wyzwalającego alarm o włamaniach, wskazanie podejrzanej aktywności po fakcie może zainicjować dalszą analizę.

Baselining

Technika określana jako **baselining** (dosł. wyznaczanie podstawy, bazy) to proces definiowania zdarzeń i wzorców normalnych w zestawieniu z nietypowymi. Proces obejmuje mierzenie zbioru znanych danych w celu obliczenia zakresu wartości normalnych. Te wartości bazowe można następnie porównać z nowymi danymi w celu wykrycia nietypowych przesunięć. Przykładami aktywności zmierzającymi do określenia poziomu bazowego są:

- natężenie ruchu sieciowego według protokołów: całkowity ruch HTTP, e-mail, FTP i tak dalej;
- logowania (wylogowania);
- dostęp do kont administracyjnych;

- zarządzanie adresami protokołu DHCP (ang. *Dynamic Host Configuration Protocol*), żądania DNS;
- całkowita ilość danych logu na godzinę (dzień);
- liczba procesów uruchomionych w dowolnym momencie.

Na przykład duży wzrost natężenia ruchu FTP może wskazywać, że napastnik przejął kontrolę nad serwerem FTP i złośliwie go wykorzystuje.

Po ustaleniu podstaw możliwe jest wykonywanie analiz porównawczych. Jednym z podejść, na które często powołujemy się w tym tekście, jest **wykrywanie anomalii**. Przykładem prostego sposobu wykrywania anomalii jest zastosowanie darmowego oprogramowania Never Before Seen (NBS) Anomaly Detection Driver[4]. Narzędzie implementuje bardzo szybki mechanizm przeszukiwanie bazy danych ciągów znaków i informuje, czy określony ciąg znaków się w niej znajduje (tzn. był widziany wcześniej).

Rozważmy następujący przykład dotyczący DHCP. Protokół DHCP służy do łatwej konfiguracji TCP/IP hostów w sieci. Po uruchomieniu systemu operacyjnego host klienta wysyła żądanie konfiguracji, które zostaje wykryte przez serwer DHCP. Serwer DHCP wybiera dla stacji klienckich odpowiednie parametry konfiguracyjne (adres IP z właściwą maską podsieci oraz inne opcjonalne parametry, takie jak adres IP domyślnej bramy, adresy serwerów DNS, nazwa domeny itp.). Serwer DHCP przypisuje adresy IP klientów w ramach predefiniowanego zakresu na określony czas (czas dzierżawy). Jeśli adres IP ma być zachowany, klient musi zażądać przedłużenia dzierżawy na kolejny okres przed jej wygaśnięciem. Jeśli klient nie zażądał przedłużenia czasu dzierżawy, adres IP jest uważany za wolny i można go przypisać innemu klientowi. Zadania te są wykonywane automatycznie i w sposób przezroczysty dla użytkownika. Dzięki NBS można łatwo monitorować sieci organizacji pod kątem nowych kombinacji adresów MAC – IP dzierżawionych przez serwery DHCP. Administrator natychmiast dowiaduje się o nowych adresach MAC i nowych dzierżawionych adresach IP, które zwykle nie były wydzierżawione. W niektórych przypadkach może to mieć wpływ na bezpieczeństwo. Mechanizm NBS pozwala również skanować zniekształcone zapisy, nowe zapytania klientów oraz wiele innych wzorców.

Inną formą techniki **baseliningu** jest tzw. **thresholding** (dosł. wyznaczanie progów, nazywane czasami progowaniem). Thresholding polega na identyfikacji danych, które przekraczają określoną wartość bazową. Prosty **thresholding** służy do identyfikowania zdarzeń, takich jak odrzucone połączenia, które wydarzyły się więcej niż określoną liczbę razy. Thresholding zamiast na prostej liczbie zdarzeń może koncentrować się na innych parametrach — na przykład na częstotliwości zdarzeń.

[4] Link do tego oprogramowania można znaleźć na stronie WWW związanej z tą książką.

Okienkowanie (ang. *windowing*) to wykrywanie zdarzeń w ramach danego zbioru parametrów — na przykład w określonym przedziale czasu lub poza danym przedziałem czasowym — na przykład wyznaczenie bazy pory dnia, w której loguje się każdy z użytkowników, a następnie oznaczenie zdarzeń logowania, które wykraczają poza ten zakres.

