بسم الله الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

نوشتن SQL پویای امن در SQL Sever

مدرس : مهندس افشین رفوآ

دوره آموزش SQL Server Administration

نوشتن SQL پویای امن در SQL Sever

تزریق SQL فرایندی است که طی آن کاربر مخرب بجای درج ورودی معتبر، دستورات Transact-SQL وارد می نماید. حال چنانچه ورودی به طور مستقیم و بدون اینکه مورد اعتبارسنجی قرار گیرد به سرویس دهنده ارسال گردد و برنامه ی کاربردی مربوطه به صورت تصادفی کد تزریق شده را اجرا کند، حمله می تواند خسارات جبران ناپذیر وارد آورده یا اطلاعات را به طور کامل از بین ببرد.

هر پروسیجری که دستورات SQL می سازد بایستی برای آسیب پذیری در برابر تزریق کد مورد بازبینی قرار گیرد، زیرا که SQL به طور پیش فرض تمامی query هایی دریافتی که از نظر ساختار نگارشی (syntax) صحیح می باشند را اجرا می کند. حتی داده های پارمتری نیز توسط یک هکر مصمم و ماهر قابل دستکاری می باشند. در صورت استفاده از SQL پویا، از پارامتری شدن دستورات خود اطمینان حاصل کنید و از گنجاندن مقادیر پارامتر به صورت مستقیم در query string خود خودداری نمایید.

تشریح آناتومی یک حمله ی SQL Injection

در پروسه ی تزریق، یک رشته ی متنی (text string) زودتر از موعد مقرر قطع شده (خاتمه یافته) و یک دستور جدید الحاق می گردد. به این خاطر که دستور درج شده ممکن است پیش از اجرا دارای رشته های اضافه بر سازمان باشد که به آن پیوست شده، مهاجم (malefactor) رشته ی تزریق شده را با علامت "--" خاتمه می

دهد. Text بعدی (که پس از علامت "--" قرا می گیرد) در زمان اجرا کاملا نادیده گرفته می شود. می توان چندین دستور را با استفاده از نقطه ویرگول به طور همزمان درج کرد.

تا زمانی که کد SQL تزریق شده از نظر ساختار نگارشی صحیح می باشد، دستکاری ها را نمی توان به صورت برنامه نویسی کشف کرد. از این رو لازم است تمامی ورودی کاربران را اعتبارسنجی کرده و کدی که دستورات ساخته شده ی SQL را در سرور مورد استفاده اجرا می کند بازبینی نمایید. هیچگاه آن دسته از ورودی کاربران که اعتبارسنجی نشده اند را با هم concatenate (متصل) نکنید. String concatenation نقطه ی اصلی و اولیه ی ورود (تزریق) اسکریپت می باشد.

برای جلوگیری از این رخداد، رهنمودهایی ارائه گردیده که به شرح زیر می باشد:

1. هیچگاه دستورات Transact-SQL را مستقیما از ورودی کاربر نسازید، همچنین به منظور اعتبارسنجی ورودی کاربر، از stored procedure ها استفاده کنید.
2. طی اعتبارسنجی ، ورودی کاربر را از نظر نوع، طول، فرمت و محدوده (range) تست کنید. با استفاده از دستور QUOTENAME() از بکار بردن نام های سیستمی جلوگیری کرده و یا اینکه با استفاده از دستور REPLACE() کاراکترهای خطرناک را با رشته خالی جایگزین می نماییم.
3. در هر یک از لایه های برنامه ی کاربردی خود چندین لایه ی اعتبارسنجی پیاده سازی کنید.
4. حجم (size) و نوع داده ی ورودی را تست کرده و محدودیت های لازم را اعمال نمایید. با این کار از سر ریز شدن حافظه ی میانجی (buffer overrun) جلوگیری نمایید.
5. محتویات متغیرهای رشته ای را آزمایش کرده و تنها مقادیر مورد انتظار را بپذیرد. از پذیرفتن ورودی هایی که دربردارنده ی مقادیر دودیی، escape sequence (توالی گریز) و comment character (کاراکتر توضیح) هستند، خودداری کنید.
6. حین کار با سندهای XML، تمامی داده ها را به مجرد ورود، با schema تطبیق داده و بر اساس آن اعتبارسنجی کنید.
7. در محیط های چند لایه ای، تمامی داده ها بایستی پیش از پذیرش در ناحیه ی قابل اطمینان (trusted zone) مورد اعتبارسنجی قرار گیرند.

