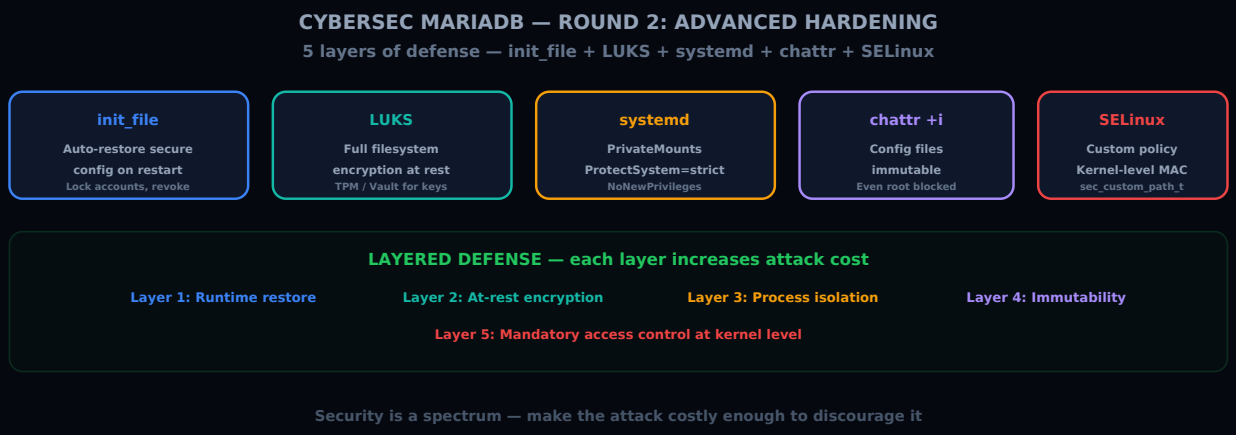
Cyberbezpieczeństwo MariaDB: runda 2

Sylvain ARBAUDIE · June 8, 2025

MARIADB SECURITY HARDENING SYSTEMD SELINUX



Poza podstawami

Pierwsza runda bezpieczeństwa MariaDB / MySQL obejmuje podstawy: silne hasła, minimalna liczba użytkowników, włączone TLS, skonfigurowany firewall. Runda 2 idzie dalej. Wkraczamy na terytorium zaawansowanego wzmacniania — technik, które niewielu DBA implementuje, ale które robią różnicę wobec zdeterminowanego atakującego.

init_file: cichy skrypt

Zmienna `init_file` MariaDB / MySQL pozwala określić plik SQL, który będzie automatycznie wykonywany przy starcie serwera. To potężne narzędzie do wzmacniania:

```
[mysqld]
init_file = /etc/mysql/conf.d/init_security.sql
```

Plik `init_security.sql` może zawierać:

```
-- Wyłączenie domyślnych kont
ALTER USER 'root'@'localhost' ACCOUNT LOCK;

-- Cofnięcie nadmiernych uprawnień
```

```
REVOKE ALL PRIVILEGES ON *.* FROM 'app_user'@'%';
GRANT SELECT, INSERT, UPDATE, DELETE ON app_db.* TO 'app_user'@'%';

-- Usunięcie testowych baz
DROP DATABASE IF EXISTS test;

-- Włączenie audytu
INSTALL SONAME 'server_audit';
SET GLOBAL server_audit_logging = ON;
```

Zaleta: nawet jeśli atakujący zmodyfikuje bazę podczas włamania, restart serwera automatycznie przywróci bezpieczną konfigurację.

LUKS: szyfrowanie systemu plików

Dane MariaDB w spoczynku powinny być zaszyfrowane. InnoDB obsługuje natywne szyfrowanie tablespace, ale LUKS (Linux Unified Key Setup) oferuje pełniejszą ochronę: szyfruje cały system plików, w tym logi, pliki tymczasowe i pliki konfiguracyjne.

```
# Tworzenie zaszyfrowanego wolumenu LUKS dla datadir
cryptsetup luksFormat /dev/sdb1
cryptsetup luksOpen /dev/sdb1 mariadb_data
mkfs.ext4 /dev/mapper/mariadb_data
mount /dev/mapper/mariadb_data /var/lib/mysql
```

Klucz LUKS nigdy nie powinien być przechowywany na tym samym dysku co dane. Użyj modułu TPM, tokena USB lub zewnętrznej usługi zarządzania kluczami (Vault, AWS KMS).

systemd: defaults-file i PrivateMounts

Plik unit systemd MariaDB może być wzmocniony na kilka sposobów:

--defaults-file jawne

```
[Service]
ExecStart=/usr/sbin/mariadbd --defaults-file=/etc/mysql/mariadb.cnf
```

Określenie `--defaults-file` uniemożliwia MariaDB odczyt innych plików konfiguracyjnych (jak złośliwy `~/.my.cnf` podrzucony przez atakującego).