Korelacja

Innym rodzajem analizy jest korelacja — technika polegająca na poszukiwaniu związków pomiędzy zdarzeniami. Prostym przykładem korelacji jest alert o obecności określonego zapisu w logu w przypadku istnienia innego zapisu w logu. Na przykład, jeśli system Snort (patrz podrozdział 8.9) zgłosił próbę przepełnienia bufora ze zdalnego hosta, to rozsądną próbą korelacji będzie odczytanie wszystkich wiadomości, które zawierają adres IP tego zdalnego hosta. Administrator może również chcieć dowiedzieć się o wszystkich próbach użycia polecenia su (od ang. *switch user* — dosł. przełącz użytkownika) na koncie, na którym zalogowano się z nieużywanego wcześniej hosta zdalnego.

18.5. ZARZĄDZANIE INFORMACJAMI O BEZPIECZEŃSTWIE I ZDARZENIACH

Istnieje zapotrzebowanie na systemy, które potrafią automatycznie przetwarzać ogromną ilość audytowych danych bezpieczeństwa generowanych przez współczesne sieci, serwery i hosty w większych organizacjach. W takich środowiskach generowanych jest tyle danych, że ogólnie rzecz biorąc, wydobycie na czas przydatnych informacji jest niemożliwe. Obejmuje to konieczność scharakteryzowania normalnej aktywności i progów. Dzięki temu system może generować ostrzeżenia w przypadku wykrycia anomalii lub złośliwych wzorców. Z tych powodów potrzebna jest pewna forma zintegrowanego, zautomatyzowanego, scentralizowanego systemu logowania. Tego rodzaju problemy można rozwiązać za pomocą produktów określanych jako **systemy zarządzania informacjami o bezpieczeństwie i zdarzeniach** (ang. *security information and event management* — SIEM).

Na potrzebę istnienia takich systemów jako kluczowych mechanizmów bezpieczeństwa wskazuje między innymi standard NIST SP 800-137 [*Information Security Continuous Monitoring (ISCM) for Federal Information Systems and Organizations* z września 2011]. [TARA11] zauważa, że system SIEM można skonfigurować w taki sposób, aby wspomagał wdrażanie wielu spośród „20 kluczowych zabezpieczeń" opracowanych przez SANS i inne instytucje, o których pisaliśmy w rozdziale 12.

Systemy SIEM

Oprogramowanie SIEM to scentralizowany pakiet oprogramowania logowania podobny do mechanizmu syslog, ale znacznie bardziej złożony. Systemy SIEM dostarczają centralnego, jednolitego system przechowywania śladu z audytu oraz zestawu programów do analizy danych audytowych. Zarządzanie logami i systemami SIEM omówiono w standardzie NIST SP 800-92. Zauważono, że istnieją dwa ogólne podejścia do konfiguracji, a wiele produktów oferuje kombinację obu:

- **Bez agentów**. Serwer SIEM odbiera dane od poszczególnych hostów generujących logi bez potrzeby instalowania na tych hostach specjalnego oprogramowania. Niektóre serwery ściągają logi z hostów, co zwykle odbywa się poprzez uwierzytelnienie serwera na każdym hoście i regularne pobieranie jego logów. W innych przypadkach hosty przesyłają swoje logi na serwer, co zwykle wymaga od każdego z nich uwierzytelnienia się na serwerze i regularnego przesyłania swoich logów. Następnie serwer SIEM przeprowadza filtrowanie i agregację zdarzeń oraz normalizację i analizę zebranych logów.
- **Z agentami**. Program agenta jest instalowany na hoście generującym logi w celu przeprowadzenia filtrowania zdarzeń oraz agregacji i normalizacji logu dla określonego typu. Następnie agent przesyła znormalizowane dane logów na serwer SIEM w celu przeprowadzenia analizy oraz do przechowywania — zwykle odbywa się to w czasie rzeczywistym lub w czasie zbliżonym do rzeczywistego. Jeśli host obsługuje wiele typów logów, to może być konieczne zainstalowanie wielu agentów. Niektóre produkty SIEM oferują również agenty dla formatów generycznych, takich jak syslog i SNMP. Generyczny agent służy przede wszystkim do pobierania danych logu ze źródeł, dla których metoda bazująca na agencie specyficznym dla formatu oraz metoda bez agentów nie są dostępne. Niektóre produkty umożliwiają także administratorom tworzenie niestandardowych agentów, co pozwala na obsługę nietypowych źródeł dzienników.

