ششمین کنفرانس بین المللی انجمن رمز ایران
۱۵ و ۱۶ مهر ماه ۱۳۸۸، دانشگاه اصفهان
انجمن رمز ایران
دانشگاه اصفهان

تشخیص آسیب‌پذیری تزریق SQL براساس راهکاری مستقل از فناوری توسعه وب

احسان اعرابی[۱]، مهدی برنجکوب[۲]، محمد علی منتظری[۲]

[۱]مرکز تخصصی آپا دانشگاه صنعتی اصفهان
e.aerabi@ec.iut.ac.ir
[۲]دانشگاه صنعتی اصفهان
brnjkb@cc.iut.ac.ir, montazer@cc.iut.ac.ir

چکیده

با افزایش بکارگیری برنامه‌های کاربردی تحت وب، مسئله امنیت اطلاعات در این زمینه از اهمیت بیشتری برخوردار شده است. یکی از مهم ترین حمله‌هایی که امنیت برنامه‌های کاربردی تحت وب را تهدید می کند، حمله به پایگاه داده‌ها است. گروه عمده‌ای از این حملات، با نام تزریق SQL شناخته شده‌اند. در این مقاله به ارائه راهکاری برای تشخیص آسیب پذیری تزریق SQL پرداخته می شود که نسبت به روش‌های قبلی، برتری‌هایی دارد. دراین روش از دو پروکسی، یکی در جلوی کارگزار وب و دیگری در جلوی پایگاه داده استفاده شده است. پروکسی اول به درهم ریزی پارامترهای درخواست http و پروکسی دوم به بازگشایی آن‌ها می پردازد. مهم ترین برتری این روش، مستقل بودن از زبان و فناوری توسعه وب است. از این رو نیاز به تغییر کدبرنامه در آن نیست. این روش، تمامی حملات تزریق SQL را پوشش داده و نیاز به مرحله یادگیری ندارد.

واژه‌های کلیدی

آسیب‌پذیری تزریق SQL، اعتبار سنجی ورودی، امنیت وب

۱- مقدمه

گسترش استفاده از اینترنت و وب در امور روزمره، باعث افزایش سرویس‌های وب در زمینه‌های مختلف شده است. این سرویس‌ها عموما با داده‌هایی سروکار دارند که برای سازماندهی و سهولت بازیابی آن‌ها، در پایگاه داده‌ها ذخیره می شوند. بنابراین امروزه پایگاه‌های داده در پشت سر سرویس‌های بسیاری در وب قرار گرفته‌اند. این ارتباط باعث بوجود آمدن تهدیدی به نام تزریق SQL [۱] شده است.

تزریق SQL زمانی اتفاق می افتد که برنامه کاربردی تحت وب که به زبان‌هایی مانند PHP، ASP،Java و ... نوشته شده است، بر روی ورودی‌های کاربر، اعتبار سنجی[۲] مناسبی نداشته باشد. عدم اعتبارسنجی ورودی-هایی که در تولید یک پرس وجو[۳] بکارگرفته می شوند، باعث این آسیب‌پذیری می شود. از آنجا که این پرس و جوها توسط پایگاه‌های داده (برای بازیابی اطلاعات) اجرا خواهندشد نفوذگرها با اِعمال ورودی‌های ماهرانه، سعی در اجرای نادرست برنامه و تحقق حملاتی مانند دور زدن مکانیزم‌های احراز اصالت، افشا و یا تغییر اطلاعات، از کار اندازی سرویس کنند.[۱]

تزریق SQL، ۱۰% از کل جرائم کامپیوتری را از سال ۲۰۰۲ تا به حال، تشکیل می دهد[۳]. موسسه NIST در پایگاه داده آسیب پذیری‌های خود، از ابتدای سال ۲۰۰۷ تا

[1] SQL Injection

[2] Input Validation

[3] Query

ماه may سال ۲۰۰۹ میلادی، بالغ بر ۲۰۰۰ مورد آسیب‌پذیری تزریق SQL را در برنامه‌های مهم کاربردی تحت وب گزارش کرده است[۳] . Michael Sutton در بررسی به‌عمل آمده توسط موتورهای جستجو نشان می دهد که حدود ۱۱% از وب سایت‌های مورد جستجو، مستعد بروز حمله تزریق وب هستند[۴].

در مقابله با این آسیب‌پذیری، روش‌هایی ارائه شده که هر کدام دارای نقاط ضعف و قوت هستند. از جمله نقاط ضعف عمده در این روش‌ها، وابستگی آن‌ها به فناوری توسعه وب است. اکثر این روش‌ها نیازمند تغییر کد منبع برنامه هستند[۱][۲][۷]. در مواردی هم که نیاز به تغییر کد منبع نیست، روش نیازمند یک مفسر است که زبان برنامه نویسی را بشناسد[۱۷] که بکارگیری این مفسر برای زبان‌های توسعه وب دیگر، امکان پذیر نیست. ما در این مقاله به ارائه روشی می پردازیم که نیاز به تغییر یا تفسیر زبان توسعه را از بین می برد و قابل بکارگیری در مکانیزم‌های تشخیص آسیب‌پذیری است.

ادامه این مقاله در بخش ۲، با معرفی پیش زمینه‌ای از حمله و روش‌های کشف و مقابله با آن و نقاط ضعفشان می‌پردازد. سپس در بخش ۳ روش پیشنهادی این مقاله معرفی می‌شود و در نهایت در بخش‌های ۴ و ۵، به نحوه پیاده‌سازی و نتایج بدست پرداخته می‌شود.

