Audyt i ślady logowań: auditd, last, lastb, journalctl

Poziom: średniozaawansowany

Dotyczy: RHEL/Rocky 9–10 (audit + auditd), różnice Debian/Ubuntu i SLES zaznaczone

Czas: ~22 min

Kto się zalogował, kto próbował i nie dał rady, kto zmienił `/etc/passwd`, kto uruchomił `sudo su` o trzeciej w nocy — trzy różne warstwy systemu odpowiadają na trzy różne części tego pytania, i ta strona pokazuje, gdzie każdej z nich szukać.

Skróty użyte na tej stronie (4)

Skrót	Co znaczy
PID	Process ID: numer procesu w systemie
RHEL	Red Hat Enterprise Linux: komercyjna dystrybucja Linuksa
SLES	SUSE Linux Enterprise Server: płatna dystrybucja Linuksa firmy SUSE, konkurent Rocky/RHEL
UID	User ID: numer użytkownika w Linuksie

Fiszka

TYLKO ODCZYT

`last` — historia logowań

```
last
```

Czyta plik `wtmp` i wypisuje historię logowań od najnowszego do najstarszego wpisu wciąż przechowywanego w pliku — kto się logował, z jakiego terminala albo adresu, kiedy i jak długo trwała sesja.

Przykład:

```
adm_user pts/0        10.20.20.15      Wt wrz 17 09:12   still logged in
root     pts/1        10.20.20.20      Pon wrz 16 14:03 - 14:47  (00:44)
reboot   system boot  6.1.0-1.el9      Pon wrz 16 08:01   still running
```

□□ **last** = pokaż historię logowań użytkowników do systemu

□□ TYLKO ODCZYT

`last -n 20` — ograniczenie liczby wpisów

```
last -n 20
```

Pokazuje tylko dwadzieścia najnowszych wpisów zamiast całej historii — przydatne na serwerach, gdzie `wtmp` sięga wiele miesięcy wstecz i pełna lista zapycha ekran.

□□ **last** = pokaż historię logowań użytkowników do systemu

□□ TYLKO ODCZYT

`last -F` — pełne daty z rokiem i sekundami

```
last -F
```

Domyślny format `last` skraca datę do dnia tygodnia i godziny bez roku — niewystarczające przy porównywaniu zdarzeń sprzed kilku miesięcy. `-F` wypisuje pełną datę i godzinę początku oraz końca sesji.

Przykład:

```
adm_user pts/0        10.20.20.15      Tue Sep 17 09:12:03 2026   still logged in
root     pts/1        10.20.20.20      Mon Sep 16 14:03:41 2026 - Mon Sep 16 14:47:52
2026  (00:44)
```

□□ **last** = pokaż historię logowań użytkowników do systemu

□□ TYLKO ODCZYT

`last reboot` — historia restartów

```
last reboot
```

Filtruje `wtmp` do samych wpisów startu systemu — szybki sposób sprawdzenia, kiedy maszyna ostatnio się restartowała i czy pokrywa się to z planowanym oknem serwisowym.

Przykład:

```
reboot   system boot  6.1.0-1.el9      Pon wrz 16 08:01   still running
reboot   system boot  6.1.0-1.el9      Sob wrz 14 02:15 - 08:01 (2+05:46)
```

□□ **last** = pokaż historię logowań użytkowników do systemu

□□ **TYLKO ODCZYT**

`lastb` — nieudane próby logowania

```
sudo lastb
```

Analogiczne do `last`, ale czyta plik `btmp` — dziennik **nieudanych** prób logowania. Wymaga roota: plik `btmp` ma uprawnienia zamykające go przed zwykłymi użytkownikami, bo zawiera informacje przydatne do rozpoznania trwającego ataku.

Przykład:

```
UNKNOWN  ssh:notty    203.0.113.5      Śr wrz 18 09:14 - 09:14  (00:00)
root     ssh:notty    203.0.113.5      Śr wrz 18 09:14 - 09:14  (00:00)
admin    ssh:notty    203.0.113.26     Śr wrz 18 08:02 - 08:02  (00:00)
```

Długa lista prób z tego samego adresu w krótkim czasie to obraz tego, jak wygląda atak siłowy z zewnątrz zanim `fail2ban` zdąży go zablokować.

□□ **TYLKO ODCZYT**

`lastlog` — ostatnie logowanie każdego konta

```
lastlog
```

W przeciwieństwie do `last`, który pokazuje historię sesji, `lastlog` pokazuje **jeden** wiersz na każde konto systemowe — datę jego ostatniego logowania, albo `**Never logged in**`, jeśli konto nigdy nie było użyte. Dobre narzędzie do znalezienia kont, które istnieją, ale nikt z nich nie korzysta.