Oprogramowanie SIEM potrafi rozpoznawać różne formaty logów, w tym pochodzące z różnych systemów operacyjnych, oprogramowania zabezpieczającego (np. systemy IDS i zapory firewall), serwerów aplikacji (np. serwerów WWW i pocztowych), a nawet urządzeń kontroli bezpieczeństwa fizycznego, takich jak czytniki identyfikatorów. Oprogramowanie SIEM normalizuje te różnorodne wpisy w logach, dzięki czemu we wszystkich wpisach w logu dla tego samego elementu danych (np. adresu IP) używany jest ten sam format. Oprogramowanie może usuwać we wpisach logu pola, które nie są potrzebne dla funkcji bezpieczeństwa, oraz wpisy w logu, które nie są istotne, co znacznie zmniejsza ilość danych w logu centralnym. Serwer SIEM analizuje połączone dane z wielu źródeł logów, koreluje zdarzenia pomiędzy wpisami logu, identyfikuje i nadaje priorytety istotnym zdarzeniom oraz, jeśli jest to pożądane, inicjuje odpowiedzi na zdarzenia. Produkty SIEM zwykle obejmują kilka funkcji ułatwiających pracę użytkowników, takich jak:

- Graficzne interfejsy użytkownika (GUI), specjalnie zaprojektowane do wspomagania analityków w identyfikowaniu potencjalnych problemów i przeglądaniu wszystkich dostępnych danych związanych z każdym z nich.
- Baza wiedzy o zabezpieczeniach z informacjami o znanych słabych punktach, prawdopodobnym znaczeniu niektórych komunikatów logów oraz innymi technicznymi danymi; często zapewnia możliwość dostosowania do indywidualnych potrzeb analityków.
- Mechanizmy śledzenia i raportowanie zdarzeń. Czasem obejmują rozbudowane mechanizmy obsługi przepływu pracy.
- Systemy przechowywania i korelacji informacji o zasobach (np. nadawanie wyższego priorytetu atakom skierowanym przeciw wrażliwym systemom operacyjnym lub ważniejszym hostom).

Dobrze zaimplementowane systemy SIEM mogą stanowić kluczowy komponent infrastruktury bezpieczeństwa organizacji. Wiele organizacji nie potrafi jednak odpowiednio planować i instalować takich systemów, a potem nimi zarządzać. [HADS10] zauważa, że właściwy proces obejmuje definiowanie zagrożeń, dokumentowanie reakcji i konfigurowanie standardowych raportów w celu spełnienia wymogów audytu i obowiązujących przepisów. Załączniki w tym dokumencie zawierają przykłady każdego z tych działań, które można dostosować i rozszerzyć zgodnie z potrzebami konkretnej organizacji. Wszystkie te działania można przeprowadzić w ramach szerszego procesu oceny ryzyka bezpieczeństwa informatycznego, które omawialiśmy w rozdziałach 14. i 15. W tym dokumencie wymieniono również wielu dostawców produktów SIEM.

18.6. PODSTAWOWE POJĘCIA, PYTANIA SPRAWDZAJĄCE I ZADANIA

Podstawowe pojęcia

analiza ścieżki audytu	biblioteka współdzielona dołączana statycznie	ścieżka audytu dostępu fizycznego
audyt	dynamiczne przepisywanie binarne	ścieżka audytu na poziomie systemu
audyt bezpieczeństwa	log	ścieżka audytu na poziomie użytkownika
baselining (bazowanie)	ładowalne moduły	ślad audytu na poziomie aplikacji
biblioteka dołączana statycznie	przegląd audytu	thresholding (progowanie)
biblioteka interpozycjonowana	syslog	windowing (okienkowanie)
biblioteka współdzielona	ścieżka audytu	wykrywanie anomalii
biblioteka współdzielona dołączana dynamicznie	ścieżka audytu bezpieczeństwa	zarządzanie informacjami o zabezpieczeniach i zdarzeniami (SIEM)

