پیشگیری و افزایش امنیت پایگاه داده در مقابل حملات SQL Injection در سیستم مدیریت پایگاه داده MS-SQL Server

یوسف رحیمی آخوندزاده[۱] ، مهدی رحیمی آخوندزاده[۲]

۱- دانشجو کارشناسی ارشد مهندسی نرم افزار، دانشکده برق و کامپیوتر و فناوری اطلاعات، واحد قزوین، دانشگاه آزاد اسلامی، قزوین، ایران

۲- دانشجو کارشناسی ارشد مهندسی نرم افزار، دانشکده برق و کامپیوتر و فناوری اطلاعات، واحد قزوین، دانشگاه آزاد اسلامی، قزوین، ایران

چکیده

اطلاعات بخش مهم و حیاتی در جوامع بشری و زندگی افراد می‌باشد. با توجه به پیشرفت تکنولوژی و همه‌گیر شدن استفاده از شبکه‌ی اینترنت، اکثر شرکت‌ها، ادارات و سازمان‌ها اطلاعات و خدمات خود را در بستر وب ارائه می‌دهند. همچنین به‌دلیل ارزشمند بودن این اطلاعات، همواره افراد و یا سازمان‌هایی وجود دارند که قصد سوء استفاده از این اطلاعات را دارند. در محیط وب، این داده‌ها اکثراً با استفاده از سیستم مدیریت پایگاه‌داده MySQL و یا MS-SQL Server و در پاره‌ای موارد پایگاه‌داده‌های دیگری نظیر Oracle و غیره، نگهداری و مدیریت می‌شوند. یکی از روش‌های متداول برای دسترسی غیر مجاز به این داده‌ها، استفاده از حملات SQL Injection است که در آن مهاجم با ارسال دستورات SQL و حتی کدهای JavaScript به سمت سرور داده، قصد مختل کردن فعالیت آنرا داشته و بواسطه‌ی این اختلال می‌تواند به اطلاعات پایگاه‌داده و حتی در گاهی مواقع به سرور نیز دسترسی داشته باشد. یکی از وظایف مدیر پایگاه داده، تشخیص و جلوگیری از بروز چنین حملاتی می‌باشد.

واژگان کلیدی: حملات SQL Injection، امنیت پایگاه داده، اسکیوال سرور، امنیت اسکیوال سرور، امنیت پایگاه داده تحت وب

مقدمه

حملات SQL Injection یک نوع از حملات پر کاربرد برروی پایگاه‌داده‌هایی است که در بستر وب فعالیت می‌کنند و به مهاجم امکان اجرای دستورات مخرب را می‌دهد تا بواسطه‌ی آنها یا بتواند به داده‌هایی که برای او قابل مشاهده نیستند دسترسی داشت و یا حتی داده‌ها را تخریب کند. این داده‌ها می‌توانند اطلاعات مشتریان، داده های حساس نظامی، اطلاعات حکومتی و هر نوع اطلاعات دیگری باشند.

انواع مختلفی از آسیب‌پذیری‌ها و تهدیدات یا حملات وجود دارد که می‌توان آن‌ها را برروی یک وب‌سایت یا برنامه‌های مبتنی بر وب انجام داد، مانند حملات XSS, SMTP Headers, LDAP, XML, SQL Injection و حملات Broken Authentication و بسیاری از آسیب‌پذیری‌ها و حملات دیگر که رایج‌ترین و پر کاربردترین آنها حملات SQL Injection می‌باشد (Benfano Soewito, et al, 2018).

متأسفانه، ثابت شده است که این نوع نفوذ یک مشکل چالش برانگیز با راه‌حل‌های بسیاری است که از معایبی مانند ناتوانی در پردازش داده‌ها در زمان واقعی به‌عنوان یک راه‌حل پیشگیرانه، عدم سازگاری با انواع مختلف حملات، و نیاز به دسترسی به اطلاعات مربوط به برنامه منبع، رنج می‌برند (Kashif Kifayat, et al, 2018).