Przykład:

```
Username         Port     From             Latest
root             pts/1    10.20.20.20      Pon wrz 16 14:03:41 +0200 2026
adm_user         pts/0    10.20.20.15      Wt wrz 17 09:12:03 +0200 2026
ftpuser                                    **Never logged in**
```

□□ TYLKO ODCZYT

`who` — kto jest zalogowany teraz

```
who
```

Pokazuje wyłącznie sesje aktywne w tej chwili — bez historii. Krótszy i szybszy w czytaniu niż `w`, gdy interesuje Cię tylko lista, kto teraz siedzi na maszynie.

Przykład:

```
adm_user pts/0        2026-09-17 09:12 (10.20.20.15)
root     pts/1        2026-09-16 14:03 (10.20.20.20)
```

□□ TYLKO ODCZYT

`w` — kto jest zalogowany i co robi

```
w
```

To samo co `who`, plus obciążenie systemu w nagłówku i to, jaki proces każdy zalogowany użytkownik aktualnie uruchomił na swoim terminalu.

Przykład:

```
 09:20:11 up 1 day,  1:19,  2 users,  load average: 0.15, 0.09, 0.05
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
adm_user pts/0    10.20.20.15      09:12    0.00s  0.04s  0.01s w
root     pts/1    10.20.20.20      14:03   45:12   0.02s  0.02s -bash
```

□□ TYLKO ODCZYT

`ac -d` — czas zalogowania w podziale na dni

```
sudo ac -d
```

Z pakietu `psacct` / `acct` — sumuje łączny czas trwania sesji logowań, dzień po dniu, dla wszystkich użytkowników. Przydaje się do rozliczeń albo do wykrycia dnia z nietypowo długą aktywnością.

Przykład:

```
Sep 15   total        2.30
Sep 16   total        8.75
Sep 17   total        1.10
```

□□ TYLKO ODCZYT

`systemctl status auditd` — czy podsystem audytu w ogóle działa

```
systemctl status auditd
```

Pierwsze sprawdzenie przed jakąkolwiek analizą regułami audytu — jeśli `auditd` nie działa, `ausearch` i `aureport` będą przeszukiwać pusty albo nieaktualny dziennik.

Przykład:

```
● auditd.service - Security Auditing Service
     Loaded: loaded (/usr/lib/systemd/system/auditd.service; enabled)
     Active: active (running) since Mon 2026-09-16 08:01:12 CEST; 1 day 1h ago
   Main PID: 780 (auditd)
```

□□ **systemctl** = system control: sterowanie usługami systemd

□□ TYLKO ODCZYT

`auditctl -l` — lista aktywnych reguł

```
sudo auditctl -l
```

Wypisuje reguły audytu aktualnie załadowane do jądra — nie te zapisane w plikach konfiguracyjnych, tylko te faktycznie działające w tej chwili.

Przykład:

```
-w /etc/passwd -p wa -k tozsamosc
-w /etc/shadow -p wa -k tozsamosc
-a always,exit -F arch=b64 -S execve -k polecenia
```

ZMIENIA SYSTEM

`auditctl -w` — obserwacja pliku

```
sudo auditctl -w /etc/passwd -p wa -k NazwaKlucza
```

> (gdzie `NazwaKlucza` to własna etykieta reguły, np. `tozsamosc` — po niej później przeszukamy dziennik przez `ausearch -k`)

Nakłada regułę obserwacji na konkretną ścieżkę: `-p wa` oznacza, że rejestrowane są próby zapisu (`w`) i zmiany atrybutów (`a`) tego pliku. `-k` nadaje regule etykietę (klucz), po której później przeszuka się dziennik przez `ausearch -k`. Reguła nałożona tym poleceniem znika po restarcie — do reguł trwałych służy plik w `/etc/audit/rules.d/`, opisany niżej.

TYLKO ODCZYT

`auditctl -a` — reguła na wywołania systemowe

```
sudo auditctl -a always,exit -F arch=b64 -S execve -k NazwaKlucza
```

Rejestruje każde wywołanie systemowe `execve` (czyli uruchomienie dowolnego programu) na architekturze 64-bitowej, przy wyjściu z wywołania (`exit`), zawsze (`always`) — innymi słowy, każde uruchomienie jakiegokolwiek polecenia w systemie, pod wskazaną etykietą. To jedna z najbardziej „gadatliwych" reguł audytu, jaką można nałożyć — patrz uwaga niżej o zalewaniu dysku.

