بررسی و دسته‌بندی حملات تزریق به زبان پرس‌وجوی ساخت‌یافته

سید محمد بیدکی[1,i]، سعیده حسینی اقبال[ii]، امیر جهانگرد رفسنجانی[iii]

i- مربی گروه کامپیوتر، دانشکده فنی و مهندسی جم، دانشگاه خلیج فارس بوشهر

ii- دانشجوی کارشناسی ارشد، دانشکده مهندسی کامپیوتر، دانشگاه یزد

iii- استادیار دانشکده مهندسی کامپیوتر، دانشگاه یزد

خلاصه

امروزه داده به عنوان یکی از دارایی‌های کلیدی هر سازمان محسوب شده و افشای اطلاعات[2] یک سازمان می‌تواند اثرات مخربی برای آن داشته باشد. حملات تزریق زبان پرس‌وجوی ساخت‌یافته[3] از جمله تهدیدهای جدی برای امنیت نرم افزارهای تحت وب می‌باشند، به طوری که به مهاجم این قابلیت را می‌دهد تا به داده‌های موجود در پایگاه داده‌ی نرم‌افزار، دسترسی نامحدود داشته باشد. محققین راهکارهای متنوعی برای مقابله با حملات تزریق اس کیو ال ارائه داده‌اند ولی متاسفانه هر راهکار در برخی گونه‌های حمله مفید و در برخی دیگر کارایی ندارد. در این مقاله انواع مختلف حمله‌های تزریق اس کیو ال[4] معرفی شده است. برای هر حمله توضیحات و چگونگی اجرای آن حمله بیان گردیده و همچنین روش-هایی که برای تشخیص و جلوگیری از این حملات وجود دارد و نقاط قوت و ضعف روش‌های جلوگیری، بررسی شده است.

کلمات کلیدی: حملات تزریق اس کیو ال، امنیت پایگاه‌داده‌ها، نرم افزار تحت وب، افشای اطلاعات، تزریق کد.

۱. مقدمه

امروزه پایگاه‌داده به عنوان قلب اغلب نرم‌افزارها در سازمان‌های مختلف محسوب می‌شود و به عنوان یک دارایی کلیدی و حساس برای آن سازمان محسوب می‌گردد. ولی متاسفانه اکثر سازمان‌ها به محافظت از این منبع با ارزش به اندازه کافی توجه ندارند. در مطالعات مشاهده شده است که سازمان‌ها تنها ۵ درصد از ۲۷ میلیارد دلاری که برای تهیه محصولات امنیتی هزینه کرده‌اند به صورت مستقیم با امنیت مراکز داده در ارتباط بوده است [۱]. لذا حوزه پایگاه‌داده‌ها همواره یکی از نقاط مورد توجه مهاجمین بوده است.

حملات تزریق اس کیو ال (SQLI) که یکی از انواع حملات تزریق کد[5] است، طی سالهای گذشته تا کنون به عنوان یکی از مهم‌ترین تهدیدات نرم‌افزارهای تحت وب [۲] و به طور کلی امنیت پایگاه داده‌ها [۳] شناخته شده است. نرم‌افزار-

[1] **Corresponding author: هضو هیئت علمی دانشگاه خلیج فارس بوشهر**
Email: s.m.bidoki@pgu.ac.ir
[2] Information Leakage
[3] Structural Query Language (SQL)
[4] SQL Injection (SQLI)
[5] Code Injection

های تحت وب که در مقابل این حملات آسیب‌پذیرند، به مهاجم اجازه می‌دهند که دسترسی کامل به پایگاه‌داده‌ی آنها داشته باشد. داده‌های موجود در پایگاه‌داده سازمان‌ها، معمولاً اطلاعات حساس کاربران و مشتریان و سایست‌های درون و یا برون‌سازمانی است. افشای این اطلاعات می‌تواند منجر به رنج وسیعی از آسیب‌ها از جمله افشای هویت، از دست رفتن اطلاعات محرمانه‌ی کاربران و سوءاستفاده از کاربران، بدنامی برای سازمان، اخاذی توسط مهاجمین، از دست دادن مشتریان یا شرکای تجاری سازمان و حتی دعاوی حقوقی برعلیه سازمان گردد.

متاسفانه نرم‌افزارهای تحت وب که درمقابل حملات SQLI آسیب‌پذیرند، بسیار گسترده می‌باشند. براساس مطالعه‌ای که توسط گروه گارتنر روی ۳۰۰ وب‌سایت اینترنتی صورت گرفت، نشان داده شد که اکثر آنها دربرابر حملات SQLI آسیب‌پذیرند و ۷۰درصد آسیب‌پذیری آنها در لایه کاربرد قرار دارد [۴]. در سال‌های گذشته وب‌سایت‌های مهم و معتبری مورد این نوع حمله قرار گرفته‌اند و سازمان‌های پشتیبان آنها را به چالش کشیده است. به عنوان نمونه در سال ۲۰۰۸ مهاجمین با استفاده از آسیب‌پذیری IIS مایکروسافت، ۵۰۰هزار وبسایت را مورد حمله قرار دادند. در سال ۲۰۱۱، هکرها اطلاعات حساب کاربری ۱۵۰هزار نفر از کاربران Sony را مورد سرقت قرار دادند. در سال ۲۰۱۲ اطلاعات ۶.۵ میلیون کاربر LinkedIn و ۴۵۰هزار کاربر Yahoo از طریق حملات تزریق SQL ربوده شد. در سال ۲۰۱۴ سرور مهندسی زیست-پزشکی دانشگاه جان هاپکینز مورد حمله قرار گرفت و اطلاعات شخصی دانشجویان و کارمندان آن موسسه به سرقت رفت [۵].

تزریق اس کیو ال به دسته‌ای از حملات تزریق کد اتلاق می‌شود که اطلاعات گرفته‌شده از کاربر را در ساخت یک عبارت اس کیو ال به‌کار می‌برد. در این صورت مهاجم می‌تواند دستورات اس کیو ال را از طریق فیلدهای ورودی، به صورت مستقیم به پایگاه‌داده ارسال کند. متاسفانه پیاده‌سازی اکثر نرم‌افزارهای تحت وب، به گونه‌ایست که از ورودی‌های کاربر در ایجاد ساختارهای اس کیو ال به منظور دسترسی به منابع پایگاه‌داده استفاده می‌کند و بدین صورت برنامه مستعد حمله تزریق اس کیو ال خواهد بود. از اینرو بدیهیست که مهمترین دلیل آسیب‌پذیری نرم‌افزار نسبت به تزریق اس کیو ال، ارزیابی و اعتبارسنجی ناکافی ورودی‌های کاربر می‌باشد.

بسیاری از روش‌های پیشگیری از حملات، به شیوه‌های کدنویسی دفاعی و اعتبارسنجی پارامترهای موجود در برنامه پرداخته‌اند که کارایی مناسبی ارائه داده‌اند. ولی این روش‌ها در عمل مبتنی بر انسان بوده و بستگی به مهارت برنامه‌نویس دارد و درنتیجه مستعد خطا می‌باشند. درنتیجه رویکردهای گوناگونی برای تشخیص و مقابله خودکار با حملات SQLI پیشنهاد شده است. اما از آنجا که گونه‌های این نوع حمله متنوع است، هر کدام از روش‌ها در مقابل بخشی از این گونه‌ها می‌تواند مقاومت نماید.