SQL Injection یا به اختصار SQLI و برنامه نویسی متقابل سایت (Cross Site Script) یا به اختصار XSS دو آسیب‌پذیری رایج و جدی برنامه‌های تحت وب در دهه‌های گذشته هستند. با توجه به پروژه‌ی امنیتی وب باز (Open Web Application Security Project) که به اختصار OWASP خوانده می‌شود، SQL Injection و XSS دو آسیب‌پذیری رایج و حیاتی برنامه‌های وب هستند (Lwin Khin Shar & Hee Beng Kuan Tan, 2013).

SQL Injection تکنیکی برای سوء استفاده‌ی مخرب برنامه‌هایی است که از داده‌های ارائه شده توسط مشتری در دستورات SQL استفاده می‌کند. بیشتر این آسیب‌پذیری‌ها ناشی از عدم اعتبارسنجی ورودی است. یعنی برنامه‌های کاربردی وب از ورودی‌های مخرب به‌عنوان بخشی از یک عملیات حساس استفاده می‌کنند بدون اینکه مقادیر ورودی را به‌درستی بررسی یا پاکسازی کنند. حملات SQL Injection پایگاه‌های داده‌ای را هدف قرار می‌دهد که از طریق یک وب Front End قابل دسترسی هستند. علاوه بر این، آنها از نقص در منطق اعتبارسنجی ورودی اجزای وب استفاده می‌کنند (Jin-Young Choi & Young-Su Jang, 2014).

به‌دلیل آسیب‌پذیری‌های مختلف وب سرورها و روش‌های غیر دقیق، علت حملات اسکریپت وب سرور در حال افزایش است، این حملات عمدتاً از طریق تزریق اسکریپت‌هایی به زبان‌های ASP (معمولاً مبتنی بر پایگاه داده MS-SQL Server) و یا PHP (معمولاً مبتنی بر پایگاه داده MySQL) به‌عنوان یک وسیله حمله‌ی اصلی صورت می‌گیرند.

حملات تزریقی تقریبا به جریان اصلی حملات تبدیل شده‌اند، و در حالی‌که فرآیند کامپایل وب سرور غالبا در سمت سرور انجام می‌شود، اسکریپت نویس یا برنامه نویس سایت معمولاً پدیده‌ی آزمایش ایمنی کد برنامه را نادیده می‌گیرد، که منجر به ارائه تعداد زیادی حفره‌های امنیتی در وب سرور می‌شود، از جمله حداقل ۷۰ درصد از این حفره‌های امنیتی مربوط به SQL Injection می‌شوند.

با وجود چنین نقص‌هایی هکرها و کاربران مخرب می‌توانند از سرور استفاده کنند و پیکربندی پایگاه‌داده، نقص‌ها و ساختار دقیق اظهارات غیرقانونی را از طریق برنامه‌ها یا اسکریپت‌هایی که به سرور تزریق می‌کنند، مجوزهای مدیر وب سایت را به‌دست آورند و محتوای پایگاه داده‌ی مربوطه را نیز در دست گیرند (XuePing-Chen, 2011).

اگرچه بسیاری از محققان طی مطالعات خود روش‌هاو ابزارهایی را برای مقابله با حملات SQL Injection ایجاد و پیشنهاد کرده‌اند، در این تحقیق سعی بر آن شده است که روش‌های مقابله و پیشگیری از حملات SQL Injection برروی پایگاه داده MS-SQL Server را مورد بررسی قرار دهیم.

روش تحقیق

در این تحقیق ابتدا به مفهوم حملات SQL Injection پرداخته شده و بطور مختصر تاریخچه اینگونه حملات مورد بررسی قرار گرفته است. سپس کارهای انجام شده و روشهای مقابله‌ی پیشنهادی سایر محققان برای جلوگیری از اینگونه حملات و مزایا و معایب برخی از آنها مورد مطالعه و بررسی قرار گرفته است که در ادامه به برخی از آنها اشاره شده است.