۲- پیش زمینه و روش‌های مرتبط

در این بخش، ابتدا یک مثال ساده و پرکاربرد برای حمله تزریق SQL ارائه می‌شود و سپس با روش‌های مطرح تشخیص و مقابله با این حملات و همچنین نقاط ضعف آن‌ها آشنا می‌شویم.

۲-۱ حمله تزریق SQL

در بسیاری از برنامه‌های کاربردی تحت وب، ورودی‌های کاربران، هنگامی‌که به سمت کارگزار وب فرستاده می‌شوند، قسمتی از یک پرس و جوی SQL [1] را تشکیل می‌دهند که سپس توسط DBMS [2] اجرا خواهد شد. برای ارائه یک مثال واقعی، صفحه احراز اصالت یک وب سایت را مطابق شکل ۱ در نظر می‌گیریم.

صفحه احراز اصالت شکل ۱، ۲ فیلد دارد: یکی نام کاربری و دیگری رمز عبور. کاربر بعد از پرکردن این دو فیلد و ارسال آن به سمت سرور می‌تواند خود را احراز اصالت کند. در سمت سرور این دو متغیر در درون یک پرس‌وجو قرار داده می شود، تا از صحت و وجود آن در جدول کاربران (که در پایگاه داده وجود دارد) اطمینان حاصل شود. پرس‌وجوی مذکور در طرف سرور به صورت زیر است:

```
statement = "SELECT * FROM users WHERE
name = '" + Username + "' AND pass = '" +
Password + "';"
```

که در این جا Username همان نام و Password هم رمز عبور وارد شده توسط کاربر خواهد بود. این پرس و جو اجرا می‌شود و اگر کاربری مطابق با نام کاربری و رمزعبور وارد شده در پایگاه داده (و جدول کاربران) موجود باشد، درایه آن کاربر توسط DBMS بازگردانده خواهد شد که‌این خود به معنای احراز اصالت کاربر است و گرنه DBMS جوابی خالی بازخواهد گرداند.

شکل ۱: یک صفحه احراز اصالت مبتنی بر وب

این کد (در صورت عدم اعتبار سنجی مناسب) دارای آسیب پذیری است و نحوه حمله به آن بدین گونه است که اگر کاربر به جای نام کاربری و رمز عبور، عبارت a' or 't' = 't را وارد کند،پرس‌وجو به شکل زیر خواهد شد:

```
statement = "SELECT * FROM users WHERE
name = 'a' or 't' = 't'AND  Pass ='a' or
't' = 't';"
```

اگر چه کاربری به نام 'a' در سیستم موجود نیست، اما به خاطر وجود or و منطق همواره درست 't' = 't'، عبارت بالا توسط DBMS، همواره درست ارزیابی می‌شود و (احتمالا) کل درایه‌های موجود در جدول را به عنوان جواب باز-می‌گرداند که‌این به معنی احراز اصالت شدن کاربر است. یعنی نفوذگر بدون داشتن نام کاربری و رمز عبور صحیح، وارد سیستم شده است و حتی اگر اولین درایه موجود در جدول، کاربر مدیر باشد، احتمالا نفوذگر با دسترسی مدیر، وارد سیستم خواهد شد.

در تمامی حملات تزریق SQL یک اتفاق اساسی وجود دارد که در میان همه حملات مشترک است و آن این است که حملات تزریق SQL زمانی موفق خواهند شد که بتوانند ساختار پرس‌وجو را تغییر بدهند. برای مثال، ساختار درست پرس‌وجوی بالا بدین صورت است:

```
Select * from users where name=* AND
Pass=*;
```

[1] query

[2] Database Management System.

این در حالی است که اگر حمله ذکر شده موفق شود، ساختار پرس‌وجو به صورت زیر تغییر می کند:

```
Select * from users where name=* AND
Pass=* OR *=*;
```

۲-۲ کارهای مرتبط

در این زیربخش یک دسته‌بندی متعارف از روش‌های موجود ارائه خواهد شد. این دسته‌بندی برگرفته و کامل شده‌ی دسته‌بندی SANIA [۵] است. در ضمن هر دسته شامل نقاط ضعفی است که در ادامه ذکر شده است.

- **گزاره‌های آماده**

روش‌هایی که بر این اساس کار می کنند، از امکانی در فناوری توسعه وب بهره می گیرند که گزاره‌های آماده[1] نام دارد. با این امکان، برنامه‌نویس می تواند در هنگام توسعه برنامه وب، ساختار پرس‌وجوی SQL را مشخص کند این ساختار در هنگام اجرای برنامه ثابت است و تنها، مقادیر ورودی را از یک مسیر امن و ساخت یافته دریافت می‌کند و در درون پرس‌و جو می‌گذارد. از آنجا که ساختار پرس‌وجو از قبل مشخص است، دیگر نمی توان آن را تغییر داد. نمونه کاربرد این روش در [۶] آمده است.

نقاط ضعف: برای بکارگیری آن نیاز به شناخت، تفسیر و تغییر کد برنامه است. این مسئله برای برنامه‌های کاربردی که تا قبل از این در حال استفاده‌اند، مشکل است. از طرفی اگر ساختار پرس‌وجو در زمان اجرا مشخص شود (و نه در زمان توسعه)، این روش عملا غیرقابل پیاده‌سازی خواهد شد.

