فصلنامه علمی-ترویجی پدافند غیرعامل
سال نهم، شماره ۳، پاییز ۱۳۹۷، (پیاپی ۳۵): صص ۱۱۷-۱۰۱

یک طبقه‌بندی از حملات تزریق SQL و روش‌های دفاع از این حملات در پدافند غیرعامل

میترا ترابی[۱]، علی شهیدی نژاد[۲*]

تاریخ دریافت: ۱۳۹۶/۰۸/۰۷
تاریخ پذیرش: ۱۳۹۶/۰۹/۲۵

چکیده

حملات تزریق SQL، یک تهدید امنیتی جدی برای برنامه‌های کاربردی تحت وب در فضای سایبری می‌باشند. حملات تزریق SQL، به مهاجمان اجازه می‌دهند تا دسترسی نامحدود به پایگاه داده‌ای که برنامه کاربردی و اطلاعات بالقوه حساس را شامل می‌شوند به‌دست آورند. اگرچه محققان و متخصصان، روش‌های مختلفی برای حل مسئله تزریق SQL، پیشنهاد کرده‌اند، اما رویکردهای فعلی یا به‌طور کامل برای حل محدوده‌ای از مشکل شکست خورده‌اند، یا محدودیت‌هایی دارند که از استفاده و پذیرش آن‌ها جلوگیری می‌کند. بسیاری از محققان و متخصصان، تنها با یک زیر مجموعه از طیف گسترده‌ای از روش‌های موجود برای دفاع در برابر حملات تزریق SQL، آشنا هستند. این مقاله، یک طبقه‌بندی، بر اساس یک بررسی گسترده از روش‌های فعلی، برای دفاع در برابر حملات تزریق SQL، ارائه می‌دهد. این طبقه‌بندی، به سازمان‌های نظامی و دولتی، برای درک بهتر روش‌های دفاع در برابر حملات تزریق SQL، کمک می‌کند. از این رو، بر اساس این طبقه‌بندی، سازمان‌های نظامی و دولتی، می‌توانند روش‌های مناسب بسته به منابع و محیط‌های موجود خود انتخاب کنند. برای مقابله با مشکل حملات تزریق SQL، این پژوهش، یک بررسی از انواع مختلف حملات تزریق SQL، که تا به امروز شناخته می‌شوند را همراه با نمونه‌هایی از نحوه حملات ارائه می‌دهد. روش‌های مختلف، برای تشخیص آسیب‌پذیری‌های تزریق SQL تشریح می‌شود و همچنین روش‌های تشخیص و پیشگیری موجود، در برابر حملات تزریق SQL، بررسی می‌شود. برای هر روش، در مورد ویژگی‌ها و نقاط قوت و ضعف آن در رسیدگی به حملات تزریق SQL، طبقه‌بندی، انجام می‌گیرد.

کلیدواژه‌ها: برنامه‌های کاربردی وب، امنیت پایگاه داده، حملات تزریق SQL، تشخیص، پیشگیری

۱- دانشجوی کارشناسی ارشد، موسسه غیر انتفاعی تعالی قم
۲- استادیار دانشگاه آزاد اسلامی واحد قم - a.shahidinejad@gmail.com - نویسنده مسئول

۱- مقدمه

تزریق SQL، یک آسیب‌پذیری رایج است که برای نفوذ در پایگاه داده‌های برنامه‌های کاربردی وب، با استفاده از اجرای یک کد SQL مخرب تزریق‌شده، توسط مهاجم، به‌کارگرفته می‌شود. همچنین تزریق SQL، به عنوان اولین آسیب‌پذیری خطرناک با توجه به آمارهای OWASP[1] طبقه‌بندی شده‌است [۱]. برنامه‌های کاربردی وب، که برای حملات تزریق SQL آسیب‌پذیرند، ممکن است به مهاجم اجازه دسترسی کامل به پایگاه داده‌های اساسی خود را بدهند. از آنجا که این پایگاه‌های داده، اغلب حاوی اطلاعات حساس مصرف‌کننده یا کاربر هستند، نقض‌های امنیتی ناشی از آن می‌تواند شامل سرقت هویت، از دست‌دادن اطلاعات محرمانه و تقلب باشد. در برخی موارد، مهاجمان حتی می‌توانند یک آسیب‌پذیری تزریق SQL را برای گرفتن کنترل و از کار انداختن سامانه‌هایی که برنامه کاربردی وب را میزبانی می‌کنند، استفاده کنند. برنامه‌های کاربردی وب که برای حملات تزریق SQL آسیب‌پذیرند گسترده هستند. در واقع، حملات تزریق SQL، به‌طور موفقیت‌آمیز، قربانیان بسیاری را هدف قرار داده‌اند.

تزریق SQL، به یک کلاس از حملات تزریق کد اشاره دارد که در آن، داده‌های ارائه‌شده توسط کاربر، در یک پرس‌وجوی SQL به گونه‌ای قرار می‌گیرند که بخشی از ورودی کاربر به عنوان کد SQL رفتار شود. با استفاده از این آسیب‌پذیری‌ها، مهاجم می‌تواند دستورات SQL را به‌طور مستقیم به پایگاه داده ارسال کند. این حملات تزریق SQL، یک تهدید جدی برای هر برنامه کاربردی وب است که از کاربران، ورودی دریافت می‌کند و آن را به درخواست‌های SQL در یک پایگاه داده پایه ترکیب می‌دهد [۲]. اکثر برنامه‌های کاربردی وب، که در اینترنت یا در سامانه‌های سازمانی استفاده می‌شوند، به این طریق کار می‌کنند و بنابراین می‌توانند به تزریق SQL آسیب‌پذیر باشند.

علت آسیب‌پذیری‌های تزریق SQL، نسبتا ساده و قابل درک است: اعتبارسنجی ناکافی از ورودی کاربر. برای حل این مشکل، توسعه‌دهندگان طیف وسیعی از دستورالعمل‌های برنامه‌نویسی که شیوه‌های برنامه‌نویسی دفاعی مانند رمزگذاری ورودی کاربر و اعتبارسنجی را ترویج می‌دهند. استفاده دقیق و اصولی از این روش‌ها، یک راه حل موثر، برای جلوگیری از آسیب‌پذیری‌های تزریق SQL است. با این حال در عمل، استفاده از چنین روش‌هایی مبتنی بر انسان است و بنابراین مستعد اشتباهات است. علاوه بر این، پاکسازی پایگاه‌های کدی که ممکن است آسیب‌پذیری‌های تزریق SQL را داشته باشند، می‌تواند یک کار بسیار کارآمد باشد. اگرچه، توجه زیادی به مشکل آسیب‌پذیری‌های تزریق SQL شده است، اما بسیاری از راه‌حل‌های پیشنهادی، برای رسیدگی به دامنه کامل مشکل موفق نشده‌اند. انواع مختلفی از حملات تزریق SQL و تغییرات بی‌شماری روی این انواع اساسی وجود دارد. محققان و متخصصان، اغلب از طیف وسیعی از روش‌های مختلف که می‌توانند برای انجام حملات تزریق SQL استفاده کنند، بی‌اطلاع هستند. بنابراین، بسیاری از راه‌حل‌های پیشنهادشده، تنها تعدادی از حملات تزریق SQL احتمالی را شناسایی یا جلوگیری می‌کنند. برای حل این مشکل، در این مقاله یک بررسی جامع از روش‌های تشخیص حملات تزریق SQL، از ابعاد مختلف مانند ماهیت دفاع، روش تجزیه و تحلیل، زمان تشخیص و مکان تشخیص انجام می‌گیرد و به طبقه‌بندی روش‌های مختلف استفاده‌شده به‌منظور دفاع در برابر حملات تزریق SQL، پرداخته می‌شود. این طبقه‌بندی، به‌منظور آشنایی با روش‌های مختلف و نقاط قوت و ضعف آن‌ها انجام می‌شود. بر اساس روش‌های تشخیص حملات تزریق SQL ذکرشده در مقاله، به همراه ارزیابی از ویژگی‌های مثبت کاذب و منفی کاذب، طبقه‌بندی صورت می‌گیرد. این طبقه‌بندی، به سازمان‌ها کمک می‌کند تا برای مقابله با مشکل حملات تزریق SQL، روش‌های مناسب، بسته به منابع و محیط‌های موجود خود انتخاب کنند. به‌طور مختصر، ساختار ادامه مقاله بدین شرح است: ابتدا در بخش دوم مقاله، پیش زمینه‌ای از حملات تزریق SQL و مفاهیم مرتبط با موضوع تحقیق شرح داده می‌شود. در بخش سوم، انواع حملات تزریق SQL، همراه با جزئیات آن‌ها بیان می‌شوند. در بخش چهارم، روش‌های دفاع در برابر حملات تزریق SQL ارائه می‌شود. در بخش پنجم، طبقه‌بندی روش‌های دفاع در برابر حملات تزریق SQL ارائه می‌شود و در نهایت، این مقاله در بخش ششم، با بیان جمع‌بندی و نتیجه‌گیری به اتمام می‌رسد.

۲- پیش زمینه‌ای از حملات تزریق SQL

به‌طور مستقیم، یک حمله تزریق SQL زمانی رخ می‌دهد که یک مهاجم، اثر دل‌خواه پرس‌وجوی SQL را با قرار دادن کلیدواژه‌های جدید یا عملگرهای SQL در پرس‌وجو تغییر می‌دهد. این تعریف غیررسمی، شامل انواع حملات تزریق SQL است که در این مقاله ارائه شده است. خوانندگان علاقه‌مند می‌توانند برای تعریف رسمی حملات تزریق SQL به [۳] مراجعه کنند. در ادامه مقاله، دو ویژگی مهم حملات تزریق SQL تعریف می‌شود که برای توصیف حملات استفاده می‌شوند: ساز و کار تزریق و هدف حمله.

1- Open Web Application Security Project

۲-۱- ساز و کار تزریق

دستورات مخرب SQL، به یک برنامه کاربردی آسیب‌پذیر، را می‌توان از طریق بسیاری از ساز و کارهای ورودی مختلف، معرفی کرد. در این بخش، ساز و کارهای رایج توضیح داده می‌شود.

- تزریق از طریق ورودی کاربر: در این مورد، مهاجمان دستورات SQL را با ارائه مناسب ورودی کاربر ساخته‌شده، تزریق می‌کنند. یک برنامه کاربردی وب، می‌تواند ورودی کاربر را در چندین روش بر اساس محیطی که برنامه کاربردی در آن مستقر است بخواند. در اکثر حملات تزریق SQL، که برنامه کاربردی وب را هدف قرار می‌دهند، ورودی‌های کاربر، معمولا از ارسال فرم‌هایی است که به برنامه کاربردی وب از طریق درخواست‌های HTTP GET یا POST ارسال می‌شوند. برنامه‌های کاربردی وب، به‌طور کلی می‌توانند به ورودی کاربر در این درخواست‌ها دسترسی پیدا کنند زیرا آن‌ها به هر متغیر دیگری در محیط دسترسی دارند.

- تزریق از طریق کوکی‌ها: کوکی‌ها، فایل‌هایی هستند که اطلاعات وضعیت تولید شده، توسط برنامه کاربردی وب را شامل می‌شوند و در دستگاه سرویس‌گیرنده ذخیره می‌شوند. هنگامی که سرویس‌گیرنده به یک برنامه کاربردی وب بازمی‌گردد، کوکی‌ها می‌توانند برای بازگرداندن اطلاعات مربوط به وضعیت سرویس‌گیرنده استفاده شوند. از آنجا که سرویس‌گیرنده، کنترل ذخیره‌سازی کوکی را دارد، یک سرویس‌گیرنده مخرب می‌تواند با محتویات کوکی، مداخله کند. اگر یک برنامه کاربردی وب، از محتویات کوکی برای ساختن پرس‌وجوهای SQL استفاده کند، مهاجم به راحتی می‌تواند یک حمله را توسط جاسازی آن در کوکی ارسال کند.

