KN0-0904-3802

تزریق کدهای SQL و راه های مقابله با آن

میلاد کرمی [۱] karamimilad46@gmail.com
علی پارسا [۲] aria.parsa@gmail.com
[۱]میلاد کرمی کارشناس ارشد مهندسی نرم افزار دانشگاه یاسین
[۲]علی پارسا کارشناس مهندسی نرم افزار دانشگاه آفرینش

چکیده: تزریق کدهای SQL (SQLi) یک آسیب پذیری امنیتی وب است که به مهاجم اجازه می‌دهد در پرس و جوهایی که یک برنامه در پایگاه داده خود ایجاد می‌کند دخالت کند. به طور کلی به مهاجم اجازه می دهد تا داده هایی را مشاهده کند که معمولاً قادر به مشاهده آنها نیستند. این ممکن است شامل داده های متعلق به سایر کاربران یا هر داده دیگری باشد که خود برنامه قادر به دسترسی به آن باشد. در بسیاری از موارد یک مهاجم می‌تواند این داده‌ها را تغییر داده یا حذف کند و باعث تغییرات مداوم در محتوا یا رفتار برنامه شود، در برخی شرایط مهاجم می‌تواند یک حمله تزریق SQL را تشدید کند تا سرور زیربنایی یا دیگر زیرساخت‌های بک اند را به خطر بیاندازد.

کلید واژه‌ها: امنیت ، هک ،نفوذ ،پایگاه داده

۱. مقدمه

یک حمله موفقیت آمیز تزریق SQL می تواند منجر به دسترسی غیرمجاز به داده های حساس مانند رمز عبور، جزئیات کارت اعتباری یا اطلاعات شخصی کاربر شود. بسیاری از سرقت داده‌ها در سال‌های اخیر نتیجه حملات تزریق SQL بوده‌اند که منجر به آسیب به شهرت و جریمه‌های قانونی سرویس دهنده ها شده است. در برخی موارد مهاجم می‌تواند یک درب پشتی دائمی در سیستم‌های یک سازمان به دست آورد که می‌تواند برای مدت طولانی مورد توجه قرار نگیرد.

نمونه‌های تزریق SQL

طیف گسترده‌ای از آسیب‌پذیری‌ها ، حملات و تکنیک‌های تزریق SQL وجود دارد که در موقعیت‌های مختلف به وجود می‌آیند.

برخی از نمونه های رایج تزریق SQL عبارتند از:

بازیابی داده های پنهان ، که در آن می توانید یک پرس و جوی SQL را برای بازگرداندن نتایج اضافی تغییر دهید.

براندازی منطق برنامه ، که در آن می توانید یک پرس و جو را تغییر دهید تا با منطق برنامه تداخل داشته باشد.

حملات UNION ، که در آن می توانید داده ها را از جداول پایگاه داده مختلف بازیابی کنید.

بررسی پایگاه داده ، که در آن می توانید اطلاعات مربوط به نسخه و ساختار پایگاه داده را استخراج کنید.

تزریق کور SQL ، که در آن نتایج جستجویی که شما کنترل می‌کنید در پاسخ‌های برنامه برگردانده نمی‌شود.

بازیابی داده های پنهان

یک اپلیکیشن خرید را در نظر بگیرید که محصولات را در دسته های مختلف نمایش می دهد. وقتی کاربر روی دسته موبایل کلیک می کند، مرورگر او URL زیر را درخواست می کند:

https://Example.com/products?category=mobile

این باعث می شود که برنامه یک پرس و جوی SQL برای بازیابی جزئیات محصولات مربوطه از پایگاه داده ایجاد کند:

```
SELECT * FROM products WHERE category = '
mobile' AND released = 1
```

نتیجه کوئری فوق به صورت زیر خواهد بود :

تمام جزئیات از جدول محصولات که در آن دسته بندی برابر با موبایل و منتشر شده برابر با ۱ است.

محدودیت • برای پنهان کردن محصولاتی که منتشر نشده اند استفاده می شود.

