УДК 004.056.5:004.65
DOI: 10.31673/2409-7292.2024.040010

Момрик Я. Б.

ЧИСЛОВІ ТИПИ КЛЮЧІВ ДЛЯ ПРОЕКТУВАННЯ ОПТИМІЗОВАНИХ БАЗ ДАНИХ І ЗАХИСТУ ВІД SQL-АТАК

Матеріал даної статті призначений для вироблення оптимального підходу до розробки баз даних (БД) у розрізі вибору типів полів. Обґрунтовується вибір для побудови ідентифікаторів первинних та зовнішніх ключів полів числового типу з точки зору оптимальної роботи з базами даних та оптимізації реалізації захисту даних. Наголошено переваги такого вибору за критерієм швидкості виконання операцій з числовими даними та економії ресурсів і спрощення вибірок даних. Запропоновано структуру та методику валідації даних, які дозволяють мінімізувати необхідність доєднання спеціальних фреймворків при захисті даних від внутрішніх загроз та атак на зразок SQL-ін'єкцій. Проведено цілісний аналіз для реалізації рекомендацій щодо роботи з СКБД – а саме принципу "не брати на стороні сервера жодних вхідних полів без перевірки". Дане питання розглядається в аспекті правильного підбору типів полів при розробці структури баз даних для оптимізації вирішення таких питань на подальших кроках розробки. Запропоновано стратегію від побудови структури до захисної методики. Розроблено приклади реалізації валідації ідентифікаторів на сервері та методику передачі ідентифікаторів від клієнта до сервера з зазначенням довжини переданого значення. Автори приділили увагу не створенню нового захисного фреймворку, а виробленню оптимального підходу до розробки структури баз даних та розробки методики валідації даних, наведено приклади протидії атакам на її основі. Побудовано метод протидії атакам на основі поєднання ключа та довжини поля у ідентифікаторах, які передаються на сервер. У цілому, рекомендовано розробникам використовувати числові типи полів для побудови первинних та зовнішніх ключів – сформульована стратегія захисту, яка може бути використана для реалізації підходу безпечного коду відповідно до сучасних стандартів розробки.

Ключові слова: проектування баз даних, безпечний код, протидія SQL-атакам, оптимізація баз даних, методи захисту web-застосунків, ключові поля, числові типи полів баз даних.

Вступ і формулювання проблеми

Оскільки загрози даним стають настільки актуальними, що автори [1] вживають термін "кібер-війна", пропонується комплексно вирішувати дане питання, враховуючи і внутрішні загрози, пов'язані та недосконалою розробкою структури і програмного забезпечення, які в сукупності з помилками роботи персоналу приводять до порушення цілісності та втрат даних, і аспекти атак на бази даних через мережу Інтернет.

При цьому важливо дотримуватися мети оптимізації роботи з базами даних. Розглядаючи техніку оптимізації баз даних автори [2] рекомендують ефективне проектування схеми бази даних як один з кроків для досягнення продуктивності: "Проектуйте схему бази даних з урахуванням продуктивності. Оптимізуйте типи даних, використовуйте відповідні обмеження і мінімізуйте зайві відносини. Добре проектована схема може значно впливати на ефективність запитів." Вцілому можливі методи оптимізації [3] (безпосередньо сама оптимізація БД і СУБД в цілому; оптимізація взаємодії програми та MS SQL Server; оптимізація запитів) залежать від правильності виконання даного кроку.

За [4] правильний вибір (професійний підхід) до етапу проектування – шлях до забезпечення цілісності та безпеки даних і дає захист даних від внутрішніх загроз на подальших етапах роботи.

Незважаючи на існуючі стратегії захисту, SQL-ін'єкції продовжують бути ефективними засобами атаки [5]. Тому потрібні подальші дослідження та розробка нових технік та рекомендацій для ефективного запобігання цим атакам. Якщо йде мова про бази даних – першочерговими є загрози атак впровадження SQL які, залежно від типу використовуваної СКБД та умов впровадження, можуть дати можливість атакуючому виконати довільний запит до бази даних [6]. Коли атаки SQL успішні, зловмисники можуть:

увійти у програму або на веб-сайт без пароля;

отримати доступ, видобувати та видаляти збережені дані із захищених баз даних;

створювати власні записи бази даних або змінювати існуючі записи, відкриваючи двері для подальших атак.