- تزریق از طریق متغیرهای سرور: متغیرهای سرور، مجموعه ای از متغیرها هستند که حاوی HTTP، سرآیندهای شبکه و متغیرهای محیطی هستند. برنامه‌های کاربردی وب، از این متغیرهای سرور به روش‌های مختلفی استفاده می‌کنند، مانند آمارهای استفاده از ورود به سیستم و شناسایی روندهای مرور. اگر این متغیرها بدون پاکسازی به یک سیستم پایگاه داده وارد شوند، این می‌تواند یک آسیب‌پذیری SQL ایجاد کند. به دلیل این‌که، مهاجمان می‌توانند مقادیری را که در HTTP و سرآیندهای شبکه قرار می‌گیرند جعل کنند. آن‌ها می‌توانند از این آسیب‌پذیری با قرار دادن یک حمله تزریق SQL، به‌طور مستقیم در سرآیندها، بهره‌برداری کنند. هنگامی‌که پرس‌وجو برای وارد کردن متغیر سرور، به پایگاه داده صادر می‌شود حمله در سرآیند جعلی می‌باشد سپس فعال می‌شود.

- تزریق مرتبه دوم: در تزریق‌های مرتبه دوم، مهاجمان، ورودی‌های مخرب را به یک سیستم یا پایگاه داده وارد می‌کنند که به‌طور غیرمستقیم به حمله تزریق SQL منجر می‌شود، زمانی‌که این ورودی در زمان بعدی استفاده شود. هدف از این نوع حمله، به‌طور قابل‌توجهی از یک حمله تزریق منظم (یعنی مرتبه اول) متفاوت است. تزریقات مرتبه دوم سعی ندارند که باعث شوند هنگامی‌که ورودی مخرب در ابتدا به پایگاه داده می‌رسد حمله رخ دهد. در عوض، مهاجمان بر دانشی که در آن ورودی متعاقبا مورد استفاده قرار می‌گیرد، تکیه می‌کنند و به آن حمله می‌کنند به‌طوری‌که، در طول استفاده از آن حمله رخ دهد. تزریق مرتبه دوم، می‌تواند به‌طور خاص برای تشخیص و پیشگیری دشوار باشد، زیرا نقطه تزریق، متفاوت از نقطه‌ای است که در آن حمله واقعا خودش را نشان می‌دهد.

۲-۲- هدف حمله

حملات نیز می‌توانند بر اساس مقصود یا هدف مهاجم مشخص شوند. بنابراین، هریک از تعاریف نوع حمله که در بخش (۳) ارائه می‌شوند، شامل فهرستی از یک یا چند هدف حمله است که در این بخش تعریف شده است. اهداف حمله به شرح زیر می‌باشند:

- شناسایی پارامترهای تزریقی: مهاجم می‌خواهد یک برنامه کاربردی وب را بررسی کند تا مشخص شود کدام پارامتر و فیلدهای ورودی کاربر به حملات تزریق SQL آسیب‌پذیر است.

- انگشت‌نگاری پایگاه داده: مهاجم تمایل دارد نوع و نسخه پایگاه داده‌ای را که یک برنامه‌نویس استفاده می‌کند، کشف کند. انواع خاصی از پایگاه‌های داده، به‌طور متفاوتی به پرس‌وجوها و حملات مختلف پاسخ می‌دهند و این اطلاعات می‌توانند برای "انگشت‌نگاری" پایگاه داده مورد استفاده قرارگیرند. دانستن نوع و نسخه پایگاه داده مورد استفاده توسط یک برنامه کاربردی وب، به مهاجم اجازه می‌دهد تا حملات خاص پایگاه داده را انجام دهد.

- تعیین شمای پایگاه داده: برای استخراج صحیح داده‌ها از یک پایگاه داده، مهاجم اغلب نیاز به دانستن اطلاعات شمای پایگاه داده مانند نام جدول، نام ستون، انواع داده‌های ستون دارد. حملات با این هدف، برای جمع‌آوری یا استنتاج این نوع از اطلاعات ایجاد می‌شوند.

- استخراج داده‌ها: این نوع از حملات، از روش‌هایی استفاده می‌کنند که مقادیر داده را از پایگاه داده استخراج می‌کنند. بسته به نوع برنامه کاربردی وب، این اطلاعات می‌تواند برای مهاجم حساس و بسیار مطلوب باشد. حملات با این تعریف، معمول‌ترین نوع حمله تزریق SQL هستند.

- افزودن یا تغییر داده‌ها: هدف این حملات، افزودن یا تغییر اطلاعات در پایگاه داده است.

- انجام حمله انکار سرویس[1]: این حملات برای تعطیل کردن یا بستن پایگاه داده یک برنامه کاربردی وب انجام می‌شوند. بنابراین، سرویس را به سایر کاربران انکار می‌کنند. حملات شامل قفل کردن یا از قلم انداختن جداول پایگاه داده نیز در این دسته قرار می‌گیرند.

- گریز زدن یا از سر بازکردن تشخیص: این دسته به روش‌های حمله خاصی که برای جلوگیری از حساب‌رسی و تشخیص، توسط ساز و کارهای حفاظت از سیستم استفاده می‌شود اشاره دارد.

- دور زدن یا عبور از احراز هویت[2]: هدف این نوع از حملات، این است که به مهاجم اجازه داده شود از ساز و کارهای احراز هویت پایگاه داده و برنامه کاربردی جلوگیری کند. دور زدن چنین ساز و کاری، می‌تواند به مهاجم اجازه دهد حقوق و امتیازات مربوط به کاربر دیگری را در نظر بگیرد.

- اجرای دستورات از راه دور: این نوع از حملات، برای اجرای دستورات دل‌خواه در پایگاه داده تلاش می‌کنند. این دستورات می‌توانند رویه‌ها یا توابع ذخیره‌شده در دسترس، برای کاربران پایگاه داده را ذخیره کنند.

- انجام ترفیع امتیازات: این حملات، از مزایای خطاهای پیاده‌سازی یا معایب منطقی در پایگاه داده به‌منظور افزایش امتیازات مهاجم استفاده می‌کنند. در مقابل حملات دور زدن احراز هویت، این حملات بر بهره‌برداری از امتیازات کاربر پایگاه داده تمرکز دارد.

۳- انواع حملات تزریق SQL

مطابق با [۴] انواع مختلفی از روش‌های تزریق SQL وجود دارند. هر نوع می‌تواند در انزوا یا به‌صورت ترکیبی انجام شوند. این به تجربیات، اهداف و رفتار مهاجم بستگی دارد. در این بخش، انواع مختلفی از حملات تزریق SQL مورد بحث قرار خواهد گرفت. علاوه بر این، برای هر نوع یک مثال گویا وجود دارد.

۳-۱- پرس‌وجوی درست‌نما[3]

دور زدن احراز هویت، شناسایی پارامترهای تزریقی و استخراج داده از اهداف این حمله می‌باشد.

در این روش، اظهارات شرطی معمولا صحیح[4] را بر می‌گردانند یا صحیح ارزیابی می‌شوند. هنگامی‌که مهاجم، اظهارات شرطی پرس‌وجوی برنامه کاربردی وب را با کد مخرب تزریق می‌کند، مهاجم قصد دارد تا مقدار اظهارات شرطی برابر با صحیح را حفظ کند. این روش را معمولا در صفحه ورود به سایت برای تزریق در فیلد ورود با " -- or 1=1" به‌کار می‌گیرند [۴-۶].

۳-۲- پرس‌وجوی Piggy- Backed

استخراج داده، افزودن یا تغییر داده‌ها، انجام انکار سرویس و اجرای دستورات از راه دور از اهداف این حمله می‌باشد.

هدف از این نوع حمله، تزریق پرس‌وجوی اصلی با یک پرس‌وجوی اضافی می‌باشد. همه پرس‌وجوها به ترتیب شروع، با یکی از پرس‌وجوی اصلی اجرا خواهند شد. این حمله از دیگر حملات متفاوت است زیرا مهاجمان، پرس‌وجوی اصلی را ویرایش یا تغییر نمی‌دهند، آن‌ها تنها سعی دارند که پرس‌وجوهای جدیدی اضافه و آن‌ها را به پرس‌وجوی اصلی پیوست کنند. بر این اساس، موتور پایگاه داده بیشتر از یک پرس‌وجو دریافت خواهد کرد، اولین پرس‌وجو به‌طور نرمال اجرا خواهد شد سپس دومی یا دیگر پرس‌وجوها اجرا خواهند شد. در نتیجه، اگر پرس‌وجوی دوم به‌طور موفقیت‌آمیز اجرا شده باشد، مهاجم می‌تواند هر دستور SQL مانند روال‌های ذخیره‌شده یا هر دستور دیگر را اجرا و تزریق کند. این نوع آسیب‌پذیری به‌طور نرمال، به یک پیکربندی موتور پایگاه داده خاص، نیاز دارد تا به مهاجم اجازه‌ی اجرای دستورات SQL زیان‌آور را بدهد. به عبارت دیگر، پیکربندی موتور پایگاه داده به سیستم پایگاه داده اجازه می‌دهد که رشته واحد متشکل از دستورات چندگانه اجرا شود. برای مثال، فرض کنید که کد روبه‌رو " ; drop table UserTable - - " در فیلد ورود به سایت از صفحه ورود به سیستم وارد شده‌است. این سناریو به شرح زیر خواهد بود:

```
SELECT * FROM UserTable WHERE username = ' any
;DROP TABLE UserTable - - 'AND
userpassword =' user_entry_password'
```

پس از ارسال صفحه ورود به سایت[5]، برنامه کاربردی این اطلاعات را به موتور پایگاه داده خواهد فرستاد. سپس، موتور پایگاه داده پرس‌وجوی ورود به سایت را طبق روال معمولی اجرا خواهد کرد. همان‌طور که پرس‌وجو اجرا می‌شود، موتور پایگاه داده جداکننده پرس‌وجو ";" را کشف خواهد کرد، همچنین پایگاه داده به‌طور پیش فرض کد تزریق‌شده را اجرا خواهد کرد. در این مرحله جدول کاربر حذف خواهد شد و سیستم داده‌ی کاربر را از دست خواهد داد.

1- Danial of Service
2- Bypassing Authentication
3- Tautology Query
4- True
5- Login

توجه داشته باشید که تفاوت، بین موتور پایگاه داده، برای جداسازی پرس‌وجو وجود دارد. بر این اساس، روش خوب برای تشخیص و پیشگیری از این نوع حمله، استفاده از یک روش موثر برای اعتبارسنجی داده کاربر در زمان اجرا توسط اسکن و تجزیه و تحلیل پرس‌وجوها، برای کشف جداکننده‌های پرس‌وجو، همچنین یک تنظیمات پایگاه داده امن می‌باشد [۷-۶].

۳-۳- پرس‌وجوی اجتماع [۱]

دور زدن احراز هویت و استخراج داده از اهداف حمله پرس‌وجوی اجتماع می‌باشد.

