تشریح حملات تزریق اس کیو ال

(SQL Injection Introduction)

بررسی اجمالی

حمله تزریق[۱] SQL شامل ورود یا "تزریق" یک جستار SQL توسط داده‌های ورودی از مشتری به برنامه می‌باشد. استفاده از[۲]تزریق SQL موفق می‌تواند داده‌های حساس را از پایگاه داده بخواند، داده‌های پایگاه داده را اصلاح (ورود، به روز رسانی، پاک‌سازی) کند، عملیات مدیریتی را روی پایگاه داده اجرا کند (مانند خاموش کردن DBMS)، محتوای یک فایل داده‌شده حاضر روی سیستم فایلی DBMS را بازیابی کند و در بعضی موارد دستورات را به سیستم عامل[۳]بفرستد. حملات تزریق SQL یک نوع حمله تزریق هستند که در آن دستورات SQL به ورودی سطح داده[۴]تزریق می‌شوند تا اجرای دستورات SQL از پیش تعریف‌شده را انجام دهد.

مدل‌سازی تهدید

- حملات تزریق به مهاجمان اجازه می‌دهد تا هویت را جعل کند، داده‌های موجود را دستکاری[۵] کند، باعث انکار اشکالاتی نظیر لغو تراکنش‌ها یا تغییر تعادلات شود، اجازه افشای[۶]کامل همه داده‌های سیستم را بدهد، داده‌ها را تخریب کند یا در غیر اینصورت آن را دسترس‌ناپذیر کند، و مدیران سرور پایگاه داده شود.

---

۱ injection attack

۲ exploit

۳ operating system

۴ data-plane

۵ tamper

۶ disclosure

- تزریق SQL توسط برنامه‌های PHP و ASP به دلیل شیوع رابط‌های عملکردی قدیمی‌تر رایج است. با توجه به طبیعت رابط‌های برنامه‌نویسی در دسترس، برنامه-های J2EE و ASP.NET به احتمال کمتر به آسانی از تزریق‌های SQL استفاده می‌کنند.
- شدت حملات تزریق SQL توسط مهارت و تخیل مهاجم ، و به میزان کمتر، دفاع در عمق اقدامات متقابل[۱]، مانند اتصالات کم‌امتیاز[۲] به سرور پایگاه داده و غیره محدود می‌شود. به طور کلی، تزریق SQL را به شدت تحت تأثیر قرا دهید.

فعالیت‌های مربوط به امنیت

چگونه از آسیب‌پذیری‌های تزریق SQL جلوگیری کنیم

برگه رمز ممانعت تزریق OWASP SQL را ببینید.

برگه رمز پارامتری کردن جستار OWASP را ببینید.

مقاله راهنمای OWASP در مورد چگونگی اجتناب از آسیب‌پذیری‌های تزریق SQL را ببینید.

نحوه بررسی کد برای آسیب‌پذیری‌های تزریق SQL

مقاله راهنمای بررسی کد[۳] OWASP در مورد چگونگی بررسی کد برای آسیب‌پذیری-های تزریق SQL را ببینید.

نحوه تست کردن آسیب‌پذیری‌های تزریق SQL

مقاله راهنمای آزمون OWASP در مورد چگونگی آزمایش آسیب‌پذیری‌های تزریق SQL را ببینید.

---

۱ countermeasures

۲ privilege

۳ Code Review Guide

چگونه با استفاده از دیوارهای آتش[1]کاربردی وب با SQLi کنار بیاییم

مقاله OWASP در مورد استفاده از تزریق SQL برای عبور از WAF را ببینید.

شرح

خطاهای تزریق SQL زمانی رخ می‌دهد که:

۱. داده‌ها از یک منبع نامعتبر وارد برنامه می‌شوند.

۲. داده‌ها به طور پویا برای ساختن یک جستار SQL مورد استفاده قرار می‌گیرند.

نتایج اصلی شامل موارد زیر هستند:

- **محرمانگی[2]:** از آنجا که پایگاه‌های داده SQL به طور کلی داده‌های حساس را نگه می‌دارند، از دست رفتن محرمانگی یک مشکل مکرر با آسیب‌پذیری‌های تزریق SQL است.
- **احراز هویت[3]:** اگر دستورات SQL ضعیف برای بررسی نام کاربری و کلمه عبور استفاده شوند، ممکن است به یک سیستم به عنوان یک کاربر دیگر بدون هیچ دانش قبلی از کلمه عبور وصل شود.
- **مجوز[4]:** اگر اطلاعات مجوز در یک پایگاه داده SQL نگهداری شود، ممکن است این اطلاعات را از طریق بهره‌برداری موفقیت‌آمیز از آسیب‌پذیری تزریق SQL تغییر دهد.

---

۱ Firewalls

۲ Confidentiality

۳ Authentication

۴ Authorization

- **یکپارچگی**[1]**:** درست همانطور که ممکن است اطلاعات حساس را بخواند، همچنین ممکن است تغییراتی انجام دهد یا حتی این اطلاعات را با یک حمله تزریق SQL حذف کند.