ZMIENIA SYSTEM

Reguły trwałe w plikach

```
sudo tee /etc/audit/rules.d/NazwaKlucza.rules <<'EOF'
-w /etc/passwd -p wa -k NazwaKlucza
-w /etc/shadow -p wa -k NazwaKlucza
```

```
-w /etc/sudoers -p wa -k NazwaKlucza
EOF
```

Reguły wpisane bezpośrednio przez `auditctl` w linii poleceń nie przetrwają restartu jądra. Do reguł, które mają obowiązywać na stałe, tworzy się plik `.rules` w `/etc/audit/rules.d/` — każdy plik w tym katalogu jest wczytywany przy starcie `auditd` i po każdym `augenrules --load`.

tee = zapisz strumień wejścia do pliku i jednocześnie wypisz go na ekran

TYLKO ODCZYT

`augenrules --load` — wczytanie reguł bez restartu

```
sudo augenrules --load
```

Składa wszystkie pliki `.rules` z `/etc/audit/rules.d/` w jeden zestaw i ładuje go do jądra bez restartowania usługi `auditd` — reguły zaczynają obowiązywać natychmiast, bez przerwy w audytowaniu.

TYLKO ODCZYT

`ausearch -k` — przeszukiwanie po kluczu reguły

```
sudo ausearch -k NazwaKlucza
```

Filtruje zdarzenia audytu po etykiecie nadanej regule przez `-k` — zamiast przekopywać się przez cały dziennik, patrzymy tylko na zdarzenia wywołane przez tę jedną, konkretną regułę.

Przykład:

```
type=PATH msg=audit(1758180245.123:501): item=0 name="/etc/passwd" ...
type=SYSCALL msg=audit(1758180245.123:501): arch=c000003e syscall=2 success=yes exit=3
... key="tozsamosc"
```

ausearch = przeszukaj log audytu jądra, tu pod kątem zdarzeń SELinux oznaczonych avc

TYLKO ODCZYT

`ausearch -m USER_LOGIN` — zdarzenia logowania w danym dniu

```
sudo ausearch -m USER_LOGIN --start today
```

Filtruje po typie zdarzenia (`USER_LOGIN`) i po zakresie czasu (`--start today` — od północy bieżącego dnia). Kombinacja typu zdarzenia i przedziału czasu to najczęstszy sposób zawężania wyników w codziennej pracy.

ausearch = przeszukaj log audytu jądra, tu pod kątem zdarzeń SELinux oznaczonych avc

TYLKO ODCZYT

`ausearch -ua` — zdarzenia konkretnego użytkownika

```
sudo ausearch -ua nazwa_usera
```

Filtruje po identyfikatorze użytkownika audytowanego (`auid`) — czyli po tym, kto **faktycznie** się zalogował na początku sesji, nawet jeśli w międzyczasie przełączył się na innego użytkownika przez `su` albo `sudo`. To jest różnica między `auid` a bieżącym `uid`, o której więcej w sekcji *Jak to działa*.

ausearch = przeszukaj log audytu jądra, tu pod kątem zdarzeń SELinux oznaczonych avc

TYLKO ODCZYT

`aureport` — zbiorcze podsumowania

```
sudo aureport
```

Zamiast surowych zdarzeń pokazuje gotowe, policzone podsumowania: liczbę logowań, liczbę uruchomionych procesów, liczbę zdarzeń każdego typu — punkt startowy przed sięgnięciem po `ausearch` po konkretne wpisy.

Przykład:

```
Summary Report
======================
Range of time in logs: 2026-09-16 08:01:12 - 2026-09-18 09:20:00
Selected time for report: 2026-09-16 08:01:12 - 2026-09-18 09:20:00
Number of changes in configuration: 4
Number of changes to accounts, groups: 0
Number of logins: 12
Number of failed logins: 5
```

```
Number of authentications: 12
Number of failed authentications: 5
```

□□ TYLKO ODCZYT

`aureport --failed` — tylko nieudane zdarzenia

```
sudo aureport --failed
```

Zawęża podsumowanie do zdarzeń zakończonych niepowodzeniem — nieudane logowania, odrzucone wywołania systemowe pilnowane regułą. Odpowiednik `lastb`, ale osadzony w pełnym kontekście audytu, nie tylko logowań.

□□ TYLKO ODCZYT

`aureport -au` — podsumowanie logowań

```
sudo aureport -au
```

Wypisuje tabelę logowań z numerem zdarzenia, datą, nazwą hosta, terminalem, wykonawcą i wynikiem (`success` / `failed`) — bardziej szczegółowy odpowiednik `last`, osadzony w tym samym dzienniku, z którego korzystają `ausearch` i pozostałe raporty `aureport`.

□□ TYLKO ODCZYT

`journalctl _COMM=sudo` — zdarzenia konkretnego polecenia

```
journalctl _COMM=sudo
```

Filtruje dziennik `systemd-journald` po polu `_COMM` — nazwie programu, który wygenerował wpis. Pokazuje każde wywołanie `sudo` zapisane przez `journald`, niezależnie od tego, jaką komendę użytkownik przez `sudo` uruchomił.