مروری بر کارهای انجام شده قبلی

همان‌گونه که در مقاله‌ی (Kashif Kifayat, et al, 2018) ذکر شده است، پایگاه های داده اغلب حاوی مقادیر قابل توجهی اطلاعات محرمانه هستند که می‌توانند برای هکرها و کاربران مخرب برنامه‌های کاربردی وب سودآور باشند. آنها می‌توانند با طرح پرس‌وجوهایی برای دستیابی به داده‌هایی که مجاز به مشاهده یا تغییر آن ها نیستند، اقدام کنند. SQL Injection یکی از جدی‌ترین تهدیدها برای برنامه‌های کاربردی وب است. راه حل آنها برای حل این مشکل، پیاده‌سازی روش‌های یادگیری ماشینی است که می‌توانند پرس‌وجوهای مخرب را بر اساس اطلاعات ساختار رشته‌های پرس‌وجو که از مجموعه‌ی آموزشی از پرس‌وجوهای تولید شده در زمان اجرا به دست می‌آیند، شناسایی کنند. این اطلاعات ساختاری با استفاده از تابع Gap Weighted String Subsequence Kernel (GWSSK) استخراج می‌شوند. این تابع شباهت رشته‌های پرس‌وجوهای ناشناخته را با رشته‌های پرس‌وجوهایی آموزشی از پیش انتخاب شده را محاسبه می‌کند. سپس یک طبقه‌بندی‌کننده‌ی ماشین بردار پشتیبان (SVM) از این اندازه‌گیری‌های شباهت استفاده می‌کند تا با تعیین یک

مرز تصمیم که فاصله‌ی بین دو کلاس را به حداکثر می‌رساند، مشخص کند که آیا پرس‌وجوی ناشناخته عادی است یا مخرب. این روش درواقع شکلی از روش جعبه سیاه است (William G.J. Halfond, et al, 2006).

این روش نیازی به مهندسی مجدد برنامه‌های کاربردی وب وابسته به SQL، یا افشای کامل سورس کد آنها ندارد که این مقوله به‌عنوان مزیت آن تلقی می‌شود زیرا به‌عنوان یک نقص برای بسیاری از روش‌های قبلی مطرح می‌شد.

از آنجایی که این روش از یک طبقه‌بندی‌کننده‌ی احتمالی در قالب طبقه‌بندی‌کننده‌ی SVM استفاده می‌کند، پرس‌وجوهای ناشناخته با ساختارهای پرس‌وجو که از مجموعه‌ی داده‌های آموزشی منحرف می‌شوند، همچنان به دلیل اطلاعات مشابه استخراج‌شده، به‌عنوان پرس‌وجوهای مخرب تعیین می‌شوند. راه حل آنها دو محدودیت واضح دارد:

اول اینکه روش آنها بایستی بین برنامه‌ی وب و پایگاه داده قرار گیرد که این سبب سربار سخت افزاری برای کار با راه حل تشخیص و پیشگیری می‌شود. علاوه بر این، الگوریتم‌های تشخیص هرگز دقت تشخیص کاملی ندارند و بنابراین مسائل مربوط به منفی‌های کاذب (حملاتی که موفق می‌شوند) که می‌توانند آسیب به پایگاه داده و مثبت‌های کاذب (حملاتی که موفق نمی‌شوند) که می‌توانند از عملکرد عادی یک پایگاه داده جلوگیری کنند، باید کاهش یابند (Abdelhamid MAKIOU, et al, 2014).

