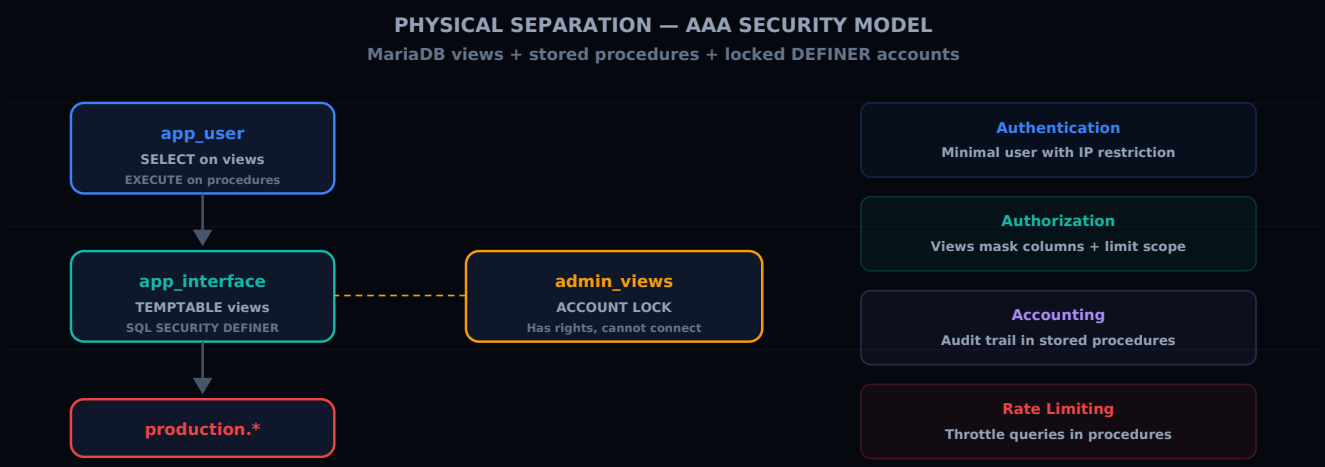
دعونا ننفصل جسديا

Sylvain ARBAUDIE · 4 نوفمبر 2024

MARIADB SECURITY ACCESS-CONTROL VIEWS



نموذج AAA المطبق على قواعد البيانات

في مجال أمن تكنولوجيا المعلومات، يعد نموذج AAA (المصادقة، الترخيص، المحاسبة) ركيزة أساسية. إنه موجود في RADIUS، وTACACS+، وجدران الحماية، والشبكات الافتراضية الخاصة... ولكن نادرًا ما يتم تطبيقه بدقة على قواعد البيانات العلائقية.

ومع ذلك، يقدم MariaDB / MySQL آليات أصلية تجعل من الممكن تنفيذ فصل مادي حقيقي بين البيانات الحساسة ومستخدمي التطبيق. لا حاجة إلى برامج وسيطة إضافية، ولا حاجة إلى وكيل باهظ الثمن. كل شيء موجود بالفعل في المحرك.

الفكرة الأساسية بسيطة: **يجب ألا يتمتع مستخدم التطبيق مطلقًا بإمكانية الوصول المباشر إلى الجداول التي تحتوي على بيانات حساسة**. يجب أن يتفاعل فقط مع طرق العرض والإجراءات المخزنة التي تم تصميمها بعناية لتعريضه فقط لما هو ضروري للغاية.

لماذا الانفصال الجسدي؟

يتكون النموذج الكلاسيكي من إعطاء مستخدم التطبيق `GRANT SELECT, INSERT, UPDATE, DELETE ON mydb.*`. إنه سريع الإعداد، لكنه يمثل كارثة أمنية:

- يمكن للمستخدم الوصول إلى **جميع** أعمدة **جميع** الجداول

- حقن SQL في التطبيق يكشف القاعدة بأكملها
- لا يوجد تتبع تفصيلي للوصول إلى البيانات الحساسة
- من المستحيل إخفاء أعمدة معينة (أرقام البطاقات، رسائل البريد الإلكتروني، كلمات المرور المجزأة)

يحل الفصل المادي هذه المشكلات عن طريق إدخال طبقة تجريد **SQL** بين التطبيق والبيانات.