ایده‌ی پشت این حمله پرس‌وجوی UNION، به انواع دیگر تزریق SQL شبیه است. مهاجمان، به دنبال یک پارامتر آسیب‌پذیر هستند و سعی دارند به بهره‌برداری از آن با استفاده از تغییر مجموعه داده‌ای که برای یک پرس‌وجو ارسال شده برگردانده می‌شود بپردازند. علاوه بر این، با استفاده از این روش، برنامه کاربردی، نتایج مختلفی از پایگاه داده به جای یک برنامه‌نویسی توسط توسعه‌دهنده دریافت خواهد کرد. این روش، توسط تزریق پارامتر آسیب‌پذیر، با استفاده از کلمه کلیدی UNION SELECT آغاز می‌شود، بنابراین، مهاجم می‌تواند پرس‌وجوی دوم را برای به‌دست آوردن اطلاعات پایگاه داده کنترل کند. علاوه بر این، داده از هر جدولی در دسترس خواهد بود و مهاجم تنها باید انتخاب کند که چه داده‌ای یا از هر جدول خاص می‌خواهد. توجه داشته باشید که مراحل حمله قبلی با دیگر انواع حمله تزریق SQL به مهاجم اجازه می‌دهد که ساختار پایگاه داده را قبل از شروع با این روش بداند مانند یک حمله پرس‌وجوی غیرقانونی [۸-۵].

۳-۴- پرس‌وجوی نادرست از لحاظ منطقی یا غیرقانونی [۲]

شناسایی پارامترهای تزریقی، انجام انگشت‌نگاری پایگاه داده و استخراج داده از اهداف این حمله می‌باشد.

پرس‌وجوی نادرست از لحاظ منطقی یا پرس‌وجوی غیرمجاز، یک نوع حمله تزریق SQL است که در مراحل اولیه از یک حمله برای جمع‌آوری اطلاعات درباره پایگاه داده مانند نوع پایگاه داده، ستون‌های جدول و نوع ستون یا دیگر اطلاعات استفاده می‌شود. در این نوع از حمله، ورودی یک جمله نادرست از لحاظ منطقی می‌باشد که موجب یک پاسخ خطای پایگاه داده می‌شود. مانند اضافه کردن ۱=۲ به جمله شرط. بنابراین، این روش معمولا، با تزریق پارامتر آسیب‌پذیر از صفحه وب با یک دستور نادرست (به لحاظ منطقی) برای ایجاد یک خطا از موتور پایگاه داده آغاز شده است. علاوه بر این، این روش می‌تواند به عنوان یک تزریق کور مورد استفاده قرار گیرد و مهاجم می‌تواند به پاسخ برنامه کاربردی وب نظارت داشته باشد. بدین ترتیب، مهاجم می‌تواند بازخوردی از موتور پایگاه داده را مطابق با خطای آن به‌دست آورد [۴، ۹ و ۱۰].

۳-۵- پرس‌وجوی روال‌های ذخیره‌شده [۳]

انجام تشدید امتیاز، انجام انکار سرویس و اجرای دستورات از راه دور از اهداف این نوع حمله می‌باشد.

این روش حمله تزریق SQL، برای اجرا یا ایجاد روال‌های ذخیره‌شده به‌کار می‌روند که توسط موتور پایگاه داده استفاده شده‌اند. روال ذخیره‌شده، معمولا توسط توسعه‌دهنده یا مدیر پایگاه داده برای کنترل پایگاه داده و استفاده از مزایای امکانات پایگاه داده استفاده شده است. مانند دسترسی به پایگاه داده یا سرویس‌های پایگاه داده. روال‌های ذخیره‌شده شبیه به یکدیگر نیستند، به عنوان مثال، پایگاه داده اوراکل [۴] شبیه به پایگاه داده MYSQL یا MS-SQL نیست. به این ترتیب، مهاجم نیاز دارد که نوع پایگاه داده را، برای بهره‌برداری این آسیب‌پذیری تعیین کند. بنابراین، مهاجم می‌تواند با یک نوع حمله پرس‌وجوی نادرست از لحاظ منطقی، نوع پایگاه داده را تعیین کند، سپس مهاجم می‌تواند حمله Stored Procedure را استفاده کند [۱۱].

۳-۶- پرس‌وجوی استنتاجی [۵]

شناسایی پارامترهای تزریقی، استخراج داده و تعیین طرح پایگاه داده از اهداف این نوع حمله می‌باشد.

این روش حمله، هنگامی‌که مهاجم قادر نیست هر گونه پیام تعاملی را از طریق یک دستور تزریق SQL به‌دست آورد، استفاده می‌شود. بنابراین، مهاجمان به دنبال پیدا کردن روش‌های دیگر برای افشای آسیب‌پذیری وب سایت هستند. مهاجم در اینجا یک پاسخ برنامه کاربردی وب را توسط تزریق آن با کلمه‌های کلیدی SQL مختلف تخمین می‌زند، تا این‌که، او اطلاعات مورد نیاز از پایگاه داده را برای شروع حمله‌ی خود به‌دست می‌آورد. این نوع از حمله به‌طور کلی به انواع زیر مجموعه‌هایی به شرح زیر تقسیم می‌شود:

۳-۶-۱- استنتاج تزریق کور [۶]

در این روش، مهاجمان به صفحه وب با یک جمله شرطی تزریق می‌کنند که به آن‌ها کمک می‌کند تا به آرایش پایگاه داده از طریق ارزیابی پاسخ موتور پایگاه داده با جمله شرطی تزریق پی‌برند، خواه

1- UNION Query
2- Illegal/Logically Incorrect Queries
3- Stored Procedures
4- Oracle
5- Inference Query
6- Blind Injection Inference

جمله درست یا غلط باشد. در این مرحله، اگر جمله درست ارزیابی شود سیستم کار را، به‌صورت نرمال ادامه خواهد داد. در نتیجه، اگر جمله تزریق‌شده غلط ارزیابی شود، صفحه وب یک پیغام خطا باز نخواهد گرداند. همان‌طورکه، صفحه وب به‌طور نرمال کار نخواهد کرد، برای مثال، تفاوت‌هایی بین رفتار صفحه قبل از جمله تزریق و بعد از آن وجود دارد. بنابراین، مهاجم اینجا اطلاعات را با استفاده از مقایسه نتایج پاسخ از پرس‌وجوها با تزریق دستور تزریق‌شده true یا false جمع‌آوری خواهد کرد [۱۲].

۳-۶-۲- پرس‌وجوی زمان بندی استنتاجی[1]

در این روش، مهاجم پرس‌وجوهایی با یک دستور مخرب تزریق می‌کند، که یک تاخیر سیستم ایجاد کند. سپس مهاجم، واکنش برنامه کاربردی وب را با نظارت بر زمان پاسخ و جمع‌آوری اطلاعات درباره پایگاه داده، طبق این پاسخ مشاهده خواهد کرد. در صورتی که تاخیری وجود داشت، دستور تزریق‌شده با موفقیت اجرا می‌شود، در غیر این‌صورت، اجرای دستور مشکل دارد و مهاجم نیازمند تغییر دستور تزریق‌شده می‌باشد. در نتیجه، روش‌های مختلفی برای تزریق به برنامه کاربردی وب با استفاده از این نوع از حمله وجود دارد مانند استفاده از تابع تاخیر [۱۰، ۱۲ و ۱۳].

۳-۷- رمزگذاری متناوب[2]

هدف از این حمله، گریز زدن از تشخیص می‌باشد. روش‌های حمله نرمال، به دنبال دانستن کاراکترها یا کلمات کلیدی هستند، که معمولا کاراکترهای بد نامیده می‌شوند. در این روش، مهاجمان از روش‌های تشخیص نرمال، با استفاده از متن تزریق‌شده که رمزگذاری متناوب به‌کار می‌گیرد، می‌گریزند. رمزگذاری متناوب، متن تزریق‌شده رمزگذاری‌شده در اسکی، یونیکد یا هگزادسیمال را استفاده می‌کند. به این ترتیب، اهداف مهاجم نمی‌تواند تعریف شده باشد، بنابراین، مهاجم می‌تواند بیشتر از یک روش رمزگذاری استفاده کند. در نتیجه، در طول توسعه برنامه کاربردی، توسعه‌دهنده باید امنیت برنامه کاربردی تحت وب را در برابر این نوع از حمله، با استفاده از روش موثری که احتمالات مختلف از متن رمزگذاری مخرب را در نظر می‌گیرد، برای پیشگیری این نوع از حمله تامین کند.

بنابراین، این روش حمله، مانند حمله در بخش‌های قبلی یکسان نیست زیرا پیشگیری موثر در برابر این نوع به در نظر گرفتن همه رمزگذاری‌های تزریق‌شده احتمالی که می‌تواند برای برنامه کاربردی تحت وب زیان‌آور باشد، نیاز خواهد داشت.

1- Timing Inference Query
2- Alternate Encoding

۳-۸- توضیحات درون خطی[3]

این حمله تزریق SQL، می‌تواند با همه روش‌های حمله قبلی استفاده شود، زیرا مهاجم می‌تواند دستور تزریق را با استفاده از قابلیت‌های برنامه‌نویسی، از توضیحات درون خطی تقسیم کند. این روش، می‌تواند مهاجم را برای دور کردن از روش‌های تشخیص و پیشگیری اولیه که به دنبال یک کاراکتر خاص هستند پشتیبانی کند.

مطابق آنچه ذکر شد، انواع مختلف از آسیب‌پذیری حمله تزریق SQL وجود دارد. این طبقه‌بندی از تزریق SQL، مفید است زیرا به توسعه‌دهنده کمک می‌کند تا آسیب‌پذیری‌های تزریق SQL در طول مرحله توسعه برنامه کاربردی را تشخیص و تعمیر کند. روش مفید دیگر برای تشخیص آسیب‌پذیری‌های تزریق SQL، تعیین همه روش‌های تزریق احتمالی می‌باشد، تا بدانید چطور انواع آسیب‌پذیری‌ها می‌تواند مورد بهره‌برداری قرار گیرند.

۴- روش‌های دفاع در برابر حملات تزریق SQL

محققان، طیف گسترده‌ای از روش‌ها را برای حل مشکل تزریق SQL، پیشنهاد کرده‌اند. این روش‌ها، از بهترین شیوه‌های توسعه تا چارچوب‌های کاملا خودکار، برای تشخیص و جلوگیری از حملات تزریق SQL را در بر می‌گیرند. در این بخش، مقاله یک طبقه‌بندی، برای روش‌های دفاع از حملات تزریق SQL، ارائه می‌دهد. روش‌های تشخیص حملات تزریق SQL را از ابعاد مختلف مانند ماهیت دفاع، روش تجزیه و تحلیل، زمان تشخیص و مکان تشخیص بررسی می‌کند. شکل (۱)، روش‌های تشخیص حملات تزریق SQL و ابعاد مختلف آن را نشان می‌دهد. سپس هریک از ابعاد در بخش‌های بعدی شرح داده خواهد شد.