□□ **journalctl** = journal control: czytanie dziennika systemd

□□ TYLKO ODCZYT

`journalctl -u NazwaUslugi | grep -i fail` — szybkie wyłuskanie nieudanych logowań

```
journalctl -u NazwaUslugi | grep -i fail
```

> (gdzie `NazwaUslugi` to nazwa jednostki systemd, np. `sshd`)

Przeszukanie logów jednostki po słowie „fail" — szybszy, choć mniej precyzyjny odpowiednik `lastb` albo `ausearch -m USER_LOGIN`, gdy potrzebny jest tylko szybki rzut oka bez pełnej analizy audytu.

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

`journalctl _AUDIT_TYPE_NAME=USER_AUTH` — zdarzenia audytu widziane przez journal

```
journalctl _AUDIT_TYPE_NAME=USER_AUTH
```

`systemd-journald` odbiera też zdarzenia z podsystemu audytu jądra i wystawia je pod polem `_AUDIT_TYPE_NAME` — sposób na przejrzenie zdarzeń audytu bez wywoływania `ausearch`, przydatny, gdy `auditd` jako osobna usługa akurat nie działa, ale sam mechanizm audytu w jądrze wciąż generuje zdarzenia odbierane przez journal.

journalctl = journal control: czytanie dziennika systemd

TYLKO ODCZYT

Wszystkie polecenia

Polecenie	Co pokazuje
`last`	historia logowań z `wtmp`
`lastb`	historia **nieudanych** logowań z `btmp`, wymaga roota
`lastlog`	ostatnie logowanie każdego konta, jeden wiersz na konto
`who` / `w`	kto jest zalogowany teraz (i co robi)
`ac -d`	suma czasu logowań dzień po dniu
`auditctl -l`	aktywne reguły audytu w jądrze
`auditctl -w`	reguła obserwacji pliku

Polecenie	Co pokazuje
`auditctl -a ...execve`	reguła na wywołania systemowe
`augenrules --load`	wczytanie reguł trwałych z `/etc/audit/rules.d/`
`ausearch -k`	zdarzenia po kluczu reguły
`ausearch -m TYP`	zdarzenia po typie
`aureport` / `aureport --failed` / `aureport -au`	zbiorcze podsumowania
`journalctl _COMM=`, `_AUDIT_TYPE_NAME=`	filtrowanie dziennika systemd po polach

Uważaj

- `lastb` wymaga roota — bez `sudo` zwraca błąd braku dostępu do `btmp`, nie pustą listę.
- Reguła `-a always,exit -F arch=b64 -S execve` bez dodatkowego filtra rejestruje **każde** uruchomienie programu w systemie — na zajętym serwerze potrafi wygenerować gigabajty logu dziennie.
- `journalctl` i `auditd` to dwa niezależne systemy zapisujące częściowo pokrywające się, ale nie identyczne zdarzenia — patrz różnica opisana w *Jak to działa*.

Z życia wzięte

Z ŻYCIA WZIĘTE

Środa, 9:00: pan Wojtek pyta podczas audytu, kto ostatnio zmieniał `/etc/passwd` na ipa01. Nakładamy `auditctl -w /etc/passwd -p wa -k tozsamosc` prosto w terminalu, pokazujemy mu działający `ausearch -k tozsamosc` i sprawa wygląda załatwiona. Miesiąc później, po rutynowym restarcie serwera, ta sama reguła nigdzie się nie pojawia w `auditctl -l`, bo nigdy nie trafiła do pliku w `/etc/audit/rules.d/`.

Po co to istnieje

Pojęcie: podsystem audytu jądra. Audyt w Linuksie to mechanizm wbudowany w jądro, który potrafi rejestrować dowolne wywołanie systemowe, dostęp do wskazanego pliku albo zmianę konfiguracji uwierzytelniania — niezależnie od tego, czy dana aplikacja sama pisze

> cokolwiek do logów. Usługa `auditd` w przestrzeni użytkownika odbiera te zdarzenia z jądra i zapisuje je w binarnym dzienniku, zwykle w `/var/log/audit/audit.log`, skąd czytają je `ausearch` i `aureport`.

Podstawowe pytanie, na które ta strona odpowiada: kto był w systemie, kto próbował i mu się nie udało, i kto zmienił coś, czego zmieniać nie powinien. `last`/`lastb` odpowiadają na pierwsze dwie części pytania na poziomie sesji logowania. `auditd` odpowiada na trzecią część — i na dużo więcej, bo potrafi śledzić dowolny plik albo dowolne wywołanie systemowe, nie tylko logowania. `journalctl` łączy w sobie fragmenty obu tych światów, bo część zdarzeń audytu trafia też do dziennika systemd.

Analogia

System audytu działa jak **budynek biurowy z trzema niezależnymi systemami rejestracji**.