Отже, на часі розглянути питання, як правильно розробити структуру бази ключових полів для таблиць реляційної бази даних, щоб вона відповідала можливості оптимального від вирішення вищезгаданих цілей та розглянути методику захисту, зокрема від SQL атак, з використанням даної структури на подальших етапах розробки та експлуатації.

Аналіз літератури

Матеріали про способи атак та протидії [10, 13, 14] дають інформацію про способи атак та рекомендації протидії. Аналіз літератури дає підстави для висновку що більшість атак сьогодні – це експлуатація вразливостей самого за стосунку [7]. Отже атака типу застосування SQL може бути можлива через некоректну обробку вхідних даних, що використовуються в SQL-запитах тому першочергова увага приділена валідації даних отриманих сервером від клієнта – в силу сучасних загроз їх практично заборонено брати дані з масиву request в запит без перевірки [15, 16].

Як правило будуються методики та рекомендації використання захисних фреймворків [6] – вони дозволяють прикладними програмістам уникнути зайвого коду при написанні програм для глибокої перевірки. Запропоновані фреймворки ефективні для захисту у випадку реалізації складних задач які потребують особливого захисту – зокрема при роботі з полями логіну [8].

Реалізацій найпростіших кроків на зразок переходів на сторінку наприклад з вибраним товаром без громіздких та часто дороговартісних і ресурсоємних розробок (фреймворків) у відкритому доступі бракує, а питання про структуру і типи полів у цьому аспекті взагалі не розглядається. Але SQl-ін’єкції в значній частині стосуються атак url, де присутні ключі полів та запитів на вибір чи модифікацію потрібної інформації за переданими користувацькою програмою ключами пошуку вибраної інформації. Тому саме етапу побудови структури у розрізі вибору типів полів присвячено основний акцент даного дослідження і побудова методики подається на підтвердження зручності та оптимально легкої реалізації захисту на таких типах полів.

В проаналізованій літературі не зустрічалося рекомендацій для такої протидії, виходячи з оптимальної побудови структури бази, зокрема полів первинних та зовнішніх ключів – дане дослідження про правильний підбір структури таблиць і ключових полів – у продовження сказаного у [4] обґрунтовуємо, що це є першим кроком і для побудови оптимального захисту і від внутрішніх помилок і від зовнішніх атак. Врахувавши вимоги професійної реалізації і особливостей самих атак [12] – приходимо до висновку, що комплексний підхід дає оптимальний захист в сучасних умовах.

Тому автори приділи увагу не створенню чергового захисного фреймворку, а виробленню оптимального підходу до розробки структури баз даних та розробки методики валідації даних для методики захисту, яка може бути використана як безпечний код і є одночасно методом оптимізації роботи з базами даних.

Мета та завдання дослідження

Мета дослідження полягає у обґрунтуванні рекомендацій про вибір числових типів полів для побудови ключових полів у базах даних для оптимізації роботи та протидії SQL-ін’єкціям, розробці практичних стратегій та побудові методу протидії цим атакам з використаннями такої структури бази даних. Завдання:

1. Розглянути доцільність побудови структури бази на основі концепції базових ідентифікаторів з використанням числових полів. Обґрунтувати та показати способи їх використання з точки зору оптимізації та захисту від внутрішніх та зовнішніх загроз.

2. Розробити оптимальний підхід – спрощену методику протидії атакам, котра може використовуватися як без задіяння фреймворків спеціального призначення так і в їх складі.

3. Розробити методику захисту від спотворення значень ідентифікаторів при пересилці ідентифікаторів від клієнта до сервера з зазначенням довжини переданого значення та продемонструвати її роботу з застосуванням приладів програмної реалізації.

© Момрик, Я. Б. (2024). Числові типи ключів для проектування оптимізованих баз даних і захисту від SQL-атак. Сучасний захист інформації, 4(60), 100–108. https://doi.org/10.31673/2409-7292.2024.040010.

