هفتمین کنفرانس بین المللی انجمن رمز ایران

۲۴ و ۲۵ شهریورماه ۱۳۸۹، دانشگاه صنعتی خواجه نصیرالدین طوسی

!

انجمن رمزایران

دانشگاه صنعتی خواجه نصیرالدین طوسی

طراحی و پیاده‌سازی یک ابزار تحلیل پویای بدون مثبت کاذب برای کشف آسیب‌پذیری تزریق SQL!

احسان اعرابی[۱]، مهدی برنجکوب[۲]، محمد علی منتظری[۳]

مرکز تخصصی آپا دانشگاه صنعتی اصفهان [۱]

e.aerabi@ec.iut.ac.ir

عضو هیئت علمی دانشگاه صنعتی اصفهان [۲و۳]

brnjkb@cc.iut.ac.ir

montazer@cc.iut.ac.ir

چکیده

با افزایش تنوع و گسترش استفاده از خدمات تحت وب، تهدیدهای مختلفی متوجه کارگزاران و کاربران آن شده است. یکی از تهدیدهای مطرح و پرمخاطره در این زمینه، حمله تزریق SQL است. حمله تزریق SQL می‌تواند منجر به افشای اطلاعات، دور زدن مکانیزم‌های احراز اصالت، حذف یا درج اطلاعات درپایگاه داده و تغییرات در سیستم شود. بدین ترتیب توسعه‌گران برنامه‌های کاربردی تحت وب تمایل دارند از امنیت محصول خود در مقابل این نوع حمله اطمینان حاصل کنند. برای اطمینان از امنیت یک برنامه کاربردی تحت وب در مقابل حمله تزریق SQL می‌بایست آن را با ابزارهای آزمون حمله تزریق SQL آزمود. در این مقاله طراحی و پیاده‌سازی ابزاری برای آزمون برنامه‌های کاربردی تحت وب شرح داده خواهد شد که نه تنها قادر به انجام آزمونی موثر و پوشاست، بلکه فاقد موارد مثبت کاذب در تشخیص آسیب‌پذیری خواهد بود. این ابزار نیازی به تفسیر یا تغییر کد منبع ندارد. همین‌طور در مواردی که برنامه کاربردی پرس‌وجوهای پویا تولید می‌کند نیز قابل استفاده است. این ابزار برای آزمون برنامه‌های کاربردی معتبری بکار گرفته شد و توانست چندین نقطه قابل تزریق را در آنان بیابد.

واژه های! کلیدی

تزریق SQL، امنیت وب، پویش امنیتی.

۱- مقدمه !

با افزایش بکارگیری برنامه‌های کاربردی تحت وب، اهمیت امنیت اطلاعات در این زمینه در حال گسترش 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. زبان SQL برای ذخیره، تغییر و بازیابی اطلاعات از جداول موجود در پایگاه داده‌ها بکار می‌رود. از آن‌جا که زبان SQL با اطلاعات موجود در وب سایت‌ها ارتباط دارد، مورد توجه و سوء استفاده نفوذگرها قرار گرفته و باعث بوجود آمدن تهدید تزریق SQL 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، اعتبارسنجی[۱] مناسبی نداشته باشد. عدم اعتبارسنجی ورودی‌هایی که در تولید یک پرس وجو[۲] بکارگرفتـه

1 -input validation

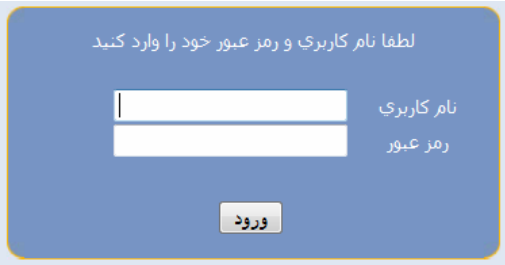
2 - query

می‌شوند، باعث ایجاد این آسیب‌پذیری می‌شود. از آنجا که این پرس‌وجوها توسط پایگاه‌های داده (برای ذخیره و 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 و از کار اندازی سرویس می‌کنند.