- **روش‌های وابسته به تحلیل ساختار پرس‌و جو**

این روش‌ها، با تحلیل برون‌خطِ کد برنامه و یا اعمال ورودی‌های بی خطر به برنامه، سعی در تشخیص ساختار درست پرس‌وجو دارند. همان طور که گفته شد، حملات تزریق SQL زمانی موفق خواهند شد که بتوانند ساختار پرس‌وجو را تغییر بدهند. این راهکارها، در زمان اجرا، پرس‌وجوهای تولید شده را بررسی خواهند کرد و از مطابقت آن‌ها با ساختاری که در زمان آموزش بدست آورده‌اند، مطمئن می شوند. اگر پرس‌وجویی در زمان اجرا، ساختاری متفاوت با ساختار درست داشته باشند، به عنوان حمله شناخته خواهند شد. AMENSIA [۷] ، به وسیله ماشین غیرقطعی متناهی[2]، سعی در نگه داری و مقایسه پرس و جوهای ایجاد شده توسط برنامه کاربردی دارد. در [۹]، برنامه کاربردی بوسیله یک روش داده کاوی، کاوش می شود تا خصوصیات پرس‌وجوهای آن بازیابی شود که این عمل را profiling می نامد. بعد از این در حین اجرای برنامه، پرس‌وجوی تولید شده در زمان اجرا، باز هم کاوش می‌شود تا از همسان بودن آن با profile تولید شده مطمئن گردد. در غیر این صورت احتمال حمله تزریق SQL وجود دارد. همچنین در [۱۶] که روش این مقاله شباهت‌هایی با آن دارد، کلاسی در جاوا موسوم به SQLGuard بوجود آمده است که استفاده از آن برای وارسی گزاره های SQL، نیاز به تغییر کد منبع برنامه دارد.

نقاط ضعف: این روش‌ها نیازمند شناخت، تفسیر و یا تغییر کد برنامه هستند. از طرفی اگر برنامه کاربردی، بسته به ورودی‌های کاربر، چندین نوع پرس‌وجو با ساختارهای مختلف تولید کند، این روش‌ها، کارایی یا دقت نخواهند داشت.

- **روش‌های یادگیری ماشین**

در [۱۰] و [۱۱] روش‌هایی مبتنی بر یادگیری ماشین ارائه شده است. ماشین، ساختار درست پرس‌وجوی SQL تولید شده را فرامی‌گیرد و با پرس‌وجوهای تولید شده جدید مقایسه می کند. Sienkiewicza و Ordona [۱۲] روشی را ارائه کردند که از شبکه‌های همروند عصبی[3] برای کشف حملات پایگاه داده استفاده می کند.

نقاط ضعف: این روش‌ها نیاز به دوره آموزش مناسب دارند، تا بتوانند در زمان اجرا دقت قابل قبولی داشته باشند. از طرفی دقت آن نیز ، در مواردی که پرس‌وجوهای SQL متغیر و پویا است نیز کمتر خواهد بود.

- **روش‌های تشخیص ناهنجاری**

در [۱۳] روشی ارائه شده است که مدل توزیع نویسه‌های پرس‌وجوی تولید شده در دوره‌ی آموزش را با مدل توزیع نویسه تولید شده در دوره‌ی عملیاتی سیستم مقایسه می کند و با کشف تفاوت در بین این دو، وجود حمله را اعلام می کند.

نقاط ضعف: نقطه ضعف این روش، عدم دقت آن در پوشش انواع حملات تزریق SQL است. دقت آن بیشتر در حملات tautology است.

- **Instruction Set Randomaization**

درروش SQLrand [۱۴]، یک چارچوب برای تولید کنندگان نرم افزار به وجود می آید که به آن‌ها اجازه می دهد که در پرس‌وجوهای خود، لغات کلیدی نرمال را با لغات کلیدی تصادفی جایگزین کنند. یک پروکسی مابین برنامه کاربردی وب و پایگاه داده، این لغات تصادفی را به لغات نرمال تبدیل می کند تا پایگاه داده بتواند آنها را

[1] Prepared Statements

[2] Non-Deterministic Finite State Automaton

[3] Recurrent Neural Networks

تفسیر و اجرا کند. بدین ترتیب حمله‌ها خنثی می شود و باعث بوجود آمدن پرس‌وجوهایی می شود که از لحاظ قواعد نادرست هستند . این روش، بر پایه یک کلید برآمده از تابعی تصادفی بنا شده که لازم است که حمله کننده، به محتوای کلید پی نبرد.

نقاط ضعف: این روش نیاز به شناخت، تفسیر و تغییر کد منبع دارد.

- **روش‌های مبتنی بر جهش**

MUSIC [۱]، روشی است که در آن از تکنیک جهش برای کشف آسیب پذیری تزریق SQL در برنامه کاربردی استفاده می شود.

نقاط ضعف: این روش نیز نیازمند تفسیر و تغییر کدبرنامه است.

