.SQL injection چیست.

**injection چیست؟**

حملات injection به دسته‌ای از حملات و هک‌ها گفته می‌شود که در آن مهاجم، ورودی غیرقابل اعتماد وارد برنامه می‌کند. این ورودی‌ها روی بخشی از دستور یا کوئری تاثیر می‌گذارد و اجرای آن برنامه را تغییر می‌دهد.

حملات injection قدیمی‌ترین و خطرناک‌ترین حمله به برنامه‌های وب است که منجر به دزدی اطلاعات، گم شدن اطلاعات، و از دست رفتن یکپارچگی اطلاعات می‌شود. بیشتر آسیب‌های اینجکشن، تایید اعتبار ورودی نامناسب کاربر است.

برخی از انواع حملات injection عبارتند از:

1. Code injection
2. CRLF injection
3. Cross-site Scripting (XSS)
4. Email Header Injection
5. Host Header Injection
6. LDAP Injection
7. OS Command Injection
8. XPath injection
9. SQL Injection (SQLi)

مفهوم SQL injection چیست؟

SQL injection یا SQLi آسیب امنیتی وب است که به هکر اجازه مداخله در کوئری‌های دیتابیس را می‌دهد. مثلا هکرها دیتایی که در حالت عادی دسترسی به آن نیست را می‌بینند. ممکن است دیتایی باشد که متعلق به کس دیگری است یا دیتایی باشد که فقط خود برنامه امکان دسترسی به آن را دارد. در بسیاری موارد مهاجم این دیتا را پاک می‌کند یا تغییر می‌دهد در نتیجه روی محتوا و یا رفتار برنامه تاثیر می‌گذارد.

هر وب سایت و برنامه وبی که از دیتابیس SQL مثل MySQL و Oracle و SQL Server استفاده می‌کند در معرض آسیب این حمله است. SQL زبان کوئری‌ها است که برای مدیریت اطلاعات ذخیره شده در دیتابیس‌ها طراحی شده است. پس با استفاده از آن می‌توان دسترسی و اصلاح و حذف اطلاعات را انجام داد. حتی می‌توان از دستورات SQL برای اجرای برخی دستورات سیستم عاملی استفاده کرد. در نتیجه حمله SQL injection موفق، عواقب زیادی دارد.

نکته مهم این است که باگ SQL اصلا به دلیل ضعف SQL نیست بلکه برنامه‌نویس‌ها با اشتباهات برنامه‌نویسی راه را برای هکرها و انجام تزریق SQL هموار می‌کنند.

هکر ابتدا به دنبال ورودی‌های کاربری آسیب‌پذیر در صفحه وب یا برنامه وب می‌گردد چون بدین ترتیب می‌تواند مستقیم وارد کوئری SQL شود. هکر، محتوای ورودی که malicious payload نامیده می‌شود و کلید اصلی حمله است را آماده می‌کند. پس از اینکه هکر این محتوا را وارد می‌کند، دستورات SQL مخرب در دیتابیس اجرا می‌شود.

در SQL Injection امکان اجرای SQL statementهای مخرب فراهم می‌شود. این استیت‌منت‌ها کنترل دیتابیس سرور پشت برنامه وب را به دست می‌گیرند. هکرها از SQLi استفاده می‌کنند تا به صفحه وب، برنامه وب و بازیابی محتوای کل دیتابیس SQL دست یابند تا بتوانند رکوردهای دیتابیس را اضافه و اصلاح و حذف کنند.

**تاثیرات حمله SQL injection موفق چیست؟**

برخی مواقع مهاجم از انواع حملات  SQL injectionاستفاده می‌کند تا سرور و دیگر زیرساخت‌ها را از سرویس خارج کند یا حمله دایس DoS – انجام دهد. همچنین دسترسی به سیستم عامل باعث می‌شود هکر بتواند به شبکه داخلی شما نفوذ کند. برای استفاده از خدمات تعمیر سرور hp و ارتقای آن روی لینک بزنید.