به‌عنوان مثال NoTamper یک روش تست جعبه سیاه است که برای تعیین آسیب‌پذیری‌ها در کدهای سمت سرور طراحی شده است. این روش اجازه می‌دهد تا آسیب‌پذیری‌ها وصله شوند، هر چند اگر آسیب‌پذیری‌ها شناسایی نشوند، هزینه‌ای سنگین در بر خواهد داشت. AMNESIA یکی دیگر از روش‌های کاوش آسیب پذیری است که تجزیه و تحلیل ایستا از کد برنامه‌ی وب را با نظارت زمان اجرا ترکیب می‌کند. SQLGuard به‌عنوان روشی برای تجزیه و تحلیل درختان تجزیه‌ی پرس‌وجو قبل و بعد از گنجاندن ورودی کاربر پیشنهاد شد و این اجازه را می‌دهد تا اجرای ورودی کاربر مورد بررسی قرار گیرد. CANDID یکی دیگر از روش‌های تجزیه و تحلیل سورس کد است که سورس کد را با درخواست‌های کاندید اضافی، اصلاح می‌کند. سپس پرس‌وجوهای زمان اجرا را می‌توان با آنها مقایسه کرد تا هرگونه اجرا غیرقانونی را مشخص کند.

دوم اینکه، این رویکرد شامل استقرار نرم‌افزار اضافی است که برای غربال کردن پرسش‌های تولید شده توسط یک برنامه‌ی وب قبل از اجرای آنها در پایگاه داده طراحی شده است. این راه‌حل‌های نرم‌افزاری از طیف گسترده‌ای از تکنیک‌ها استفاده می‌کنند و معمولاً برای استقرار در یک سیستم فعال هزینه‌ی بسیار کمتری دارند. متأسفانه، آنها اغلب راه حل کاملی برای مشکل نیستند. بسیاری از راه‌حل‌ها قادر به شناسایی هر نوع حمله‌ی SQL Injection نیستند و مسیری را برای مهاجمان ایجاد می‌کنند تا از آنها سوء استفاده کنند. آنها همچنین می‌توانند مستعد رویدادهای منفی و مثبت کاذب باشند که در آن الگوریتم‌های تشخیص، پرس‌وجوهای قانونی را مخرب تشخیص می‌دهند و آنها را مسدود می‌کنند و یا به درخواست‌های مخرب اجازه می‌دهند که منجر به ایجاد نقض امنیتی شود.

به‌عنوان مثال SQLProb یک معماری مبتنی بر پروکسی برای جلوگیری از حملات SQL Injection است. این راه حل لیستی از پرس‌وجوهای تولید شده توسط یک برنامه‌ی وب را تعریف می‌کند و تمام پرس‌وجوهای ممکن تولید شده توسط عملیات معمولی برنامه وب را پردازش می‌کند. سپس این پرسش‌ها توسط نرم‌افزار پراکسی جمع‌آوری می‌شوند تا مجموعه‌ای از پرس‌وجوهای SQL را از برنامه‌ی وب تولید کنند. در نهایت فیلتر پروکسی، پرس‌وجوهای ورودی را شناسایی کرده و آنها را رهگیری می‌کند.

یک روش جدید دیگر استفاده از داده کاوی گزارش‌های پایگاه داده (Database Log) برای شناسایی حملات SQL Injection پیشنهاد شد است (Dong Hoon Lee & Mi-Yeon Kim, 2014) که از فایل‌های گزارش پایگاه داده برای شناسایی پرس‌وجوهای اجرا شده در پایگاه‌داده استفاده شد. این فایل‌ها حاوی اطلاعاتی در مورد رشته‌ی پرس‌وجو و عملیات انجام شده توسط اجرا کننده‌ی پرس‌وجو است. این روش ابتدا یک درخت پرس‌وجو تولید می‌کند. این روش دارای دقت بسیار بالایی در حد ۹۹ درصد بود ولی نقطه ضعف اصلی این بود که فقط برای تشخیص حملات انجام شده و نه برای پیشگیری قابل استفاده بود و دلیل آن فقط به این خاطر بود که گزارش‌های پرس‌وجو که معیارهای آزمایش بودند، تنها زمانی تولید می‌شدند که یک پرس‌وجو اجرا شده باشد.