1. Обґрунтування вибору числового типу ідентифікаторів для захисту БД

1.1. Обґрунтування підходів до проектування структури та демонстрація прикладів захисту. Якщо ідентифікатори первинних та зовнішніх ключів, які як правило використовуються для переходу на потрібну сторінку чи вибору певної інформації, побудовані з використанням цілого числа, то для запобігання атакам легко без зайвих фреймворків відсікти більшість загроз. Це особливо актуально, якщо працюємо з базою даних з використанням вразливих технологій (технології відмінні від ORM) – тому приклади побудовано на такій технології (php+MySql). Далі продемонстровано, що get посилання у web з передачею параметрів чисельного типу набагато легше контролювати, і захищати від SQL-атак, ніж символьні.

Виконання такої задачі реалізується з меншими зусиллями і меншим навантаженням на ресурси, ніж коли працюємо з символьними полями. Якщо у web додатках для навігації передавати числові значення – код для валідації не потребує значних зусиль ні від програміста, ні від середовища виконання.

Найпростіша валідація захищає від цілого ряду атак при числових типах ідентифікаторів.

Розглянемо стратегію, яка приходить на думку з досвіду розробки та здається найпростішою але необхідною після аналізу літератури [10, 13, 14] про способи атак – приклади нижче містять пояснення щодо способів атак яким вони протидіють а саме:

атаки вкраплення SQL,

атаці з використанням UNION,

атаки з використанням розділювача SQL-запитів,

атаки дописування числових даних до ідентифікаторів,

атаку прикладного рівня на вичерпання ресурсів – передача пробілу.

Розглянемо переваги використання числових ідентифікаторів на прикладі коли клієнтська форма надсилає ідентифікатор на сервер з допомогою наступного коду php (метод Get):

```
<a href=\"form4.php?pid={$row['id_person']}\">Edit</a>
```

Тоді параметри будуть передаватися через url сторінки (рис. 1).

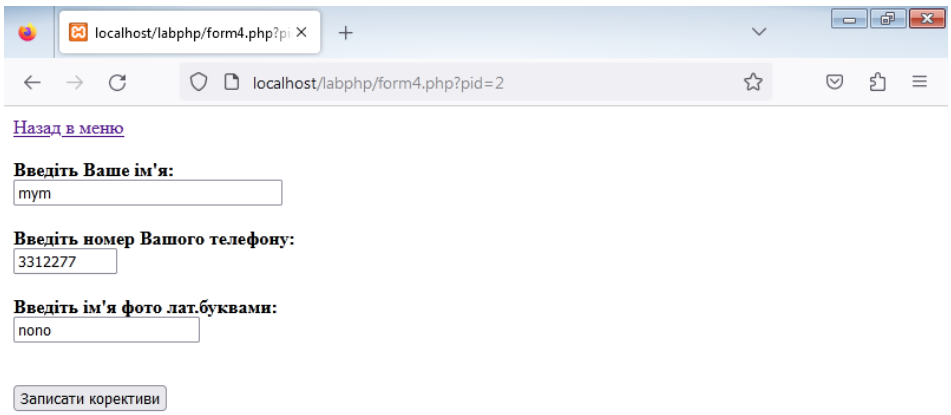

Рис. 1. Передача ідентифікаторів через url сторінки

Використовуючи числові ідентифікатори, легко відсікти вищевказані атаки, якщо використати найпростішу валідацію на зразок наведеної в блоці коду нижче (виділено шрифтом) – перевірка на наявність непустого ідентифікатора і чи він є числом відкидає можливість дописування виконуваного шкідливого коду. Причому цей метод валідації буде спрацьовувати для різних СКБД – незалежно від того, чи перетворюють вони символьний параметр у числовий у випадку невідповідності типу.

© Момрик, Я. Б. (2024). Числові типи ключів для проектування оптимізованих баз даних і захисту від SQL-атак. Сучасний захист інформації, 4(60), 100–108. https://doi.org/10.31673/2409-7292.2024.040010.