ادامه این مقاله در بخش ۲، با معرفی پیش زمینه‌ای از حمله و دسته‌های مختلف آن و در بخش ۳ با ذکر ابزارها و راهکارهای موجود برای آزمون و کشف آسیب‌پذیری تزریق SQL ادامه می‌یابد. سپس در بخش ۴ روش پیشنهادی این مقاله معرفی می‌شود و در نهایت در بخش‌های ۵ و۶، به نحوه آزمون و نتایج بدست آمده پرداخته می‌شود.

۲- پیش زمینه و ابزارهای مرتبط

در این بخش حملات تزریق SQL در یک دسته بندی چهارگانه معرفی خواهد شد. 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 توسط DBMS [2] اجرا خواهد شد. اگر ورودی‌های کاربر توسط یک نفوذگر به گونه‌ای ماهرانه تغییر یابد، ممکن است پس از قرار گرفتن در درون پرس‌جو، باعث تغییر ساختار آن و انجام عملی ناخواسته گردد.



شکل۱ - یک صفحه احراز اصالت

ما این نوع تغییرات را از دیدگاه نحوه حمله و تاثیر آن‌ها، به چهار دسته تقسیم کرده‌ایم. این دسته‌ها عبارتند از:

دور زدن مکانیزم‌های احراز اصالت: برای احراز اصالت کاربران یک پایگاه اینترنتی از مکانیزم‌های احراز اصالت استفاده می‌شود. این مکانیزم‌ها، نام کاربری و کلمه عبور را توسط قالبی شبیه به شکل ۱ دریافت می‌کنند. با فرض این‌که برنامه کاربردی با زبان php تهیه شده باشد، مکانیزم‌های احراز اصالت شبیه به این خواهد بود:

```
$sqlquery = "SELECT * From LoginTable WHERE
username='$username' and password =
'$password' ";
$process=odbc_exec($sqlconnect, $sqlquery);
```

این قطعه کد پس از دریافت نام کاربری و کلمه عبور، آن‌ها را در پرس‌جوی ذکر شده قرار می‌دهد و سپس پرس‌وجو را اجرا می‌کند. در صورتی که چنین نام کاربری و رمز عبوری در پایگاه داده موجود باشد، آن درایه توسط پایگاه داده بازگردانده خواهد شد و این به معنای احراز اصالت کاربر خواهد بود. در غیر این صورت از ورود کاربر به سیستم جلوگیری خواهد شد. این قطعه کد به ورودی زیر آسیب‌پذیر است:

```
abcd ' or 1=1;--
```

اگر این ورودی به عنوان نام کاربری وارد شود، پرس‌وجو به صورت زیر در خواهد آمد:

```
SELECT * From LoginTable WHERE
username = 'abcd' or 1=1; -- ' and
password = '$password'
```

این حمله بر اساس تبدیل پرس‌وجو به یک عبارت درستنما[3] بوجود می‌آید. بنابراین عمدتا شامل عباراتی شبیه به «or 1=1» خواهد بود. از آنجا که عبارت «or 1=1» شرط where را به یک شرط *همواره صحیح* تبدیل می‌کند، این پرس‌وجو تمامی درایه‌های جدول را به عنوان جواب بازمی‌گرداند که این به معنی احراز اصالت نفوذگر خواهد بود (زیرا که پاسخ پایگاه داده تهی نیست). برای مطالعه بیشتر در مورد اشکال مختلف نحوه دور زدن اصالت به [۳] رجوع شود.

جمع آوری اطلاعات براساس آشکارسازی خطا

این نوع حمله براساس تحمیل ورودی‌های نامعتبر به منظور مشاهده خطا از برنامه کاربردی صورت می‌گیرد. وارد کردن عباراتی مانند «having 1=1;--» در مثال احراز اصالت بالا، باعث تغییر در ساختار پرس‌وجو می‌شود به گونه ای که پایگاه داده MS-SQL Server در قبال این پرس‌وجو خطایی مانند این را تولید خواهد کرد:

```
Column 'news.news_id' is invalid in the
select list because it is not contained in
an aggreate function and there is no GROUP
BY clause.
```

این خطا، نام جدول و یکی از ستون‌های آن را آشکار کرده است (قسمتی از خطا که پررنگ‌تر نشان داده شده است). این خطا و خطاهای مشابه به نفوذگر کمک می‌کنند که اطلاعاتی در مورد جداول و صفت‌های آنان بدست آورد. این حمله برای بدست آوردن اطلاعات مختلف از پایگاه داده، روش‌های متنوعی دارد که برای مطالعه بیشتر می‌توان به [۳] [۴] [۵] رجوع کرد.