Pierwszy to książka przy wejściu, w której ochrona zapisuje każdego, kto wszedł i wyszedł, z godziną — to jest `last`/`lastb`, prosta lista sesji. Drugi to kamery przy konkretnych drzwiach — nie przy wszystkich, tylko przy tych, które ktoś świadomie wskazał jako wymagające nadzoru, na przykład wejście do serwerowni albo do archiwum z dokumentacją kadrową; to jest `auditd` z regułami `-w` na konkretne pliki. Trzeci to ogólny dziennik zdarzeń budynku, prowadzony przez centralę, która zbiera zarówno wpisy z książki przy wejściu, jak i alarmy z niektórych kamer, ale nie wszystkich — to jest `journalctl`, który widzi część, ale nie całość tego, co widzi pełny system audytu.

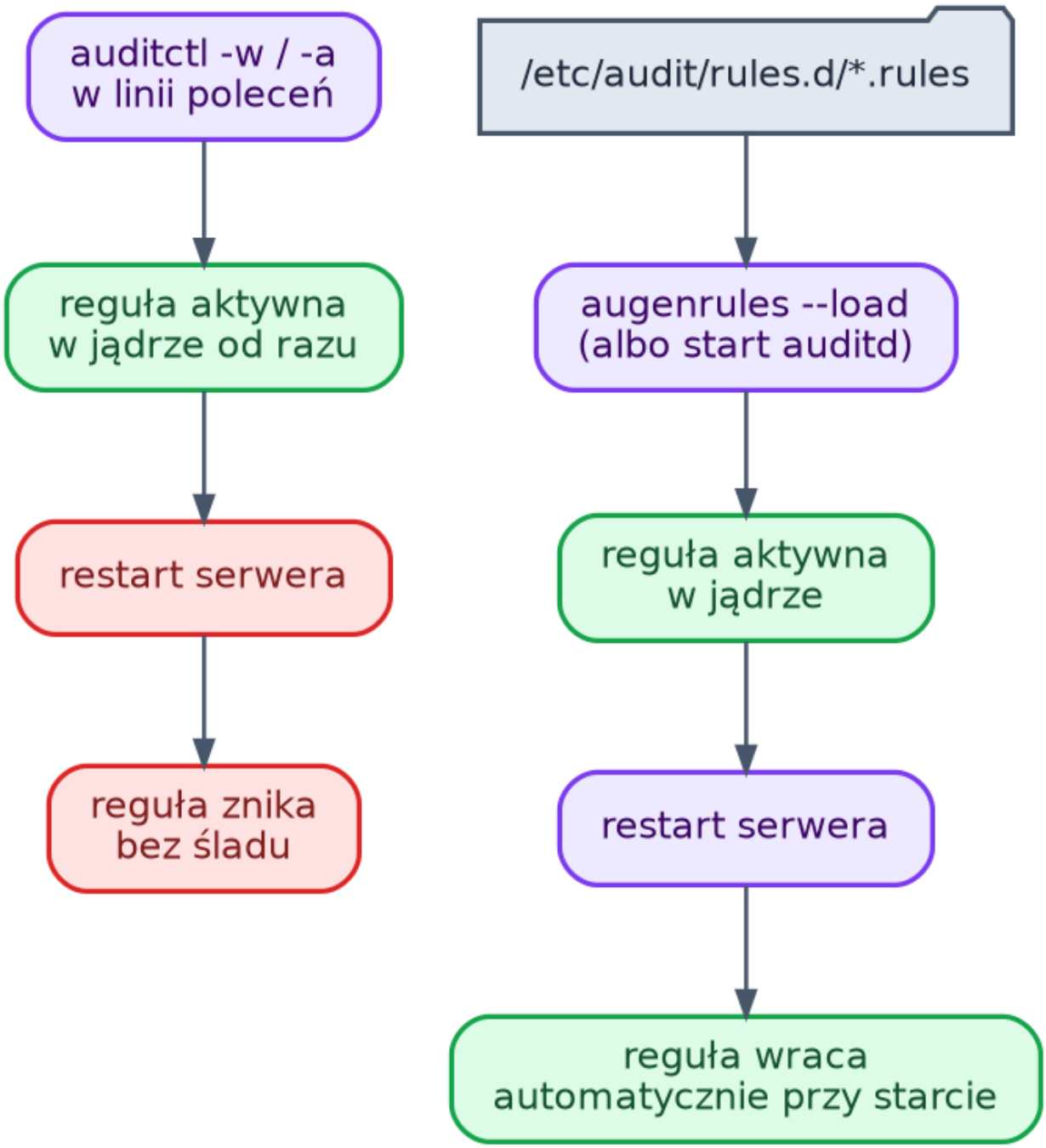
Pułapka, którą ta analogia dobrze pokazuje: jeśli ktoś zainstaluje kamerę przy **każdych** drzwiach w budynku, łącznie z tymi do zwykłej sali konferencyjnej, centrala rejestrująca nagrania zapełni się w ciągu dnia setkami godzin materiału, z którego 99% nie ma żadnej wartości śledczej. Dokładnie to samo dzieje się z regułą audytu obejmującą każde `execve` w systemie bez dodatkowego zawężenia.