برای محصولات منتشر نشده این برنامه هیچ گونه دفاعی در برابر حملات تزریق SQL اجرا نمی کند، بنابراین یک مهاجم می تواند با تغییر URL حمله ای مانند زیر را انجام دهد :

https:// Example.com/products?category='mobile '--'

با تغییر URL دستور SQL که روی پایگاه داده اجرا میشود به صورت زیر خواهد بود :

SELECT * FROM products WHERE category = ' mobile'--' AND released = 1

نکته کلیدی در اینجا این است که دو خط تیره در SQL نشانگر نظر یا همان کامنت است ، به زبان ساده تر دستوراتی که بعد از دو خط تیره قرار میگیرند در نتیجه ی جستجو اعمال نمیشوند و SQL آنها را نادیده میگیرد و به طور موثر بقیه جستجو را حذف میکند ، بنابر این دیگر شامل AND محصولات منتشر شده نمیشود و همه محصولات از جمله محصولات منتشر نشده نمایش داده میشوند

در ادامه، یک مهاجم می‌تواند باعث شود که برنامه همه محصولات را در هر دسته‌ای نمایش دهد، از جمله دسته‌هایی که از آنها اطلاعی ندارد:

https://Example.com/products?category=mobile'+OR+1=1--

این منجر به پرس و جوی زیر می شود:

SELECT * FROM products WHERE category = 'mobile' OR 1=1--' AND released = 1

پرس و جو اصلاح شده همه مواردی را که در آن دسته موبایل برابر ۱۵ است برمی گرداند. از آنجایی که۱=۱ همیشه درست است، پرس و جو همه موارد را برمی گرداند.

براندازی منطق برنامه

برنامه ای را در نظر بگیرید که به کاربران امکان می دهد با نام کاربری و رمز عبور وارد شوند. اگر کاربر نام کاربری user1 و رمز عبور 12345 را ارسال کند، برنامه با انجام پرس و جوی SQL زیر اعتبار را بررسی می کند:

SELECT * FROM users WHERE username = 'user1' AND password = '12345'

اگر پرس و جوی فوق جزئیات یک کاربر را برگرداند، ورود موفقیت آمیز است ، در غیر این صورت اجازه ورود را ندارد.

در اینجا یک مهاجم می تواند به عنوان هر کاربر بدون رمز عبور به سادگی با استفاده از دنباله نظرات SQL وارد شود ، به عنوان مثال با نام کاربری administrator و با استفاده از دو خط تیره برای حذف بررسی رمز عبور از عبارت WHERE منجر به درخواست زیر میشود :

```
SELECT * FROM users WHERE username = 'administrator' -- AND password = '12345'
```

این کوئری کاربری را که نام کاربری آن مدیر است برمی گرداند و مهاجم را با موفقیت به عنوان آن کاربر وارد می کند.

بازیابی داده ها از سایر جداول پایگاه داده

در مواردی که نتایج یک پرس و جوی SQL در پاسخ های برنامه برگردانده می شود، مهاجم می تواند از آسیب پذیری تزریق SQL برای بازیابی داده ها از جداول دیگر در پایگاه داده استفاده کند. این کار با استفاده از کلمه کلیدی UNION انجام می شود که به شما امکان می دهد یک کوئری SELECT اضافی را اجرا کنید و نتایج را به پرس و جو اصلی اضافه کنید. به عنوان مثال، اگر یک برنامه پرس و جو زیر حاوی ورودی کاربر برای دسته بندی "mobile" را اجرا کند:

SELECT name, description FROM products WHERE category = 'mobile'

سپس یک مهاجم می تواند با اضافه کردن دستور زیر :

' UNION SELECT username, password FROM users—

باعث شود که اپلیکیشن تمامی نام های کاربری و رمزهای عبور را به همراه نام و توضیحات محصولات برگرداند.