1 - query
2 - Database Management System.
3 - tautology

فاش‌سازی اطلاعات با استفاده از union

این حملات در زمانی بکار می‌روند که دو شرط برقرار باشد: «پرس‌وجو به منظور بازیابی اطلاعات از دستور Select استفاده کند» و «اطلاعات بازگردانده شده از پایگاه داده به کاربر نمایش داده شود(اطلاعات به منظور نمایش بازیابی شوند)». این گونه از حملات باعث می‌شوند که اطلاعاتی از درون پایگاه داده به همراه اطلاعات اصلی آشکار شوند.

برای مثال فرض کنیم، یک برنامه کاربردی، عددی را به منظور نمایش خبر دریافت کرده و خبر متناظر با آن عدد را نمایش دهد. بدین منظور برنامه کاربردی، قطعه کد زیر را برای پردازش عدد وارد شده توسط کاربر، اجرا می‌کند. در این‌جا $id عددی است که توسط کاربر وارد شده است:

```
$sqlquery = "SELECT * From news WHERE id =
$id";
$process=odbc_exec($sqlconnect, $sqlquery);
echo odbc_result($process,2);
```

در خط سوم این کد، پاسخ پایگاه داده نمایش داده می‌شود. اگر به جای مقدار عدد، کاربر عبارت زیر را وارد کند:

```
1 and 1=2 union select 1,2,3,@@version--
```

پرس‌وجو به صورت زیر تغییر می‌کند:

```
SELECT * From news WHERE id = 1 and 1=0
union select 1,@@version,3 --
```

عبارت «and 1=0» یک عبارت (منطقی) همواره غلط است و باعث می‌شود که اولین Select دارای جوابی تهی باشد، اما دومین Select در پرس‌وجوی بالا شامل «@@version» است که نسخه پایگاه داده را بازخواهد گرداند. حاصل هر دو عبارت Select با دستور union مجتمع خواهد شد و با عبارت echo به نفوذگر نشان داده خواهد شد. نفوذگر بدین ترتیب از نسخه پایگاه داده باخبر می‌شود. برای کسب اطلاعات بیشتر در مورد جزئیات این حمله، به [۶] [۷] [۸] رجوع شود.

استنتاج و کشف اطلاعات توسط تزریق کور SQL

گاه نفوذگران توجه خود را به عکس‌العمل‌های برنامه در مقابل عبارات همیشه درست یا همیشه غلط معطوف می‌کنند. برای مثال اگر عکس العمل برنامه در قبال بردارهای حمله زیر متفاوت باشد، احتمالا برنامه به حمله تزریق کور آسیب پذیر است:

```
5 or  1=1--
5 and 1=0--
```

عکس‌العمل‌های متفاوت برنامه بدین معنی تعبیر می‌شود: اگر در ازای وارد کردن عبارت همواره درست «5 or 1=1--» به عنوان ورودی یک برنامه کاربردی تحت وب، در صفحه بازگردانده شده، متن یا تصویر خاصی (که نام آن را «symptom» می‌گذاریم) نمایش داده شود، اما همین متن یا تصویر در ازاء بکارگیری عبارت همواره غلط «5 and 1=0--» نمایش داده نشود، احتمال آسیب‌پذیری برنامه به تزریق کور SQL وجود دارد. اینک با «نمایش یا عدم نمایش symptom » می‌توان درست بودن برخی گزاره‌ها را در سیستم مورد استنتاج قرار داد. برای مثال با این عبارت می‌توان دانست که آیا حرف اول نام پایگاه داده، با «m» شروع می‌شود یا خیر(کد ASCII حرف m ۱۰۹ است):

```
1 and SUBSTRING(SELECT db_name(),1,1)=109
```

اگر Symptom نمایش داده شود، می‌توان نتیجه گرفت که حرف اول نام پایگاه داده «m»است.