- **روش CANDID**

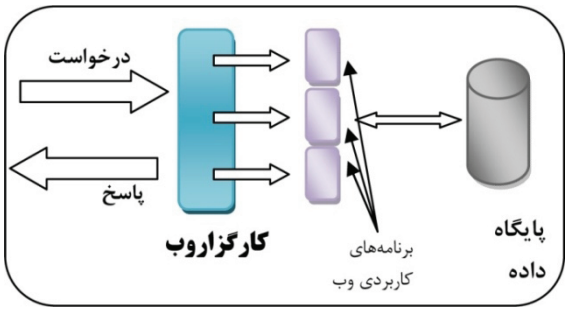
روش CANDID [۱۵] روشی است که در آن، متغیری با همین نام در کد برنامه بوجود می آید. این متغیر، هم زمان با ورودی کاربر، مقداردهی می شود. مقدار این متغیر باید بی خطر و مساوی با تعداد کاراکترهای ورودی کاربر باشد. این متغیر ، سایه به سایه با ورودی کاربر در برنامه قرار داده می‌شود و همه تغییراتی که ورودی کاربر می پذیرد ، بر آن اعمال می‌شود. در انتها، دو پرس‌وجو، با این دو مقدار ساخته می‌شود و باهم مقایسه می شود. اگر ورودی کاربر سعی در به هم زدن ساختار پرس‌وجو داشته باشد، با مقایسه با پرس‌وجوی ساخته شده با متغیر CANDID، که مقداری بی خطر داشت، آشکار می شود.

نقاط ضعف: این روش نیز نیازمند تفسیر و تغییر کدبرنامه است.

۳- روش PARS

در این بخش به روش پیشنهادی این مقاله ارائه می شود که برای جبران برخی از ضعف‌های دیگر روش‌ها، از در هم ریختن[1] پارامترهای کاربر بهره می‌گیرد. به همین خاطر نام آن را PARS[2] نهاده‌ایم. برای تشریح این روش، بازهم نیاز است که یاد آور شویم که حمله تزریق SQL زمانی اتفاق می‌افتد که نفوذگر بتواند با ترفندی ساختار پیش بینی شده و مشخص یک پرس‌وجو را تغییر بدهد. این تغییر در نهایت باعث رفتاری متفاوت با رفتار مورد انتظار از پایگاه داده می شود. که این رفتار می تواند افشای اطلاعات یا از دست رفتن و پاک‌شدن آن و مانند این‌ها باشد.

برای تبیین این روش، ابتدا معماری یک کارگزار وب مجهز به پایگاه داده در شکل ۲ نشان داده شده است.



شکل ۲: معماری کارگزار وب مجهز به پایگاه داده

این معماری، معماری پایه تمامی کارگزاران وب است. درخواست زیر را در نظر بگیرید که به کارگزار وب فرستاده می‌شود:

http://example.com/login.php?username='Ali'&password='abc'

(البته، نام‌های کاربری و رمزعبور معمولا با متد get فرستاده نمی شوند. این مثال جهت سادگی ارائه شده است.)

درخواست توسط کارگزار وب، به برنامه کاربردی که در صفحه login.php وجود دارد، سپرده می شود. برنامه کاربردی نیز از پارامتر username و pass برای پرس‌وجو در پایگاه‌داده استفاده می کند و پرس‌وجو زیر را تولید می کند:

```
statement = "SELECT * FROM users WHEHRE
name= 'Ali' AND  pass ='abc'
```

روش PARS برای تشخیص آسیب پذیری، نیازمند معماری خاصی است که در شکل ۳ دیده می‌شود. این معماری با اضافه‌کردن یک پروکسی وب، در مقابل کارگزار وب و یک پروکسی SQL، در مقابل کارگزار پایگاه‌داده شکل گرفته است.

اکنون به وظائف این دو پروکسی می پردازیم:

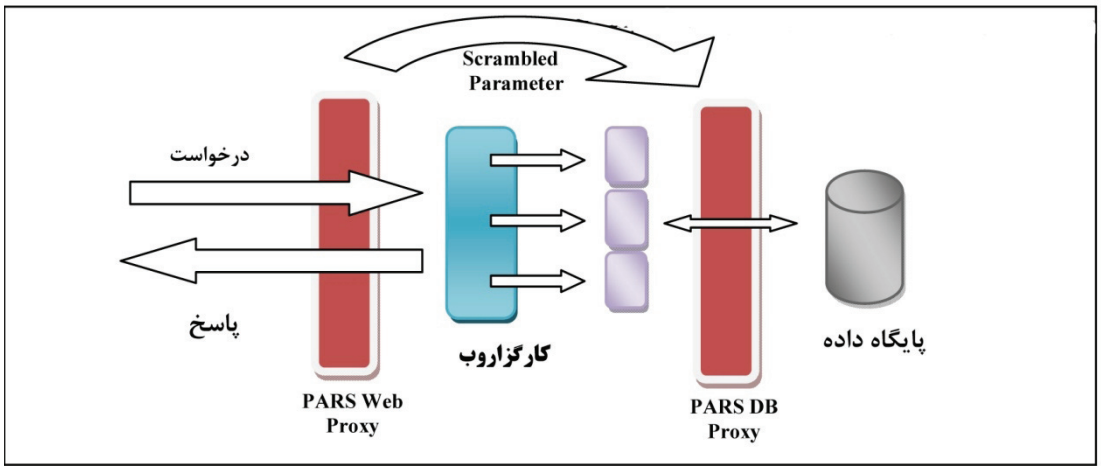
PARS Web Proxy: این پروکسی به محض دریافت یک درخواست، پارامترهای (متغیرهای Http) را درهم می‌ریزد. برای مثال، در نمونه ذکرشده، دو پارامتر username و password به صورت زیر قرار می‌گیرند:

http://example.com/login.php?username='***a&sd2re4%***'&password='***$3da&r!jh6***'

تابع درهم ریزی ، یک تابع قابل بازگشت است و برای کاهش بار محاسباتی می‌تواند یک تابع xor ساده با یک الگوی مشخص، به عنوان کلید باشد.