در این مقاله مطالعه‌ای روی حملات تزریق SQL و روش‌های مقابله با آن صورت گرفته است. ابتدا مختصری درمورد مکانیزم‌ها و اهداف حملات SQLI بحث شده است. سپس نمونه‌ای از یک نرم‌افزار تحت وب آسیب‌پذیر درمقابل حملات SQLI و گونه‌های مختلف حملات تزریق به همراه یک مثال ساده بیان گردیده است. در بخش بعد به رویکردهای مختلف برخورد با حملات SQLI پرداخته شده و نهایتاً ارزیابی و مقایسه این رویکردها از جنبه‌های مختلف ارائه گردیده است.

۲. مفاهیم اولیه حملات SQLI

حمله SQLI زمانی رخ می‌دهد که مهاجم تأثیر مورد انتظار یک پرسش اس کیو ال را با واردکردن دستورات یا عملگرهای جدید اس کیو ال، داخل پرس‌وجوی[1] موجود در پیاده‌سازی نرم‌افزار، تغییر می‌دهد. در این بخش دو مفهوم بیان شده مکانیزم تزریق و هدف از حمله.

۲.۱. مکانیزم تزریق

به طور کلی هر پارامتری از جمله پارامترهای ورودی کاربر که در ایجاد یک ساختار SQL در بدنه نرم‌افزار موثر باشد می-تواند راهی برای نفوذ مهاجم باشد. مکانیزم‌های متنوعی برای تزریق ساختارهای اس کیو ال مخرب در نرم‌افزارهای آسیب-پذیر وجود دارد که در ذیل مختصراً تعدادی از آنها بیان شده است.

- تزریق از طریق ورودی کاربر: هرکدام از اجزاء فرم‌های HTML یا پارامترهای URL می‌تواند مسیری برای تزریق در اختیار مهاجم قرار دهد. در این روش مهاجم با فراهم‌کردن ورودی کاربر به صورت یک ترکیب ماهرانه، ساختار اس کیو ال موردنظرش را تزریق می‌کند.
- تزریق توسط Cookieها: Cookieها فایل‌هایی شامل اطلاعات تنظیمات دلخواه کاربر سیستم هستند که توسط نرم افزار وب تولید شده و در کامپیوتر کلاینت ذخیره می‌گردد. به همین خاطر یک کاربر مهاجم می‌تواند به آنها به صورت محلی دسترسی مخرب داشته باشد. اگر نرم‌افزار وب برای ساخت پرس‌وجوی اس کیو ال از Cookie استفاده کند، مهاجم می‌تواند حمله را از طریق Cookie تغییر یافته خود انجام دهد.
- تزریق از طریق متغیرهای Server: متغیرهای سرور مجموعه‌ای از متغیرها شامل اطلاعات HTTP، سرآیندهای شبکه و متغیرهای محیطی هستند. نرم‌افزار وب از این اطلاعات برای فرایند logging و آمارگیری و تحلیل رفتار و اهداف کاربران استفاده می‌کنند. حال اگر این اطلاعات بدون بررسی و پاکسازی در پایگاه‌داده ذخیره شوند، می‌توانند به عنوان یک نقطه آسیب‌پذیری در مقابل تزریق اس کیو ال تلقی گردند.
- تزریق مرتبه دوم[2]: اکثر حملات اصطلاحاً مرتبه اول هستند بدین معنا که مهاجم در یک مرحله اقدام به تزریق کرده و همان موقع از نتیجه اجرای ساختار اس کیو ال تولید شده بهره می‌برد. ولی در حملات تزریق مرتبه دوم، مهاجم اطلاعاتی را از طریق یک فعالیت نرمال، در پایگاه‌داده وارد می‌کند تا بعداً از طریق آن، حمله مورد نظر خود را صورت دهد [۶].

[1] Query
[2] Second-Order Injection

۲.۲. اهداف حملات

اهداف حملات مهاجمین می‌تواند متفاوت باشد. هر نوع حمله یک یا چند مورد از این اهداف را دنبال می‌کند. در ادامه لیستی از اهداف ارائه شده است.

- یافتن پارامترهای قابل تزریق: مهاجم سعی دارد پارامترها و ورودی‌های کاربر آسیب‌پذیر موجود در نرم‌افزار وب را یافته تا از طریق آنها بتواند ورودی دلخواه خود را تزریق نماید.
- ترسیم اثر انگشت پایگاه‌داده[۱]: مهاجم اطلاعات نوع و نسخه پایگاه‌داده نرم‌افزار وب را یافته و بدین صورت می‌تواند رفتار سیستم مدیریت پایگاه‌داده را در برابر حملات مختلف پیش‌بینی کرده و یا حملاتی متناسب با نوع پایگاه-داده صورت دهد.
- تعیین شِمای پایگاه‌داده[۲]: برای دستیابی به داده‌ها، مهاجم نیاز دارد ساختار و قالب پایگاه‌داده را بداند مثلاً نام جداول و ستون‌ها و نوع مقادیر ستون‌ها و ... .
- استخراج داده‌ها: اکثر مهاجمین با این هدف حملات خود را انجام می‌دهند که اطلاعات موجود در جداول پایگاه-داده را استخراج کرده و مورد سوءاستفاده قرار دهند.

- افزودن داده‌ها و یا تغییر داده‌های موجود در پایگاه‌داده
- اختلال در ارائه سرویس[۳]: این حملات با هدف خاموش و یا مشغول کردن پایگاه‌داده به منظور عدم توانایی نرم‌افزار در ارائه سرویس به کاربران صورت می‌گیرد (مثل حذف کردن یک جدول و یا قفل کردن آن).
- دور زدن عملیات احراز هویت[۴]: با دور زدن فرایندهای اصالت‌سنجی، مهاجم می‌تواند سطح دسترسی کاربران متفاوت سیستم را برای خود داشته باشد.
- اجرای دستورات راه دور: با این حملات مهاجم می‌تواند دستورات دلخواه خود را روی پایگاه‌داده از راه‌دور اجرا کند مثل فراخوانی رویه‌های ذخیره‌شده[۵] و یا توابع در دسترس کاربران پایگاه‌داده.
- افزایش امتیازات دسترسی: این حملات از حفره‌های منطقی و خطاهای پیاده‌سازی پایگاه‌داده استفاده می‌کنند تا امتیازات دسترسی مهاجم را به پایگاه‌داده، افزایش دهند.