PrivateMounts i przestrzenie nazw

```
[Service]
PrivateMounts=yes
ProtectHome=yes
ProtectSystem=strict
ReadWritePaths=/var/lib/mysql /var/run/mysqld /tmp
NoNewPrivileges=yes
PrivateTmp=yes
```

- **PrivateMounts=yes**: MariaDB widzi własną przestrzeń nazw montowania. Zmiany montowania dokonane przez inne procesy nie są widoczne.
- **ProtectHome=yes**: Katalog /home jest niedostępny.
- **ProtectSystem=strict**: System plików jest tylko do odczytu z wyjątkiem jawnie dozwolonych ścieżek.
- **NoNewPrivileges=yes**: Proces MariaDB nie może uzyskać nowych uprawnień (brak setuid).
- **PrivateTmp=yes**: MariaDB ma własny izolowany /tmp.

chattr: niezmienność

Atrybut `+i` (immutable) systemu plików ext4 uniemożliwia modyfikację pliku, nawet przez roota:

```
# Uczynienie pliku konfiguracyjnego niezmiennym
chattr +i /etc/mysql/mariadb.cnf
chattr +i /etc/mysql/conf.d/init_security.sql

# Weryfikacja
lsattr /etc/mysql/mariadb.cnf
# ----i--------e-- /etc/mysql/mariadb.cnf
```

Atakujący, który uzyska dostęp root, nie będzie mógł zmodyfikować konfiguracji MariaDB bez wcześniejszego usunięcia atrybutu niezmienności — co pozostawia ślady w logach audytu.

Aby legalnie zmodyfikować plik:

```
chattr -i /etc/mysql/mariadb.cnf
# ... modyfikacja pliku ...
chattr +i /etc/mysql/mariadb.cnf
```

```
systemctl restart mariadb
```

SELinux: niestandardowe polityki

SELinux w trybie enforcing to najpotężniejsza i najbardziej zaniedbywana warstwa bezpieczeństwa. MariaDB jest dostarczana z domyślną polityką SELinux, ale niestandardowa polityka może pójść znacznie dalej.

Tworzenie niestandardowego typu SELinux

```
# Definiowanie typu dla wrażliwych plików konfiguracyjnych
semanage fcontext -a -t sec_custom_path_t "/etc/mysql/conf.d(/.*)?"
restorecon -Rv /etc/mysql/conf.d/
```

Niestandardowa polityka modułu

Utwórz plik `.te` (Type Enforcement) ograniczający dostępy MariaDB:

```
# mariadb_custom.te
module mariadb_custom 1.0;

require {
    type mysqld_t;
    type sec_custom_path_t;
    class file { read open getattr };
}

# MariaDB może czytać konfiguracje, ale nie modyfikować
allow mysqld_t sec_custom_path_t:file { read open getattr };
# Brak dozwolonego zapisu do konfiguracji
```

Kompilacja i instalacja:

```
checkmodule -M -m -o mariadb_custom.mod mariadb_custom.te
semodule_package -o mariadb_custom.pp -m mariadb_custom.mod
semodule -i mariadb_custom.pp
```

Z tą polityką, nawet jeśli atakujący skompromituje proces MariaDB, nie może zmodyfikować plików konfiguracyjnych — SELinux blokuje dostęp na poziomie jądra.

Obrona warstwowa

Każda przedstawiona tu technika to warstwa obrony. Żadna nie jest wystarczająca sama w sobie. Razem tworzą znaczącą ochronę:

Warstwa	Ochrona	Przed czym
init_file	Automatyczne przywracanie	Modyfikacje konfiguracji w runtime
LUKS	Szyfrowanie w spoczynku	Kradzież fizycznego dysku
Przestrzenie nazw systemd	Izolacja procesu	Eskalacja uprawnień
chattr +i	Niezmienność konfiguracji	Modyfikacja przez skompromitowanego roota
SELinux	Obowiązkowa kontrola dostępu	Eksploitacja procesu MariaDB

Podsumowanie

Zaawansowane wzmacnianie MariaDB wymaga czasu i wiedzy. Każda dodana warstwa czyni atak trudniejszym, wolniejszym i łatwiej wykrywalnym.

Bezpieczeństwo nie jest stanem binarnym — to spektrum. Celem nie jest bycie niewrażliwym (to niemożliwe), lecz uczynienie ataku na tyle kosztownym, by atakujący przeszedł do łatwiejszego celu.

Ten artykuł został pierwotnie opublikowany na Medium.