8. از پذیرش رشته های زیر در فیلدهایی که اسم فایل ها می تواند از آن ساخته شود اجتناب کنید: AUX, CLOCK$، از COM1 تا COM8، CON، CONFIG$، از LPT1 تا LPT8، NUL و PRN.
9. جهت ایجاد امکان بررسی نوع و اعتبارسنجی طول، از اشیا SqlParameter به همراه stored procedure ها و دستورات استفاده کنید.
10. با بهره گیری از عبارات Regex در کد کاربر (client code)، کاراکترهای نامعتبر را فیلتر کنید.

راهبردهایی در خصوص Dynamic SQL

اجرای دستورات SQL پویا (که به صورت پویا ایجاد شده اند) درون کد پروسیجر منجر به شکسته شدن زنجیره ی مالکیت شده و سبب می شود SQL مجوزهای فراخواننده (caller) را با مجوز اشیا مورد دسترسی توسط dynamic SQL تطبیق داده و بررسی کند.

SQL Server روش هایی برای فراهم آوردن امکان اعطای مجوز دسترسی به اطلاعات به کاربران دارد. این کار را با بهره برداری از stored procedure ها و توابع کاربر (UDF) صورت می دهد که قابلیت اجرای SQL پویا را دارا می باشند.

1. جعل هویت (impersonation) با استفاده از عبارت EXECUTE AS.
2. امضای stored procedure ها به وسیله ی گواهینامه ها (certificate).

EXECUTE AS

عبارت EXECUTE AS مجوزهای فراخواننده را با مجوزهای کاربر مشخص شده در عبارت EXECUTE AS جایگزین می کند. Stored procedureهای تودرتو یا trigger ها تحت بستر امنیتی proxy user اجرا می شوند. این کار می تواند برنامه هایی که متکی بر (مبتنی بر) امنیت در سطح سطر (row-level) هستند یا به نظارت امنیتی (auditing) نیاز دارند هک شوند. توابعی که هویت کاربر را برمی گردانند، در واقع بجای فراخواننده ی اصلی کاربری را بازگردانی می نمایند که در عبارت EXECUTE AS تعریف شده است. بستر اجرایی (execution context) تنها پس از اجرای پروسیجر و یا هنگامی که دستور REVERT صادر می شود، به فراخواننده ی اصلی بازگردانده می شود. (شما می‌توانید هنگامی که فراخواننده مجوزی بر روی اشیای مورد ارجاع ندارد، عبارت EXECUTE AS را در یک روال ذخیره شده مورد استفاده قرار دهید. تأثیر عبارت

EXECUTE AS این است که بستر اجرا به کاربر پراکسی منتقل می‌گردد. تمامی کد و تمامی فراخوانی‌ها به روال‌های ذخیره شده تودرتو یا تریگرها، تحت بستر امنیتی کاربر پراکسی اجرا می‌گردد. بستر اجرا فقط پس از اجرای روال یا هنگامی که عبارت REVERT مورد استفاده قرار گیرد، به فراخواننده اصلی بازمی‌گردد).

امضای گواهینامه (Certificate Signing)

زمانی که یک stored procedure که با یک گواهینامه امضا شده اجرا می گردد، مجوزهای اعطا شده به certificate user با مجوزهای تخصیص داده شده به فراخواننده ادغام می گردد. بستر اجرا تغییری نمی کند؛ به عبارت دیگر کاربر هویت فرخواننده را جعل نمی کند. پیاده سازی فرایند امضای stored procedure لازمه ی اجرای چندین مرحله ی متعدد می باشد. پس از هر بار اصلاح، پروسیجر بایستی مجددا امضا گردد.

دسترسی پایگاه داده متقابل (Cross Database Access)

زنجیره ی مالکیت پایگاه داده متقابل (Cross-database ownership chaining) در مواردی که دستورات پویای SQL اجرا می شوند، کارساز نخواهد بود. می توان به راحتی با ایجاد یک stored procedure که به اطلاعات حاضر در یک پایگاه داده ی دیگر دسترسی دارد و امضای پروسیجر با یک گواهینامه که در هر دو پایگاه داده موجود می باشد، این مسئله را برطرف ساخت (دور زد). این کار بدون اعطای مجوز و دسترسی به کاربران، امکان دستیابی به منابع پایگاه داده مورد استفاده ی پروسیجر را (برای کاربران) فراهم می آورد.