الخطوة 1: إنشاء رسم تخطيطي للواجهة

```
CREATE DATABASE app_interface;
```

لن يحتوي هذا المخطط على أية جداول. طرق العرض والإجراءات المخزنة فقط. هذا هو "سطح الهجوم" المرئي للتطبيق.

الخطوة 2: إنشاء طرق العرض باستخدام ALGORITHM=TEMPTABLE

يكمن مفتاح الفصل المادي في اختيار خوارزمية العرض:

```
CREATE
  ALGORITHM = TEMPTABLE
  DEFINER = 'admin_views'@'localhost'
  SQL SECURITY DEFINER
VIEW app_interface.v_customers AS
SELECT
    customer_id,
    first_name,
    last_name,
    city,
    country
FROM production.customers;
```

ثلاثة عناصر حاسمة هنا:

- **ALGORITHM=TEMPTABLE**: MariaDB يجسد العرض في جدول مؤقت. لا يمكن للمستخدم "الرجوع" إلى الجدول الأساسي عبر `SHOW CREATE VIEW` لإنشاء استعلام مباشر.

- **تعريف**: يتم تشغيل العرض بحقوق حساب `admin_views`، وليس حقوق مستخدم التطبيق.
- **SQL محدد الأمان**: يتم إجراء عمليات التحقق من الامتيازات على المحدد، وليس على INVOKER. يحتاج مستخدم التطبيق فقط إلى حقوق العرض، وليس الجدول المصدر.

لاحظ ما هو مفقود من العرض: لا يوجد بريد إلكتروني، ولا رقم هاتف، ولا عنوان كامل. **إخفاء البيانات أمر جوهري في تصميم العرض.**

الخطوة 3: الإجراءات المخزنة للكتابة

بالنسبة لعمليات الكتابة، توفر الإجراءات المخزنة تحكمًا أفضل:

```
DELIMITER //
CREATE PROCEDURE app_interface.sp_update_customer_city(
    IN p_customer_id INT,
    IN p_city VARCHAR(100)
)
SQL SECURITY DEFINER
BEGIN
    -- Validation métier
    IF p_city IS NULL OR LENGTH(TRIM(p_city)) = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'City cannot be empty';
    END IF;

    UPDATE production.customers
    SET city = p_city,
        updated_at = NOW()
    WHERE customer_id = p_customer_id;

    -- Audit trail
    INSERT INTO production.audit_log(
        table_name, record_id, field_name,
        action, performed_by, performed_at
    )
    VALUES (
        'customers', p_customer_id, 'city',
        'UPDATE', CURRENT_USER(), NOW()
    );
END //
DELIMITER ;
```

يمكن لمستخدم التطبيق تعديل المدينة فقط. ليس الاسم، وليس البريد الإلكتروني، وليس حالة الحساب. ويتم تدقيق كل تغيير تلقائيا.

الخطوة 4: قفل حساب المسؤول

ال ينبغي أبدًا استخدام حساب DEFINER الخاص بالمشاهدات والإجراءات للاتصال:

```
CREATE USER 'admin_views'@'localhost'
    IDENTIFIED BY 'impossible_to_guess_random_string';

GRANT SELECT, INSERT, UPDATE ON production.* TO 'admin_views'@'localhost';

ALTER USER 'admin_views'@'localhost' ACCOUNT LOCK;
```

ال يمكن للحساب المقفل (`ACCOUNT LOCK`) تسجيل الدخول، لكن امتيازاته تظل نشطة للعروض والإجراءات في الوضع `SQL SECURITY DEFINER`. هذه هي النقطة الحاسمة في البنية: **الحساب الذي لديه الحقوق ال يمكنه الاتصال، والحساب الذي يتصل ليس لديه حقوق مباشرة**.