Захист від атак емуляції навантаження. Якщо йдеться про професійну реалізацію, яка буде захищати від внутрішніх загроз через неуважність працівника і зовнішніх – через свідому атаку прикладного рівня [9], що має на меті вичерпати ресурси сервера баз даних, то слід також звернути увагу на дії, коли сервер просто завантажують запитами для створення навантаження. Тому якщо програміст пропускає у запит пусті поля – наприклад коли пробіл поставили у поле пошуку чи передали пустий ідентифікатор у get параметрі пошуку, то він стає провідником тої атаки, тому необхідно перевіряти такі моменти відразу і не формувати зайвого запиту до сервера. У прикладах нижче цей момент перевіряється перед формуванням запитів і в числових, і в символьних змінних переданих на сервер.

```
    ?php
    include_once 'connecta.php';
    $usertable="userdata";
    // вибрати значення по ідентифікатору і перевірити
    if(isset($_REQUEST['pid']) && trim($_REQUEST['pid']) != ''&&
is_numeric($_REQUEST['pid'])){
    $idv=trim($_REQUEST['pid']);
    print "$idv";
    $rr=intval($idv);
    $query="select * from $usertable where id_person=$idv";
    if ($result=@mysqli_query($dbc, $query))
    {
    $row = $result->fetch_row();
    print "<form method=post action='form7.php'> ";
    print "<br><b>Введіть Ваше ім'я:</b>";
    print "<br><input name='user_name' value='{$row[1]}' size=30>";
    print "<br><br><b>Введіть номер Вашого телефону:</b>";
    print "<br><input name='phone' value='{$row[2]}' size=10>";
    print "<br><br><b>Введіть ім'я фото лат.буквами:</b>";
    print "<br><input name='img' value='{$row[3]}' size=20>";
    print  "<br><input  type='hidden'  name='pid'  value='{$idv}'
size=20>";
    print     "<br><br><input     type='submit'     value='Записати
корективи'>";
    print "</form>";
    $result->close();
    }
    }
    else
    {
    print "Ідентифікатор не пройшов валідацію";
```

Принцип атаки вкраплення SQL. Якщо програма, яка обробляє запит, використає наведену нижче валідацію – то зловмисник нічого не доб’ється, заміняючи `id=2` на `id=-1 OR 1=1` (рис. 2).

Рис. 2. Перехоплення атаки вкраплення SQL

При атаці з використанням UNION. Якщо хакер передасть в id щось на зразок `-1 UNION SELECT select * from userdata` запит теж не виконається бо він містить символи – валідація його відкине (рис. 3).

Рис. 3. Перехоплення атаки з використанням UNION

При атаці з використанням розділювача SQL-запитів. Для розділення команд у мові SQL використовується символ “;” (крапка з комою) впроваджуючи цей символ у запит, зловмисник отримує можливість виконати кілька команд в одному запиті, проте не всі діалекти SQL підтримують таку можливість. Наприклад якщо хакер спробує добавити запит на знищення – `pid=2; delete * from userdata` – він також не спрацює (рис. 4).

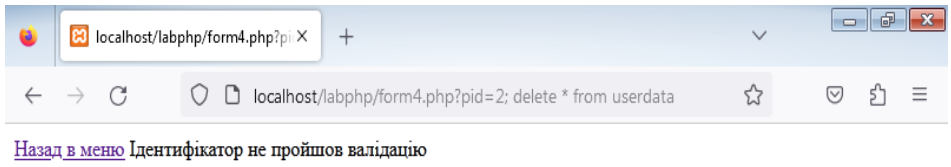

Рис. 4. Перехоплення атаки з використанням розділювача SQL-запитів

Цю валідацію також можна використати і для створення виключень і переривання роботи у випадку виявлення невідповідності довжини – якщо це потрібно для реалізації задачі.

Для символьних значень краще використовувати плейсхолдери – параметризовані запити. Однак підготовлені параметризовані запити сповільнюють сервер і створюють більше навантаження – тому числові ідентифікатори дають оптимізацію при програмуванні роботи з базами даних. Цей спосіб дозволяє працювати лиш з валілними ідентифікаторами. Якщо ідентифікатор не пройшов валідацію, робота програми переривається. Окрім того, якщо розглянути передачу ідентифікаторів через get у логіці PEN-тестування та врахувати що SQL-ін’єкції передбачають можливість дописування даних до ідентифікаторів ([10] пропонує для таких випадків метод усічення до максимально дозволеної довжини) – слід врахувати випадок, коли розробленої вище валідації недостатньо. Така ситуація виникне, якщо хакер спробує перенаправивши на інший запис – дописувати цифри до сформованого шляху url з переданим параметром, щоб спровокувати порушення цілісності та зіпсувати дані приміром при редагуванні запису.