۴-۱- طبقه‌بندی بر اساس ماهیت دفاع

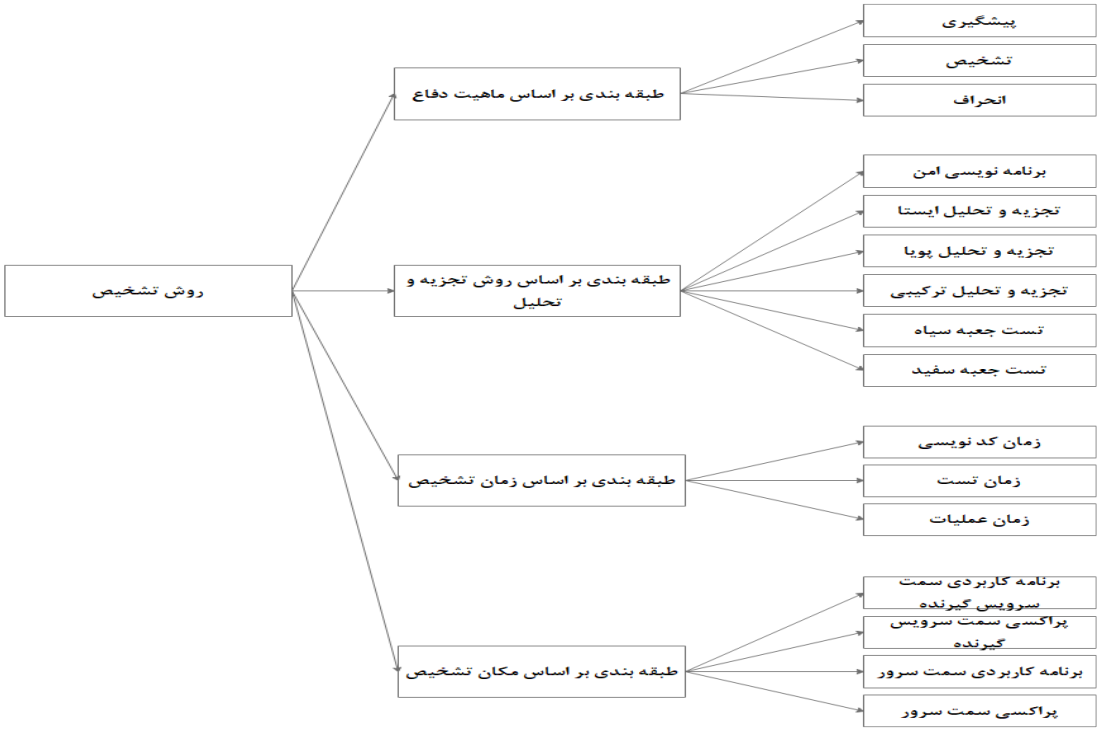
محققان، روش‌های دفاع مختلف بسیاری را برای حل مشکل حملات تزریق SQL، پیشنهاد داده‌اند. این روش‌ها، می‌توانند در ماهیت دفاع متفاوت باشند. ماهیت دفاع، به این مفهوم است که چطور یک روش، قصد دفاع در برابر حملات تزریق SQL، به برنامه کاربردی وب را دارد. در طول این مقاله، سه نوع ماهیت دفاع، در زمینه روش‌های دفاع از حملات تزریق SQL شناسایی شده است، که عبارت اند از: پیشگیری، تشخیص و انحراف.

۴-۱-۱- پیشگیری

روش پیشگیری، احتمال موفقیت حملات تزریق SQL را، با استفاده از شناسایی ایستای آسیب‌پذیری‌هایی در کد، ارائه یک الگوی توسعه متفاوت برای تولید پرس‌وجوهای SQL، یا بازرسی برنامه کاربردی

3- Inline Comments

برای اجرا کردن بهترین شیوه‌های برنامه‌نویسی دفاعی، در طول توسعه، دفع یا به شدت منع می‌کند.



شکل (۱) - طبقه‌بندی روش‌های تشخیص حملات تزریق SQL

۴-۱-۲- تشخیص

روش تشخیص، بین تلاش‌های تزریق SQL و آماده‌سازی از فعالیت و هشدار سیستم تبعیض قائل می‌شود و عمدتا، حملات تزریق SQL را، در طول زمان عملیات، تشخیص می‌دهد. پس از تشخیص حملات، این روش، به متصدی امور هشدار می‌دهد، به‌طوری‌که، آن‌ها بتوانند عملیات خاصی را همچون رد کردن و فرار از حملات انجام دهند.

۴-۱-۳- انحراف[1]

روش انحراف، منجر به هدایت مهاجمی می‌شود، که بر این باور است که در یک تلاش تزریق موفق شده است در حالی‌که، واقعیت این است که او تنها موفق به مصالحه اطلاعات نادرست شده است. این روش به گونه ای طراحی شده است که مهاجمان، به راحتی به سمت آن جلب می‌شوند. این روش به یادگیری بیشتر درباره حملات تزریق SQL مختلف و عادات حمله آن‌ها کمک می‌کند. هانی‌پات، تنها یک روشی است که تحت این دسته اتفاق می‌افتد. هانی‌پات‌ها، یک مفهوم پیشرفته در امنیت شبکه می‌باشند. هدف از چنین سیستمی کسب اطلاعاتی در مورد تلاش‌های نفوذ و یا نفوذهایی از یک منبع است. این اطلاعات می‌تواند بسیار متفاوت باشد. به عنوان مثال زمان، تاریخ، آدرس IP نفوذگر، سیستم‌عامل نفوذگر، فهرست واژه‌های به‌کارگرفته شده، بهره‌برداری‌ها و دستورات پس از نفوذ [۱۴].

۴-۲- طبقه‌بندی بر اساس روش تجزیه و تحلیل

روش‌های تشخیص آسیب‌پذیری‌های تزریق SQL، چندین روش تجزیه و تحلیل مختلف را برای تشخیص آسیب‌پذیری‌های موجود در برنامه‌های کاربردی وب، به‌کار می‌گیرند. در این پژوهش، شش نوع مختلف از روش‌های تجزیه و تحلیل ارائه شده است: برنامه‌نویسی امن، تجزیه و تحلیل ایستا، تجزیه و تحلیل پویا، تجزیه و تحلیل ترکیبی، آزمون جعبه سیاه، آزمون جعبه سفید.

۴-۲-۱- برنامه‌نویسی امن[2]

برنامه‌نویسی امن، یک رویکرد برنامه‌نویسی دفاعی، به‌منظور کاهش آسیب‌پذیری‌های تزریق SQL، با اجرای روال‌های اعتبارسنجی یا با استفاده از استاندارد API یا کتابخانه کلاس‌های موجود برای ساخت دستور در کد منبع برنامه کاربردی در طول توسعه می‌باشد. بسیاری

1- Deflection
2- Secure Programming

از کتابخانه‌های امن و در دسترس ارائه‌شده توسط فروشندگان وجود دارد. SQLDOM، توسعه‌دهندگان را به استفاده از مجموعه‌ای از کلاس‌های به شدت ماشینی‌شده برای شمای پایگاه داده، تشویق می‌کند. به جای استفاده از دستکاری رشته برای ساخت پرس‌وجوهای SQL پویای برنامه‌ریزی شده توسط توسعه‌دهندگان، از API امن، برای تولید دستورهای SQL استفاده می‌کند، که محافظت از امنیت و در نتیجه گریز از مشخصه‌های مخرب برای جلوگیری از حملات تزریق SQL را در بر خواهد داشت. اشکال اصلی برنامه‌نویسی امن، این است که نیازمند آموزش توسعه‌دهنده برای یادگیری استفاده مناسب از کتابخانه‌های امن می‌باشد.

۴-۲-۲- تجزیه و تحلیل ایستا[1]

روش‌های تجزیه و تحلیل ایستا، آثار برنامه‌های کاربردی مانند کد منبع، کد باینری، بایت کد و پیکربندی فایل‌ها به‌منظور رسیدن به اطلاعات، در مورد برنامه کاربردی را تجزیه و تحلیل می‌کنند. اطلاعات می‌توانند مانند این باشند، که چطور داده در زمان اجرا، بدون اجرایی‌کردن کد، جریان دارد. برخی از روش‌ها، روی تجزیه و تحلیل ایستای مجتمع، به‌منظور شناسایی آسیب‌پذیری‌های ممکن در کد تکیه می‌کنند، برای مثال [۱۶-۱۵]. چنین تجزیه و تحلیل ایستای محافظه‌کار، می‌تواند تعداد زیادی مثبت کاذب را تولید کند. مثالی از تجزیه و تحلیل ایستا، مدل ADMIRE می‌باشد. نویسندگان، مدل ریسک تهدید جامع و مرحله‌ای ADMIRE، برای شناسایی و مقابله با حملات تزریق SQL، توسط محافظت از پایگاه داده زیرین را پیشنهاد کرده‌اند. مدل ADMIRE، ارزیابی خطر را تجزیه و تحلیل می‌کند. این مدل، شش مرحله را برای مقابله با حملات تزریق SQL، دنبال می‌کند: ۱- اهداف امنیتی را تجزیه و تحلیل می‌کند. ۲- برنامه کاربردی را تقسیم می‌کند. ۳- آسیب‌پذیری‌ها را علامت‌گذاری می‌کند. ۴-تهدیدات را شناسایی می‌کند. ۵- تهدید را رتبه‌بندی می‌کند. ۶- تهدید را از بین می‌برد. مدل تهدید ADMIRE، به توسعه‌دهندگان و طراحان، یک درک بهتری از برنامه کاربردی ارائه می‌دهد و کمک به پیدا کردن اشکالات می‌کند. این یک رویکرد پیشگیرانه و یک بخش حیاتی از فرآیند طراحی می‌باشد.

۴-۲-۳- تجزیه و تحلیل پویا[2]

روش‌های تجزیه و تحلیل پویا، اطلاعات به‌دست‌آمده در طول اجرای برنامه را برای تشخیص آسیب‌پذیری‌های تزریق SQL، تجزیه و تحلیل می‌کنند. اطلاعات ممکن است درخواست‌ها، الگوهای پاسخ و ساختار پرس‌وجو باشد. تجزیه و تحلیل پویا، می‌تواند در زمان آزمون، در طول توسعه و یا در زمان اجرا، پس از انتشار انجام گیرد. اشکال تجزیه و تحلیل پویا، این است که، تنها آسیب‌پذیری‌هایی در مسیرهای اجرایی را تشخیص می‌دهد؛ اما نمی‌تواند قسمت‌های اجرانشده در کد را تشخیص دهد. به عنوان مثالی از تجزیه و تحلیل پویا، روش X-log authentication توضیح داده می‌شود. این روش، برای پیشگیری از حملات تزریق SQL، در برنامه کاربردی وب پیشنهاد شده‌است. ایده اصلی این روش، استفاده از روش‌های سه‌گانه پالایش برای پیشگیری از تزریق SQL است، که در زیر شرح داده می‌شود:

- محافظ آسیب‌پذیری: برای شناسایی فراکاراکترها، کاراکترهایی که نشان‌دهنده مجموعه‌ای از کاراکترها می‌باشند مانند txt.*، برای پیشگیری از حملات تزریق SQL می‌باشد.

- احراز هویت ورودی (X-log authentication): برای بررسی ورودی کاربر، از X- Log Generator، که در آن یک پایگاه داده معتبر به‌طور جداگانه ذخیره شده است و سپس، فیلد ورودی کاربر اعتبارسنجی‌شده، مجاز به ادامه دادن، می‌شود.

- رویه ذخیره‌شده: برای بررسی اندازه و نوع داده ورودی کاربر و انجام اعتبارسنجی سمت سرور می‌باشد. نویسندگان، این روش را در بسیاری از برنامه‌های کاربردی وب، آزمایش کرده‌اند و استدلال می‌کنند که از انواع حملات تزریق SQL پیشگیری می‌کند.