۳. یک نمونه نرم‌افزار آسیب‌پذیر

قبل از معرفی انواع مختلف حمله‌های SQLI، یک نمونه نرم‌افزار کاربردی آسیب‌پذیر در برابر تزریق اس کیو ال در شکل ۱ معرفی شده است. از این نرم‌افزار در بخش‌های بعد برای بیان نمونه حملات استفاده گردیده است. این مثال یک قطعه کد را نشان می‌دهد که فرایند احراز هویت را پیاده‌سازی کرده است. قطعه کد زیر از پارامترهای ورودی "login" و "pass" و "pin" برای ساخت عبارت پرس‌وجوی پویا[۶] به صورت یک رشته استفاده می‌کند و عبارت حاصل را برای اجرا به پایگاه‌داده ارسال می‌نماید. اگر کاربری با مشخصات داده‌شده در پایگاه‌داده وجود داشته باشد، اطلاعات حساب کاربری او نمایش داده می‌شود وگرنه پیغام خطایی تولید خواهد شد.

[1] Database Finger Print
[2] Database Schema
[3] Denial of Service
[4] Authentication
[5] Stored Procedure
[6] Dynamic Query

```
1. String login, Password, pin, query;
2. Login=getparameter("login");
3. Password=getparameter("pass");
4. Pin=getparameter("pin");
5. Connection conn.CreatConnection("MyDataBase");
6. Query=SELECT account From USERS Where login=' "+login+" 'AND
   pass='" +password+" ' AND pin=' "+pin;
7. ResultSet result    =conn.Executequery(Query);
8. If(result=!Null)
9. DisplayAccounts(result);
10. Else
11. Displayfailed();
```

شکل (۱) نمونه‌ای از کد اجرایی یک برنامه آسیب‌پذیر

۴. گونه‌های مختلف حملات SQLI

در این بخش برخی از معروفترین انواع حملات SQLI معرفی و درمورد آنها بحث شده است. برای هر نوع حمله یک نام مستعار، اهداف، شرح عملکرد و مثالی از آن نوع بیان گردیده است.

۴.۱. حملات درست‌نما

اهداف: گریز از احراز هویت، یافتن پارامترهای قابل نفوذ، استخراج داده.

توضیح: رایج‌ترین کاربرد حملات درست‌نما[1]، دور زدن فرایندهای احراز هویت و سپس استخراج اطلاعات است. عملکرد اصلی اینگونه حملات، تزریق کد در گزاره‌های شرطی بخش WHERE است به طوری که همواره نتیجه عبارت گزاره، به‌صورت درست[2] ارزیابی شود یا به عبارتی شرط بازیابی همواره برقرار باشد. بدین صورت مهاجم یک جمله شرطی را به یک درست‌نما تبدیل می‌کند. حمله زمانی موفقیت‌آمیز است که تمام رکوردها برگردانده شده یا در صورت برگرداندن حداقل یک رکورد برخی اقدامات انجام شود.

مثال: در این مثال، مهاجم در فیلد ورودی login، عبارت x' or 1=1 -- را به عنوان کلمه عبور وارد می‌کند. پرس-وجوی حاصل به فرم زیر خواهد بود:

SELECT accounts FROM users WHERE login='x' or 1=1 -- AND pass='???' AND pin='???'

از عبارت فوق مشخص است که کد تزریق شده، بخش‌های دوم و سوم شرط را کامنت کرده و گزاره شرطی where را به یک درست‌نما تبدیل کرده است. نتیجه این مثال بی‌اثرشدن فرایند احراز هویت و دسترسی مهاجم به نرم‌افزار بدون نیاز به داشتن نام کاربری معتبر خواهد بود.

۴.۲. پرس‌وجوهای غیر مجاز یا نادرست

اهداف: شناسایی پارامترهای آسیب‌پذیر، تعیین اثرانگشت پایگاه‌داده، استخراج داده.

توضیح: مهاجم این نوع حمله می‌تواند اطلاعاتی در مورد ساختار و نوع پایگاه‌داده زیربنایی نرم‌افزار بدست آورد. اطلاعات مورد نیاز مهاجم از طریق خطاهای سرور پایگاه‌داده به او داده می‌شود. این خطاها به صورت نرمال برای راهنمایی

[1] Tautology
[2] True

کاربر برنامه‌نویس تولید می‌شود ولی مهاجم می‌تواند از آنها سوءاستفاده نماید. برای این نوع حمله، مهاجم ساختارهایی را تزریق می‌کند که منجر به یک خطای نحوی یا منطقی شود.

مثال: هدف مهاجم در این مثال ایجاد خطای تبدیل نوع است که می‌تواند اطلاعات محرمانه‌ای را در فیلد ورودی وارد می‌کند:pinرا به او نشان دهد. برای این کار مهاجم متن زیر را در

convert(int,(select top 1 name from sysobjects where xtype='u'))

درنتیجه عبارت پرس‌وجوی زیر به پایگاه‌داده ارسال می‌شود:

SELECT accounts FROM users WHERE login='' AND pass='' AND pin= convert (int,(select top 1 name from sysobjects where xtype='u'))"

در این حمله، عبارت تزریق شده سعی دارد به نام یکی از جداول تعریف شده‌توسط کاربر (xtype='u') که در جدول فراداده‌های[1] پایگاه‌داده است، دسترسی نماید. پرس‌وجو توسط تابع convert اقدام به تبدیل نام این جدول به یک عدد صحیح می‌کند. از آنجا که این تبدیل نوع در مایکروسافت اس کیو ال سرور مجاز نیست، پایگاه داده پیام خطایی را به فرم زیر نمایش می‌دهد:

"Microsoft OLE DB Provider for SQL Server (0x80040E07) Error converting nvarchar value 'CreditCards' to a column of data type int".

دو بخش اطلاعاتی مهم برای مهاجم در این پیام خطا وجود دارد. اول اینکه مهاجم پی می‌برد پایگاه‌داده زیربنایی مایکروسافت اس کیو ال سرور است. در ادامه پیغام خطا، مقدار رشته‌ای *CreditCards* به عنوان دلیل بروز خطای تبدیل نوع ذکر شده است که نام اولین جدول تعریف‌شده توسط کاربر در پایگاه‌داده را برای مهاجم فاش می‌کند [۷]. مهاجم می‌تواند به شیوه مشابه، دیگر جداول موجود را شناسایی کرده و حملات بعدی خود را روی آنها اعمال نماید.

۴.۳. پرس‌وجوهای اجتماع

اهداف: دور زدن احراز هویت، استخراج داده.

توضیح: در این نوع حمله، مهاجم ساختار پرس‌وجو را به نحوی تغییر می‌دهد که اطلاعاتی را از جدولی غیر از جدول مورد انتظار پرس‌وجو استخراج کند. حمله توسط تزریق یک ساختار select که با پرس‌وجوی نرمال موجود در نرم‌افزار، اجتماع می‌شود، صورت می‌گیرد. مهاجم کنترل کامل روی این ساختار داشته و می‌تواند اطلاعات را از جدول دلخواهش استخراج کند.