Наприклад, користувацька програма вибрала запис з ідентифікатором 2 (рис. 5), а хакер дописав ще цифри – ідентифікатор став 22 і беруться в роботу зовсім інші дані (рис. 6).

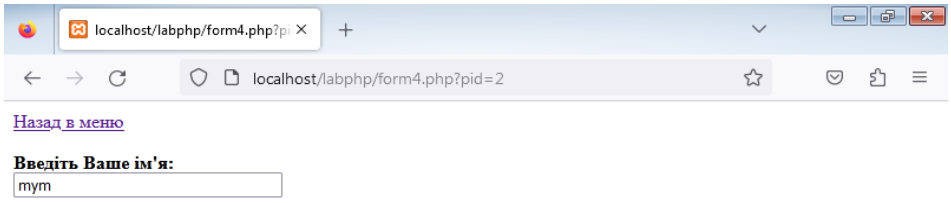

Рис. 5. Передача користувацькій програмі коректного ідентифікатора

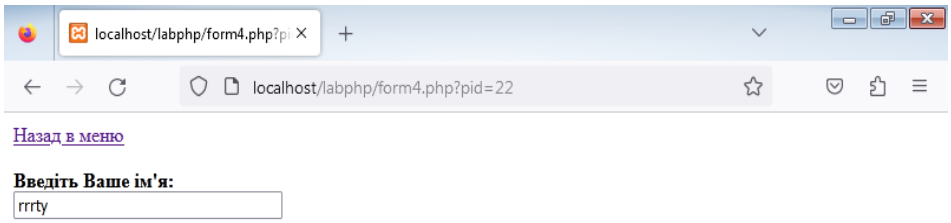

Рис. 6. Вибір інших даних при дописуванні даних до ідентифікатора (=22)

1.2. Метод додавання дробової частини. Розвиток найпростішого методу валідації числових ідентифікаторів для захисту параметрів шляхом фіксації і контролю довжини параметра. Для захисту від атак потрібно переконатися, що отримали саме той id, який вибрав користувач. Щоб контролювати довжину параметра – можна видозмінити числовий ідентифікатор-код для передачі його у програму, перетворивши його у тип float – де в дробовій частині міститиметься довжина ідентифікатора.

Ідентифікатор буде надсилатися на сервер у вигляді – “значення ідентифікатора”.“значення довжини”.

```
    print "<form method=post action='form4.php'>";
    print "<table>";
    print "<tr><td>Назва користувача
</td><td>Телефон</td><td></td></tr>";
    $rows = $result->fetch_all(MYSQLI_ASSOC);
    foreach ($rows as $row)
    {
     $ilen=0;
     $ilen=strlen((string)(trim($row['id_person'])));
     $st=strlen((string)(trim($ilen)));
     $llen=(int)$row['id_person']+$ilen/10**$st;
    print
"<tr><td>{$row["name"]}</td><td>{$row["pfone"]}</td><td><a
href=\"form4.php?pid=$llen\">Edit</a></td></tr>";
    }
    print "</table>";
    print "</form>";
```

Тоді програма, котра обробляє дані на сервері, може перевірити довжину і або від’єднати зайві символи або перервати виконання – у прикладі нижче – виводиться попередження.

```
    <?php
    include_once 'connecta.php';
    $usertable="userdata";
    // вибрати значення по pid
    if(isset($_REQUEST['pid']) && trim($_REQUEST['pid']) != ''&&
is_numeric($_REQUEST['pid'])){
      $idv=floor(trim($_REQUEST['pid']));
      $len=round(trim($_REQUEST['pid'])-$idv,2);
      $llen=(int)(substr((string)($len),2));
      if(strlen((string)$idv)!= $llen)
      {
         print "LEN ID IS BAD";
      }
```

Ось на рис. 7 проілюстрована робота при коректній довжині ідентифікатора (=2) на рис. 7 – в url 19.2, де 2 – довжина ідентифікатора.