Pytania sprawdzające

18.1. Wyjaśnij różnicę między komunikatem audytu bezpieczeństwa a alarmem bezpieczeństwa.

18.2. Wymień i krótko opisz elementy modelu audytu bezpieczeństwa i alarmów.

18.3. Wymień i krótko opisz główne funkcje audytu bezpieczeństwa.

18.4. W jakich obszarach (kategoriach danych) należy zbierać dane z audytu?

18.5. Wymień i wyjaśnij różnice pomiędzy czterema różnymi kategoriami ścieżek audytu.

18.6. Jakie są główne elementy uniksowego mechanizmu syslog?

18.7. Wyjaśnij, w jaki sposób można wykorzystać do audytu na poziomie aplikacji biblioteki interpozycjonowane.

18.8. Wyjaśnij różnicę między przeglądem audytu a analizą audytu.

18.9. Co to jest system zarządzania informacjami o bezpieczeństwie i zdarzeniach (SIEM)?

Zadania

18.1. Porównaj tabele 18.2 i 18.3. Omów obszary wspólne i odrębne oraz ich znaczenie.

a) Czy w tabeli 18.2 znajdują się pozycje, których nie ma w tabeli 18.3? Podaj uzasadnienie.

b) Czy w tabeli 18.3 znajdują się pozycje, których nie ma w tabeli 18.2? Podaj uzasadnienie.

18.2. Kolejną listę zdarzeń podlegających audytowi pochodzącą z [KUPE04] zaprezentowano w tabeli 18.6. Porównaj ją z listami w tabelach 18.2 i 18.3.

a) Czy w tabelach 18.2 i 18.3 są pozycje, których nie ma w tabeli 18.6? Podaj uzasadnienie.

b) Czy są pozycje w tabeli 18.6, których nie ma w tabelach 18.2 i 18.3? Podaj uzasadnienie.

18.3. Przedstaw zalety i wady opisanych w podrozdziale 18.5 systemów oprogramowania SIEM z agentami i bez agentów.

Tabela 18.6. Sugerowana lista zdarzeń podlegających audytowi

Identyfikacja i uwierzytelnianie

- hasło zostało zmienione
- zdarzenia nieudanego logowania
- udane próby logowania
- typ terminala
- lokalizacja logowania
- zapytano o tożsamość użytkownika
- próby logowania na nieistniejących kontach
- użycie terminala
- typ logowania (interaktywny/automatyczny)
- metoda uwierzytelniania
- czas wylogowania
- całkowity czas połączenia
- powód wylogowania

Operacje w OS

- włączenie audytu
- próba wyłączenia audytu
- próba zmiany konfiguracji audytu
- umieszczenie obiektu w przestrzeni pamięci innych użytkowników
- usunięcie obiektów z przestrzeni pamięci innych użytkowników
- zmiany uprawnień
- zmiana etykiety grupy
- użycie „wrażliwego" polecenia

Udany dostęp do programu

- nazwy poleceń i argumenty
- czas korzystania
- dzień użycia
- wykorzystany czas procesora
- czas zegarowy korzystania z programu
- pliki, do których uzyskano dostęp
- liczba plików, do których uzyskano dostęp
- maksymalna ilość wykorzystanej pamięci

Nieudany dostęp do programu

Parametry systemowe

- ogólnosystemowa aktywność procesora (obciążenie)
- ogólnosystemowa aktywność dysku
- ogólnosystemowe wykorzystanie pamięci

Dostęp do plików

- utworzenie pliku
- odczyt pliku
- zapis do pliku
- usunięcie plików
- próba uzyskania dostępu do plików innych użytkowników
- próba dostępu do „wrażliwych" plików
- nieudany dostęp do plików
- zmiana uprawnień
- zmiana etykiety
- modyfikacja katalogu

Informacje o plikach

- nazwa
- znaczniki czasu
- rodzaj
- zawartość
- właściciele
- grupa
- uprawnienia
- etykieta
- urządzenie fizyczne
- blok dysku