عوامل خطر

زیرساخت[2]تحت تأثیر واقع شده می‌تواند موارد زیر باشد:

- زبان: SQL
- زیرساخت: هر (چیزی که نیاز به تعامل با پایگاه داده SQL دارد.)

تزریق SQL به یک موضوع رایج با وب‌سایت‌های مبتنی بر پایگاه داده تبدیل شده است. این عیب به راحتی شناسایی و به آسانی مورد سوء استفاده قرار می‌گیرد، و به همین ترتیب، هر سایت یا بسته نرم‌افزاری با حتی یک پایه کاربری حداقل، احتمالا با حمله تلاش‌شده از این نوع مواجه شود.

اساسا، حمله با قرار دادن یک فوق‌کاراکتر[3]در ورودی داده‌ها انجام می‌شود تا آنگاه دستورات SQL در صفحه کنترل قرار گیرد، که قبلا وجود نداشت. این نقص به این واقعیت بستگی دارد که SQL هیچ تمایزی بین صفحه‌های کنترل و داده‌ها ایجاد نمی‌کند.

مثال‌ها

مثال ۱

در SQL:

```
select id, firstname, lastname from authors
```

---

۱ Integrity

۲ platform

۳ meta character

اگر یکی ارائه شد:

```
Firstname: evil'ex
Lastname: Newman
```

رشته[1] جستار تبدیل می‌شود به:

```
select id, firstname, lastname from authors where forename = 'evil'ex' and surname
='newman'
```

که پایگاه داده تلاش می‌کند که به صورت زیر اجرا شود:

```
Incorrect syntax near il' as the database tried to execute evil.
```

نسخه امن دستور SQL بالا می‌تواند در جاوا به صورت زیر کدگذاری شود:

```
String firstname = req.getParameter("firstname");
String lastname = req.getParameter("lastname");
// FIXME: do your own validation to detect attacks
String query = "SELECT id, firstname, lastname FROM authors WHERE forename = ? and
surname = ?";
PreparedStatement pstmt = connection.prepareStatement( query );
pstmt.setString( 1, firstname );
pstmt.setString( 2, lastname );
try
{
        ResultSet results = pstmt.execute( );
}
```

[1] string

مثال ۲

کد C# زیر به صورت پویا یک جستار SQL را می‌سازد و اجرا می‌کند که موارد را با منطبق کردن یک نام مشخص‌شده جستجو می‌کند. جستار عناصر نمایش داده شده را به آن‌هایی که در آن مالک، بر نام کاربری کاربر تایید شده در حال حاضر منطبق است، محدود می‌کند.

```
...
        string userName = ctx.getAuthenticatedUserName();
        string query = "SELECT * FROM items WHERE owner = '"
                                        + userName + "' AND itemname = '"
                                        + ItemName.Text + "'";
        sda = new SqlDataAdapter(query, conn);
        DataTable dt = new DataTable();
        sda.Fill(dt);
        ...
```

جستاری که این کد در نظر دارد اجرا شود، به شرح زیر است:

```
SELECT * FROM items
        WHERE owner =
        AND itemname = ;
```

با این حال، چون جستار به صورت پویا بوسیله اتصال[۱] یک رشته جستار پایه ثابت و یک رشته ورودی کاربر ایجاد می‌شود، جستار فقط وقتی به درستی رفتار می‌کند که نام عنصر حاوی هیچ کاراکتر تک-عبارتی[۲]نباشد. اگر مهاجم با نام کاربری wiley وارد رشته "name' OR 'a'='a" برای نام عنصر شود، آنگاه جستار به شرح زیر می‌باشد:

```
        SELECT * FROM items
```

---

۱ concatenating

۲ single-quote

```
WHERE owner = 'wiley'
AND itemname = 'name' OR 'a'='a';
```

علاوه بر این، شرط OR 'a'='a' باعث می‌شود که شرط[۱] همیشه به درستی ارزیابی شود، بنابراین جستار به طور منطقی با جستار ساده‌تر معادل می‌شود:

```
SELECT * FROM items;
```

این ساده‌سازی جستار به مهاجم اجازه می‌دهد تا از الزامی عبور کند که بر طبق آن جستار فقط مواردی را که متعلق به کاربر تأیید شده است، برمی‌گرداند; در حال حاضر، جستار همه ورودی‌های ذخیره شده در جدول موارد را بدون در نظر گرفتن صاحب مشخص آن‌ها بازمی‌گرداند.

مثال ۳

این مثال اثرات مقدار مخرب[۲] مختلف گذرا به جستار ساخته شده و اجرا شده در مثال ۱ را بررسی می‌کند. اگر یک مهاجم با نام کاربری هکر وارد رشته "name'); DELETE FROM items; --" برای نام موردی شود، آنگاه جستار به دو جستجوی زیر تبدیل می‌شود:

```
SELECT * FROM items
WHERE owner = 'hacker'
AND itemname = 'name';

DELETE FROM items;

--'
```

بسیاری از سرورهای پایگاه داده، از جمله مایکروسافت ® SQL Server 2000، اجازه می‌دهد تا چندین دستور SQL جدا شده توسط نقطه‌ویرگول به طور همزمان اجرا شوند.