بررسی پایگاه داده

پس از شناسایی اولیه یک آسیب‌پذیری تزریق SQL، به‌دست آوردن اطلاعاتی در مورد خود پایگاه داده معمولاً مفید است. این اطلاعات اغلب می تواند راه را برای بهره برداری بیشتر هموار کند. شما می توانید جزئیات نسخه را برای پایگاه داده استعلام کنید. روش انجام این کار به

نوع پایگاه داده بستگی دارد، بنابراین می توانید نوع پایگاه داده را از هر تکنیکی که کار می کند استنباط کنید. به عنوان مثال، در اوراکل می توانید موارد زیر را اجرا کنید:

SELECT * FROM v$version

همچنین می توانید تعیین کنید که چه جداولی در پایگاه داده وجود دارد و چه ستون هایی را شامل می شوند. به عنوان مثال، در اکثر پایگاه های داده می توانید پرس و جوی زیر را برای لیست کردن جداول اجرا کنید:

SELECT * FROM information_schema.tables

آسیب‌پذیری‌های کور تزریق SQL

بسیاری از نمونه‌های تزریق SQL آسیب‌پذیری‌های کور هستند. این بدان معنی است که برنامه نتایج جستجوی SQL یا جزئیات خطاهای پایگاه داده را در پاسخ های خود باز نمی گرداند. آسیب‌پذیری‌های کور همچنان می‌توانند برای دسترسی به داده‌های غیرمجاز مورد سوء استفاده قرار گیرند، اما تکنیک‌های درگیر معمولاً پیچیده‌تر و انجام آن‌ها دشوار است. بسته به ماهیت آسیب‌پذیری و پایگاه داده می‌توان از تکنیک‌های زیر برای بهره‌برداری از آسیب‌پذیری‌های تزریق SQL کور استفاده کرد:

می‌توانید منطق پرس و جو را تغییر دهید تا بسته به صحت یک شرط، تفاوت قابل تشخیصی در پاسخ برنامه ایجاد شود. این ممکن است شامل تزریق یک شرط جدید به منطق بولی یا ایجاد یک خطای مشروط مانند تقسیم بر صفر باشد.

شما می توانید به صورت مشروط یک تاخیر زمانی در پردازش پرس و جو ایجاد کنید، که به شما امکان می دهد صحت شرایط را بر اساس زمانی که برنامه برای پاسخ دادن نیاز دارد استنباط کنید.

با استفاده از تکنیک های OAST می توانید یک تعامل شبکه خارج از باند را راه اندازی کنید. این تکنیک بسیار قدرتمند است و در شرایطی کار می کند که سایر تکنیک ها این کار را نمی کنند. اغلب می‌توانید مستقیماً داده‌ها را از طریق کانال خارج از باند استخراج کنید، برای مثال با قرار دادن داده‌ها در جستجوی DNS برای دامنه‌ای که کنترل می‌کنید.

با استفاده از تکنیک های OAST می توانید یک تعامل شبکه خارج از باند را راه اندازی کنید. این تکنیک بسیار قدرتمند است و در شرایطی کار می کند که سایر تکنیک ها این کار را نمی کنند. اغلب می‌توانید مستقیماً داده‌ها را از طریق کانال خارج از باند استخراج کنید، برای مثال با قرار دادن داده‌ها در جستجوی DNS برای دامنه‌ای که کنترل می‌کنید.

چگونه آسیب پذیری های تزریق SQL را شناسایی کنیم

اکثر آسیب‌پذیری‌های تزریق SQL را می‌توان به سرعت و با اطمینان با استفاده از اسکنر آسیب‌پذیری وب Burp Suite پیدا کرد. تزریق SQL را می توان با استفاده از مجموعه ای سیستماتیک از تست ها در برابر هر نقطه ورودی در برنامه به صورت دستی تشخیص داد. این معمولاً شامل موارد زیر است:

ارسال کاراکتر نقل قول تکی ' و جستجوی خطاها یا ناهنجاری های دیگر.

ارائه برخی از نحو خاص SQL که به ارزش پایه (اصلی) نقطه ورودی و به یک مقدار متفاوت ارزیابی می‌شود و به دنبال تفاوت‌های سیستماتیک در پاسخ‌های برنامه شده است.

ارائه شرایط بولی مانند OR 1=1 و OR 1=2 و جستجوی تفاوت در پاسخ های برنامه.