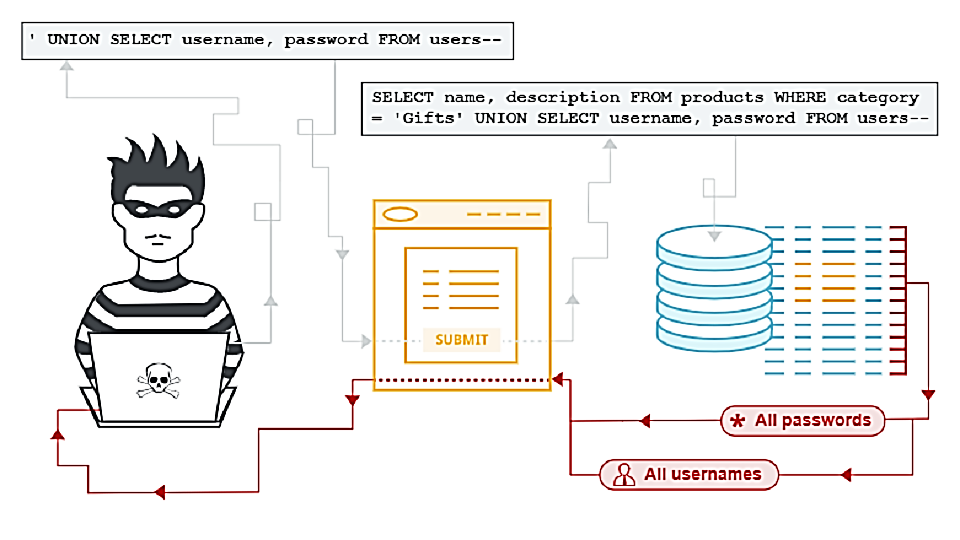
یک حمله موفق SQL injection دسترسی کامل به تمام اطلاعات دیتابیس سرور را به هکر می‌دهد و روی دسترسی‌های غیرمجاز به اطلاعات حساس تاثیر می‌گذارد مثلا پسوردها، اطلاعات کارت اعتباری، اطلاعات مشتری و اطلاعات شخصی کاربر. بسیاری از هک‌های اطلاعات محرمانه در سال‌های اخیر با استفاده از انواع حملات  SQL injectionانجام شده است. برخی مواقع، مهاجم به سیستم‌های دولتی نفوذ می‌کند و در اصطلاح بک دور دارد در نتیجه مدت‌ها می‌تواند از اطلاعات استفاده کند بدون اینکه کسی متوجه شود.

هکر با تغییر در رکوردهای دیتابیس به اهداف خود می‌رسد مثلا در برنامه‌های مالی، انتقال وجه بین حساب‌ها انجام می‌دهد. با حذف رکوردی‌های دیتابیس و حتی جدول‌های آن، ممکن است برنامه از دسترس خارج شود تا زمانی که دیتابیس با استفاده از بکاپ‌ها ری‌استور شود. حتی ممکن است با بکاپ‌ها هم کامل نتوان دیتای کامل را به دست آورد.

**انواع تزریق SQL**

طیف گسترده‌ای از آسیب پذیری‌ها، حملات و تکنیک‌های SQL Injection وجود دارد که در شرایط مختلف ایجاد می‌شوند. برخی از نمونه‌های معمول تزریق SQL عبارتند از:

1. بازیابی داده‌های پنهان، جایی که می‌توانید یک درخواست SQL را برای بازگشت نتایج اضافی اصلاح کنید.
2. برهم‌زدن منطق برنامه، جایی که می توانید یک سوال را برای تغییر در منطق برنامه تغییر دهید.
3. حمله به UNION، که در آن می توانید داده‌ها را از جداول پایگاه‌داده‌های مختلف بازیابی کنید.
4. بررسی بانک اطلاعاتی، جایی که می توانید اطلاعات مربوط به نسخه و ساختار بانک اطلاعات را استخراج کنید.
5. تزریق کور SQL، که در آن نتایج جستجوی شما کنترل شده در پاسخ برنامه نیست.