در مقاله‌ای دیگر (Lwin Khin Shar & Hee Beng Kuan Tan, 2013)، برای مقابله با تهدیدات SQL Injection و XSS استفاده از رویکردهای تشخیص آسیب‌پذیری مبتنی بر تکنیک‌های تحلیل لکه‌های استاتیک و پویا پیشنهاد شده است. هر چند نشان داده شده است که این رویکردهای مبتنی بر تجزیه و تحلیل لکه‌ها در تشخیص آسیب‌پذیری‌های SQLI و XSS مؤثر هستند، اما رویکردهای استاتیک عموماً هشدارهای نادرست زیادی تولید می‌کنند. رویکردهای پویا معمولاً دقیق‌تر هستند. اما در حالی‌که تجزیه و تحلیل پویا به‌طور معمول نیازمند چارچوب‌های تحلیل پیچیده است، پیاده سازی کامل برخی از این رویکردها نه به لحاظ تجاری و نه به لحاظ عمومی در دسترس هستند. در نتیجه، تیم‌های توسعه‌ی نرم افزار در اتخاذ این رویکردهای موجود با مشکلاتی مواجه می‌شوند.

از سوی دیگر، کاربرد تکنیک‌های یادگیری ماشین در پیش‌بینی نقص نرم‌افزار و پیش‌بینی آسیب‌پذیری، به نتایج امیدوارکننده‌ای دست یافته است. به‌طور کلی، آنها مدل‌های پیش‌بینی می‌سازند که ماژول‌ها، کامپوننت‌ها، برنامه‌های نرم‌افزاری و غیره را که با مجموعه‌ای از ویژگی‌های نرم‌افزار نشان داده می‌شوند، در یکی از کلاس‌ها (مثلاً معیوب و غیرمعیوب) با استفاده از طبقه‌بندی‌کننده‌هایی که روی همان مجموعه‌ی ویژگی‌های به دست آمده از ماژول‌های نرم افزاری با اطلاعات نقص یا آسیب‌پذیری شناخته شده آموزش داده شده‌اند، دسته‌بندی می‌کنند.

در مقاله‌ای دیگر (Jin-Young Choi & Young-Su Jang, 2014) بیان می‌کند که بیشتر آسیب‌پذیری‌ها SQL Injection، ناشی از عدم اعتبارسنجی ورودی است. یعنی برنامه‌های کاربردی وب از ورودی مخرب به‌عنوان بخشی از یک عملیات حساس استفاده می‌کنند بدون اینکه مقادیر ورودی را به‌درستی بررسی یا پاکسازی کنند. با استفاده از

آسیب‌پذیری‌های SQL Injection، مهاجم ممکن است اطلاعات پایگاه داده را بخواند، تغییر دهد یا حتی حذف کند. این حملات زمانی رخ می‌دهند که توسعه‌دهندگان رشته‌های کدگذاری‌شده را با ورودی ارائه‌شده توسط کاربر ترکیب می‌کنند تا پرس‌وجوهایی ایجاد کنند. اگر منابع ورودی به درستی تایید نشده باشند، مهاجمان ممکن است بتوانند پرس‌وجوی SQL مورد نظر توسعه دهنده را با درج کلمات کلیدی SQL جدید از طریق رشته‌های ورودی ساخته شده‌ی خاص، تغییر دهند.

ابزارهای عمومی، مانند پروکسی یا سیستم تشخیص دستورالعمل Instruction Detection System (IDS)، نیز تا حد زیادی در برابر حملات SQL Injection که از طریق پورت‌های مورد استفاده برای ترافیک وب معمولی انجام می‌شوند، بی اثر هستند (Halfond WG & Orso A. Amnesia, 2005). همچنین تکنیک‌های تطبیق ساختاری، مانند درخت تجزیه، هنگامی که درخت تجزیه‌ی پرس‌وجوی حمله با ساختار مورد انتظار مطابقت دارد، از منفی کاذب رنج می‌برند.