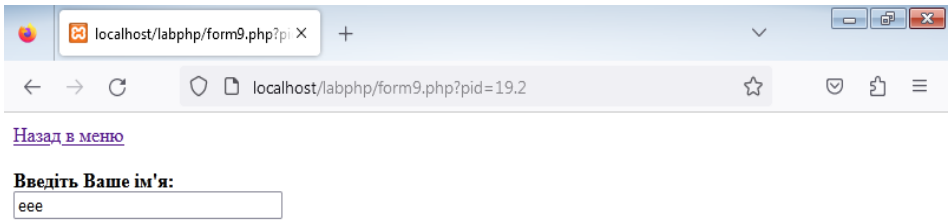

Рис. 7. Робота програми з застосуванням методу додавання дробової частини при коректній довжині ідентифікатора (=2)

Проілюстрована робота додатка при коректній довжині ідентифікатора (=1) – на рис. 8 – в url 1.1, де 1 у дробовій частині – довжина ідентифікатора.

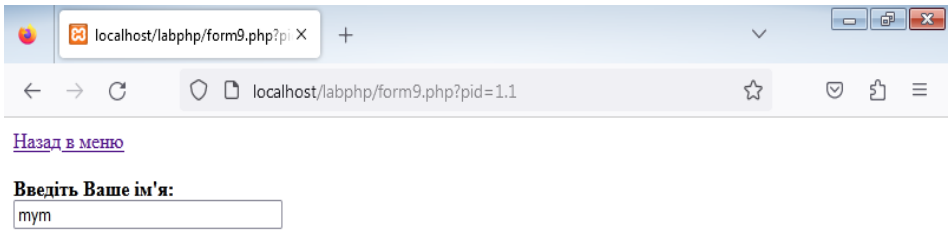

Рис. 8. Робота програми з застосуванням методу додавання дробової частини при коректній довжині ідентифікатора (=1)

На рис. 9 показано, як програма видає попередження, оскільки довжина числового ідентифікатора не рівна числу в дробовій частині 1 (дописано 9 до 1 у цілій частині числа).

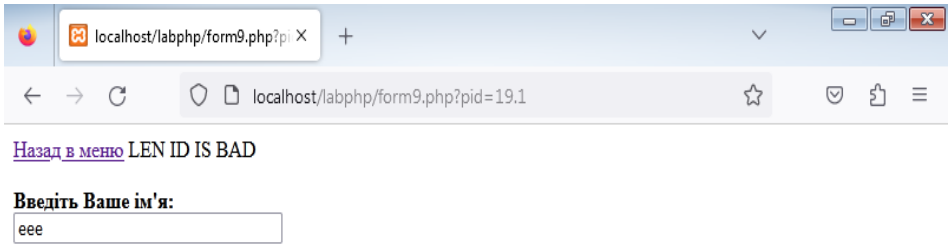

Рис. 9. Перехоплення хакерської зміни ідентифікатора методом додавання дробової частини

Для ілюстрації роботи методу запрограмовано видачу попередження, але можна запроектувати відповідно до потреб проекту створення переривань або обробку таких ситуацій в залежності, чи ідентифікатор за довжиною більший, чи менший за задане у дробовій частині число. Але краще протоколювати – збирати інформацію про такі збої, щоб адміністратор міг проаналізувати спосіб атаки та вибрати стратегію як описано [15] – у розрізі корисності використання протоколювання.

Якщо вектор атаки побудовано таким чином, що вкінці вже сформованого дробового числа дописуються цифри – то це не зашкодить роботі вибірок – ідентифікатори будуть правильні, бо дописуватимуться цифри в дробову частину – а вона відкидається автоматично в даному методі – тому стандартно видається лише попередження, а не переривається робота.

Отже цей метод покращує продуктивність захисту – у результаті валідації можна продовжувати роботу, навіть якщо була спроба атаки – у всіх випадках, коли до ідентифікатора додається шкідливий код. Йдеться про первинні SQL-інфекції, передані методом GET, які базуються на дописуванні шкідливого коду до даних, а не на повній заміні

© Момрик, Я. Б. (2024). Числові типи ключів для проектування оптимізованих баз даних і захисту від SQL-атак. Сучасний захист інформації, 4(60), 100–108. https://doi.org/10.31673/2409-7292.2024.040010.

переданого значення. Дописаний код ігнорується, і сервер може отримати правильний ідентифікатор з цілої частини числа.

Попередження про спроби атаки можна протоколювати та аналізувати журнал збоїв.