مثال: مهاجم می‌تواند عبارتی به فرم زیر را در فیلد ورودی *login* وارد نماید:

'UNION SELECT cardNo from CreditCards where acctNo=10032 --

که باعث ایجاد پرس‌وجوی زیر می‌شود:

SELECT accounts FROM users WHERE login='' UNION SELECT cardNo from CreditCards where acctNo=10032 -- AND pass='' AND pin=

بخش اول پرس‌و جو مربوط به کد پیاده‌سازی نرم‌افزار است و مقدار تهی را برمی‌گرداند، در حالی که پرس‌وجوی دوم، داده‌ها را از جدول "CreditCards" برمی‌گرداند. در این مثال، پایگاه‌داده مقدار ستون "cardNo" را به ازای شماره‌حساب ۱۰۰۳۲، برمی‌گرداند [۸].

[1] Metadata

۴.۴. پرس‌وجوی سوارشده

اهداف: استخراج داده، درج یا تغییر داده، اختلال در ارائه سرویس، اجرای دستورات راه دور.

توضیح: در حملات پرس‌وجوی سوارشده[1]، مهاجم پرس‌وجوهای اضافه‌ای را در کد تزریق می‌کند. هدف او در این نوع حمله تغییر دادن پرس‌وجوی اصلی نیست بلکه پرس‌وجوهای جدید و مستقلی را تزریق می‌کند (بر پشت پرس‌وجوی اصلی سوار می‌کند). در نتیجه عبارت پرس‌وجو که شامل چندین پرس‌وجو است به مفسر پایگاه‌داده ارسال و اجرا می‌شود. این نوع حمله می‌تواند بسیار مخرب باشد. در صورت موفقیت، مهاجم عملاً می‌تواند هر نوع دستور اس کیو ال را تزریق کند.

مثال: اگر مهاجم عبارت ';drop table users;-- را در فیلد *password* وارد کند، نرم‌افزار پرس‌وجوی زیر را تولید می‌نماید:

SELECT accounts FROM users WHERE login='doe' AND pass='''; drop table users; -- ' AND pin=123

پس از اتمام اولین پرس‌وجو، پایگاه‌داده انتهای پرس‌وجو را با مشاهده علامت (;) تشخیص می‌دهد و ادامه عبارت را به عنوان یک ساختار مستقل اجرا می‌کند. نتیجه اجرای پرس‌وجوی دوم حذف جدول users خواهد بود که منجر به از بین رفتن اطلاعات کاربران نرم‌افزار و درنتیجه عدم امکان اجازه ورود به آنها خواهد شد و سرویس‌دهی نرم‌افزار را مختل می‌نماید [۹].

۴.۵. رویه‌های ذخیره‌شده

اهداف: افزایش امتیازات دسترسی، اختلال در ارائه سرویس، اجرای دستورات راه دور.

توضیح: حمله‌های از این نوع، تلاش می‌کنند رویه‌های ذخیره‌شده در پایگاه‌داده را اجرا کنند. اغلب سیستم‌های مدیریت پایگاه‌داده با رویه‌های ذخیره‌شده‌ی استانداردی ارائه می‌شوند. بنابراین وقتی مهاجم بداند نوع پایگاه‌داده نرم‌افزار چیست، می‌تواند رویه‌های همراه با آن را فراخوانی کند. از طرفی اکثر توسعه‌دهندگان نرم‌افزار بر این باورند که استفاده از رویه‌های ذخیره‌شده برای تعامل با پایگاه‌داده، نرم افزار وب را در مقابل حملات تزریق اس کیو ال مصون می‌دارد. ولی این تصور نادرست است. اگر بدنه رویه‌های ذخیره‌شده به صورت پویا از طریق عملیات الحاق رشته‌ای ساخته شود، آن روال به همان اندازه که عبارات اس کیو ال پویا در بدنه برنامه آسیب‌پذیرند، می‌تواند مورد حمله قرار گیرد [۹].

مثال: فرض کنید مهاجم از طریق پارامتر *password* عبارت ';SHUTDOWN;-- را تزریق نماید. این کار موجب تولید پرس‌وجوی زیر خواهد شد:

SELECT accounts FROM users WHERE login='doe' AND pass=' '; SHUTDOWN; -- AND pin=

این حمله می‌تواند مانند یک حمله سوارشده عمل کند. اولین پرس‌وجو به صورت نرمال اجرا می‌شود و سپس عبارت مخرب دوم اجرا می‌شود. در این حالت، روال ذخیره‌شده SHUTDOWN موجود در پایگاه‌داده، توسط مهاجم از راه‌دور فراخوانی گردیده و منجر به خاموش شدن پایگاه داده خواهد شد [۱۰].

۴.۶. حملات استنتاج

اهداف: یافتن پارامترهای قابل تزریق، استخراج داده، تعیین قالب پایگاه داده.

[1] Piggy-Backed Queries

توضیح: در مواقعی که تنظیمات پایگاه‌داده به اندازه‌ی کافی امن و مناسب پیکربندی شده است، اطلاعات خطاها به مهاجم نشان داده نمی‌شود. در حملات استنتاج[1] مهاجم دستور موردنظرخود را تزریق می‌کند و رفتار نرم‌افزار در پاسخ به عملیات نرمال و حملات مختلف را بررسی می‌کند. با توجه به تغییر در رفتار نرم‌افزار اطلاعات مورد نیاز خود را استنتاج می‌کند. دو نوع مشهور تکنیک حمله از این نوع وجود دارد: تزریق کورکورانه[2] و حملات زمان‌بندی‌شده[3].

- تزریق کورکورانه: در این نوع تکنیک، اطلاعات با توجه به رفتار نرم‌افزار (وب‌سایت) در جواب به عبارات پرس‌وجوی همواره درست/نادرست از سرور استنتاج شود. در این حالت اگر ساختار تزریق شده منجر به جواب درست شود، نرم-افزار به عملکرد نرمال خود ادامه می‌دهد. اگر منجر به جواب نادرست شود، حتی با وجود عدم نمایش پیغام خطا، عملکرد نرم‌افزار مثل حالت نرمال نخواهد بود.
- حملات زمان‌بندی‌شده: مهاجم در این نوع حمله از طریق مشاهده‌ی تأخیرهای زمانی پاسخ گویی پایگاه‌داده، اطلاعات مورد نیازش را استنتاج می‌کند. در این گونه حملات مهاجم پرس‌وجوی تزریق‌شده خود را به فرم if-then وارد می‌کند به نحوی که گزاره‌ی آن براساس یک جنبه ناشناخته از پایگاه‌داده، به مسیر خاصی منشعب شود و در آن شاخه یک عملیات زمان‌گیر انجام شود. از طریق اندازه‌گیری افزایش یا کاهش زمان پاسخگویی، مهاجم متوجه می-شود که شرایط موجود در گزاره درست یا نادرست بوده است.

مثال: در اینجا دو نمونه حمله روی نرم‌افزار شکل ۱ مطرح می‌شود. حمله اول با هدف تعیین پارامتر قابل تزریق توسط یک حمله کورکورانه صورت گرفته است. به عنوان مثال یک بار عبارت legalUser’ and 1=0 -- و بعداً عبارت legalUser’ and 1=1 -- را در پارامتر ورودی *login* وارد می‌کند. دو عبارت زیر نتیجه این تزریق خواهد بود.