در مقاله‌ای دیگر (Benfano Soewito, et al, 2018) بیان می‌کند حملات SQL Injection مهمترین (پر انجام شده ترین و آسان‌ترین) نوع حملات برروی وب سایت‌ها یا برنامه‌های کاربردی مبتنی بر وب است که دارای پایگاه داده هستند، می‌باشند. حملات SQL Injection رایج‌ترین نوع حمله است و همچنین انجام آن توسط افراد ماهر و غیر ماهر بسیار آسان است. برای اینکه یک وب سایت و برنامه‌ی تحت وب عاری از حملات SQL Injection باشد، لازم است ورودی‌ها، پارامترهای URL و سایر متغیرهایی را که به‌صورت پویا در پایگاه داده جستجو می‌شوند اعتبار سنجی کنید تا محتویات ورودی‌های با ورودی مورد نظر مطابقت داشته باشد.

حملات SQL Injection به مهاجمان اجازه می‌دهد تا به پایگاه داده‌ی برنامه دسترسی داشته باشند و به مهاجمان اجازه می‌دهد تا داده‌های حساس را بخوانند و حتی دستکاری (درج، تغییر و حذف) کنند و در نتیجه تهدیدات امنیتی مانند: جعل هویت، تخریب داده‌ها، معاملات غیرمجاز، تصاحب حقوق مدیریت پایگاه داده و غیره را ایجاد کنند.

مفهوم مدیریت SQL Injection در واقع بسیار ساده است، یعنی اطمینان از فرمت داده‌های ورودی یا محتویات پارامترها در قالب فرمت مورد نظر پرس‌وجو می‌باشد. برای مثال وقتی می‌خواهیم یک شناسه به‌شکل یک عدد صحیح را پاس بدهیم، قبل از پرس‌وجو، باید مطمئن شویم که مثلاً شناسه‌ی id به شکل یک عدد صحیح، واقعاً درست است، یا مثلاً می‌خواهم نام یک شخص را در پرس‌وجوی بررسی کنیم، بنابراین قبل از ارسال داده‌ها به پرس‌وجو باید مطمئن شویم که داده‌ی ارسالی یک رشته‌ی واقعی است که فقط از حروف الفبای A تا Z یا a به z و اعداد و یا فاصله خالی تشکیل شده است. بر همین اساس می‌توان از عبارات منظم Regular Expression که می‌توانند یک الگو را در یک رشته جستجو کنند، استفاده کرد. با عبارت منظم می‌توان الگوهایی برای فیلتر ورودی کاربر یا پارامترهای URL ورودی طراحی کرد مانند الگوی قالب ایمیل، الگوی نام کاربری و غیره.

راهکارهای پیشنهادی

در این مقاله با بررسی تحقیقات انجام شده قبلی که اکثراً برای پایگاه داده‌های MySQL می‌باشند و براساس مطالعات صورت گرفته، برای مقابله و کاهش احتمال دسترسی غیر مجاز به داده‌های موجود در پایگاه‌داده و بخصوص سیستم مدیریت پایگاه داده MS-SQL Server راهکارهای زیر پیشنهاد می‌شود:

- **ایزوله کردن سرور پایگاه داده:** با ایزوله کردن سرور پایگاه داده از سرور میزبان فایل‌های وب، می‌توان تا حد زیادی از نفوذ به پایگاه‌های داده جلوگیری کرد. در این حالت دسترسی به داده‌ها بایستی از API ها و از طریق وب سرویس‌های اینترنتی و کنترل شده صورت گیرد. از جمله معایبی که به این روش وارد است، افزایش هزینه بدلیل استفاده از سرور مجزا و کاهش سرعت پاسخگویی به درخواست‌های داده می‌باشد. در این حالت می‌توان از Sanitizer ها نیز استفاده کرد زیرا ماهیت Sanitizer ها به‌گونه‌ای است که باید بعنوان واسط بین صفحه‌وب و پایگاه داده قرار گیرند.
- **استفاده از فایروال:** فایروال‌ها می‌توانند به‌عنوان یکی از ابزارهای کارآمد برای جلوگیری از نفوذ به سرورها باشند. زمانیکه سرور پایگاه داده از سرور وب مجزا باشد، می‌توان به استفاده از فایروال دسترسی احتمالی به آن سرور را حتی درصورت نفوذ به سرور میزبان فایل‌های وب را مسدود کرد.
- **استفاده از وب سرویس:** بهتر است برقراری ارتباط بین صفحات وب و پایگاه‌داده در قالب وب سرویس باشد. در صورت استفاده از وب سرویس، دامنه دستورات قابل اجرا محدود شده و تمامی درخواست‌های داده در قالب خاصی به سمت پایگاه داده ارسال شده و خروجی آنها نیز تحت قالب خاصی برگردانده می‌شود. همچنین درصورت استفاده از وب سرویس می‌توان برای برقراری ارتباط بین پایگاه داده و صفحات وب از روش‌های اعتبار سنجی نیز استفاده کرد.
- **استفاده از اسامی مستعار:** یکی دیگر از روش‌های کاهش احتمال دسترسی به داده‌های با ارزش در مقابل به‌خصوص حملات SQL Injection استفاده از اسامی مستعار برای ستون‌ها و جداول داده در پایگاه‌داده می‌باشد. با استفاده از این روش، درصورتی‌که پایگاه داده مورد حملات SQL Injection قرار گیرد، شخص مهاجم به اسامی واقعی ستون‌ها و جداول دسترسی نخواهد داشت که این خود می‌تواند تا حد زیادی از موفقیت حملات بکاهد.
- **استفاده از View ها:** برای انجام عملیات برروی داده‌ها بهتر است از View هایی استفاده شود که در آنها نیز ستون‌ها و جداول پایگاه‌داده با اسامی مستعار فراخوانی شوند. از View ها می‌توان برای مخفی نگه داشتن بخش‌های حساس پایگاه‌داده از دید کاربران نیز استفاده کرد.

- **استفاده از Stored Procedure:** بهتر است برای انجام عمالی مانند درج، ویرایش و حذف داده‌ها در پایگاه‌داده از Stored Procedure ها استفاده شود. بزرگترین مزیت استفاده از روال‌ها این است که شخص مهاجم نمی‌تواند دستورات مخرب خود را در قالب دستورات SQL به سایت یا برنامه وب شما تزریق کند.
- **استفاده از Function:** همچنین برای استخراج داده‌ها و انجام پرس‌وجو برروی داده‌ها بهتر است از Function ها استفاده شود و حتی‌الامکان در این توابع از View ها و اسامی مستعار برای ستون‌های داده و جداول استفاده شود تا به بتوان نوعی اصل داده‌ها و اطلاعات حساس را از دید مهاجم مخفی نگه داشت.
- **عدم استفاده از Relation:** اکثر طراحان پایگاه داده از Relation برای برقراری ارتباط بین جداول اصلی با جداول فرعی براساس کلیدهای اصلی و کلیدهای خارجی جداول استفاده می‌کنند. این ویژگی می‌تواند منجر به افشای جداولاصلی و فرعی براساس ماهیت ارتباط بین آنها، درصورت موفقیت در حمله به یکی از این جداول شود. لذا توصیه می‌شود بجای استفاده از Relation، از طریق پرس‌وجو و دستوراتی مانند Join و حتی‌الامکان در قالب توابع، روال‌ها و View ها داده ها را از جداول مرتبط استخراج کرد.
- **رمزنگاری داده‌های حساس:** با استفاده از تکنیک‌های رمزنگاری داده‌های، بخصوص درمورد داده‌های حساسی مانند شماره تماس‌ها، شماره کارت‌های اعتباری، آدرس‌ها و غیره، می‌توان تا حد زیادی از قابل استفاده بودن آنها درصورت نفوذ به پایگاه داده، جلوگیری کرد. برای این منظور هم می‌توان از ابزارهای رمزنگاری خود MS-SQL Server استفاده کرد و هم از کتابخانه‌ها و کامپونت‌های طراحی شده برای این منظور.
- **پردازش استثنائات:** پردازش استثنائات و خطاها می‌تواند یکی دیگر از موارد مؤثر در جلوگیری از حملات SQL Injection باشد. معمولاً هکرها با ارسال پرس‌وجوها و پارامترهای مد نظر خود به سمت سرور وب، با دریافت خطاهایی که از سمت سرور بخصوص IIS صادر می‌شود، به سرور نفوذ می‌کنند. در این مورد می‌توان با پردازش استثنائات و حالاتی که در وب سایت یا برنامه تحت وب منجر به بروز خطا می‌شوند، تا حد زیادی از موفقیت اینگونه حملات جلوگیری کرد.