[1] Scrambling

[2] PARameter Scrambling



شکل ۳: معماری PARS

اکنون پروکسی وب، دو کار دیگر را انجام می دهد:

- درخواست تغییر یافته را برای وب سرور می فرستد.
- لیستی از پارامترهای در هم ریخته را برای پروکسی دیگر یا همان PARS DB Proxy می‌فرستد.

PARS Database Proxy: این پروکسی پارامترهای درهم‌ریخته‌شده را از پروکسی وب گرفته و آن‌ها را در لیستی به نام «*لیست پارامترهای درهم ریخته شده*»، جای می‌دهد. این لیست حاوی نام پارامترها و مقدار درهم‌ریخته-شده‌ی آن‌ها است.

وظیفه اصلی این پروکسی زمانی ایفا می گردد که یک پرس‌وجو برای پایگاه داده فرستاده می‌شود.

این پرس‌وجو منطقا پس از پردازش درخواست‌های http، توسط برنامه کاربردی وب صادر می شود.

در این هنگام، پروکسی پارامترهای درهم ریخته شده‌ی درون پرس‌وجو را در میان پارامترهای درون جدول، جستجو می کند. اگرپارامتری(هایی) را در درون پرس‌وجو و جدول یکسان یافت، سعی می‌کند که پرس‌وجو را با بازگرداندن پارامترها به مقدار اولیه خودشان، باز سازی کند. برای مثال، در نمونه بالا، پروکسی پایگاه داده، از پروکسی وب، این دو پارامتر را دریافت می کند:

username=’***a&sd2re4%***’

password=’***$3da&r!jh6***’

سپس، از برنامه کاربردی، پرس‌وجوی زیر را دریافت می‌کند:

```
statement = "SELECT * FROM users WHEHRE
name= 'a&sd2re4%' AND  pass ='$3da&r!jh6'
```

پروکسی پس از جستجوی جدولی که شامل پارامترهای درهم‌ریخته‌شده است، دو پارامتر username و password را می‌یابد که مقادیرشان در پرس‌وجو بکار رفته است. اکنون پروکسی با بازگرداندن مقدار پارامترها به مقدار اولیه، می تواند پرس‌وجوی اصلی را بازسازی کند. برای نمونه بالا، پرس‌وجو بدین صورت خواهد شد.

```
statement = "SELECT * FROM users WHEHRE
name= 'Ali' AND  pass ='abc'
```

حال پروکسی می تواند بروز حمله را تشخیص دهد. حمله زمانی صورت گرفته است که پرس‌وجوی اصلی با پرس‌وجویی که شامل مقادیر درهم ریخته شده است، ساختاری متفاوت داشته باشد. برای تشخیص و مقایسه ساختارها، از درخت تجزیه [۱۶] استفاده می شود.

اگر درخت تجزیه‌ی ساخته شده از دو پرس‌وجو، باهم متفاوت باشند بدین معنی است که پارامترهایی که کاربر وارد کرده است، شامل مقادیری است که باعث تغییر ساختار پرس‌وجو خواهد شد.

برای مثال، فرض کنید که نفوذگری مقدار زیر را وارد کند:

username=’**Ali**’

Password=’**a’ OR 1=1;--**’

پس درخواست به صورت زیر خواهد شد.

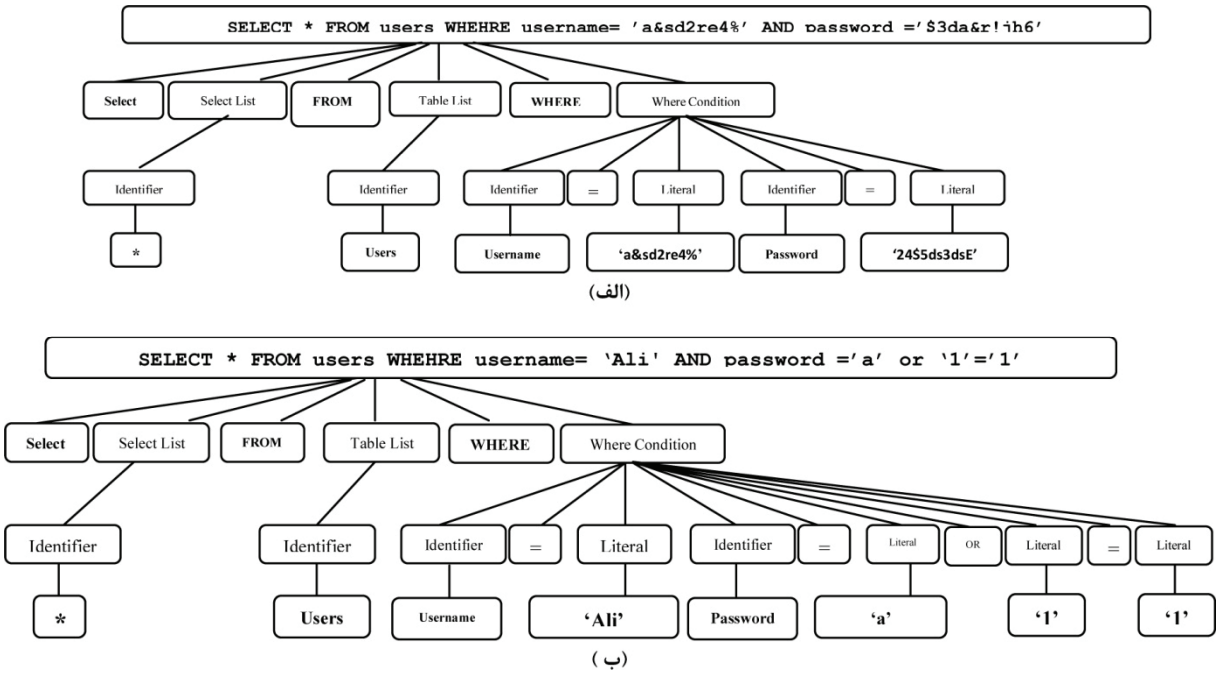
http://example.com/login.php?username=’Ali’&password=’a’ OR 1=1’