Jak to działa

Rozwiń: jak to działa pod spodem

> **Pojęcie: reguła audytu i klucz.** Reguła audytu to instrukcja dla jądra: obserwuj tę konkretną ścieżkę pod kątem tych konkretnych operacji (`-w plik -p uprawnienia`) albo obserwuj to konkretne wywołanie systemowe pod tymi warunkami (`-a always,exit -F`

`warunek -S wywołanie`). Każdej regule można nadać etykietę przez `-k nazwa` — klucz, który nie wpływa na to, co reguła robi, tylko ułatwia późniejsze przeszukiwanie: `ausearch -k nazwa` zwróci wyłącznie zdarzenia wygenerowane przez tę jedną regułę, zamiast przeszukiwać cały, potencjalnie ogromny dziennik. Reguły nałożone bezpośrednio przez `auditctl` żyją tylko do najbliższego restartu jądra; reguły, które mają przetrwać restart, zapisuje się w plikach `.rules` w `/etc/audit/rules.d/` i wczytuje przez `augenrules --load` (albo automatycznie przy starcie `auditd`).



Reguła nałożona auditctl w linii poleceń działa od razu, ale znika po restarcie; reguła zapisana w pliku i wczytana augenrules przetrwa restart (kliknięcie otwiera pełny rozmiar)

> **Pojęcie: dziennik logowań (wtmp/btmp).** Niezależnie od pełnego podsystemu audytu jądra, system utrzymuje dwa osobne, znacznie prostsze pliki binarne dedykowane wyłącznie logowaniom: `/var/log/wtmp` (udane logowania i wylogowania, restarty

> systemu) oraz `/var/log/btmp` (nieudane próby logowania). `last` czyta pierwszy, `lastb` drugi. Te pliki istnieją niezależnie od tego, czy `auditd` w ogóle działa — to osobny, starszy mechanizm, wywodzący się jeszcze z klasycznego Uniksa, długo sprzed powstania podsystemu audytu jądra Linux.

Różnica między tym, co widzi `journal`, a tym, co rejestruje `auditd`, jest źródłem częstych pomyłek. `systemd-journald` zbiera standardowe wyjście i błędy usług uruchamianych przez systemd, komunikaty jądra i — na systemach z odpowiednią konfiguracją — również pewną część zdarzeń audytu przekazywanych mu przez jądro (widoczną pod polami `_AUDIT_TYPE_NAME` i podobnymi). Nie zastępuje to jednak pełnego `auditd`: reguły `-w` i `-a` nałożone przez `auditctl` generują zdarzenia zapisywane do `/var/log/audit/audit.log`, z pełnym kontekstem (PID, UID, dokładna ścieżka, wynik operacji), których `journalctl` bez włączonego, poprawnie skonfigurowanego `auditd` po prostu nie zobaczy. Innymi słowy: `journalctl` to dobre pierwsze przybliżenie i szybki podgląd, `ausearch`/`aureport` to właściwe narzędzie, gdy potrzebny jest pełny, precyzyjny obraz.

Osobny szczegół wart zapamiętania: pole `auid` (audit user ID) w zdarzeniach audytu zapamiętuje, kto **pierwotnie** się zalogował, nawet jeśli ten użytkownik później przełączył się przez `su` albo `sudo` na inne konto. Dzięki temu `ausearch -ua adm_user` znajdzie także działania wykonane przez `adm_user` po przełączeniu się na roota — coś, czego zwykłe `who`/`w`, patrzące tylko na bieżący `uid` sesji, by nie pokazały.

Krok po kroku

1. Sprawdź, czy `auditd` w ogóle działa.

```
systemctl status auditd
```

2. Zobacz, jakie reguły są aktualnie załadowane.

```
sudo auditctl -l
```

3. Dodaj celowaną regułę z kluczem — nie regułę na wszystko.

```
sudo tee /etc/audit/rules.d/tozsamosc.rules <<'EOF'
-w /etc/passwd -p wa -k tozsamosc
-w /etc/shadow -p wa -k tozsamosc
EOF
sudo augenrules --load
```

4. Zweryfikuj, że reguła faktycznie łapie zdarzenia.

```
sudo touch /etc/passwd
sudo ausearch -k tozsamosc --start recent
```

5. Przejrzyj logowania i próby logowania za ostatnią dobę.

```
last -n 20
sudo lastb -n 20
sudo aureport -au --start today
```