SELECT accounts FROM users WHERE login=’legalUser’ and 1=0 -- ’ AND pass=’’ AND pin=0
SELECT accounts FROM users WHERE login=’legalUser’ and 1=1 -- ’ AND pass=’’ AND pin=0

اگر نرم‌افزار به درستی پارامترهای ورودی را ارزیابی کند، هر دو عبارت فوق منجر به خطای عدم امکان ورود به نرم‌افزار می‌شوند و مهاجم می‌فهمد که فیلد *login* قابل تزریق نیست. اگر نرم‌افزار آسیب‌پذیر باشد چون عبارت اول شامل گزاره همواره نادرست است اجرای آن منجر به خطای ورود می‌شود. با اجرای عبارت دوم اگر خطای ورود ظاهر نشد متوجه می-شود که پارامتر *login* قابل تزریق است و می‌تواند از آن در حملات خود سوءاستفاده نماید.

در ادامه، حمله دوم از نوع زمانبندی‌شده صورت می‌گیرد. مهاجم در این حمله قصد دارد نام یکی از جداول پایگاه‌داده را پیدا کند. پس عبارت زیر را در پارامتر آسیب‌پذیر *login* وارد می‌نماید.

legalUser’ AND ASCII(SUBSTRING((select top 1 name from sysobjects),1,1))=65 WAITFOR 5 --

درنتیجه عبارت پرس‌وجویی به فرم ذیل در بدنه نرم‌افزار تولید خواهد شد.

SELECT accounts FROM users WHERE login=’legalUser’
AND ASCII(SUBSTRING((select top 1 name from sysobjects),1,1)) = 65 WAITFOR 5 -- ’
AND pass=’’ AND pin=0

در این حمله تابع SUBSTIRNG کاراکتر اول از نام اولین جدول پایگاه‌داده را برمی‌گرداند و تابع ASCII کد اسکی آن را می‌دهد. زمانی که شرط گزاره بررسی می‌شود اگر مقدار این کاراکتر برابر با ۶۵ یعنی کد اسکی کاراکتر A باشد، تابع WAITFOR اجرا شده و ۵ ثانیه تاخیر در پاسخگویی نرم‌افزار ایجاد می‌نماید [۱۱].

[1] Inference Attacks
[2] Blind Injection
[3] Timing Attacks

۴.۷. کدگذاری جایگزین

اهداف: تشخیص راه‌های گریز.

توضیح: در حملات کدگذاری جایگزین[1]، رشته تزریق‌شده به نحوی تغییر می‌یابد که توسط راهکارهای دفاعی و تکنیک‌های خودکار فیلترینگ در نرم‌افزار، به عنوان حمله تشخیص داده نشود. این نوع حمله در ترکیب با انواع دیگر حمله صورت می‌گیرد.

مثال: فرض کنید عبارت زیر در فیلد ورودی *login* تزریق شده است.

legalUser';exec(char(0x73687574646f776e))--

درنتیجه ساختار تولیدشده توسط نرم‌افزار به صورت زیر می‌باشد.

SELECT accounts FROM users WHERE login='legalUser'; exec(char(0x73687574646f776e)) --
AND pass='' AND pin=

تابع CHAR به عنوان پارامتر ورودی عدد صحیح یا کد مبنای شانزده یک سری کاراکتر را دریافت می‌کند و معادل آن کاراکتر را برمی‌گرداند. دنباله اعداد در بخش دوم تزریق، کد اسکی مربوط به رشته SHUTDOWN می‌باشد. بنابراین، هنگامی که پرس‌وجو توسط پایگاه‌داده اجرا می‌شود، روال ذخیره‌شده "SHUTDOWN" فراخوانی می‌گردد [۱۰].

۵. پیشگیری از حملات تزریق اس کیو ال

به منظور جلوگیری از حملات تزریق اس کیو ال رویکردهای مختلفی را می‌توان در پیش گرفت که در این بخش برخی از آنها برشمرده شده است.

۵.۱. شیوه‌های کدنویسی دفاعی

ریشه آسیب‌پذیری در برابر تزریق اس کیو ال، اعتبارسنجی ناکافی ورودی‌ها است. پس اولین و موثرترین راهکار مقابله، استفاده از شیوه‌های برنامه‌نویسی دفاعی می‌باشد. برخی از اقدامات مهم در ادامه آورده شده است [۱۱].

- کنترل نوع ورودی‌ها: حملات تزریق اس کیو ال توسط تزریق یک رشته در پارامترهای ورودی عددی یا رشته‌ای صورت می‌گیرد که با کنترل نوع و فیلترینگ ورودی، بسیاری از حملات ساده قابل کشف است.
- رمزنگاری ورودی‌ها: تزریق در یک پارامتر رشته‌ای اغلب توسط استفاده از کاراکترهایی صورت می‌گیرد که پارسر اس کیو ال را فریب دهد تا ورودی کاربر را به عنوان یک ساختار اس کیو ال تفسیر کند. پس می‌توان استفاده از این کاراکترها را در ورودی محدود کرد. راه حل بهتر استفاده از توابعی به منظور رمزنگاری ورودی کاربر است به گونه‌ای که توسط پارسر اس کیو ال مانند کاراکترهای معمولی با آنها رفتار شود.
- تطابق الگوی مثبت: از آنجایی که تمام حالات ورودی‌های مخرب قابل پیش‌بینی نیست، نمی‌توان تابع جامعی برای تطابق الگوهای منفی داشت. ولی هر برنامه‌نویس حالات درست مقادیر ورودی را می‌داند و می‌تواند روتین‌هایی برای تشخیص این الگوهای مثبت، تعبیه نماید.

[1] Alternate Encoding

- شناسایی تمام منابع ورودی: هر منبع ورودی می‌تواند راهی برای تزریق باشد پس باید تمام نقاط ورودی بررسی شوند و نقطه‌ای از قلم نیفتد. بسیاری از آسیب‌پذیری‌ها ناشی از فراموش‌کردن کنترل یک نقطه از کد است.

با اینکه شیوه‌های برنامه‌نویسی دفاعی بهترین راه مقابله با حملات تزریق اس کیو ال است ولی تحقیقات نشان داده است با توجه به اینکه در عمل این شیوه‌ها مبتنی بر مهارت انسان است، همواره مستعد خطا بوده و به طور کامل قابلیت خودکارسازی را ندارند [۱۲].

۵.۲. تکنیکهای تشخیص و جلوگیری از حمله

در این بخش راهکارهای کلی جلوگیری از حملات تزریق اس کیو ال که در مقالات مختلف ارائه شده، ذکر گردیده است.