Interakcje z użytkownikiem

- szybkość pisania
- błędy literowe
- okresy pisania
- rytm pisania
- analogie nacisku
- zdarzenia związane z oknami
- wiele zdarzeń w tej samej lokalizacji
- wiele lokalizacji zdarzeń
- ruchy myszą
- kliknięcia myszą
- okresy bezczynności
- czas połączenia
- dane wysyłane z terminala
- dane wysyłane do terminala

Wydrukowane kopie

Aktywność w sieci

- odebrano pakiet
- protokół
- adres źródłowy
- adres docelowy
- port źródłowy
- port docelowy
- rozmiar
- rozmiar ładunku
- ładunek danych
- suma kontrolna
- flagi
- otwarto port
- zamknięto port
- zażądano połączenia
- zamknięto połączenie
- zresetowano połączenie
- wyłączenie maszyny

SKOROWIDZ

A

B

C

D

E

F

G

H

I

J

K

L

Ł

M

N

O

P

R

S

Ś

T

U

W

Z

Ź

Ż

CYBEROBRONA: BĄDŹ CZUJNY I PRZYGOTUJ SIĘ!

Bezpieczeństwo systemu informatycznego od dawna nie jest problemem wyłącznie administratora IT i jego zespołu. Różnorodność metod, socjotechnik i wyrafinowanych narzędzi wykorzystywanych do ataków oraz zacierania śladów sprawia, że zabezpieczenie usług czy zasobów musi być obowiązkiem całego personelu firmy — od prezesa po stażystę. Co więcej, bezpieczeństwo zasobów informatycznych wymaga systematycznej kontroli i systemowego podejścia, co wcale nie jest łatwym zadaniem. Z jednej strony polityka bezpieczeństwa informatycznego powinna zostać sprzężona z pozostałymi elementami strategii przedsiębiorstwa, z drugiej — podlegać ciągłej aktualizacji i przeglądom z uwagi na szybki rozwój technik ataków i ich nieprzewidywalność.

Ta książka jest drugim tomem znakomitego podręcznika projektowania, wdrażania i utrzymywania systemów bezpieczeństwa informatycznego. Poruszono w niej dość różnorodne zagadnienia: problemy zarządzania bezpieczeństwem systemu, algorytmy kryptograficzne i bezpieczeństwo sieci. Zaprezentowano różne podejścia do oceny ryzyka bezpieczeństwa, a także do tworzenia planów reagowania w przypadku wystąpienia zagrożeń, w tym klęski żywiołowej. Sporo uwagi poświęcono zapobieganiu szkodom wyrządzanym przez ludzi i reagowaniu na incydenty bezpieczeństwa. W przystępny sposób wyjaśniono standardy bezpieczeństwa sieci bezprzewodowych oraz systemów linuksowych i opartych na MS Windows. Książkę wzbogacono o szereg interesujących studiów przypadków, pytań sprawdzających, projektów i uzupełnień.

Najciekawsze zagadnienia:

- Zarządzanie bezpieczeństwem i ryzykiem IT w organizacji
- Praktyki, procedury i zasady zwiększające bezpieczeństwo oprogramowania i infrastruktury
- Standardy szyfrowania i rodzaje ataków na zaszyfrowane dane
- Bezpieczeństwo sieci bezprzewodowych i urządzeń mobilnych
- Środowiska zaufane i zabezpieczenia wielopoziomowe

Dr William Stallings jest autorem kilkudziesięciu książek i artykułów wielokrotnie publikowanych przez ACM i IEEE. Trzynastokrotnie zdobył nagrodę za najlepszy podręcznik informatyczny roku. Działa w branży od ponad 30 lat, projektował i implementował zestawy protokołów sieciowych dla różnych systemów.

Dr Lawrie Brown wykłada w School of Engineering and Information Technology w Australii. Specjalizuje się w zagadnieniach komunikacji oraz bezpieczeństwa systemów, a także kryptografii i projektowania bezpiecznych środowisk zdalnego wykonywania kodu.

Helion
helion.pl
HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-8322-558-6

Cena: 119,00 zł

Pearson