دو رویکرد مرتبط SQLGuard و SQLCheck، نیز پرس‌وجوها را در زمان اجرا بررسی می‌کنند تا ببینند که آیا آن‌ها مطابق با یک مدل از درخواست‌های مورد انتظار است. در این رویکردها، مدل به عنوان یک گرامر بیان می‌شود که تنها پرس‌وجوهای قانونی را می‌پذیرد. در SQLGuard، مدل در زمان اجرا، با بررسی ساختار پرس‌وجو قبل و پس از اضافه کردن ورودی کاربر محاسبه می‌شود. در SQLCheck، مدل به‌طور مستقل توسط توسعه‌دهنده مشخص‌شده است. هر دو روش، از یک کلید مخفی، برای تعیین ورودی کاربر، در هنگام تجزیه، توسط کنترل‌کننده زمان اجرا، استفاده می‌کنند، بنابراین، امنیت این رویکرد، بستگی به مهاجمانی دارد که قادر به کشف کلید نیستند. علاوه بر این، استفاده از این دو رویکرد، نیاز به توسعه‌دهنده دارد که یا کد را برای استفاده از یک کتابخانه واسط خاص، بازنویسی کند، یا به‌طور دستی نشانگرهای خاصی را به کد ورودی کاربر، به یک پرس‌وجوی ایجادشده، که به‌طور پویا اضافه شده است، وارد کند.

1- Static Analysis
2- Dynamic Analysis

۴-۲-۴- تجزیه و تحلیل ترکیبی[1]

روش‌های تجزیه و تحلیل ترکیبی، ترکیبی از هر دو تجزیه و تحلیل ایستا و پویا را، برای آنالیز اطلاعات به‌دست‌آمده در طول اجرای برنامه به‌کار می‌گیرد. برخی روش‌ها، تجزیه و تحلیل ترکیبی را به‌منظور کاهش سربار عملکرد و افزایش بهره‌وری برای تشخیص آسیب‌پذیری، استفاده کرده‌اند. AMNESIA ترکیبی از تجزیه و تحلیل ایستا و پویا را، برای مقابله در برابر حمله تزریق SQL استفاده می‌کند. در بخش تجزیه و تحلیل ایستا، کد برنامه کاربردی را تجزیه و تحلیل می‌کند و به‌طور خودکار، مدلی از پرس‌وجوهای قانونی را تولید می‌کند. درحالی‌که در بخش تجزیه و تحلیل پویا، تمام پرس‌وجوهای تولیدشده را به‌طور پویا، در زمان اجرا نظارت می‌کند و با پرس‌وجوهای قانونی ساخته‌شده با مدل ایستا مقایسه می‌کند.

۴-۲-۵- آزمون جعبه سیاه[2]

آزمون جعبه سیاه، یک آزمونی است که روش‌هایی برای تشخیص آسیب‌پذیری‌ها، به‌وسیله آزمون برنامه کاربردی، بر اساس مشخصات مورد نیاز طراحی می‌کند. مشخصات مورد نیاز بدین معناست که ورودی‌های در دسترس و خروجی‌های مورد انتظار چه هستند که باید از هر ورودی حاصل شوند و به کد منبع برنامه کاربردی مربوط نیست. برای مثال روش‌های اسکن خودکار، به این دسته متعلق‌اند مانند WAVES، SecuBat. روش آزمون جعبه سیاه در طول زمان آزمون برنامه کاربردی استفاده می‌شود. مثال دیگر از آزمون جعبه سیاه، viper است که نویسندگان، آن را به عنوان یک ابزار آزمون نفوذ خودکار، برای آزمون برنامه کاربردی وب توسعه داده‌اند. آن بر اساس الگوی تطبیق پیام های خطا و صفحات خروجی معتبر متکی است. بنابراین، این بستگی به دانش گسترده‌ای از اکتشافات برای هدایت تولید پرس‌وجوهای SQL است. این ابزار شامل چهار مرحله جمع‌آوری اطلاعات، شناسایی پارامترهای ورودی، ایجاد حملات و نتیجه گزارش است.

۴-۲-۶- آزمون جعبه سفید[3]

روش‌های آزمون جعبه سفید، آزمونی است که روش‌هایی را برای تشخیص آسیب‌پذیری‌ها توسط آزمون برنامه کاربردی با موارد آزمون طراحی می‌کند. موارد آزمون از ساختار داخلی سیستم مانند کد منبع تولید شده‌اند. برای مثال، Ardilla یک ابزار آزمون جعبه سفید است.

۴-۳- طبقه‌بندی بر اساس زمان تشخیص

حملات تزریق SQL و آسیب‌پذیری‌هایشان، می‌توانند در زمان کدنویسی، زمان آزمون یا زمان عملیات تشخیص داده شوند.

۴-۳-۱- زمان کدنویسی

اگر حملات تزریق SQL، در طول زمان کدنویسی، برای چرخه توسعه یک برنامه کاربردی، تشخیص داده شوند، آنگاه یک تشخیص زمان کدنویسی در نظر گرفته می‌شود. تشخیص آسیب‌پذیری‌ها، در این مرحله اولیه، کمک می‌کند تا هزینه‌های ناشی از تشخیص دیرهنگام، کاهش یابد. روش‌های تجزیه و تحلیل ایستا، آسیب‌پذیری‌های حملات تزریق SQL را، در طول زمان کدنویسی، بدون نیاز به کد اجرایی تشخیص می‌دهند [۱۷].

۴-۳-۲- زمان آزمون

اگر حملات تزریق SQL و آسیب‌پذیری‌هایشان، در طول زمان آزمون، از چرخه توسعه یک برنامه کاربردی، تشخیص داده شوند، آنگاه به عنوان یک تشخیص زمان آزمون، در نظر گرفته می‌شود. رویکردهای مختلف آزمون، مانند آزمون جعبه سیاه و آزمون جعبه سفید، می‌توانند به عنوان روش‌های تجزیه و تحلیل، در زمان آزمون، برای تشخیص حملات تزریق SQL و آسیب‌پذیری‌هایشان استفاده شوند.

۴-۳-۳- زمان عملیات (زمان اجرا)

اگر حملات تزریق SQL و آسیب‌پذیری‌هایشان، در طول زمان اجرا[4]، در عرصه دنیای واقعی، پس از عرضه محصول، تشخیص داده شوند، آنگاه به عنوان یک تشخیص زمان عملیات، در نظر گرفته می‌شود. روش‌های دفاع زمان اجرا، معمولا با پایان دادن به اجرای حملات و یا پاکسازی آن‌ها، از حملات تزریق SQL پیشگیری می‌کنند. با این حال، در صورت مثبت کاذب، پایان دادن به اجرا، منجر به ناراحتی‌های قابل توجهی برای کاربران خواهد شد.

۴-۴- طبقه‌بندی بر اساس مکان تشخیص

حملات تزریق SQL و آسیب‌پذیری‌هایشان، در مکان‌های مختلفی از سیستم تشخیص داده می‌شوند. در این مقاله، این مکان‌های مختلف، به چهار دسته طبقه‌بندی شده‌اند: برنامه کاربردی سمت سرویس‌گیرنده، پراکسی سمت سرویس‌گیرنده، برنامه کاربردی سمت سرور، پراکسی سمت سرور.

۴-۴-۱- برنامه کاربردی سمت سرویس‌گیرنده

روش‌های برنامه کاربردی سمت سرویس‌گیرنده، حملات تزریق SQL را با تجزیه و تحلیل صفحات HTML، تشخیص می‌دهند. درحالی‌که، اسکریپت‌های سمت سرویس‌گیرنده نیز با استفاده از روش‌هایی که برای شناسایی حملات تزریق SQL و cross-site scripting (عبور از

1- Hybrid Analysis
2- Black Box Testing
3- White Box Testing
4- Run Time

اسکریپت نویسی سایت) استفاده می‌شوند، مورد تجزیه و تحلیل قرار می‌گیرند.

۴-۴-۲- پراکسی سمت سرویس‌گیرنده

پراکسی سمت سرویس‌گیرنده، به عنوان یک دروازه یا سرور واسط، بین کاربر و یک سرور وب عمل می‌کند. آن، درخواست‌ها و پاسخ‌های کاربر را از سرور وب، به‌منظور شناسایی حملات تزریق SQL متوقف می‌کند. پس از شناسایی ورودی‌های مخرب، یا درخواست را رد می‌کند یا ورودی‌های مخرب را به ورودی‌های بی‌خطر تغییر می‌دهد.

۴-۴-۳- برنامه کاربردی سمت سرور

روش برنامه کاربردی سمت سرور، با تجزیه و تحلیل برنامه کاربردی سمت سرور نوشته‌شده در زبان های برنامه‌نویسی و اسکریپتی، حملات تزریق SQL را تشخیص می‌دهند.

۴-۴-۴- پراکسی سمت سرور

پراکسی سمت سرور، به عنوان سرور تکمیلی، بین یک سرور برنامه کاربردی و یک سرور پایگاه داده عمل می‌کند. آن، پرس‌وجوهای SQL، از یک برنامه کاربردی را قبل از رسیدن به سرور پایگاه داده متوقف می‌کند. این، در مسدود کردن اجرای پرس‌وجوی مخرب، در پایگاه داده کمک می‌کند.

۵- طبقه‌بندی روش‌های دفاع در برابر حملات تزریق SQL

در این مقاله روش‌های تشخیص حملات تزریق SQL، از ابعاد مختلف مانند ماهیت دفاع، روش تجزیه و تحلیل، زمان تشخیص و مکان تشخیص بررسی شدند. در این بخش، به بررسی روش‌های مختلف استفاده‌شده به‌منظور دفاع از حملات تزریق SQL، پرداخته می‌شود. این بررسی، بر اساس طبقه‌بندی روش‌های تشخیص حملات تزریق SQL ذکرشده در مقاله، به همراه ارزیابی از ویژگی‌های مثبت کاذب و منفی کاذب صورت گرفته است. جدول (۱)، طبقه‌بندی روش‌های مختلف دفاع در برابر حملات تزریق SQL، بر اساس طبقه‌بندی روش‌های تشخیص ذکرشده در مقاله را نشان می‌دهد. در طبقه‌بندی روش‌ها، علاوه بر روش‌های تشخیص، ویژگی مثبت و منفی کاذب نیز در جدول (۱)، ذکر شده است. دقت یک روش را می‌توان از نظر تولید مثبت کاذب و منفی کاذب، اندازه‌گیری کرد. مثبت کاذب[1]، زمانی اتفاق می‌افتد که روش، اجرایی از ورودی غیرمخرب را مسدود کند. منفی کاذب[2]، زمانی اتفاق می‌افتد که روش، به ورودی واقعا مخرب، اجازه دسترسی به پایگاه داده را می‌دهد. در جدول (۱)، مثبت کاذب با FP و منفی کاذب با FN نشان داده شده است. در این پژوهش، بسیاری از مقاله‌ها، جزئیات دقیقی در مورد مثبت کاذب و منفی کاذب، را ذکر نکرده‌اند. بر اساس سطح جزئیات ارائه‌شده توسط نویسندگان، در مورد مثبت کاذب و منفی کاذب، این مقاله آن‌ها را به سه گروه تقسیم کرده است: واقعی، صریح و ضمنی. که در زیر به تعریف هریک می‌پردازیم:

- واقعی[3]: مقالاتی که مثبت کاذب و منفی کاذب را نشان می‌دهند و با ارزیابی، نتایجی از مقادیر واقعی، مانند اعداد یا درصد را ارائه می‌دهند، در این گروه قرار می‌گیرند.

- صریح[4]: مقالاتی که به صراحت، در مورد مثبت کاذب و منفی کاذب، بدون هیچ نتیجه ارزیابی، بحث می‌کنند، در این گروه قرار می‌گیرند.

- ضمنی[5]: مقالاتی که در مورد مثبت کاذب و منفی کاذب، هیچ بحثی نکرده‌اند، در این گروه قرار می‌گیرند، اگرچه می‌توان نتایج مثبت کاذب و منفی کاذب، را از اطلاعاتی که آن‌ها همراه با روش ارائه داده‌اند نتیجه گرفت.