**بازیابی داده‌های پنهان**

یک برنامه خرید را در نظر بگیرید که محصولات را در دسته‌های مختلف نمایش می‌دهد. وقتی کاربر بر روی دسته هدایا کلیک کند، مرورگر، URL را درخواست می‌کند:

“https://insecure-website.com/products?category=Gifts”

این امر باعث می‌شود تا یک نرم‌افزار SQL برای بازگرداندن جزئیات مربوط به محصولات از پایگاه داده، از یک SQL Query استفاده کند:

SELECT \* FROM products WHERE category = 'Gifts' AND released = 1

این پرس‌و‌جو(SQL query) از database اطلاعات زیر را درخواست می‌کند:

* all details (\*)
* from the products table
* where the category is Gifts
* and released is 1.

“released = 1” برای مخفی کردن محصولاتی که عرضه نمی‌شوند، استفاده می‌شود. برای محصولات منتشر نشده “released = 0” استفاده می‌شود.

این برنامه هیچ‌گونه دفاعی را در برابر حملات تزریق SQL پیاده‌سازی نمی‌کند، بنابراین یک مهاجم می‌تواند حمله‌ای مانند:

“https://insecure-website.com/products?category=Gifts’–”

نتیجه جستجوی SQL به صورت زیر می‌باشد:

.SELECT \* FROM products WHERE category = 'Gifts'--' AND released = 1.

نکته اصلی در اینجا این است که توالی دو خط “–” یک شاخص در SQL است و به این معنی است که بقیه پرس و جو به عنوان یک تفسیر تعبیر می‌شود.

این موضوع به طور تاثیرگذاری باقی‌مانده پرس‌وجو را از بین می‌برد، بنابراین دیگر “1=AND released” منتشر نمی‌شود. این بدان معنی است که همه محصولات از جمله محصولات غیرمجاز نمایش داده می‌شوند.

با پیشروی بیشتر، یک مهاجم می‌تواند باعث شود، برنامه تمام محصولات را در هر گروه از جمله دسته‌هایی که از آن‌ها هیچکس خبر ندارد نمایش دهد:

” https://insecure-website.com/products?category=Gifts’+OR+1=1– “

نتیجه در جستجوی SQL به صورت زیر می‌باشد:

.SELECT \* FROM products WHERE category = 'Gifts' OR 1=1--' AND released = 1.

پرس‌وجوی اصلاح شده، تمام مواردی را که در آن دسته از هدایا 1 برابر با 1 است را نشان می‌دهد، از آن‌جا که 1 = 1 همیشه درست است، پرس‌وجو تمام موارد را برمی‌گرداند.

**برهم‌زدن منطق برنامه**

برنامه‌ای را در نظر بگیرید که به کاربران اجازه می‌دهد با یک نام کاربری و رمز عبور وارد شوند. اگر کاربر نام کاربری و رمز عبور را ارسال کند، برنامه با انجام SQL زیر، اعتبار نامه را بررسی می‌کند:

.'SELECT \* FROM users WHERE username = 'wiener' AND password = 'bluecheese.

اگر پرس‌وجو، جزئیات یک کاربر را بازگرداند، ورود به سیستم موفقیت آمیز خواهد بود. در غیر این صورت درخواست پذیرفته نمی‌شود.

در این بخش، یک مهاجم می‌تواند مانند هر کاربر و بدون رمز ورود به سادگی با استفاده از”--SQL comment” وارد سیستم شود.

بخش چک‌کردن رمز عبور را از قسمت “WHERE” انجام می‌دهد و آن را حذف می‌کند. به عنوان مثال، ارسال نام کاربری ” administrator’– “و یک رمز عبور خالی به صورت خط زیر:

.' '=SELECT \* FROM users WHERE username = 'administrator'--' AND password.

می‌تواند کاربر را که نام کاربری آن administrator است از دیتابیس دریافت کند و هکر خود را به جای کاربر ثبت کند.

**حمله به UNION**

در مواردی که نتایج یک پرس و جو SQL در پاسخ برنامه ها برگردانده شود، مهاجم می تواند از آسیب پذیری SQL Injection برای بازیابی داده ها از جداول دیگر در پایگاه داده استفاده کند.