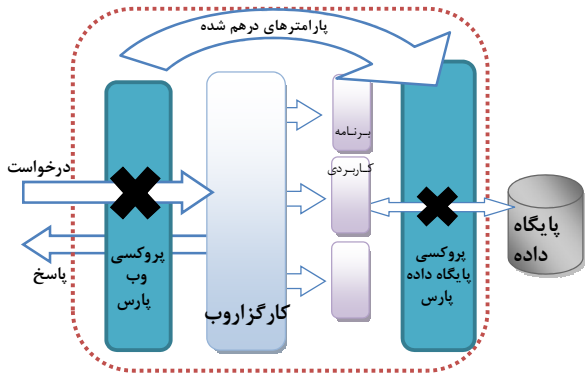
۳-کارهای مرتبط

در این بخش راهکار PARS به همراه راهکارهای مرتبط در زمینه آزمون و کشف آسیب‌پذیری تزریق SQL معرفی می‌شود.

۳-۱- روش PARS [۱]

روش PARS که قبلا توسط مولفین همین مقاله ارائه شده است، روشی است برای تشخیص موفقیت حمله تزریق SQL. این روش می‌تواند موفقیت یا عدم موفقیت حمله را بدون خطا و همچنین بدون تفسیر یا تغییر کد منبع تشخیص دهد. همچنین این روش محدودیتی در استفاده از پرس‌وجوهای پویا و همزمان ندارد. دیگر مزیت مهم این روش عدم نیاز به دوره یادگیری است.

روش PARS برای تشخیص موفقیت حمله تزریق SQL، 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.



شکل ۳ معماری PARS

وظایف این دو پروکسی بدین شرح است:

PARS Web Proxy: این پروکسی به محض دریافت یک درخواست، پارامترهای (متغیرهای) Http را درهم می‌ریزد. برای مثال، در نمونه احراز اصالت ذکرشده در بخش ۲-۱، مقدار دو پارامتر username و password بدین صورت درهم می‌شود:

```
username='a&sd2re4%'&password='$3da&r! jh6'
```

پروکسی وب درخواست تغییر یافته را برای وب سرور می فرستد.

PARS Database Proxy: وظیفه اصلی این پروکسی زمانی ایفا می گردد که یک پرس‌وجو برای پایگاه داده فرستاده می‌شود. در این هنگام، پروکسی پارامترهای درهم ریخته‌شده‌ی درون پرس‌وجو را یافته و سعی می‌کند که پرس‌وجو را با بازگرداندن پارامترها به مقدار اولیه خودشان، بازسازی کند. برای مثال، در نمونه مذکور، پروکسی پایگاه‌داده، از پروکسی وب، این دو پارامتر را دریافت می‌کند:

```
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'
```

پرس‌وجو پس از بازگشایی پارامترهای درهم شده، بدین صورت خواهد شد:

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

حال پروکسی می‌تواند 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. دو پرس‌وجو با هم در تعداد عبارات Where_Conditions (شرط-های مقابل where) متفاوت هستند. جزئیات استفاده از درخت تجزیه برای مقایسه پرس‌وجوها در [۸] آمده است این راه حل نیازی به تفسیر یا تغییر کد منبع ندارد.

۳-۲- Secubat [۱۰]

Secubat یک ابزار پیمانه‌ای [۴] تحت ویندوز است که توانایی کشف آسیب‌پذیری تزریق SQL را دارد. این ابزار می‌تواند مواردی از آسیب‌پذیری‌های پایه و ساده را در برنامه‌های کاربردی تحت وب بیابد. این ابزار یک ابزار آزمون جعبه سیاه[۵] است که نیاز به تفسیر یا تغییر کد منبع ندارد. همچنین این ابزار در مواردی سعی می-کند تا جهت اثبات آسیب‌پذیری یافت شده ، کد سوء استفاده[۶] تولید کند. از آن‌جا که این ابزار برپایه مشاهده خطاهای بیرونی عمل می‌کند، درصورتی که خطاهای برنامه به صورت کامل توسط برنامه نویس پوشش داده شود، این ابزار کارایی خود را از دست می‌دهد و موارد منفی کاذب[۷] آن افزایش می‌یابد. به دلیل مشابه، موارد مثبت کاذب[۸] نیز در گزارشات این گونه ابزارها به چشم می‌خورد.