Запропонований підхід можна застосувати і як удосконалення до методу усічення – коли у параметрі відсікаються зайві знаки, щоб запобігти переповненню змінних. Наприклад можна обрізати ідентифікатор – щоб його довжина мала 1 знак як зафіксовано у його дробовій частині. Тоді у запит буде передано сааме той ідентифікатор який надіслав клієнт. Хакер може розгадати спосіб кодування – але такий спосіб допоможе запобігти векторним атакам з допомогою роботів та атакам шляхом дописування знаків вкінці числа.

2. Інші переваги, в тому числі в розрізі оптимізації, при виборі числових типів ідентифікаторів БД

Окрім зручності та оптимальності використання числових ідентифікаторів для валідації і захисту даних від внутрішніх та зовнішніх загроз розглянемо ще ряд аргументів оптимізації роботи з базами даних виходячи з яких можна рекомендувати розробникам використовувати числові типи полів для побудови первинних та зовнішніх ключів.

Ось практичні аргументи на користь такого підходу:

1**. Для формування ключів**. З числом менша ймовірність помилок, в тому числі помилок при побудові зовнішнього з’єднання – якщо це ключове поле – його або генерує процедура котра використовує щось на зразок secuence або в структурі задано як автоінкрементне або є процедура котра генерує шифр – тобто це поле не набирається руками – відповідно не виникає помил від клієнта на сервер через випадкову зміну мови шрифта та кодування.

2**. Для індексації (один з найважливіших способів оптимізації).** Якщо індексувати – бо на первинний ключ та зовнішні ключі створюються індекси на рівні СКБД – такий індекс (в тому числі кластерний) буде займати менше місця. Таким чином уникнемо створення [3] індексів, які для свого обслуговування використовують більше витрат, ніж приносять виграш.

3. **Для швидкості пошуку** (в тому числі з використанням індексних полів). Для того, щоб СКБД порівняло дві стрічки – потрібно посимвольно порівняти дві стрічки. За числовими ідентифікаторами пошук відбувається швидше в силу самої технології виконання такої операції для чисел.

4. **Для зменшення ймовірності помилок.** При вибірці даних по join, при зв’язку за допомогою символьних полів є більша ймовірність помилок через неуважність, (якщо допускається, що такі поля корегуються вручну), зміну кодування, чи мови (деякі букви такі як 'а', однаково виглядають на різних мовах, але мають різні коди), отже ймовірність порушення цілісності через внутрішні помилки набагато менша для числових полів, тому саме їх варто закладати як поля зовнішніх ключів.

5. **Для зменшення громіздкості.** щоб уникнути партиціювання [2] – яке рекомендується як метод оптимізації Таблиці, котрі будуть мати в посиланнях на інші таблиці саме числа будуть менш громіздкими за розміром.

Висновки і рекомендації

Вищенаведений аналіз дозволяє стверджувати, що використання для полів первинних ключів та зовнішніх ключів індексних типів при проектуванні таблиць баз даних дає змогу полегшити захист від SQL-ін’єкцій та оптимізувати роботу з базою даних. Виходячи з такої структури побудовано схематичну методику валідаціїі та методику захисту від підміни чи спотворення ключових ідентифікаторів при передачі від клієнта до сервера. Метод гнучкий: у дробовій частині можуть бути застосовані інші критерії для перевірки валідності, наприклад контрольна сума. Даний матеріал втілює застосування підходу “безпечний код”, починаючи з етапу проектування баз даних – при закладенні структури таблиць.

На майбутнє планується удосконалити метод для розпізнавання способу SQL атаки: якщо викривлюються дані саме в цілій частині – надати можливості прийняття рішення на

© Момрик, Я. Б. (2024). Числові типи ключів для проектування оптимізованих баз даних і захисту від SQL-атак. Сучасний захист інформації, 4(60), 100–108. https://doi.org/10.31673/2409-7292.2024.040010.

основі запротокольованої статистики таких збоїв про вибір способу протидії – наприклад чи обрізати ідентифікатор перед формуванням запиту до потрібної величини або чи формувати переривання, метод можна оформити наприклад у вигляді функції, яка дозволяє вибір режиму обробки помилок або має інтелектуальну складову, що пропонує такий спосіб. Іншим напрямком розвитку такого підходу може бути використання контрольної суми цілочисельного ідентифікатора, що дозволить запобігти спробам неавторизованого доступу до даних шляхом повної підміни цифр ідентифікатора.