6. Do szybkiego przeglądu bez `ausearch` — sprawdź, co widać w journalu.

```
journalctl _AUDIT_TYPE_NAME=USER_AUTH --since today
```

Częste błędy

Reguła bez klucza i bez zawężenia zalewa dysk.

```
$ df -h /var/log/audit
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/vg-audit  10G  9.8G  120M  99% /var/log/audit
```

Reguła `-a always,exit -F arch=b64 -S execve` nałożona bez dodatkowego filtra (np. po użytkowniku) na zajętym serwerze generuje zdarzenie dla **każdego** uruchomionego procesu — także tych wywoływanych setki razy na sekundę przez zwykłe działanie systemu. Partycja `/var/log/audit` zapełnia się w ciągu godzin, a bez klucza (`-k`) przeglądanie tego, co się nazbierało, jest praktycznie niewykonalne. Naprawa: zawężać reguły do konkretnych ścieżek albo dodawać filtry (np. `-F auid!=unset` albo `-F euid=0`), zawsze z kluczem, i pilnować `SystemMaxUse` w konfiguracji audytu (podobnie jak przy `journald`, opisane na osobnej stronie o retencji logów).

`lastb` bez `sudo`.

```
$ lastb
lastb: /var/log/btmp: Permission denied
```

Plik `btmp` ma restrykcyjne uprawnienia, bo ujawnia informacje o próbach logowania, które mogłyby pomóc atakującemu ocenić, czy trafił na istniejące konto. Naprawa: `sudo lastb`.

Reguła nałożona `auditctl` znika po restarcie.

Administrator nakłada regułę bezpośrednio: `sudo auditctl -w /etc/passwd -p wa -k tozsamosc`, sprawdza, że działa, i uznaje sprawę za załatwioną. Po najbliższym restarcie serwera reguła znika bez śladu, bo `auditctl` w linii poleceń modyfikuje wyłącznie bieżący stan jądra, nie żaden plik konfiguracyjny. Naprawa: ta sama reguła musi trafić do pliku w `/etc/audit/rules.d/` i zostać wczytana przez `augenrules --load`, żeby przetrwała restart.

Mylenie braku wyniku w `journalctl` z brakiem zdarzenia.

```
journalctl _AUDIT_TYPE_NAME=USER_AUTH --since today
```

```
-- No entries --
```

Administrator wnioskuje, że nikt się nie logował. W rzeczywistości `auditd` mógł zarejestrować zdarzenie, ale konfiguracja przekazywania zdarzeń audytu do `journald` na tej maszynie jest wyłączona albo niepełna. Naprawa: sprawdzić właściwym narzędziem — `sudo ausearch -m USER_AUTH --start today` — zanim wyciągnie się wniosek z negatywnego wyniku w `journalctl`.

Filtrowanie po `uid` zamiast po `auid` przy `sudo`.

Administrator szuka działań użytkownika `adm_user` po przełączeniu się na roota przez `sudo -i`, ale filtruje po bieżącym `uid` procesu (który po `sudo -i` wynosi `0`, czyli root) i nie znajduje niczego przypisanego do `adm_user`. Naprawa: `ausearch -ua adm_user` filtruje po `auid` — tożsamości, z jaką użytkownik się zalogował na samym początku — i znajduje zdarzenia niezależnie od tego, na jakie konto przełączył się w międzyczasie.

□□ Ćwiczenia

> Wszystko poniżej jest wyłącznie do odczytu — żadne polecenie nie zmienia reguł audytu ani polityki hosta. Jeśli chcesz przetestować własną regułę `auditctl -w`, zrób to na maszynie testowej, nigdy na współdzielonym hoście, którego nie kontrolujesz.

Ćwiczenie 1 — czytanie własnej historii logowań

```
last -F | head -10
who
w
```

Porównaj czas trwania Twojej bieżącej sesji w `w` (kolumna `LOGIN@`) z tym, co pokazuje `last -F` dla tego samego terminala. Czy się zgadzają?

Odpowiedź

Powinny się zgadzać co do minuty — `w` i `last` czerpią dane z tego samego mechanizmu (`utmp`/`wtmp`), tylko `w` pokazuje wyłącznie sesje aktywne teraz, a `last` całą historię łącznie z zakończonymi sesjami. Różnica pojawia się tylko wtedy, gdy między jednym a drugim poleceniem minęło na tyle dużo czasu, że zdążyłeś się na nowo zalogować.