در جدول (۱)، تنها ۲۴ روش، دقت را با نتایج آماری واقعی، از لحاظ مثبت کاذب، و ۱۵ روش، از نظر منفی کاذب شرح داده‌اند درحالی‌که، بقیه مقالات، نتایج آماری را نشان نداده‌اند. این طبقه‌بندی، به سازماندهی، برای درک بهتر روش‌های مختلف کمک می‌کند. از این رو، بر اساس این تفاوت‌ها، سازمان‌ها می‌توانند روش‌های مناسب بسته به منابع و محیط‌های موجود خود را انتخاب کنند. بر اساس جدول (۱)، اکثریت روش‌ها، از تجزیه و تحلیل پویا به دلیل مزایایی چون شناسایی آسیب‌پذیری‌ها و حملات در زمان اجرا، شناسایی حملات بدون نیاز به تغییر برنامه تحت وب، بهتر و دقیق عمل کردن به دلیل مطلع بودن از فرآیند اجرای برنامه کاربردی استفاده کرده‌اند. درحالی‌که، آزمون جعبه سفید، کم‌ترین روش تجزیه و تحلیل استفاده شده می‌باشد. تجزیه و تحلیل ترکیبی، دومین روش تجزیه و تحلیل است که به دلیل برخورداری از مزایای هر دو روش تجزیه و تحلیل ایستا و پویا، بیشتر استفاده شده است. البته باید توجه داشت که در تجزیه و تحلیل ترکیبی، موفقیت کار وابسته به کامل بودن مدل پرس‌وجوهای ایستا در فاز تجزیه و تحلیل ایستا

1- False Positive
2- False Negative
3- Concrete
4- Explicit
5- Implicit

می‌باشد. هر اشکالی در فاز ایستا، باعث تولید مثبت کاذب و منفی کاذب می‌شود.

از معایب دیگر آن، افزایش پیچیدگی زمانی و محاسباتی می‌باشد. همچنین با توجه به جدول (۱)، تجزیه و تحلیل ایستا و برنامه‌نویسی امن، تقریبا به نسبت برابر در روش‌های دفاع مورد استفاده قرار گرفته‌اند.

جدول ۱- طبقه‌بندی روش‌های مختلف دفاع در برابر حملات تزریق SQL

روش	ماهیت دفاع	روش تجزیه و تحلیل	زمان تشخیص	مکان تشخیص	ارزیابی مثبت و منفی کاذب
Security Gateway[18]	اقدام متقابل	پویا	زمان اجرا	پراکسی سمت سرور	FP: ضمنی FN: ضمنی
WAVES[19]	تشخیص	آزمون جعبه سیاه + پویا	زمان آزمون	برنامه کاربردی سمت سرویس گیرنده	FP: ضمنی FN: واقعی
AUSELSQL[20]	اقدام متقابل	پویا	زمان اجرا	پراکسی سمت سرویس گیرنده	FP: صریح FN: ضمنی
SQLrand[21]	اقدام متقابل	پویا	زمان اجرا	پراکسی سمت سرور	FP: ضمنی FN: ضمنی
JDBC Checker[15]	تشخیص	ایستا	زمان کد نویسی	برنامه کاربردی سمت سرور	FP: صریح FN: صریح
WebSSARI[22]	پیشگیری	ترکیبی از ایستا و پویا	زمان اجرا	برنامه کاربردی سمت سرور	FP: واقعی FN: ضمنی
Tautology checker[23]	پیشگیری	ایستا	زمان اجرا	برنامه کاربردی سمت سرور	FP: ضمنی FN: ضمنی
SQLGuard[24]	اقدام متقابل	پویا	زمان اجرا	برنامه کاربردی سمت سرور	FP: ضمنی FN: ضمنی
Variable Normalization[25]	اقدام متقابل	پویا	زمان اجرا	پراکسی سمت سرور	FP: ضمنی FN: ضمنی
Java Dynamic Tainting[26]	اقدام متقابل	پویا	زمان اجرا	برنامه کاربردی سمت سرور	FP: ضمنی FN: ضمنی
AMNESIA[27]	اقدام متقابل	ترکیبی از ایستا و پویا	زمان اجرا	برنامه کاربردی سمت سرور	FP: واقعی FN: واقعی
Java Static Tainting[16]	پیشگیری	ایستا	زمان اجرا	برنامه کاربردی سمت سرور	FP: واقعی FN: واقعی
SecuriFly[28]	اقدام متقابل	پویا	زمان اجرا	برنامه کاربردی سمت سرور	FP: صریح FN: صریح
SQL DOM[29]	اقدام متقابل	برنامه نویسی امن	زمان کامپایل	برنامه کاربردی سمت سرور	FP: ضمنی FN: ضمنی
Anomaly Detection[30]	اقدام متقابل	پویا	زمان اجرا	برنامه کاربردی سمت سرور	FP: واقعی FN: ضمنی
Pixy[31]	پیشگیری	ایستا	زمان کد نویسی	برنامه کاربردی سمت سرور	FP: ضمنی FN: ضمنی
SecBat[32]	تشخیص	آزمون جعبه سیاه	زمان آزمون	برنامه کاربردی سمت سرویس گیرنده	FP: صریح FN: صریح
AntiMaliciousInjection[33]	اقدام متقابل	برنامه نویسی امن	زمان کد نویسی	برنامه کاربردی سمت سرور	FP: ضمنی FN: ضمنی
ESIATDM[34]	اقدام متقابل	ترکیبی از ایستا و پویا	زمان اجرا	برنامه کاربردی سمت سرور	FP: واقعی FN: واقعی

CSSE[35]	اقدام متقابل	پویا	زمان اجرا	برنامه کاربردی سمت سرور	صریح :FP صریح:FN
SQLCHECK[36]	اقدام متقابل	پویا	زمان اجرا	پراکسی سمت سرور	واقعی :FP واقعی:FN
Stored Procedure[37]	اقدام متقابل	ترکیبی از ایستا و پویا	زمان اجرا	پراکسی سمت سرور	صریح :FP ضمنی:FN
Swaddler[38]	تشخیص	پویا	زمان اجرا	برنامه کاربردی سمت سرور	واقعی :FP ضمنی:FN
DMSQL[39]	اقدام متقابل	پویا	زمان اجرا	پراکسی سمت سرور	واقعی :FP ضمنی:FN
AProSec[40]	تشخیص	پویا	زمان کامپایل	پراکسی سمت سرویس گیرنده + پراکسی سمت سرور	ضمنی :FP ضمنی:FN
Smask[41]	اقدام متقابل	پویا	زمان اجرا	برنامه کاربردی سمت سرور	صریح :FP صریح:FN
Sania[42]	تشخیص	پویا	زمان آزمون	پراکسی سمت سرویس گیرنده + پراکسی سمت سرور	واقعی :FP ضمنی:FN
APPASIA[43]	اقدام متقابل	پویا	زمان اجرا	برنامه کاربردی سمت سرور	واقعی :FP واقعی:FN
AFGSSS[44]	پیشگیری	برنامه نویسی امن	زمان اجرا	برنامه کاربردی سمت سرور	صریح :FP صریح:FN
Secure PHP[45]	پیشگیری	برنامه نویسی امن	زمان اجرا	برنامه کاربردی سمت سرور	صریح :FP صریح:FN
SAFELI[46]	اقدام متقابل	ترکیبی از ایستا و پویا	زمان کامپایل	برنامه کاربردی سمت سرور	ضمنی :FP ضمنی:FN
WASP[47]	اقدام متقابل	پویا	زمان اجرا	برنامه کاربردی سمت سرور	واقعی :FP واقعی:FN
SQL-IDS[48]	اقدام متقابل	پویا	زمان اجرا	پراکسی سمت سرور	واقعی :FP واقعی:FN
ABCDM[49]	اقدام متقابل	پویا	زمان اجرا	پراکسی سمت سرویس گیرنده	واقعی :FP واقعی:FN
AMSMI[50]	اقدام متقابل	پویا	زمان اجرا	پراکسی سمت سرور	واقعی :FP واقعی:FN
Sdriver[51]	اقدام متقابل	پویا	زمان اجرا	پراکسی سمت سرور	صریح :FP صریح:FN
MUSIC[52]	تشخیص	آزمون جعبه سفید + پویا	زمان آزمون	برنامه کاربردی سمت سرور	ضمنی :FP ضمنی:FN
CVSID[53]	تشخیص	ایستا	زمان آزمون	برنامه کاربردی سمت سرور	ضمنی :FP ضمنی:FN
HoneyPot[54]	انحراف	پویا	زمان اجرا	برنامه کاربردی سمت سرور	ضمنی :FP ضمنی:FN
CAPSIA[55]	اقدام متقابل	پویا	زمان اجرا	برنامه کاربردی سمت سرور	ضمنی :FP ضمنی:FN
WBSCA[56]	اقدام متقابل	پویا	زمان اجرا	برنامه کاربردی سمت سرور	واقعی :FP صریح:FN
SQLInjectionGen[57]	تشخیص	ترکیبی از ایستا و پویا	زمان آزمون	برنامه کاربردی سمت سرور	واقعی :FP واقعی:FN
Ardilla[58]	تشخیص	آزمون جعبه سفید	زمان آزمون	برنامه کاربردی سمت سرور	واقعی :FP واقعی:FN

SQLProb[59]	اقدام متقابل	پویا	زمان اجرا	پراکسی سمت سرور	واقعی :FP واقعی:FN
ADMIRE[60]	پیشگیری	ایستا	زمان کد نویسی	برنامه کاربردی سمت سرور	ضمنی :FP ضمنی:FN
Phan[61]	اقدام متقابل	ترکیبی از ایستا و پویا	زمان اجرا	برنامه کاربردی سمت سرور	ضمنی :FP ضمنی:FN
MDWAA[62]	پیشگیری	پویا	زمان آزمون	برنامه کاربردی سمت سرور	واقعی :FP واقعی:FN
SBSQLID[63]	اقدام متقابل	پویا	زمان اجرا	برنامه کاربردی سمت سرور	ضمنی :FP ضمنی:FN
Evolutionary Approach[64]	اقدام متقابل	پویا	زمان اجرا	برنامه کاربردی سمت سرور	واقعی :FP واقعی:FN
Prepared Statement[65]	تشخیص	برنامه نویسی امن	زمان اجرا	برنامه کاربردی سمت سرور	ضمنی :FP ضمنی:FN
IAAT[66]	تشخیص	آزمون جعبه سیاه + پویا	زمان آزمون	برنامه کاربردی سمت سرور	واقعی :FP ضمنی:FN
CANDID[67]	اقدام متقابل	پویا	زمان اجرا	برنامه کاربردی سمت سرور	واقعی :FP ضمنی:FN
TAPS[68]	اقدام متقابل	پویا	زمان اجرا	برنامه کاربردی سمت سرور	ضمنی :FP ضمنی:FN
Syntax Embeddings[69]	اقدام متقابل	برنامه نویسی امن	زمان کد نویسی	برنامه کاربردی سمت سرور	صریح :FP صریح:FN
Viper[70]	تشخیص	آزمون جعبه سیاه	زمان آزمون	برنامه کاربردی سمت سرویس گیرنده	صریح :FP صریح:FN
DUD[71]	اقدام متقابل	پویا	زمان اجرا	برنامه کاربردی سمت سرور	صریح :FP صریح:FN
ACMWDPC[72]	اقدام متقابل	پویا	زمان اجرا	برنامه کاربردی سمت سرور	ضمنی :FP ضمنی:FN
Query Parser[73]	اقدام متقابل	پویا	زمان اجرا	برنامه کاربردی سمت سرور	ضمنی :FP ضمنی:FN
ANNWAF[74]	اقدام متقابل	پویا	زمان اجرا	پراکسی سمت سرویس گیرنده	صریح :FP ضمنی:FN
D-WAV[75]	پیشگیری	پویا	زمان آزمون	برنامه کاربردی سمت سرویس گیرنده	صریح :FP ضمنی:FN
X LOG AUTHENTICATION[76]	اقدام متقابل	پویا	زمان اجرا	برنامه کاربردی سمت سرور	صریح :FP صریح:FN
SIAPBDTC[77]	پیشگیری	ترکیبی از ایستا و پویا	زمان اجرا	برنامه کاربردی سمت سرور	واقعی :FP ضمنی:FN
PTAFMESIV[78]	تشخیص	آزمون جعبه سیاه	زمان آزمون	برنامه کاربردی سمت سرویس گیرنده	واقعی :FP ضمنی:FN
DATDSI[79]	تشخیص	پویا	زمان اجرا	برنامه کاربردی سمت سرور	ضمنی :FP ضمنی:FN
RPHoneypot[80]	انحراف	پویا	زمان اجرا	برنامه کاربردی سمت سرور	ضمنی :FP ضمنی:FN
Dynamic Analyzer[81]	تشخیص	پویا	زمان اجرا	برنامه کاربردی سمت سرور	ضمنی :FP ضمنی:FN