پروکسی پایگاه داده، پرس‌وجویی مانند زیر را از برنامه کاربردی دریافت می کند و آن را تجزیه می‌کند(شکل ۴- الف):

```
statement = "SELECT * FROM users WHEHRE
name= 'a&sd2re4%' AND  pass ='24$5ds3dsE'
```

پروکسی پرس‌وجو را به حالت اولیه بازمی‌گرداند و و آن را نیز تجزیه می‌کند(شکل ۴- ب):

```
statement = "SELECT * FROM users WHEHRE
name= 'Ali' AND  pass ='a' OR 1=1;-- '
```



شکل۴: تفاوت ساختاری دو پرس‌وجو با استفاده از درخت تجزیه

استفاده از درخت تجزیه به پروکسی اجازه می دهد که ساختار دو چرس‌وجو را باهم مقایسه کند. همان‌گونه که در شکل ۴ نیز مشهود است، دو پرس‌وجو با هم در تعداد عبارات Where_Conditions متفاوت هستند. جزئیات استفاده از درخت تجزیه برای مقایسه پرس‌وجوها در [۱۶] آمده است.

اکنون پروکسی می تواند بروز حمله را تشخیص دهد، زیرا که ساختار درخت تجزیه‌ی دو پرس‌وجو با هم متفاوت است. این راه حل نیازی به تفسیر یا تغییر کد منبع ندارد و همان طور که در بخش بعدی مقاله خواهیم دید، قابلیت پیاده‌سازی در سناریوهای تشخیص آسیب پذیری ومقابله با نفوذ را دارد.

۴- تحلیل مقایسه‌ای

دراین بخش دو ساختار برای استفاده از روش PARS پیشنهاد خواهد شد. اولین ساختار، استفاده برون خط PARS برای تشخیص آسیب‌پذیری است و دومین ساختار استفاده برخط آن جهت کشف نفوذ.

۴-۱ سیستم برون خط برای تشخیص آسیب پذیری

استفاده برون خط از PARS در زمان توسعه سیستم کاربرد دارد و در حوزه تحلیل پویای[1] سیستم قرار می‌گیرد. این گرایش برخلاف روش های تحلیل ایستا[2]، نیاز به دانستن کد منبع ندارد. بدین ترتیب سیستم مورد آزمون، در مقابل گروهی از مقادیر بدخواه قرار می‌گیرد و به ازای هر مقدار، PARS پاسخی مبنی بر موفق بودن با نبودن حمله صادر می‌کند.

برای تاثیر بیشتر و دقیق‌تر مکانیزم تشخیص آسیب پذیری، باید ملاحظاتی در نحوه پیاده سازی آن مدنظر قرار گیرد. بدین صورت که در بکارگیری این راه حل باید توجه داشت که همه پارامترهایی که توسط کاربر به سمت سرور فرستاده می‌شوند، در پرس‌وجو بکار گرفته نخواهند شد. این بدین معنی است که گروهی از پارامترها برای مقاصد دیگر تولید می شوند. برای مثال، فرض کنید که کاربری در یک صفحه ثبت نام، گزینه‌ای را که مربوط به پذیرش قوانین عضویت[3] است را تایید کند. این گزینه که به صورت یک "*check box*" است، تنها می تواند حاوی مقدار "yes" یا "no" باشد. در هم ریختن این پارامتر، می تواند مقدار آن را، از این دو مقدارمجاز خارج سازد؛ مشخصا سیستم ثبت نام فقط با مقدار yes عمل خواهد کرد و نه هیچ مقدار دیگری. پس این آزمون تشخیص آسیب پذیری، با درهم‌ریختن مقدار check box، فاقد کارایی است، چرا که هیچگاه سیستم با پایگاه داده (جهت ثبت نام) تماس نخواهد گرفت. برای غلبه بر این مشکل که در برخی از برنامه ها وجود دارد، ما هر آزمون را به m آزمون مختلف تقسیم کرده‌ایم؛ که در آن m تعداد پارامترهایی است که در درخواست http وجود دارد. هر آزمون تنها شامل یک پارامتر درهم شده