- آنالیز ایستای کد: تمرکز متدهای آنالیز ایستا به اعتبارسنجی نوع ورودی‌های کاربر به منظور کاهش شانس حملات تزریق اس کیو ال می‌باشد و نه تشخیص آنها. پس در حالتی که حمله دارای ساختار مناسب و تطابق نوع است، نمی‌تواند حمله را تشخیص دهد. JDBC-Checker تکنیکیست برای چک کردن آماری درستی نوع عبارت‌هایی که به صورت پویا در بدنه نرم‌افزار ساخته می‌شوند [۱۳]. این تکنیک می‌تواند برای جلوگیری از حملاتی که از عدم تطابق نوع بهره می‌برند، استفاده شود.
- آنالیز پویا: در آنالیز پویا، پاسخ‌های نرم‌افزار تحت وب آنالیز می‌شود یعنی هر نوع ورودی به نرم‌افزار ارسال شده و پاسخ نرم‌افزار به آن دریافت می‌گردد. لذا در آنها نیاز به کد حملات از پیش تعریف شده می‌باشد. مثلاً Sania [۱۴] عبارات اس کیو ال نرمال را بین کاربر و نرم‌افزار وب و بین نرم‌افزار وب و پایگاه‌داده جمع‌آوری می‌کند و آسیب-پذیری‌ها را می‌یابد. سپس کدهای حملات تزریق اس کیو ال را که آسیب‌پذیری‌ها را آشکار می‌کنند، تولید می‌نماید و با آنها اقدام به حمله کرده و عبارات اس کیو ال حاصل را جمع می‌کند. درخت پارس عبارات نرمال و عبارات حاصل از حملات را با هم مقایسه و بررسی می‌نماید و مشخص می‌کند کدام حملات موفق می‌شوند و کدام خیر.
- ترکیب آنالیز پویا و ایستا: در این شیوه از فواید هر دو نوع آنالیز پویا و ایستا استفاده می‌شود. به طوری که ساختار صفحات وب برای یافتن آسیب‌پذیری‌ها آنالیز شده و همزمان عبارات اس کیو ال برای تست کردن نتایج نیز تولید می‌گردد. AMNESIA تکنیکیست که در فاز ایستا، تمام عبارات ممکن که نرم افزار می‌تواند در نقاط مختلف دسترسی به پایگاه‌داده تولید کند را با آنالیز ایستا به صورت آتاماتای متناهی مدلسازی می‌نماید [۱۵]. در فاز پویا، قبل از ارسال عبارت‌ها به پایگاه‌داده، ابتدا هر عبارت را با مدل‌های ساخته‌شده در فاز ایستا مقایسه می‌کند. آنهایی که مدل را نقض می‌کنند به پایگاه‌داده ارسال نشده و به عنوان حمله تشخیص داده می‌شوند.

دو رویکرد SQL Guard [۱۶] و SQL Check [۱۷] نیز به طور مشابه ساختار پرس‌وجوها را در زمان اجرا با مدل مدنظر مقایسه می‌کنند با این تفاوت که مدل توسط درخت پارس یا گرامرهایی که فقط پرس‌وجوهای مجاز را می‌پذیرد، بیان می‌شود.

Lee و همکارانش روش ساده‌ای را برای تشخیص حملات SQLI ارائه داده‌اند که با حذف مقادیر صفات موجود در گزاره پرس‌وجوها، فرایند تشخیص حمله را انجام می‌دهد [۱۸]. در بخش آنالیز ایستا، تمام پرس‌وجوهای موجود در کد صفحات وب استخراج شده و مورد پردازش اولیه قرار می‌گیرند و مقادیر صفات از عبارت پرس‌وجو حذف می‌گردد. لیستی از پرس‌وجوهای پردازش‌شده برای استفاده در زمان اجرای نرم‌افزار نگه داشته می‌شود. در زمان اجرا عبارات پرس‌وجوهایی که با ورودی‌های کاربر تکمیل شده‌اند به روش مشابه در فاز ایستا، مورد پردازش قرار گرفته و مقادیر صفات از آنها حذف می‌گردد. حال عبارت پرس‌وجوی حاصله در زمان اجرا با لیست پرس-وجوهای فاز آنالیز ایستا مقایسه می‌شود و اگر با الگوی پرس‌وجوی متناظرش قبل از اجرا، مطابقت نداشته باشد، به عنوان حمله تشخیص داده می‌شود.

- رویکردهای مبتنی بر آلوده‌شدن[1]: WEBSSARI خطاهای مربوط به اعتبارسنجی ورودی‌های کاربر را با استفاده از آنالیز جریان اطلاعات[2] می‌یابد [۱۹]. در این رویکرد از آنالیز ایستا برای کنترل جریان‌های آلوده برای توابع حساس با توجه به برخی پیش‌شرط‌ها استفاده شده است. آنالیزور نقاطی را که در آنها پیش‌شرط‌ها ملاقات نمی‌شوند، یافته و به صورت خودکار فیلترها و عملیات پاکسازی مناسب را به نرم‌افزار اضافه می‌کند تا این پیش‌شرط‌ها را ارضاء نماید. اشکال اصلی این روش چنین است که فرض می‌کند پیش‌شرط‌های کافی برای توابع حساس بیان شده‌اند و همچنین فرض می‌کند ارسال ورودی به یک نوع خاص فیلتر، برای رفع آلودگی ورودی کافیست؛ که در بسیاری از نرم افزارهای واقعی چنین فرض‌هایی قابل تصور نیست.
- الگوهای جدید توسعه عبارات پرس‌وجو: این رویکردها از بسته‌بندی[3] عبارات به عنوان یک راه ارتباط امن و سالم با پایگاه‌داده استفاده می‌کنند. این تکنیک‌ها به وسیله تغییر در رویه ایجاد عبارات اس کیو ال از حملات تزریق اس کیو ال جلوگیری می‌نمایند. به عنوان مثال می‌توان استفاده از تکنولوژی ORM [۲۰] را در پیاده‌سازی سیستم‌های شیئ‌گرای مبتنی بر پایگاه‌داده نام برد. سیستم Safe Query Objects [۲۱] از این گونه رویکرد پیروی کرده است. مشکل چنین شیوه‌هایی در این است که توسعه‌دهنده نرم‌افزار باید این الگوهای جدید برنامه‌نویسی و توسعه عبارات را بیاموزند. همچنین چون بر الگوهای جدید برنامه‌نویسی تاکید دارند، امکان محافظت از نرم‌افزارهای قدیمی را تامین نمی‌کنند.
- سیستم‌های تشخیص نفوذ[4] (IDS): سیستم‌های IDS معمولاً بر اساس تکنیک‌های یادگیری ماشینی کار می‌کند. این تکنیک‌ها روی نمونه‌های عبارات اس کیو ال موجود در نرم افزارها آموزش می‌بینند و مدل عبارات را می‌سازند. در زمان اجرای نرم‌افزار، عباراتی را که با مدل مطابقت نداشته باشند، تشخیص می‌دهند. این سیستم‌ها حملاتی را که با نرخ بالا رخ می‌دهند، به خوبی تشخیص می‌دهد [۲۲]. ولی مشکل اصلی این رویکرد نیاز آن به مجموعه آموزشی قوی می‌باشد و اگر مجموعه آموزشی آن ضعیف باشد، هیچ تضمینی در تشخیص حملات وجود نخواهد داشت.
- فیلترهای پروکسی: یک سیستم فیلتر پروکسی، مجموعه‌ای از قوانین اعتبارسنجی ورودی را روی جریان داده‌های نرم‌افزار تحت وب اعمال می‌کند. این قوانین و محدودیت‌ها توسط توسعه‌دهنده نرم‌افزار مشخص می‌شود که باید روی پارامترهای ورودی‌ای که از کاربر به پایگاه‌داده جریان دارند، اعمال گردد. مشکل این روش وابستگی به انسان است. توسعه‌دهندگان باید علاوه بر شناسایی داده‌هایی که نیاز به فیلتر دارند، الگوها و فیلترهای مناسب را نیز بشناسند. Security Gateway [۲۳] یکی از سیستم‌هاییست که بر این مبنا ارائه شده است.
- آرایش تصادفی مجموعه دستورات: این متدها، مقادیر تصادفی را در ساختارهای عبارات اس کیو ال وارد می‌کنند. این روش به توسعه دهنده اجازه می‌دهد تا عبارات را با استفاده از آرایش تصادفی از دستورات به جای کلمات کلیدی نرمال اس کیو ال بسازد. به عنوان مثال می‌توان قرارداد کرد که درهنگام ایجاد عبارات پرس‌وجو، در انتهای تمام کلمات کلیدی SQL یک عدد تصادفی مثل ۵ قرار داد. مثل عبارت زیر.