با توجه به بررسی‌های انجام‌شده، معیارهای انتخاب یک روش تشخیص تزریق SQL خوب، باید موارد زیر را شامل شود:

- امکان تشخیص انواع حملات تزریق SQL را داشته باشد.
- دارای پیچیدگی زمانی کم باشد.
- دارای رنج مثبت کاذب و منفی کاذب کمی باشد.
- وابستگی به پایگاه داده خاصی نداشته باشد و بتواند با هر نوع پایگاه داده‌ای کار کند.
- قابلیت انعطاف بالا در زمان توسعه سیستم تشخیص نفوذ برای حملات جدید را داشته باشد.
- امکان شناسایی حملات در هنگام اجرای برنامه را داشته باشد.
- صحت روش تشخیص تزریق SQL، به کد منبع وابستگی کامل نداشته باشد و در صورت تغییر کد منبع، مشکلی در شناسایی حملات و مجوز دادن به ورودی‌های نرمال را نداشته باشد [۸۲].

۶- نتیجه‌گیری

در این مقاله، یک بررسی و طبقه‌بندی از روش‌های فعلی، برای تشخیص و پیشگیری از حملات تزریق SQL، ارائه شد. برای انجام این بررسی، ابتدا یک طبقه‌بندی از روش‌های تشخیص حملات تزریق SQL ارائه شد و سپس ۶۶ روش دفاع مختلف، در برابر حملات تزریق SQL، مورد بررسی قرار گرفت. این طبقه‌بندی، شامل ابعاد مختلفی مانند ماهیت دفاع، روش تجزیه و تحلیل، زمان تشخیص و مکان تشخیص می‌باشد و نیز برای هر روش، به بیان مثبت کاذب و منفی کاذب، پرداخته شد. این طبقه‌بندی، به سازمان‌ها، برای درک بهتر روش‌های مختلف دفاع کمک می‌کند. از این رو، بر اساس این تفاوت‌ها، سازمان‌ها می‌توانند بسته به منابع و محیط‌های موجود خود روش‌های مناسب را انتخاب کنند. به عنوان کار آینده، باید به دنبال روش‌هایی باشیم که انواع جدیدی از حملات تزریق SQL را تشخیص دهد و نیز دارای محدوده مثبت و منفی کاذب کمی باشد.

۷- مراجع

1. OWASP Top 10 Application Security Risks – 2017. Available: https://www.owasp.org/index.php/Top_10_2017-Top_10
2. K. V. P. R. Sheth, “Survey on Prevention of Web Injection using WAF and Input Whitelisting,” 2017.
3. Z. Su and G. Wassermann, “The Essence of Command Injection Attacks in Web Applications,” in In The 33rd Annual Symposium on Principles of Programming Languages, 2006.
4. W. G. Halfond, J. Viegas, and A. Orso, “A classification of SQL-injection attacks and countermeasures,” in Proceedings of the IEEE International Symposium on Secure Software Engineering, vol. 1, pp. 13-15, IEEE, 2006.
5. X. Fu, X. Lu, B. Peltsverger, S. Chen, K. Qian, and L. Tao, “A static analysis framework for detecting SQL injection vulnerabilities,” in Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International, vol. 1, pp. 87-96: IEEE, 2007.
6. H. K. Kim, “Frameworks for SQL Retrieval on Web Application Security,” in Proceedings of the International MultiConference of Engineers and Computer Scientists, vol. 1, pp. 1781-2006, 2010.
7. I. Lee, S. Jeong, S. Yeo, and J. Moon, “A novel method for SQL injection attack detection based on removing SQL query attribute values,” Mathematical and Computer Modelling, vol. 55, no. 1, pp. 58-68, 2012.
8. W. G. Halfond and A. Orso, “Preventing SQL injection attacks using AMNESIA,” in Proceedings of the 28th international conference on Software engineering, ACM, pp. 795-798, 2006.
9. J. Wang, R. C.-W. Phan, J. N. Whitley, and D. J. Parish, “Augmented attack tree modeling of SQL injection attacks,” in Information Management and Engineering (ICIME), 2010 The 2nd IEEE International Conference on, IEEE, pp. 182-186, 2010.
10. A. Yeole and B. Meshram, “Analysis of different technique for detection of SQL injection,” in Proceedings of the International Conference & Workshop on Emerging Trends in Technology, ACM, pp. 963-966, 2011.
11. Y. V. N. Manikanta and A. Sardana, “Protecting web applications from SQL injection attacks by using framework and database firewall,” in Proceedings of the International Conference on Advances in Computing, Communications and Informatics, ACM, pp. 609-613, 2012.
12. A. Tajpour, M. Z. Heydari, M. Masrom, and S. Ibrahim, “SQL injection detection and prevention tools assessment,” in Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, vol. 9, IEEE, pp. 518-522, 2010.
13. J. Clarke-Salt, “SQL injection attacks and defense,” Elsevier, 2009.
14. D. Fraunholz, M. Zimmermann, and H. D. Schotten, “An adaptive honeypot configuration, deployment and maintenance strategy,” in Advanced Communication Technology (ICACT), 2017 19th International Conference on, pp. 53-57: IEEE, 2017.
15. C. Gould, Z. Su, and P. Devanbu, “JDBC checker: A static analysis tool for SQL/JDBC applications,” in Proceedings of the 26th International Conference on Software Engineering, pp. 697-698, IEEE Computer Society, 2004.
16. V. B. Livshits and M. S. Lam, “Finding Security Vulnerabilities in Java Applications with Static Analysis,” in USENIX Security Symposium, vol. 14, pp. 8-18, 2005.
17. Y. Shin and L. A. Williams, “Towards a taxonomy of techniques to detect cross-site scripting and SQL injection vulnerabilities,” North Carolina State University Dept. of Computer Science, 2008.
18. D. Scott and R. Sharp, “Abstracting application-level web security,” in Proceedings of the 11th international conference on World Wide Web, pp. 396-407, ACM, 2002.

19. Y.-W. Huang, S.-K. Huang, T.-P. Lin, and C.-H. Tsai, "Web application security assessment by fault injection and behavior monitoring," in Proceedings of the 12th international conference on World Wide Web, pp. 148-159, ACM, 2003.

20. A. A. Alfantookh, "An automated universal server level solution for SQL injection security flaw," in Proceedings of the 2004 International Conference on Electrical, Electronic and Computer Engineering (ICEEC'04), pp. 131-135, 2004.

21. S. Boyd and A. Keromytis, "SQLrand: Preventing SQL injection attacks," in Applied Cryptography and Network Security, pp. 292-302, Springer, 2004.

22. Y.-W. Huang, F. Yu, C. Hang, C.-H. Tsai, D.-T. Lee, and S.-Y. Kuo, "Securing web application code by static analysis and runtime protection," in Proceedings of the 13th international conference on World Wide Web, pp. 40-52, ACM, 2004.

23. G. Wassermann and Z. Su, "An analysis framework for security in web applications," in Proceedings of the FSE Workshop on Specification and Verification of component-Based Systems (SAVCBS 2004), pp. 70-78, 2004.

24. G. Buehrer, B. W. Weide, and P. A. Sivilotti, "Using parse tree validation to prevent SQL injection attacks," in Proceedings of the 5th international workshop on Software engineering and middleware, pp. 106-113, ACM, 2005.

25. M. Sam and N. SQLBlock, "SQL Injection Protection by Variable Normalization of SQL Statement," Online http://www. sqlblock. com/sqlblock. pdf, 2005.

26. V. Haldar, D. Chandra, and M. Franz, "Dynamic taint propagation for Java," in Computer Security Applications Conference, 21st Annual, pp. 9-311, IEEE, 2005.

27. W. G. Halfond and A. Orso, "AMNESIA: analysis and monitoring for NEutralizing SQL-injection attacks," in Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering, pp. 174-183, ACM, 2005.

28. M. Martin, B. Livshits, and M. S. Lam, "Finding application errors and security flaws using PQL: a program query language," in ACM SIGPLAN Notices, vol. 40, no. 10, pp. 365-383, ACM, 2005.

29. I. Kruger and R. McClure, "SQL DOM: compile time checking of dynamic SQL statements," in 27th International Conference on Software Engineering, pp. 88-96: IEEE.

30. F. Valeur, D. Mutz, and G. Vigna, "A learning-based approach to the detection of SQL attacks," Detection of intrusions and malware, and vulnerability assessment, pp. 533-546, 2005.

31. N. Jovanovic, C. Kruegel, and E. Kirda, "Pixy: A static analysis tool for detecting web application vulnerabilities," in Security and Privacy, 2006 IEEE Symposium on, , pp. 6 -263, IEEE, 2006.

32. S. Kals, E. Kirda, C. Kruegel, and N. Jovanovic, "Secubat: a web vulnerability scanner," in Proceedings of the 15th international conference on World Wide Web, pp. 247-256, ACM, 2006.

33. J.-C. Lin and J.-M. Chen, "An automatic revised tool for anti-malicious injection," in Computer and Information Technology, 2006. CIT'06. The Sixth IEEE International Conference on, pp. 164-164: IEEE, 2006.

34. M. Muthuprasanna, K. Wei, and S. Kothari, "Eliminating SQL injection attacks-A transparent defense mechanism," in Web Site Evolution, 2006. WSE'06. Eighth IEEE International Symposium on, pp. 22-32: IEEE, 2006.

35. T. Pietraszek and C. V. Berghe, "Defending against injection attacks through context-sensitive string evaluation," in RAID, vol. 3858, pp. 124-145, Springer, 2005.

36. Z. Su and G. Wassermann, "The essence of command injection attacks in web applications," in ACM SIGPLAN Notices, vol. 41, no. 1, pp. 372-382: ACM, 2006.

37. K. Wei, M. Muthuprasanna, and S. Kothari, "Preventing SQL injection attacks in stored procedures," in Software Engineering Conference, Australian, pp. 8-198, IEEE, 2006.

38. M. Cova, D. Balzarotti, V. Felmetsger, and G. Vigna, "Swaddler: An approach for the anomaly-based detection of state violations in web applications," in Recent Advances in Intrusion Detection, pp. 63-86, Springer, 2007.