[1] Dynamic Analysis

[2] Static Analysis

[3] terms of agreement

است و باقی پارامترها به صورت رمز نشده باقی می‌مانند. ایده اصلی این تصمیم این است که در اکثر حملات تزریق SQL، تنها یک پارامتر باعث بروز حمله می‌شود و هیچ رابطه‌ای مابین آن و متغیرهای دیگر وجود ندارد. بنابراین هنگامی یک ورودی بدخواه در بروز یک حمله موفق خواهد بود که حداقل یک درخواست از m درخواست فرستاده شده، بتواند ساختار پرس‌وجو را تغییر دهد. این نتیجه را می توان به صورت زیر نشان داد:

$$R = \bigwedge_0^m T_i \qquad (۱)$$

که در آن R نتیجه موفقیت آزمون در بروز آسیب پذیری برای یک ورودی بدخواه از لیست ورودی‌ها، m تعداد پارامترهای مورد نیاز برای ارسال درخواست در آن آزمون و T_i نتیجه آزمون برای درخواستی که تنها پارامتر i ام آن درهم‌سازی شده است که درصورت موفقیت حمله مقدار یک و درصورت شکست آن، مقدار صفر می‌گیرد.

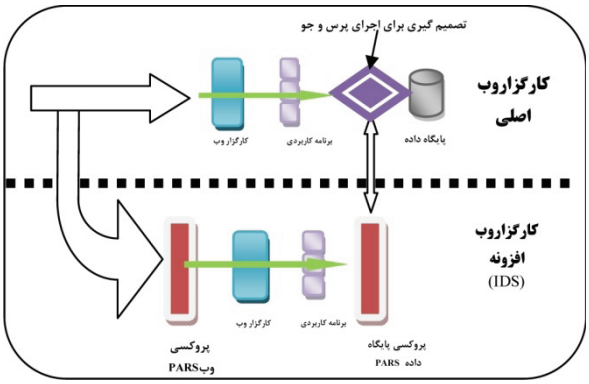
از آن‌جا که مسئله اثبات موفقیت یک حمله در راه‌حل‌های برون خط همواره مورد سوال بوده است، این سیستم می تواند در مقایسه با راه‌حل MUSIC [۱] مطرح شود که بر خلاف آن، تشخیص موفقیت حمله، نیاز به تغییر کد منبع ندارد.

۲-۴ سیستم برخط برای تشخیص و جلوگیری نفوذ

از آنجا که در راه حل PARS، تمامی پارامترها، درهم می‌شوند (حتی پارامترهایی که در ساختن پرس‌وجوها استفاده نمی‌شوند)، آن را نمی‌توان، به تنهایی و بی هیچ افزونه‌ای، در زمان اجرا بکار برد؛ چرا که تمامی پارامترهایی که به کارگزار وب می‌رسد، درهم‌ریخته و بی‌معنی خواهد بود و صحت کار برنامه کاربردی را دچار مشکل خواهد کرد.

سیستم مناسبی که برای پیاده‌سازی PARS در زمان اجرا پیشنهاد می‌شود مطابق شکل ۵ است. در این سیستم، از یک کارگزار وب افزونه کمک گرفته شده است، که کار تشخیص نفوذ را انجام می‌دهد. پارامترهای درهم نشده به کارگزار وب اصلی، و پارامترهای درهم شده به کارگزار وب افزونه داده می‌شود. در نهایت، این کارگزار وب افزونه است که به عنوان سیستم تشخیص نفوذ، اجازه اجرای پرس‌وجو از طرف کارگزار وب اصلی را می‌دهد.

تمامی درخواست‌هایی که برای کارگزار وب فرستاده می‌شود، به سیستم افزونه نیز فرستاده می‌شود. این سیستم افزونه، دقیقا شبیه به سیستم اصلی است، با این تفاوت که در آن از پروکسی‌های معماری PARS استفاده شده است.



شکل۵ : سیستم تشخیص و جلوگیری از نفوذ با استفاده از معماری PARS

سیستم افزونه به کمک سیستم PARS، می‌تواند حمله را کشف کند و به سیستم اصلی در تصمیم‌گیری برای اجرای پرس‌وجو فرمان دهد. از آن‌جا که درخواست‌های درهم‌نشده وارد سیستم اصلی می‌شود، صحت سیستم به خطر نخواهد افتاد.

این راه، علی رغم این‌که نیاز به یک سیستم افزونه دارد، در مقابل راه‌حل‌هایی که نیاز به تغییر کد برنامه، و فراخوانی یک مانیتور در زمان اجرا دارند، مانند AMNESIA [۷] و یا راه‌حل‌هایی مانند CANDID [۱۵] که با اجرای یک روند مشترک برای دو متغیر انجام می‌شود، عمومی‌تر است.

اگرچه ساختاری پیشنهادی برون خط PARS به خاطر ماهیت تشخیصی آن در زمان توسعه، فاقد محدودیت زمان پاسخ است و تقسیم هر آزمون آن به m آزمون مجزا، اهمیتی نخواهد کرد، اما در زمان اجرا و در ساختار برخط، تقسیم آزمون به m آزمون، زمان تصمیم گیری سیستم PARS برای صدور مجوز اجرای پرس‌جو را افزایش می‌دهد و بدین ترتیب زمان پاسخ کل سیستم کارگزار وب را به درخواست‌های ورودی افزایش می‌دهد.