این کار با استفاده از کلمه کلیدی UNION انجام می شود و به شما امکان می دهد “Select query” اضافی را نیز انجام دهید و نتایج را به پرس و جو اصلی اضافه کنید.

به عنوان مثال، اگر برنامه ای درخواست زیر را که حاوی ورودی کاربر “gifts” است، اجرا کند:

.'SELECT name, description FROM products WHERE category = 'Gifts.

سپس هکر می تواند ورودی زیر را ارسال کند:

.––UNION SELECT username, password FROM users.

این امر باعث می شود برنامه تمام نام های کاربری و کلمات عبور را به همراه نام و توضیحات محصولات بازگرداند.

بررسی بانک اطلاعاتی

پس از شناسایی اولیه آسیب پذیری SQL Injection، به طور کلی دستیابی به برخی اطلاعات در مورد خود پایگاه داده مفید است. این اطلاعات اغلب می تواند راه را برای بهره برداری بیشتر هموار کند.

می توانید جزئیات نسخه مربوط به پایگاه داده را پرس و جو کنید. نحوه انجام این کار بستگی به نوع بانک اطلاعاتی دارد، بنابراین می توانید نوع پایگاه داده را از هر تکنیک دیگری بدست آورید. به عنوان مثال، در Oracle می توانید این کار را انجام دهید:

.SELECT \* FROM v$version.

همچنین می توانید تعیین کنید که جداول بانک اطلاعاتی وجود دارد و کدام ستون ها را شامل می شود. به عنوان مثال، در اکثر بانک های اطلاعاتی می توانید عبارت زیر را برای لیست کردن جدول ها اجرا کنید:

.SELECT \* FROM information\_schema.tables.

تزریق کور SQL

بسیاری از موارد تزریق SQL آسیب پذیری کور است. این بدان معناست که برنامه نتایج جستجوی SQL یا جزئیات هرگونه خطای پایگاه داده را در پاسخ های خود باز نمی گرداند.

از آسیب پذیری های کور هنوز هم می توان برای دسترسی به داده های غیر مجاز استفاده کرد، اما تکنیک های موجود عموماً پیچیده تر و دشوار هستند.

بسته به ماهیت آسیب پذیری و پایگاه داده، از تکنیک های زیر می توان برای سوءاستفاده از آسیب پذیری های تزریق کور SQL استفاده کرد:

1. شما می توانید منطق پرس و جو را تغییر دهید تا بسته به حقیقت یک شرط واحد، اختلاف قابل تشخیص در پاسخ برنامه ایجاد شود. این ممکن است شامل تزریق یک شرایط جدید به برخی از منطق بولی و یا ایجاد مشروط مانند خطای تقسیم بر صفر باشد.
2. شما می توانید به طور مشروط یک تأخیر زمانی در پردازش پرس و جو ایجاد کنید که به شما این امکان را می دهد تا واقعیت شرط را بر اساس زمانی که برنامه پاسخ می دهد استنباط کنید.
3. با استفاده از تکنیک های OAST می توانید تعامل خارج از باند شبکه را آغاز کنید. این روش بسیار قدرتمند است و در شرایطی کار می کند که تکنیک های دیگر این کار را انجام ندهند. شما می توانید مستقیماً داده ها را از طریق کانال خارج از باند، مجزا کنید، به عنوان مثال با قرار دادن داده ها در جستجوی DNS برای دامنه ای که شما کنترل می کنید.

نحوه تشخیص آسیب پذیری های تزریق SQL

اکثر آسیب پذیری های تزریق SQL را می توان به سرعت و با اطمینان با استفاده از اسکنر آسیب پذیری وب “Burp Suite” یافت.