---

۱ clause

۲ malicious

در حالی که این رشته حمله منجر به خطا در اوراکل[1] و سایر سرورهای پایگاه داده می‌شود که اجازه اجرای گروهی دستورات جدا شده توسط نقطه ویرگول را نمی‌دهند، در پایگاه داده‌هایی که اجازه اجرای گروهی را می‌دهند، این نوع حمله به مهاجم اجازه می‌دهد که دستورات اختیاری دلخواه را علیه پایگاه داده اجرا کند.

توجه داشته باشید که جفت خط‌تیره انتهایی[2] (--)، که در اکثر سرورهای پایگاه داده مشخص می‌کند که با باقی‌مانده عبارت به عنوان یک نظر و نه اجرا شده، رفتار شود. در این مورد، کاراکتر نظر به منظور حذف تک نقل قول انتهایی از سمت چپ جستار اصلاح-شده به کار می‌رود. در پایگاه داده‌ای که نظرات اجازه استفاده از این روش را ندارند، حمله عمومی هنوز می‌تواند با استفاده از فن مشابه با آنچه که در مثال ۱ نشان داده شد، مؤثر باشد. اگر یک مهاجم وارد رشته `"name'); DELETE FROM items; SELECT * FROM items WHERE 'a'='a"` شود، سه دستور معتبر زیر ایجاد خواهد شد:

```
        SELECT * FROM items
AND itemname = 'name';

        DELETE FROM items;

        SELECT * FROM items WHERE 'a'='a';
```

یک رویکرد سنتی برای جلوگیری از حملات تزریق SQL این است که آن‌ها را به عنوان مسئله اعتبارسنجی ورودی بکار برد و یا فقط کاراکترهایی از لیست سفید[3] مقادیر ایمن را قبول کند و لیست سیاهی از مقادیر بالقوه مخرب فرار کنند. لیست سفید می‌تواند یک ابزار مؤثر برای اجرای قوانین اعتبارسنجی محض ورودی باشد، اما دستورات SQL پارامتری شده نیاز به نگهداری کمتری دارند و می‌توانند تضمین‌های بیشتری را در رابطه

---

[1] Oracle

[2] trailing pair of hyphens

[3] whitelist

با امنیت ارائه دهند. همانطور که تقریبا همیشه اینطور است، لیست سیاه توسط روزنه-هایی غربال می‌شود که آن را در جلوگیری از حملات تزریق SQL غیرمؤثر می‌کند. برای مثال، مهاجمان می‌توانند:

- زمینه‌هایی که نقل قول نشده‌اند، هدف‌گیری کنند.
- راه‌هایی برای عبور از نیاز به فوق کاراکترهای فراری را بیابند.
- از روش‌های ذخیره شده برای مخفی کردن فوق کاراکترهای تزریقی استفاده کنند.

کاراکترهایی که به صورت دستی در ورود به جستارهای SQL فرار کرده‌اند، می‌توانند کمک کنند، اما برنامه شما را از حملات تزریق SQL ، ایمن نمی‌کند.

یکی دیگر از راه‌حل‌های معمول پیشنهاد شده برای مقابله با حملات تزریق SQL استفاده از روش‌های ذخیره‌شده است. هرچند روش‌های ذخیره‌شده از بعضی انوع حملات تزریق SQL ممانعت می‌کنند، اما در حفاظت علیه بسیاری دیگر شکست می‌خورند. برای مثال، روش PL/SQL زیر به همان حمله تزریق SQL نشان داده شده در مثال اول، آسیب‌پذیر است.

```
procedure get_item (
                itm_cv IN OUT ItmCurTyp,
                usr in varchar2,
                itm in varchar2)
        is
                open itm_cv for ' SELECT * FROM items WHERE ' ||
                                'owner = '''|| usr ||
                                ' AND itemname = ''' || itm || '''';
        end get_item;
```

روش‌های ذخیره شده[1] به طور معمول به جلوگیری از حملات تزریق SQL با محدود کردن انواع دستوراتی که می‌تواند به پارامترهای آن‌ها منتقل شود، کمک می‌کنند.

[1] Stored procedures

با این حال، راه‌های بسیاری در حوالی محدودیت‌ها و بسیاری از دستورات جالب وجود دارد که هنوز هم می‌تواند به روش‌های ذخیره‌شده منتقل شود. مجددا، روش‌های ذخیره شده می‌تواند از بعضی سوء استفاده‌ها جلوگیری کند، اما آن‌ها برنامه شما را علیه حملات تزریق SQL ایمن نخواهند کرد.

ترجمه و تدوین توسط: نیما اخلاقی

منبع: مقاله OWASP