۳-۳- Acunetix [۱۴]

یکی از ابزارهای تجاری معروف که جزء ۱۰ ابزار برتر پویش امنیتی در سایت insecure.org ذکر شده است. این ابزار می‌تواند انواع آسیب‌پذیری‌های وب را بواسطه‌ی یک آزمون جعبه سیاه بیابد و در زمینه کشف آسیب‌پذیری تزریق SQL نیز ابزاری موثر است.

۳-۴- Wapiti [۱۵]

یک ابزار پویش امنیتی که بسیاری از آسیب‌پذیری‌های برنامه‌های کاربردی تحت وب را می‌شناسد. این ابزار فاقد واسط گرافیکی کاربر است و باید توسط پوسته اجرا شود.

۳-۵- Scrawler [۱۶]

ابزاری که توسط شرکت hp ارائه شد و به طور اختصاصی می‌تواند آسیب‌پذیری تزریق SQL را شناسایی کند. این ابزار در محیط ویندوز اجرا می‌شود و دارای واسط گرافیکی است.

۴- ابزار پیشنهادی

در این بخش نحوه طراحی ابزار آزمون تزریق SQL پیشنهادی بازگو می‌شود. این ابزار از ایده PARS بهره می‌گیرد که به همین علت نام آن را PARSgen گذارده‌ایم. این ابزار شامل مولفه‌هایی است که در این بخش معرفی می شود. معماری ابزار PARSgen دارای معماری خاصی است که آن را در شکل ۴ می‌توان دید.

۴-۱- پایگاه دانش تزریق SQL

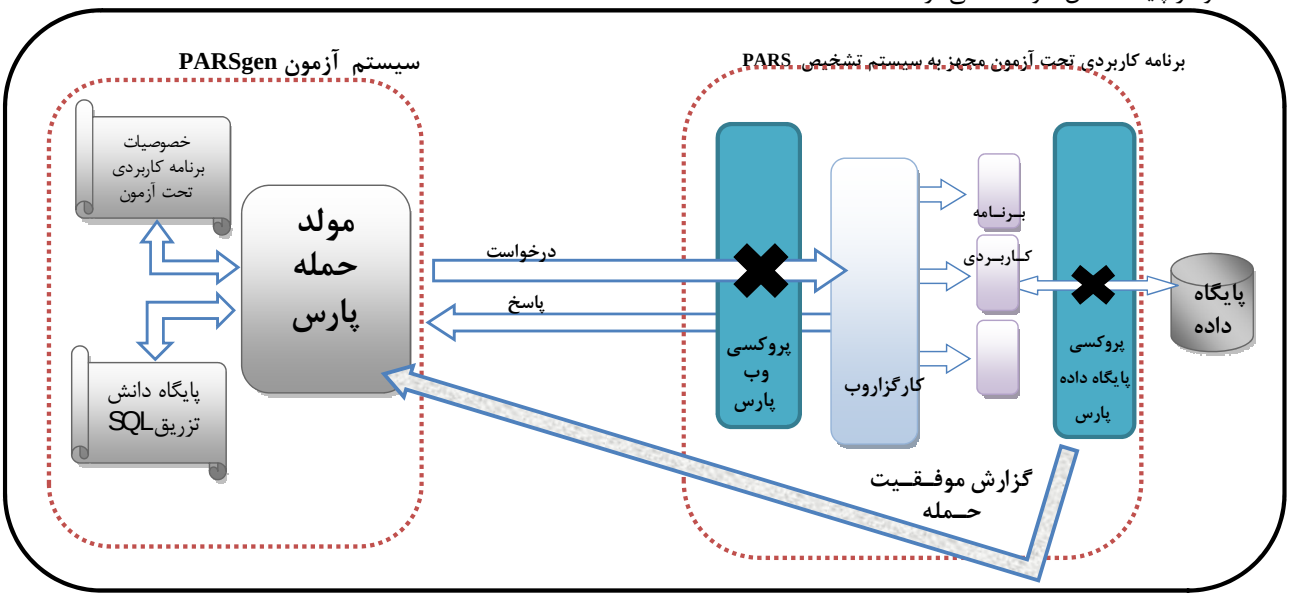
سیستم آزمون PARSgen به گونه‌ای طراحی می‌شود که بتواند به صورت پویا، بردارهای حملات را تولید کند. تولید بردارهای پویا، انعطاف ابزار را به منظور عبور از فیلترهای برنامه کابردی و کشف نقاط آسیب‌پذیر افزایش می‌دهد. بدین منظور زبانی نمادین برای توصیف بردارهای حمله تهیه می شود که برای تولید پویای حملات تزریق SQL لازم است. با این زبان نمادین می‌توان پایگاه دانشی از انواع حملات تزریق SQL را در اختیار PARSgen قرار داد تا بتواند حملات خود را به صورت افزایشی تکمیل نماید(شکل ۴). مثال ذکر شده در بخش دوم با عنوان یک صفحه احراز اصالت نوعی را در نظر می‌گیریم. این صفحه در پارامتر « نام کاربری » در مقابل بردار حمله --; A' OR 1=1 آسیب پذیر بود. این بردار را می توان به صورت زیر نمایش داد:

```
<literal><space><OR><space><tautology>
<SemiColon><comment>
```