39. J. Fonseca, M. Vieira, and H. Madeira, "Detecting malicious SQL," Trust, Privacy and Security in Digital Business, pp. 259-268, 2007.

40. G. Hermosillo, R. Gomez, L .Seinturier, and L. Duchien, "Using aspect programming to secure web applications," Journal of Software, vol. 6, no. 2, pp. 53-63, 2007.

41. M. Johns and C. Beyerlein, "SMask: preventing injection attacks in web applications by approximating automatic data/code separation," in Proceedings of the 2007 ACM symposium on Applied computing, pp. 284-291: ACM, 2007.

42. Y. Kosuga, K. Kono, M. Hanaoka, M. Hishiyama, and Y. Takahama, "Sania: Syntactic and semantic analysis for automated testing against sql injection," in Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual, pp. 107-117, IEEE, 2007.

43. E. Merlo, D. Letarte, and G. Antoniol, "Automated protection of php applications against SQL-injection attacks," in Software Maintenance and Reengineering, 2007. CSMR'07. 11th European Conference on, pp. 191-202, IEEE, 2007.

44. S. Thomas and L. Williams, "Using automated fix generation to secure SQL statements," in Proceedings of the Third International Workshop on Software Engineering for Secure Systems, p. 9, IEEE Computer Society, 2007.

45. F. Dysart and M. Sherriff, "Automated fix generator for sql injection attacks," in Software Reliability Engineering, 2008. ISSRE 2008. 19th International Symposium on, pp. 311-312, IEEE, 2008.

46. X. Fu and K. Qian, "SAFELI: SQL injection scanner using symbolic execution," in Proceedings of the 2008 workshop on Testing, analysis, and verification of web services and applications, pp. 34-39, ACM, 2008.

47. W. Halfond, A. Orso, and P. Manolios, "WASP: Protecting web applications using positive tainting and syntax-aware evaluation," IEEE Transactions on Software Engineering, vol. 34, no. 1, pp. 65-81, 2008.

48. K. Kemalis and T. Tzouramanis, "SQL-IDS: a specification-based approach for SQL-injection detection," in Proceedings of the 2008 ACM symposium on Applied computing, pp. 2153-2158, ACM, 2008.

49. M. Kiani, A. Clark, and G. Mohay, "Evaluation of anomaly based character distribution models in the detection of SQL injection attacks," in Availability, Reliability and Security, 2008. ARES 08. Third International Conference on, pp. 47-55, IEEE, 2008.

50. J.-C. Lin, J.-M. Chen, and C.-H. Liu, "An automatic mechanism for sanitizing malicious injection," in Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for, pp. 1470-1475, IEEE, 2008.

51. D. Mitropoulos and D. Spinellis, "SDriver: Location-specific signatures prevent SQL injection attacks," computers & security, vol. 28, no. 3, pp. 121-129, 2009.

52. H. Shahriar and M. Zulkernine, "MUSIC: Mutation-based SQL injection vulnerability checking," in Quality Software, 2008. QSIC'08. The Eighth International Conference on, pp. 77-86, IEEE, 2008.

53. Z. Zhang, Q. Zheng, X. Guan, Q. Wang, and T. Wang, "A method for detecting code security vulnerability based on variables tracking with validated-tree," Frontiers of Electrical and Electronic Engineering in China, vol. 3, no. 2, pp. 162-166, 2008.

54. T. M. Chen and J. Buford, "Design considerations for a honeypot for SQL injection Attacks," in Local Computer Networks, 2009. LCN 2009. IEEE 34th Conference on, pp. 915-921, IEEE, 2009.

55. R. Ezumalai and G. Aghila, "Combinatorial approach for preventing SQL injection attacks," in Advance Computing Conference, 2009. IACC 2009 .IEEE International, pp. 1212-1217, IEEE, 2009.

56. M. Ficco, L. Coppolino, and L. Romano, "A weight-based symptom correlation approach to SQL injection attacks," in Dependable Computing, 2009. LADC'09. Fourth Latin-American Symposium on, pp. 9-16, IEEE, 2009.

57. M. Junjin, "An approach for SQL injection vulnerability detection," in Information Technology: New Generations, 2009. ITNG'09. Sixth International Conference on, pp. 1411-1414, IEEE, 2009.

58. A. Kieyzun, P. J. Guo, K. Jayaraman, and M. D. Ernst, "Automatic creation of SQL injection and cross-site scripting attacks," in Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference on, pp. 199-209, IEEE, 2009.

59. A. Liu, Y. Yuan, D. Wijesekera, and A. Stavrou, "SQLProb: a proxy-based architecture towards preventing SQL injection attacks," in Proceedings of the 2009 ACM symposium on Applied Computing, pp. 2054-2061, ACM, 2009.

60. S. Madan and S. Madan, "Shielding against sql injection attacks using admire model," in Computational Intelligence, Communication Systems and Networks, 2009. CICSYN'09. First International Conference on, pp. 314-320, IEEE, 2009.

61. M. Monga, R. Paleari, and E. Passerini, "A hybrid analysis framework for detecting web application vulnerabilities," in Proceedings of the 2009 ICSE Workshop on Software Engineering for Secure Systems, pp. 25-32, IEEE Computer Society, 2009.

62. A. Razzaq, A. Hur, N. Haider, and F. Ahmad, "Multi-layered defense against web application attacks," in Information Technology: New Generations, 2009. ITNG'09. Sixth International Conference on, pp. 492-497, IEEE, 2009.

63. S. V. Shanmughaneethi, S. C. E. Shyni, and S. Swamynathan, "SBSQLID: Securing web applications with service based SQL injection detection," in Advances in Computing, Control, & Telecommunication Technologies, 2009. ACT'09. International Conference on, pp. 702-704, IEEE, 2009.

64. J. Skaruz and F. Seredynski, "Intrusion detection in web applications: evolutionary approach," in Computer Science and Information Technology, 2009. IMCSIT'09. International Multiconference on, pp. 117-123, IEEE, 2009.

65. S. Thomas, L. Williams, and T. Xie, "On automated prepared statement generation to remove SQL injection vulnerabilities," Information and Software Technology, vol. 51, no. 3, pp. 589-598, 2009.

66. A. Anchlia and S. Jain, "A Novel Injection Aware Approach for the Testing of Database Applications," in Recent Trends in Information, Telecommunication and Computing (ITC), 2010 International Conference on, pp. 311-313, IEEE, 2010.

67. P. Bisht, P. Madhusudan, and V. Venkatakrishnan, "CANDID: Dynamic candidate evaluations for automatic prevention of SQL injection attacks," ACM Transactions on Information and System Security (TISSEC), vol. 13, no. 2, p. 14, 2010.

68. P. Bisht, A. P. Sistla, and V. Venkatakrishnan, "Taps: automatically preparing safe sql queries," in Proceedings of the 17th ACM conference on Computer and communications security, pp. 645-647, ACM, 2010.

69. M. Bravenboer, E. Dolstra, and E. Visser, "Preventing injection attacks with syntax embeddings," Science of Computer Programming, vol. 75, no. 7, pp. 473-495, 2010.

70. A. Ciampa, C. A. Visaggio, and M. Di Penta, "A heuristic-based approach for detecting SQL-injection vulnerabilities in Web applications," in Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems, pp. 43-49, ACM, 2010.

71. D. Das, U. Sharma, and D. Bhattacharyya, "An approach to detection of SQL injection attack based on dynamic query matching," International Journal of Computer Applications, vol. 1, no. 25, pp. 28-34, 2010.

72. Z. Jan, M. Shah, A. Rauf, M. A. Khan, and S. Mahfooz, "Access Control Mechanism For Web Databases By Using Parameterized Cursor," in Future Information Technology (FutureTech) ,5th International Conference on, pp. 1-6, IEEE, 2010.

73. L. Ntagwabira and S. L. Kang, "Use of Query Tokenization to detect and prevent SQL Injection Attacks," in Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, vol. 2, pp. 438-440, IEEE, 2010.

74. A. Moosa, "Artificial neural network based web application firewall for sql injection," World Academy of Science, Engineering and Technology, vol. 40, pp. 12-21, 2010.

75. L. Zhang, Q. Gu, S. Peng, X. Chen, H. Zhao, and D. Chen, "D-WAV: A web application vulnerabilities detection tool using Characteristics of Web Forms," in Software Engineering Advances (ICSEA), 2010 Fifth International Conference on, pp. 501-507, IEEE, 2010.

76. B. Indrani and E. Ramaraj, "X–Log Authentication Technique to Prevent SQL Injection Attacks,"

International Journal of Information Technology and Knowledge Management, vol. 4, no. 1, pp. 323-328, 2011.

77. B. Hanmanthu, B. R. Ram, and P. Niranjan, "SQL Injection Attack prevention based on decision tree classification," in Intelligent Systems and Control (ISCO), 2015 IEEE 9th International Conference on, pp. 1-5, IEEE, 2015.

78. L. Liu et al., "An effective penetration test approach based on feature matrix for exposing SQL Injection Vulnerability," in Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual, vol. 1, pp. 123-132, IEEE, 2016.

79. A. Khalid and M. M. Yousif, "Dynamic Analysis Tool for Detecting SQL Injection," International Journal of Computer Science and Information Security, vol. 14, no. 2, p. 224, 2016.

80. S. Djanali, F. Arunanto, B. A. Pratomo, H. Studiawan, and S. G. Nugraha, "SQL injection detection and prevention system with raspberry Pi honeypot cluster for trapping attacker," in Technology Management and Emerging Technologies (ISTMET), 2014 International Symposium on, pp. 163-166, IEEE, 2014.

81. R. M. Nadeem, R. M. Saleem, R. Bashir, and S. Habib, "Detection and Prevention of SQL Injection Attack by Dynamic Analyzer and Testing Model," International JournalOURNAL of Advanced Computer Science and Applications, vol. 8, no. 8, pp. 209-214, 2017.

۸۲. تجلی پور، ب.، اصغر صفایی، ع.، تحلیل ساختاری و معنایی پرسوجو برای تشخیص حملات تزریقSQL، مجله پدافند الکترونیکی و سایبری، سال دوم، شماره یک، صفحات ۹۷-۸۳، بهار ۱۳۹۳.

A Classification of SQL Injection Attacks and Techniques to Defend These Attacks in the Passive Defense

M. Torabi, A. Shahidinejad*

Abstract

SQL injection attacks are a serious security threat to web applications in cyberspace. SQL injection attacks allow attackers to gain unlimited access to a database that includes applications and potentially sensitive information. Although researchers and practitioners have proposed different methods to solve the SQL injection problem, current approaches either fail to solve the full scope of the problem or have limitations that prevent their use and adoption. Many researchers and practitioners are familiar with only a subset of a wide range of available techniques to defend against SQL injection attacks. This paper provides a classification based on a comprehensive review of current techniques to defend against SQL injection attacks. This classification helps military and government organizations to understand the techniques of defense against SQL injection attacks. Hence, based on this classification, military and government organizations can choose appropriate techniques depending on their resources and environments. To deal with the problem of SQL injection attacks, this study provides a survey on various types of SQL injection attacks that are known today, with examples of how attacks can be made. Various methods are described to diagnose SQL injection vulnerabilities, and also existing detection and prevention techniques against SQL injection attacks are investigated. For each technique, a classification is made about its features, its strengths and weaknesses in dealing with SQL injection attacks.

Key Words: *Web Applications, Database Security, SQL Injection Attacks, Detection, Prevention*

* Islamic Azad University, Qom Branch (a.shahidinejad@gmail.com) - Writer-in-Charge