SQL Injection می تواند به صورت دستی و با استفاده از یک مجموعه سیستماتیک از تست در برابر هر نقطه ورود به برنامه تشخیص داده شود. به طور معمول شامل موارد زیر است:

* ارسال شخصیت نقل قول واحد و به دنبال خطا یا ناهنجاری های دیگر.
* ارسال برخی از نحوه های خاص SQL که به مقدار پایه (اصلی) از نقطه ورود و به یک مقدار دیگر ارزشیابی می کنند و به دنبال تفاوت های سیستماتیک در پاسخ های برنامه کاربردی هستند.
* ارسال شرایط بولی مانند 1 = 1 و 1 = 2 و جستجوی تفاوت در پاسخ برنامه.
* ارسال بار های با هدف ایجاد تأخیر در هنگام اجرای درخواست در SQL و به دنبال تفاوت در زمان لازم برای پاسخگویی هستید.
* ارسال بار های OAST با هدف ایجاد تعامل شبکه خارج از باند هنگام اجرای یک پرس و جو SQL و نظارت بر هرگونه تعامل حاصل از آن، طراحی شده است.

تزریق SQL در قسمت های مختلف پرس و جو

بیشتر آسیب پذیری های تزریق SQL در بند “WHERE” از “SELECT Query” بوجود می آیند. این نوع تزریق SQL معمولاً توسط آزمایش کنندگان باتجربه قابل درک است.

اما آسیب پذیری های تزریق SQL در اصل می توانند در هر مکانی از پرس و جو و در انواع مختلف نمایش داده شود. رایج ترین مکان ها که در آن تزریق SQL بوجود می آید یه شرح زیر می باشد:

* در بیانیه های UPDATE، در مقادیر به روز شده یا بند WHERE.
* در عبارت INSERT، درون مقادیر درج شده.
* در عبارت SELECT، در جدول یا نام ستون آمده است.
* در بیانیه های SELEC، در بند ORDER BY.

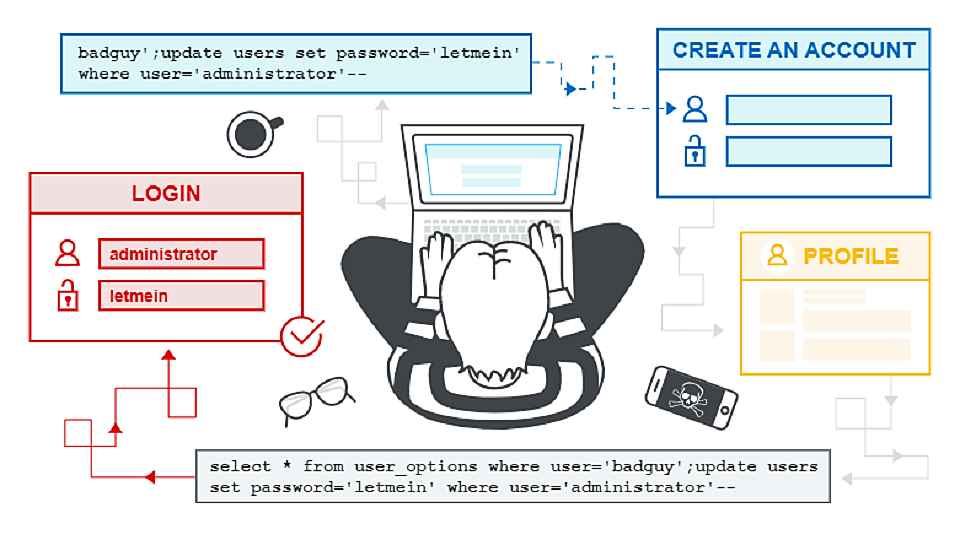
تزریق مرتبه دوم SQL

تزریق مرتبه اول SQL در جایی بوجود می آید که برنامه از کاربر درخواست HTTP دریافت کرده و در طی پردازش آن، ورودی را در یک پرس و جو SQL به روشی ناامن وارد می کند.

در تزریق SQL مرتبه دوم (که به عنوان تزریق SQL نیز شناخته می شود)، برنامه از درخواست HTTP ورودی کاربر را می گیرد و آن را برای استفاده بعدی ذخیره می کند.

این کار معمولاً با قرار دادن ورودی در یک پایگاه داده انجام می شود، اما در نقطه ای که داده ها ذخیره می شوند، آسیب پذیری ایجاد نمی شود.