نتیجه‌گیری

از آنجایی که در عصر کنونی اکثر فعالیت‌های افراد به‌صورت آنلاین و از طریق محیط‌های وب انجام می‌گیرد، و از آنجایی که داده‌های نقش حیاتی را در این میان ایفا می‌کنند، هرگونه افشا و یا خرابکاری داده‌ها می‌تواند منجر به بروز خسارات سنگین و حتی جبران ناپذیری شود. از طرفی به‌واسطه‌ی پیشرفت سریع تکنولوژی و تجهیزات و روش‌های دسترسی و ذخیره سازی داده‌ها، همواره ممکن است حفره‌های امنیتی جدیدی بوجود آید، لذا حفاظت از این داده‌ها در مقابل کاربران مخرب و هکرها، کاری بسیار دشوار می‌باشد. تقریبا می‌توان چنین بیان کرد که هیچ سرور و پایگاه داده‌ای نمی‌تواند برای همیشه ایمن و غیر قابل نفوذ باشد زیرا هکرها همواره در کمین بوده و به دنبال استفاده از حفره‌های امنیتی جدید می‌باشند و با استفاده از تکنیک‌ها و روش‌های جدید قصد نفوذ به سرورها را دارند، لذا همواره بایستی برای مقابله با این‌گونه حملات آمادگی داشت و بررسی نقاط احتمالی نفوذ را جزو وظایف روزمره خود قرار داد تا بتوان حتی‌الامکان از نفوذ به سرورها جلوگیری کرد و یا درصورت نفوذ بتوان سریعاً آنرا برطرف کرد و میزان خسارت را به حداقل رساند.

منابع

Benfano Soewito, Fergyanto E.Gunawan, Hirzi, Frumentius. (2018). Prevention Structured Query Language Injection Using Regular Expression and Escape String. 3rd International Conference on Computer Science and Computational Intelligence 135 (2018). 678-687.

Kashif Kifayat, Paul R.McWhirter, Qi Shi, Bob Askwith. (2018). SQL Injection Attack classification through the feature extraction of SQL query strings using a Gap-Weighted String Subsequence Kernel. Journal of Information Security and Applications 40 (2018). 199-216.

Lwin Khin Shar, Hee Beng Kuan Tan. (2013). Predictiong SQL injection and cross site scripting vulnerabilities through mining input sanitization patterns. Information and Software Technology 55 (2013). 1767-1780.

Jin-Young Choi, Young-Su Jang. (2014). Detecting SQL injection attacks using query result size. ScienceDirect 44 (2014). 104-118.

XuePing-Chen. (2011). SQL injection attack and guard technical research. Elsevier 15 (2011). 4131-4135.

William G.J. Halfond, Alessandro Orso, Panagiotis Manolios. (2006). Using positive tainting and syntax-aware evaluation to counter SQL injection attacks. Proceedings of the 14th ACM SIGSOFT international symposium on foundation of software engineering (2006). PP 175-85.

Abdelhamid MAKIOU, Youcef BEGRICHE, Ahmed SERHROUCHNI. (2014). Improving Web Application Firewalls to Detect Advanced SQL Injection Attacks. 10th International Conference on Information Assurance and Security (IAS) 2014. IEEE (2014). P 35-40.

Dong Hoon Lee, Mi-Yeon Kim. (2014). Data-mining based SQL injection attack detection using internal query trees. Expert Systems with Applications (2014). ESWA 9201, 10 March 2014.