ارسال بارهای طراحی شده برای ایجاد تأخیر زمانی هنگام اجرا در پرس و جوی SQL، و جستجوی تفاوت در زمان صرف شده برای پاسخ.

ارسال بارهای OAST طراحی شده برای راه اندازی یک تعامل شبکه خارج از باند هنگام اجرا در یک پرس و جوی SQL، و نظارت بر هرگونه تعامل حاصل.

تزریق SQL در قسمت های مختلف پرس و جو

اکثر آسیب‌پذیری‌های تزریق SQL در عبارت WHERE یک کوئری SELECT ایجاد می‌شوند. این نوع تزریق SQL عموماً توسط آزمایش کنندگان با تجربه به خوبی درک می شود. اما آسیب پذیری‌های تزریق SQL در اصل می‌توانند در هر مکانی در پرس و جو و در انواع مختلف پرس و جو رخ دهند. متداول ترین مکان های دیگری که در آن تزریق SQL ایجاد می شود عبارتند از: در دستورات UPDATE، در مقادیر به روز شده با عبارت WHERE. در دستورات INSERT، در داخل مقادیر درج شده. در عبارات SELECT، در نام جدول یا ستون. در عبارات SELECT، در بند ORDER BY.

تزریق SQL در زمینه های مختلف

در تمام آزمایش‌هایی که تاکنون انجام شده است، از رشته query برای تزریق بار SQL مخرب خود استفاده کرده‌اید. با این حال، مهم است که توجه داشته باشید که می توانید حملات تزریق SQL را با استفاده از هر ورودی قابل کنترلی که به عنوان یک پرس و جوی SQL توسط برنامه پردازش می شود، انجام دهید. به عنوان مثال، برخی از وب سایت ها ورودی را با فرمت JSON یا XML دریافت می کنند و از آن برای جستجو در پایگاه داده استفاده می کنند. این فرمت‌های مختلف حتی ممکن است راه‌های جایگزینی را برای شما فراهم کنند تا حملاتی را که در غیر این صورت به دلیل WAF و سایر مکانیسم‌های دفاعی مسدود شده‌اند، پنهان کنید. پیاده‌سازی‌های ضعیف اغلب فقط به دنبال کلمات کلیدی رایج تزریق SQL در درخواست می‌گردند، بنابراین ممکن است بتوانید این فیلترها را با رمزگذاری یا فرار از کاراکترها در کلمات کلیدی ممنوعه دور بزنید. به عنوان مثال، تزریق SQL مبتنی بر XML زیر از یک دنباله فرار XML برای رمزگذاری کاراکتر S در SELECT استفاده می کند:

```
<stockCheck>
  <productId>
    123
  </productId>
  <storeId>
    999 &#x53;ELECT * FROM
information_schema.tables
  </storeId>
</stockCheck>
```

قبل از ارسال به مفسر SQL، در سمت سرور رمزگشایی می شود.

تزریق SQL مرتبه دوم

تزریق مرتبه اول SQL زمانی ایجاد می شود که برنامه ورودی کاربر را از یک درخواست HTTP می گیرد و در طول پردازش آن درخواست، ورودی را به روشی ناامن در یک پرس و جوی SQL وارد می کند. در تزریق SQL مرتبه دوم (همچنین به عنوان تزریق SQL ذخیره شده نیز شناخته می شود)، برنامه ورودی کاربر را از یک درخواست HTTP می گیرد و آن را برای استفاده در آینده ذخیره می کند. این معمولاً با قرار دادن ورودی در یک پایگاه داده انجام می شود، اما هیچ آسیب پذیری در نقطه ای که داده ها ذخیره می شوند ایجاد نمی شود. بعداً، هنگام رسیدگی به درخواست HTTP متفاوت، برنامه داده‌های ذخیره‌شده را بازیابی می‌کند و آن‌ها را به روشی ناامن در یک کوئری SQL وارد می‌کند.