Ćwiczenie 2 — symulacja analizy `lastb` na przygotowanym pliku

```
cd /tmp
cat > proby.txt <<'EOF'
09:14:01 FAILED 203.0.113.5 root
09:14:03 FAILED 203.0.113.5 root
09:14:05 FAILED 203.0.113.5 admin
09:20:11 FAILED 203.0.113.26 test
09:25:00 OK      10.20.20.15 adm_user
EOF
awk '$2=="FAILED"{print $3}' proby.txt | sort | uniq -c | sort -rn
```

Który adres wygenerował najwięcej nieudanych prób w tym przykładowym pliku? Gdyby to był realny plik `btmp`, jaka komenda `fail2ban` (z poprzedniej strony) prawdopodobnie zadziałałaby na ten adres pierwsza?

Odpowiedź

Adres `203.0.113.5` ma trzy nieudane próby, `203.0.113.26` jedną — `203.0.113.5` jest tu wyraźnym kandydatem do zbanowania. Przy typowym `maxretry = 5` fail2ban jeszcze by go nie zbanował na podstawie tej próbki (za mało prób), ale przy niższym progu (np. `maxretry = 3`, częstym dla `sshd`) właśnie ten adres przekroczyłby próg jako pierwszy.

Ćwiczenie 3 — odczyt istniejących reguł i podsumowań, bez ich zmiany

```
sudo auditctl -l
sudo aureport --summary 2>/dev/null | head -20
```

Jeśli `auditd` nie działa w Twoim środowisku, obie komendy zwrócą błąd albo pusty wynik — to sam w sobie ważny wynik ćwiczenia. Jaka byłaby Twoja pierwsza komenda diagnostyczna w takiej sytuacji?

Odpowiedź

`systemctl status auditd` — sprawdzenie, czy usługa w ogóle działa, zanim zacznie się szukać przyczyny pustych wyników w samych regułach czy w dzienniku. Pusty wynik `aureport` przy nieaktywnym `auditd` nie oznacza braku zdarzeń w systemie, tylko brak ich rejestrowania w tym okresie.

Pytania kontrolne

1. Czym różni się to, co pokazuje `last`, od tego, co pokazuje `lastb`?

Odpowiedź

`last` czyta `wtmp` i pokazuje udane logowania oraz restarty systemu. `lastb` czyta osobny plik, `btmp`, i pokazuje wyłącznie **nieudane** próby logowania — inny plik, inny cel, i w przeciwieństwie do `last`, wymaga uprawnień roota.

2. Dlaczego reguła `-a always,exit -F arch=b64 -S execve` bez dodatkowych warunków jest ryzykowna na produkcyjnym serwerze?

Odpowiedź

Bo rejestruje absolutnie każde uruchomienie dowolnego programu w systemie, także te wywoływane wielokrotnie w ciągu sekundy przez zwykłe działanie usług. Na zajętej maszynie potrafi to wygenerować na tyle dużo zdarzeń, że partycja `/var/log/audit` zapełni się w ciągu godzin, a sam dziennik stanie się praktycznie nieprzeszukiwalny bez dodatkowego zawężenia przez klucz i filtry.

3. Reguła nałożona przez `auditctl -w /etc/passwd -p wa -k tozsamosc` znika po restarcie serwera. Dlaczego i jak temu zapobiec?

Odpowiedź

`auditctl` w linii poleceń modyfikuje wyłącznie bieżący, działający stan jądra, nie zapisuje niczego trwale na dysku. Żeby reguła przetrwała restart, trzeba ją zapisać w pliku `.rules` w `/etc/audit/rules.d/` i wczytać przez `augenrules --load` (albo poczekać na automatyczne wczytanie przy starcie `auditd`).

4. `journalctl _AUDIT_TYPE_NAME=USER_AUTH` nie pokazuje żadnych zdarzeń, ale podejrzewasz, że ktoś próbował się logować. Jakiego narzędzia użyjesz zamiast tego i dlaczego może pokazać więcej?

Odpowiedź

`sudo ausearch -m USER_AUTH` (ewentualnie `aureport --failed`) — `ausearch` czyta bezpośrednio pełny dziennik `auditd` (`/var/log/audit/audit.log`), podczas gdy `journalctl` widzi tylko tę część zdarzeń audytu, która została przekazana do `systemd-journald`; ta ścieżka bywa niekompletna albo wyłączona niezależnie od tego, czy sam `auditd` rejestruje zdarzenia poprawnie.

Powiązane: SELinux: podstawy, tryby, konteksty · fail2ban: automatyczne blokowanie po nieudanych próbach · journalctl i zarządzanie dziennikiem systemowym · sudo i /etc/sudoers: konfiguracja i audyt · Retencja i czyszczenie dzienników

Źródła: `man 8 auditd` · `man 8 auditctl` · `man 8 ausearch` · `man 8 aureport` · `man 1 last` · `man 1 lastb`

Revision #15
Created 2026-09-18 06:57:26 UTC by Tomasz Konieczny
Updated 2026-10-03 19:10:06 UTC by Tomasz Konieczny