بعداً هنگام رسیدگی به درخواست HTTP، برنامه داده های ذخیره شده را بازیابی می کند و آن را به روش نا امن در یک پرس و جو SQL می گنجاند.



تزریق مرتبه دوم SQL اغلب در شرایطی بوجود می آید که توسعه دهندگان از آسیب پذیری های تزریق SQL آگاه هستند و بنابراین با اطمینان می توانید محل اولیه ورود به پایگاه داده را کنترل کنید.

وقتی داده ها بعدا پردازش می شوند، بی خطر تلقی می شوند، زیرا در گذشته با خیال راحت در پایگاه داده قرار گرفته اند. در این مرحله، داده ها به شیوه ای نا امن اداره می شوند، زیرا توسعه دهنده فکر می کند که این موضوع قابل اعتماد است.

عوامل خاص در پایگاه داده

برخی از ویژگی های اصلی زبان SQL به همان روش در سراسر سیستم عامل های پایگاه داده محبوب اجرا می شود، بنابراین بسیاری از روش های شناسایی و بهره برداری از آسیب پذیری های تزریق SQL به طور یکسان در انواع مختلف بانک اطلاعاتی کار می کنند.

با این وجود تفاوت های زیادی بین بانک های اطلاعاتی مشترک نیز وجود دارد. این بدان معنی است که برخی از تکنیک های تشخیص و بهره برداری از تزریق SQL بر روی سیستم عامل های مختلف کار می کنند. مثلا:

* Syntax for string concatenation
* Comments
* Batched (or stacked) queries
* Platform-specific APIs
* Error messages

نحوه جلوگیری از تزریق SQL

بیشتر موارد تزریق SQL با استفاده از پرس و جو های پارامتری (همچنین به عنوان عبارت های آماده شناخته شده) به جای جمع کردن رشته در پرس و جو قابل پیشگیری هستند.

کد زیر در برابر تزریق SQL آسیب پذیر است زیرا ورودی کاربر مستقیماً در پرس و جو محاسبه می شود:

.String query="SELECT \* FROM products WHERE category="'"+input+"'".

.Statement statement = connection.createStatement();.

.ResultSet resultSet = statement.executeQuery(query);.

این کد به راحتی می تواند به شکلی نوشته شود که از دخالت ورودی کاربر در ساختار پرس و جو جلوگیری کند:

.PreparedStatement statement = connection.prepareStatement(.

. "SELECT \* FROM products WHERE category = ?");

.statement.setString(1, input);.

.ResultSet resultSet = statement.executeQuery();.

نمایش داده های پارامتری برای هر شرایطی که ورودی غیرقابل اعتماد به عنوان داده در پرس و جو ظاهر می شود، از جمله بند WHERE و مقادیر در عبارت INSERT یا UPDATE استفاده می شود.

از آن ها نمی توان برای کنترل ورودی غیرقابل اعتماد در سایر قسمت های پرس و جو، مانند نام جدول یا ستون یا بند ORDER BY استفاده کرد.

عملکرد برنامه های کاربردی که داده های غیرقابل اعتماد را در آن بخش از پرس و جو قرار می دهد، نیاز به رویکرد متفاوتی دارد، مانند لیست ورودی مقادیر مجاز یا استفاده از منطق مختلف برای ارائه رفتار مورد نیاز.

برای اینکه یک پرس و جو پارامتری در جلوگیری از تزریق SQL موثر باشد، رشته ای که در پرس و جو استفاده می شود باید همیشه ثابت دارای کدگذاری سخت باشد و هرگز نباید حاوی داده های متغیر از هر مبدأ باشد.

وسوسه نشوید که در مورد اینکه آیا به یک مورد داده اعتماد دارید تصمیم بگیرید و همچنان برای مواردی که از نظر امنیت در نظر گرفته شده است، استفاده از اتصال رشته را در پرس و جو ادامه دهید.

اشتباه کردن در مورد منشأ احتمالی داده ها یا تغییر در کد دیگر برای پیش فرض درباره داده هایی که رنگ می شوند، بسیار ساده است.