SELECT5 * FROM5 users WHERE5 login=? AND5 pass=? AND5 pin=?

در این صورت کدهایی که توسط مهاجم تزریق می‌شود منجر به خطا خواهند شد. SQLrand [۲۴] سیستمیست که از آرایش تصادفی دستورات استفاده می‌کند.

[1] Taint-Based
[2] Information Flow
[3] Encapsulation
[4] Intrusion Detection Systems (IDS)

۶. ارزیابی و مقایسه تکنیک‌ها با توجه به نوع حمله

در این بخش خلاصه‌ای از مقایسه تکنیک‌های معرفی‌شده در بخش قبل، آورده شده است. برای مقایسه، تکنیک‌ها در دو گروه تقسیم شده‌اند: تکنیک‌های بازدارنده و تکنیک‌های تشخیص. تکنیک‌های بازدارنده آنهایی هستند که به تشخیص نقاط آسیب‌پذیر در کد می‌پردازند، یک الگوی توسعه متفاوت برای ایجاد پرس و جو های اس کیو ال در برنامه‌های کاربردی پیشنهاد می‌کنند، یا یک سری مراقبت‌ها به برنامه کاربردی می‌افزایند تا قابلیت‌های کدگزاری دفاعی را تقویت کنند. تکنیک‌های تشخیص، آنهایی هستند که حملات را در زمان اجرا شناسایی می‌کنند.

برای مقایسه روش‌ها به بررسی این موضوع می‌پردازیم که هر تکنیکی قادر به پردازش کدام نوع از حملات است. جداول شماره ۱ و ۲ نتایج ارزیابی را به طور خلاصه نشان می‌دهند. در این جداول از چهار علامت استفاده شده است که نشان می‌دهید هر تکنیک در مقابل یک نوع حمله چگونه عمل می‌کند. علامت "●" نشان می‌دهد یک تکنیک تمام حملات از یک نوع خاص را با موفقیت متوقف می‌کند. علامت "×" نمایانگر شکست یک تکنیک در مقابل حملات از یک نوع خاص است. علامت " ○" نشان‌دهنده تکنیکیست که می‌تواند حمله در نظر گرفته شده را پردازش کند، اما هیچ تضمینی برای پردازش تمام حملات از آن نوع نمی‌دهد. علامت " - " نشانگر تکنیکی است که فقط نیمی از حملات در نظر گرفته شده را پردازش می‌کند.

نیمی از تکنیک‌های بازدارنده، انواع حملات بررسی‌شده را به طور موثر پردازش می‌کنند و بعضی از تکنیک‌ها تا قسمتی موثر واقع می‌شوند. JDBC-Cecker فقط درمقابل زیرمجموعه‌ای از حملات SQLI پاسخگوست؛ زیرا JDBC-Checker خطاهای مربوط به نوع را که موجب آسیب‌پذیری تزریق اس کیو ال می‌شوند، تشخیص می‌دهد. اما خطاهای مرتبط با نوع، فقط یکی از چند دلیل آسیب‌پذیری‌های تزریق اس کیو ال است. پس نمی‌تواند تمام منابع و عوامل تزریق را تشخیص دهد. اما به طور کلی تکنیک‌های بازدارنده عملکرد خوبی دارند زیرا اکثر آنها شیوه‌های کدگذاری دفاعی را وارد مکانیزم بازدارندگی خود کرده‌اند.

جدول ۱) مقایسه تکنیک‌های پیشگیری از حمله با توجه به نوع حمله

تکنیک	درست‌نما	پرس‌وجوهای نادرست	پرس‌وجوهای سوارشده	پرس‌وجوهای اجتماع	روال ذخیره-شده	حملات استنتاج	کدگذاری جایگزین
JDBC-Checker (Gould et al, 2004)	-	-	-	-	-	-	-
Safe Query Objects (Cook and Rai, 2005)	●	●	●	●	×	●	●
Security Gateway (Scott and Sharp, 2002)	-	-	-	-	-	-	-
WebSSARI (Huang et al, 2004)	●	●	●	●	●	●	●

جدول شماره ۲ مقایسه تکنیک‌های تشخیص حملات را نشان می‌دهد. بیشتر تکنیک‌های تشخیص‌دهنده به طور یکنواخت روی همه انواع حملات موثر هستند. فقط دو نوع حمله روال‌های ذخیره‌شده و کدگذاری جایگزین برای اکثر تکنیک‌ها چالش‌زاست. در مورد رویه‌های ذخیره‌شده، کدی که ایجاد کننده پرس‌وجو است روی پایگاه داده ذخیره و اجرا می شود. درحالی که اغلب تکنیک‌ها فقط روی پرس‌وجوهای موجود در برنامه کاربردی متمرکز هستند. توسعه تکنیک‌ها به منظور

پوشش دادن پرس‌وجوهای تولیدشده و اجرا شده در پایگاه‌داده کار ساده‌ای نیست. به این دلیل حملاتی که براساس روال های ذخیره‌شده هستند، برای بسیاری از تکنیک‌ها مشکل‌ساز می‌باشند.

پردازش حملاتی که براساس کدگذاری جایگزین هستند نیز دشوار است. تنها چهار تکنیک AMNESIA، SQLCheck، SQLGuard و Attributes Removing صراحتاً به این نوع حملات پرداخته‌اند. دلیل موفقیت سه تکنیک اول استفاده از پارسر پایگاه‌داده برای تفسیر یک رشته پرس‌وجو دقیقاً به همان شیوه‌ای که پایگاه‌داده عمل می‌کند، می-باشد [۱۵]. تکنیک‌های دیگری که در این گروه از رتبه خوبی برخوردار هستند یا تکنیک‌های مبتنی بر توسعه‌دهنده هستند و یا تکنیک‌هایی هستند که به وسیله یک API استاندارد مشکل را حل می‌کنند.