تزریق SQL مرتبه دوم اغلب در شرایطی ایجاد می‌شود که توسعه‌دهندگان از آسیب‌پذیری‌های تزریق SQL آگاه هستند و بنابراین با خیال راحت قرار دادن اولیه ورودی را در پایگاه داده انجام می‌دهند. وقتی داده‌ها بعداً پردازش می‌شوند، ایمن تلقی می‌شوند، زیرا قبلاً به صورت ایمن در پایگاه داده قرار داده شده بودند. در این مرحله، داده ها به روشی ناامن مدیریت می شوند، زیرا توسعه دهنده به اشتباه آنها را قابل اعتماد می داند.

عوامل خاص پایگاه داده

برخی از ویژگی‌های اصلی زبان SQL به روشی مشابه در پلتفرم‌های پایگاه داده محبوب پیاده‌سازی می‌شوند، و بسیاری از راه‌های شناسایی و بهره‌برداری از آسیب‌پذیری‌های تزریق SQL به طور یکسان در انواع مختلف پایگاه داده کار می‌کنند. با این حال، تفاوت های زیادی نیز بین پایگاه های داده رایج وجود دارد. این بدان معناست که برخی از تکنیک‌ها برای شناسایی و بهره‌برداری از تزریق SQL در پلتفرم‌های مختلف متفاوت عمل می‌کنند. به عنوان مثال: نحو برای الحاق رشته. نظرات. پرس و جوهای دسته ای (یا انباشته شده). API های مخصوص پلتفرم پیغام خطا.

چگونه از تزریق SQL جلوگیری کنیم

بیشتر نمونه‌های تزریق SQL را می‌توان با استفاده از پرس‌وجوهای پارامتری (همچنین به عنوان عبارات آماده‌شده شناخته می‌شود) به جای الحاق رشته‌ها در پرس‌وجو، جلوگیری کرد. کد زیر در برابر تزریق SQL آسیب پذیر است زیرا ورودی کاربر مستقیماً به پرس و جو متصل می شود:

```
String query = "SELECT * FROM products
WHERE category = '"+ input + "'";
Statement statement = connection.createStatement();
ResultSet resultSet =
statement.executeQuery(query);
```

این کد را می توان به راحتی به گونه ای بازنویسی کرد که از تداخل ورودی کاربر با ساختار پرس و جو جلوگیری کند:

```
PreparedStatement statement =
connection.prepareStatement("SELECT * FROM
products WHERE category = ?");
statement.setString(1, input);
ResultSet resultSet = statement.executeQuery();
```

پرس و جوهای پارامتری را می توان برای هر موقعیتی که ورودی نامعتبر به عنوان داده در پرس و جو ظاهر می شود، از جمله عبارت WHERE

و مقادیر در عبارت INSERT یا UPDATE استفاده کرد. نمی‌توان از آن‌ها برای مدیریت ورودی نامعتبر در بخش‌های دیگر پرس و جو، مانند نام جدول یا ستون، یا عبارت ORDER BY استفاده کرد. عملکرد برنامه‌ای که داده‌های غیرقابل اعتماد را در آن بخش‌های پرس‌وجو قرار می‌دهد، باید رویکرد متفاوتی داشته باشد، مانند مقادیر ورودی مجاز در فهرست سفید، یا استفاده از منطق متفاوت برای ارائه رفتار مورد نیاز. برای اینکه یک پرس و جوی پارامتری شده در جلوگیری از تزریق SQL موثر باشد، رشته ای که در پرس و جو استفاده می شود باید همیشه یک ثابت کدگذاری شده باشد و هرگز نباید حاوی داده های متغیری از هر مبدا باشد. وسوسه نشوید که مورد به مورد تصمیم بگیرید که آیا یک مورد از داده ها قابل اعتماد است یا خیر، و به استفاده از الحاق رشته در پرس و جو برای مواردی که ایمن در نظر گرفته می شوند ادامه دهید. اشتباه کردن در مورد منشاء احتمالی داده ها یا تغییرات درکدهای دیگر برای نقض فرضیات در مورد اینکه چه داده هایی آلوده شده اند بسیار آسان است.

مراجع

۱ : کتاب حملات و دفاع تزریق SQL
نویسنده جاستین کلارک
۲ : کتاب راهنمای هکر برنامه های وب
نویسنده : مارکوس پینتو