Перелік посилань

1. Haiduk, O., & Zverev, V. (2024). Analysis Of Cyber Threats In The Context Of Rapid Development Of Information Technology. Cybersecurity: Education, Science, Technique, 3(23), 225–236. https://doi.org/10.28925/2663-4023.2024.23.225236.

2. Foo, D. (2023, 13 листопада). 11 Database Optimization Techniques. Medium. https://danielfoo.medium.com/11-database-optimization-techniques-97fdbed1b627.

3. Чихіра, І., Левицький, В., & Микитишин, А. (2019). Етапи оптимізації баз даних. У Матеріали IV Міжнародної науково-технічної конференції “Теоретичні та прикладні аспекти радіотехніки, приладобудування і комп’ютерних технологій” присвячена 80-ти річчю з дня народження професора Я.І. Проця (с. 256–257). ФОП Паляниця В. А. https://elartu.tntu.edu.ua/bitstream/lib/28848/2/TPARP_2019_Chyhira_I Stages_optimization_based_256-257.pdf.

4. Momryk, Y., Yashchuk, Y., & Tuchapskyi, R. (2024). A Professional Approach As A Method Of Protecting Information At The Stages Of Development Of Relational Databases And Software For Working With Them. Cybersecurity: Education, Science, Technique, 3(23), 42–55. https://doi.org/10.28925/2663-4023.2024.23.4255.

5. Starchikov, S. (2023, 30 травня). Протидія кібернетичним загрозам у вигляді SQL-ін’єкцій. Medium. https://medium.com/@serhii.starchikov/протидія-кібернетичним-загрозам-у-вигляді-sql-інєкцій-9bad2164fb7e

6. Харківський національний університет внутрішніх справ & Наукове товариство студентів, курсантів, слухачів, аспірантів, ад’юнктів, докторантів і молодих вчених. (2019). У Сучасна наука і правоохоронна діяльність. ХНУВС. http://dspace.univd.edu.ua/xmlui/handle/123456789/6855.

7. Shibghatullah, A. S. B., Fatlawi, H. K., Kadhim, S., Falih, M., Ali, N. S., & Alhilali, A. H. (2020). A Comparative Analysis and Performance Evaluation of Web Application Protection Techniques against Injection Attacks. International Journal of Mobile Communications, 18(1), 1. https://doi.org/10.1504/ijmc.2020.10019530.

8.. Куперштейн, Л., Луцишин, Г., & Кренцін, М. (2024). Інформаційна технологія моніторингу безпеки даних програмного забезпечення. Електронне фахове наукове видання “Кібербезпека: освіта, наука, техніка”, 3(23), 71–84. https://doi.org/10.28925/2663-4023.2024.23.7184

9. Haiduk, O., & Zverev, V. (2024). Analysis Of Cyber Threats In The Context Of Rapid Development Of Information Technology. Cybersecurity: Education, Science, Technique, 3(23), 225–236. https://doi.org/10.28925/2663-4023.2024.23.225236.

10. SQL Injection. (б. д.). http://sites.znu.edu.ua/webprog/lect/1230.ukr.html.

11. Secure Coding Best Practices Handbook | Veracode. (б. д.). Resources - Reports, Success Stories, and More | Veracode. https://info.veracode.com/secure-coding-best-practices-hand-book-guide-resource.html.

12. SET University. (2022, 27 жовтня). Найвідоміші вразливості веб застосунків. XSS та SQL ін’єкції, вразливості автентифікації. Спільнота програмістів | DOU. https://dou.ua/forums/topic/40613/.

13. Що таке SQL-ін'єкція? (б. д.). UKEY WAF - Надійний захист веб-сайту. https://ukeywaf.com/baza/shho-take-sql-inyekcziya/.

14. What is SQL Injection? Tutorial & Examples | Web Security Academy. (б. д.). Web Application Security, Testing, & Scanning - PortSwigger. https://portswigger.net/web-security/sql-injection.

15. PHP: SQL Injection - Manual. (б. д.). PHP: Hypertext Preprocessor. https://www.php.net/manual/en/security.database.sql-injection.php.

16. SQL injection - SQL Server. (б. д.). Microsoft Learn: Build skills that open doors in your career. https://learn.microsoft.com/en-us/sql/relational-databases/security/sql-injection?view=sql-server-ver16.

Надійшла 23.11.2024