جدول ۲) مقایسه تکنیک‌های تشخیص حمله با توجه به نوع حمله

تکنیک	درست‌نما	پرس‌وجوهای نادرست	پرس‌وجوهای سوارشده	پرس‌وجوهای اجتماع	روال ذخیره-شده	حملات استنتاج	کدگذاری جایگزین
AMNESIA (Halfond and Orso, 2005)	●	●	●	●	×	●	●
IDS (Valeur et al, 2005)	○	○	○	○	○	○	○
SQLCheck (Su and Wassermann, 2006)	●	●	●	●	×	●	●
SQLGuard (Bueher et al, 2005)	●	●	●	●	×	●	●
SQLrand (Boyd and Keromytis, 2004)	●	×	●	●	×	●	×
Removing Attributes (Lee et al, 2012)	●	●	●	●	●	●	●

۷. بحث و نتیجه‌گیری

در این مقاله، بررسی و مقایسه‌ای از تکنیک‌های موجود برای شناسایی و جلوگیری از حملات تزریق اس کیو ال معرفی گردید. برای انجام این ارزیابی، انواع مختلف حملات تزریق شناخته شده معرفی گردید. سپس توانایی تکنیک‌های متفاوت برای شناسایی و یا جلوگیری از چنین حملاتی مورد ارزیابی قرار گرفت. در نهایت، تکنیکهای پیشگیری و تشخیص از نظر شناسایی خودکار حملات و ملزومات استقرار در محیط عملیاتی نیز مورد بررسی قرار گرفت.

مطالعات نشان‌داده است عملکرد اغلب تکنیک‌های تشخیص و پیشگیری در مقابل اکثر انواع حملات قابل قبول بوده است و فقط حملات روال‌های ذخیره‌شده و کدگذاری جایگزین تکنیک‌ها را دچار مشکل نموده‌اند. چون کد مربوط به روال‌های ذخیره‌شده در سمت پایگاه‌داده است، تکنیک‌ها دسترسی به آن ندارند تا بتوانند کنترل مناسبی بر آن اعمال نمایند. از اینرو برخی مهاجمین از کدنویسی ضعیف روال‌های ذخیره‌شده به عنوان یک مزیت در حمله‌ی خود استفاده می-کنند. درمورد حملات از نوع کدگذاری جایگزین، چون چنین حملاتی ماهیت ابتکاری داشته و تمام حالات کدگذاری مخرب قابل پیش‌بینی نیست، لذا تکنیک‌ها در پیشگیری از این نوع حمله نیز ضعف دارند.

در این مقاله به مقایسه معیارهای کمی از جمله دقت عملکرد تکنیک‌های مختلف پرداخته نشد. متاسفانه در بسیاری از موارد دسترسی به پیاده‌سازی تکنیک‌ها وجود نداشت و همچنین در مقالات مختلف معیارهای متفاوتی برای

ارزیابی وجود داشت که اجماع آنها را دشوار می‌نمود. لذا در اینجا به مقایسه کلی پرداخته شد و بررسی‌های کمی به کارهای آینده موکول گردید.

۸. منابع

1. Spacey, J., (2011) *Big List of Information Security Vulnerabilities*, http://simplicable.com/new/the-big-list-of-information-security-vulnerabilities.

2. OWASP: *Top Ten Most Critical Web Application Vulnerabilities*, (2013). https://www.owasp.org/index.php/Top10#OWASP_Top_10_for_2013.

3. Imperva, *Top Ten Database Threats, The Most Significant Risks and How to Mitigate Them*, Imperva's Application Defense Center white paper, (2014). www.imperva.com/docs/wp_topten_database_threats.pdf.

4. Chopra, A., & Kaufman, M. (2014). *Rise in Web Application Intrusions* 2013-2014.

5. Wikipedia, *SQL Injection: Examples*, (Retrieved Aug, 2015). https://en.wikipedia.org/wiki/SQL_injection.

6. Anley, C., (2002) *Advanced SQL Injection In SQL Server Applications*. White paper, Next Generation Security Software Ltd.

7. Srivastava, M. (2014, March). *Algorithm to prevent back end database against SQL injection attacks*. In Computing for Sustainable Global Development (INDIACom), 2014 International Conference on (pp. 754-757). IEEE.

8. Appelt, D., Alshahwan, N., & Briand, L. (2014). *Assessing the Impact of Firewalls and Database Proxies on SQL Injection Testing*. In Future Internet Testing (pp. 32-47). Springer International Publishing.

9. Wong, S., V., (2013) *How to prevent SQL Injection in Stored Procedures*, (Retrieved Aug 2015), http://www.codeproject.com/Tips/586207/How-to-prevent-SQL-Injection-in-Stored-Procedures..

10.Labs, S., (2002) *SQL Injection*. White paper, SPI Dynamics, Inc.. http://www.spidynamics.com/assets/documents/WhitepaperSQLInjection.pdf.Srivastava, M. (2015, Aug).

11.Halfond, W. G., Viegas, J., & Orso, A. (2006, March). *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.

12.Livshits, V. B., and Lam, M. S., (Aug. 2005) *Finding Security Errors in Java Programs with Static Analysis*. In Proceedings of the 14th Usenix Security Symposium, pages 271Ð286.

13.Gould, C., Su, Z., & Devanbu, P. (2004, May). *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.

14.Kosuga, Y., Kernel, K., Hanaoka, M., Hishiyama, M., & Takahama, Y. (2007, December). *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.

15.Halfond, W. G., & Orso, A. (2005, November). *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.

16.Buehrer, G., Weide, B. W., & Sivilotti, P. A. (2005, September). *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.

17.Su, Z., & Wassermann, G. (2006, January). *The essence of command injection attacks in web applications*. In ACM SIGPLAN Notices (Vol. 41, No. 1, pp. 372-382). ACM.

18.Lee, I., Jeong, S., Yeo, S., & Moon, J. (2012). *A novel method for SQL injection attack detection based on removing SQL query attribute values*. Mathematical and Computer Modelling, 55(1), 58-68.

19.Huang, Y. W., Yu, F., Hang, C., Tsai, C. H., Lee, D. T., & Kuo, S. Y. (2004, May). *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.

20.O'Neil, E. J. (2008, June). *Object/relational mapping 2008: hibernate and the entity data model (edm)*. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data (pp. 1351-1356). ACM.

21.Cook, W. R., & Rai, S. (2005, May). *Safe query objects: statically typed objects as remotely executable queries*. In Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on (pp. 97-106). IEEE.

22.Valeur, F., Mutz, D., & Vigna, G. (2005). *A learning-based approach to the detection of SQL attacks*. In Detection of Intrusions and Malware, and Vulnerability Assessment (pp. 123-140). Springer Berlin Heidelberg.

23.Scott, D., & Sharp, R. (2002, May). *Abstracting application-level web security*. In Proceedings of the 11th international conference on World Wide Web (pp. 396-407). ACM.

24.Boyd, S. W., & Keromytis, A. D. (2004, January). *SQLrand: Preventing SQL injection attacks*. In Applied Cryptography and Network Security (pp. 292-302). Springer Berlin Heidelberg.