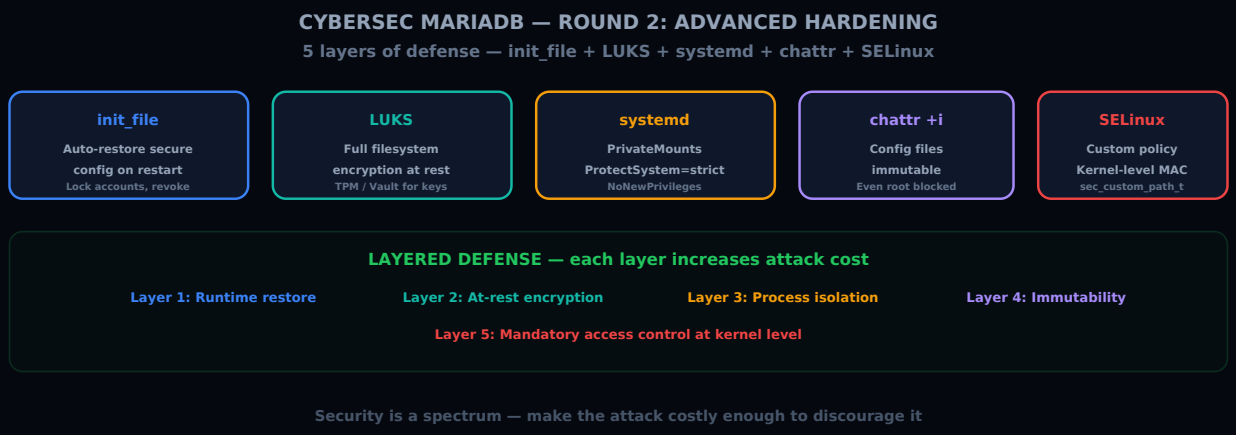
Кибербезопасность MariaDB: раунд 2

Sylvain ARBAUDIE · June 8, 2025

MARIADB SECURITY HARDENING SYSTEMD SELINUX



За пределами основ

Первый раунд безопасности MariaDB / MySQL покрывает базовые вещи: надёжные пароли, минимальные пользователи, включённый TLS, настроенный межсетевой экран. Раунд 2 идёт глубже. Мы входим на территорию продвинутого усиления защиты — техник, которые мало кто из DBA реализует, но которые делают разницу при столкновении с целеустремлённым злоумышленником.

init_file: тихий скрипт

Переменная `init_file` в MariaDB / MySQL позволяет указать SQL-файл, который будет автоматически выполнен при запуске сервера. Это мощный инструмент для усиления безопасности:

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

Файл `init_security.sql` может содержать:

```
-- Отключить учётные записи по умолчанию
ALTER USER 'root'@'localhost' ACCOUNT LOCK;
```

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

-- Удалить тестовые базы
DROP DATABASE IF EXISTS test;

-- Активировать аудит
INSTALL SONAME 'server_audit';
SET GLOBAL server_audit_logging = ON;
```

Преимущество: даже если злоумышленник изменит базу во время вторжения, перезапуск сервера автоматически восстановит безопасную конфигурацию.

LUKS: шифрование файловой системы

Данные MariaDB в состоянии покоя должны быть зашифрованы. InnoDB поддерживает нативное шифрование табличных пространств, но LUKS (Linux Unified Key Setup) обеспечивает более полную защиту: он шифрует всю файловую систему, включая логи, временные файлы и файлы конфигурации.

```
# Создать зашифрованный том LUKS для 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
```

Ключ LUKS никогда не должен храниться на том же диске, что и данные. Используйте модуль TPM, USB-токен или внешний сервис управления ключами (Vault, AWS KMS).

systemd: defaults-file и PrivateMounts

Unit-файл systemd для MariaDB можно усилить несколькими способами:

Явный --defaults-file

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

Указание `--defaults-file` не позволяет MariaDB читать другие файлы конфигурации (например, вредоносный `~/.my.cnf`, подброшенный злоумышленником).

PrivateMounts и пространства имён

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

- **PrivateMounts=yes**: MariaDB видит собственное пространство имён монтирования. Изменения монтирования, сделанные другими процессами, невидимы.
- **ProtectHome=yes**: каталог /home недоступен.
- **ProtectSystem=strict**: файловая система доступна только для чтения, кроме явно разрешённых путей.
- **NoNewPrivileges=yes**: процесс MariaDB не может получить новые привилегии (нет setuid).
- **PrivateTmp=yes**: MariaDB имеет собственный изолированный /tmp.

chattr: иммутабельность

Атрибут `+i` (immutable) файловой системы ext4 предотвращает изменение файла, даже пользователем root:

```
# Сделать файл конфигурации иммутабельным
chattr +i /etc/mysql/mariadb.cnf
chattr +i /etc/mysql/conf.d/init_security.sql

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

Злоумышленник, получивший root-доступ, не сможет изменить конфигурацию MariaDB, не сняв предварительно атрибут immutable — что оставит следы в логах аудита.

Для легитимного изменения файла:

```
chattr -i /etc/mysql/mariadb.cnf
# ... изменить файл ...
chattr +i /etc/mysql/mariadb.cnf
systemctl restart mariadb
```

SELinux: пользовательские политики

SELinux в режиме enforcing — самый мощный и наиболее игнорируемый уровень защиты. MariaDB поставляется с политикой SELinux по умолчанию, но пользовательская политика может пойти значительно дальше.

Создание пользовательского типа SELinux

```
# Определить тип для конфиденциальных файлов конфигурации
semanage fcontext -a -t sec_custom_path_t "/etc/mysql/conf.d(/.*)?"
restorecon -Rv /etc/mysql/conf.d/
```

Пользовательская политика модуля

Создайте файл `.te` (Type Enforcement), ограничивающий доступ MariaDB:

```
# mariadb_custom.te
module mariadb_custom 1.0;

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

# MariaDB может читать конфигурации, но не изменять их
allow mysqld_t sec_custom_path_t:file { read open getattr };
# Запись в конфигурации не разрешена
```

Скомпилируйте и установите:

```
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
```

С этой политикой, даже если злоумышленник скомпрометирует процесс MariaDB, он не сможет изменить файлы конфигурации — SELinux заблокирует доступ на уровне ядра.

Эшелонированная защита

Каждая представленная техника — это слой защиты. Ни одна не достаточна сама по себе. Вместе они формируют значительный бронированный щит:

Слой	Защита	Против
init_file	Автоматическое восстановление	Изменения конфигурации в runtime
LUKS	Шифрование в состоянии покоя	Физическая кража диска
systemd namespaces	Изоляция процесса	Эскалация привилегий
chattr +i	Иммутабельность конфигурации	Модификация скомпрометированным root
SELinux	Мандатный контроль доступа	Эксплуатация процесса MariaDB

Заключение

Продвинутое усиление защиты MariaDB требует времени и экспертизы. Каждый добавленный слой делает атаку сложнее, медленнее и более обнаруживаемой.

Безопасность — не бинарное состояние, а спектр. Цель — не быть неуязвимым (это невозможно), а сделать атаку достаточно дорогостоящей, чтобы злоумышленник перешёл к более лёгкой цели.

Эта статья была первоначально опубликована на Medium.