۵- نتیجه گیری

دراین مقاله راهکاری برای مقابله با حملات تزریق وب پیشنهاد شد که از فناوری توسعه وب، مستقل است. این ویژگی باعث می‌شود که راه حل را بتوان در تمامی محیط‌ها و با فناوری‌های مختلف بکار برد. ما نام این راه حل را بدلیل استفاده از توابع در هم ریزی PARS یا Parameter Scrambling گذارده‌ایم. PARS را می‌توان در دو ساختار برخط و برون خط پیاده‌سازی کرد. ساختار برون خط برای تشخیص آسیب‌پذیری و ساختار برخط برای جلوگیری از نفوذ کاربرد دارد. کاربرد عمده این روش می‌تواند در هنگام توسعه برنامه کاربردی تحت وب، برای آزمون برنامه در مقابل حملات تزریق وب باشد. اگرچه با پاره‌ای ملاحظات

می‌توان از آن در محیط عملیاتی برای کشف و مقابله با آسیب پذیری نیز استفاده کرد.

ما درآینده سعی می کنیم که زمان اجرای ساختاربرون خط و برخط PARSرا با پیاده‌سازی آن بر روی یک یا چند برنامه کاربردی بدست آورده با زمان اجرای راه‌کارهای دیگر مقایسه کنیم.

مراجع

[1] H. Shahriar, M. Zulkernine, "MUSIC Mutation-based SQL Injection Vulnerability Checking", Proceedings of the 2008 The Eighth International Conference on Quality Software - Volume 00, 2008,Pages 77-86

[2] Y. Shin, L. Williams, and T. Xie, "SQLUnitGen: SQL Injection Testing Using Static and Dynamic Analysis", In Proceedings of the International Symposium on Software Reliability Engineering (ISSRE), 2006

[3] "National Vulnerability Database Version 2.2", available at nvd.nist.gov

[4] "Michael Sutton's Blog", available at: http://www.communities.hp.com/securitysoftware/blogs/msutton/archive/tags/SQL+Injection/default.aspx

[5] Y. Kosuga, K. Kono, M. Hanaoka, M. Hishiyama, Y. Takahama, "Sania: Syntactic and Semantic Analysis for Automated Testing against SQL Injection", In Proceedings of 23rd Annual Computer Security Applications Conference,2007 (ACSAC 2007), Miami, Dec 2007, pp. 107-117.

[6] S. Thomas and L. Williams, "Using Automated Fix Generation to Secure SQL Statements", In Proceedings of Third International Workshop on Software Engineering for Secure Systems (SESS'07), Minneapolis, 2007, pp. 9-14.

[7] William G. J. Halfond , Alessandro Orso ,"AMNESIA: analysis and monitoring for Neutralizing SQL-injection attacks", Year of Publication: 2005, ISBN:1-59593-993-4

[8] K. Kemalis and T. Tzouramanis, "SQL-IDS: A Specification-based Approach for SQL-Injection Detection", In Proceedings of 23rd ACM Symposium on Applied Computing (SAC'08), Mar 2008, Fortaleza, pp. 2153- 2158.

[9] Elisa Bertino, Ashish Kamra, James P. Early, " Profiling Database Application to Detect SQL Injection Attacks", Performance, Computing, and Communications Conference, 2007,pages 1097-2641

[10] Y. Huang, S. Huang, T. Lin, and C. Tsai. Web Application Security Assessment by Fault Injection and Behavior Monitoring. In Proceedings of the 12th International World Wide Web Conference (WWW03), pages 148–159, 2003.

[11] F. Valeur, D. Mutz, and G. Vigna. A Learning-Based Approach to the Detection of SQL Attacks. In Proceedings of the Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA), pages 123–140, 2005.

[12] Skaruz, J. Seredynski, F. ,"Recurrent neural networks towards detection of SQL attacks", Parallel and Distributed Processing Symposium,. IPDPS 2007. IEEE International, 2007

[13] Evaluation of Anomaly Based Character Distribution Models in the Detection of SQL Injection Attacks", Proceedings of the 2008 Third International Conference on Availability, Reliability and Security 2008 , Pages 47-55

[14] S. Boyd and A. Keromytis. SQLrand: Preventing SQL injection attacks. In Proceedings of the Applied Cryptography and Network Security (ACNS), pages 292–304, 2004.

[15] Sruthi Bandhakavi ," CANDID, preventing sql injection attacks using dynamic candidate evaluations", Conference on Computer and Communications Security, , 2007, Pages: 12 – 24

[16] G. Buehrer, B. W. Weide, P. A. G. Sivilotti, "Using parse tree validation to prevent SQL injection attacks", Proceedings of the 5th International Workshop on Software Engineering and Middleware (SEM '05), Lisbon, Portugal, 2005, pp.106-113.

[17] Xiang Fu , "SAFELI: SQL injection scanner using symbolic execution", International Symposium on Software Testing and Analysis, 2008, Pages 34-39

[18] Black, Rex Pragmatic Software Testing: Becoming an Effective and Efficient Test Professional. New York: Wiley. 2007. p. 240. ISBN 978-0-470-12790-2.

[19] D.R. Kuhn, D.R. Wallace, A.J. Gallo, Jr., "Software Fault Interactions and Implications for Software Testing", IEEE Trans. on Software Engineering, vol. 30, no. 6, June, 2004