الخطوة 5: الحد الأدنى لمستخدمي التطبيق

```
CREATE USER 'app_user'@'10.0.%'
    IDENTIFIED BY 'strong_password_here';

GRANT SELECT ON app_interface.v_customers TO 'app_user'@'10.0.%';
GRANT EXECUTE ON PROCEDURE app_interface.sp_update_customer_city
    TO 'app_user'@'10.0.%';

-- Aucun GRANT sur production.*
```

ال يستطيع مستخدم التطبيق الوصول إلى أي شيء في مخطط `production`. حتى مع نجاح حقن SQL، يمكن للمهاجم فقط رؤية البيانات المكشوفة من خلال طرق العرض ويمكنه فقط تنفيذ الإجراءات المصرح بها.

إخفاء البيانات المتقدمة

تتيح المشاهدات أيضًا تقنيات إخفاء متطورة:

```
CREATE VIEW app_interface.v_customer_contacts AS
SELECT
    customer_id,
    CONCAT(LEFT(email, 3), '***@***.',
           SUBSTRING_INDEX(email, '.', -1)) AS masked_email,
    CONCAT('***-***-', RIGHT(phone, 4)) AS masked_phone
FROM production.customers;
```

يمكن لدعم العملاء التعرف على العميل من خلال آخر 4 أرقام من هاتفه دون رؤية الرقم الكامل على الإطلاق.

تحديد السعر لكل طلب

أسلوب يتم تجاهله غالبًا: استخدام الإجراءات المخزنة لتنفيذ تحديد المعدل على المستوى الأساسي:

```
CREATE PROCEDURE app_interface.sp_search_customers(
    IN p_search_term VARCHAR(100)
)
SQL SECURITY DEFINER
BEGIN
    DECLARE v_count INT;

    SELECT COUNT(*) INTO v_count
    FROM production.rate_limit
    WHERE user = CURRENT_USER()
      AND action = 'search'
      AND created_at > NOW() - INTERVAL 1 MINUTE;

    IF v_count > 10 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Rate limit exceeded: max 10 searches/minute';
    END IF;

    INSERT INTO production.rate_limit(user, action, created_at)
    VALUES (CURRENT_USER(), 'search', NOW());

    SELECT customer_id, first_name, last_name, city
    FROM production.customers
```

```
    WHERE last_name LIKE CONCAT(p_search_term, '%')
    LIMIT 50;
END;
```

ملخص الهندسة المعمارية

طبقة	مكون	الدور
التطبيق	`app_user`	تسجيل الدخول، تنفيذ طرق العرض/الإجراءات
الواجهة	`app_interface` (رسم بياني)	يعرض البيانات الضرورية فقط
الأمن	`admin_views` (مقفل)	لديه حقوق، لا يمكن الاتصال
الإنتاج	`production` (رسم بياني)	الجداول الحقيقية، لا يمكن الوصول إليها مباشرة

الحدود

هذا النهج ليس مثاليا:

- **الأداء**: `ALGORITHM=TEMPTABLE` ينشئ نسخة مؤقتة. بالنسبة للطاولات الكبيرة، قد يكون هذا مكلفاً.
- **التعقيد**: من المحتمل أن تتطلب كل وظيفة تطبيق جديدة عرضاً أو إجراءً جديداً.
- **الصيانة**: يجب أن تتطور طرق العرض مع مخطط الجداول الأساسية.

لكن هذه القيود هي ثمن الأمن. وفي سياق تكلف فيه خروقات البيانات ما متوسطه 4.5 مليون دولار لكل حادثة، يعد هذا استثماراً معقولاً.

الخلاصة

إن الفصل الفعلي عبر طرق عرض `TEMPTABLE` وإجراءات `DEFINER` المخزنة ليس ميزة غامضة في MariaDB / MySQL. إنها بنية أمنية قوية ومبتكرة وغير مستغلة في كثير من الأحيان.

خمس خطوات كافية: رسم تخطيطي للواجهة، وطرق عرض باستخدام الخوارزمية الصحيحة، وإجراءات الكتابة، وحساب محدد مقفل، والحد الأدنى من مستخدم التطبيق. والنتيجة هي قاعدة بيانات حيث حتى الحقن SQL الناجح يوفر فقط الوصول إلى جزء متحكم فيه من البيانات.

تم نشر هذه المقالة في الأصل على متوسط.