در این‌جا<space> تنها به معنی یک فاصله خالی نیست، بلکه مجموعه ای از کدگذاری‌های مختلف از فاصله نیز می تواند باشد. بنابراین به ازای هر مؤلفه از زبان توصیف حمله، ممکن است که چندین معادل وجود داشته باشد برای مثال نماد <tautology>

[4]- modular
[5] - black box
[6] -exploit code
[7] - false negative
[8] - false positive

شامل تمامی عبارات همیشه درست مانند «1=1»، «--;»، «'a'='a'» یا «'a' = 'a' » و امثال اینها است. سیستم آزمون PARSgen به مجموعه‌ای از این عبارات معادل مجهز شده است، که همگی از کدهای سوء استفاده و همچنین مستندات مرتبط استخراج شده‌اند. این سیستم تمامی حملات تزریق SQL چهارگانه ذکر شده در بخش ۲-۱ را با زبان نمادین مستند کرده است و در پایگاه دانش خود نگه می‌دارد.



شکل ۴- معماری سیستم آزمون PARSgen

۴-۲- خزنده[9] PARSer

«سیستم آزمون PARSgen » با استفاده از یک خزنده مناسب سعی در پیدا کردن تمامی مدخل‌ها، فرم‌ها و پارامترهای موجود در وب سایت تحت آزمون دارد. خزنده پس از ذخیره تمامی صفحات یافت شده از برنامه کاربردی تحت آزمون، آن‌ها را تجزیه کرده، خصوصیات آن صفحات را به صورت یک ورودی قابل فهم برای مولد حمله PARS (در شکل ۴) ذخیره خواهد کرد. این فایل شامل آدرس تمامی فرم‌های برنامه کاربردی، پارامترها و مقادیر اولیه آنها و بالاخره متد فرستادن اطلاعات به آن‌ها (GET, POST) خواهد بود.

۴-۳- تولید حمله

مراحل انجام الگوریتم تولید حمله را می‌توان در شکل ۴ دنبال کرد. در این شکل و در سمت چپ، «سیستم آزمون PARSgen» وجود دارد که با دسترسی به مولفه‌ی «خصوصیات برنامه کاربردی تحت آزمون» می‌تواند صفحات مورد آزمون را به همراه فرم‌ها، متدها و فیلدهای آن صفحات بیابد. سپس مولد با دسترسی به پایگاه دانش تزریق SQL که مجموعه‌ای از مولفه‌های تشکیل دهنده حمله است می‌تواند به صفحات برنامه کاربردی تحت آزمون حمله کند.

مولد حمله بر خلاف دیگر ابزارها که برپایه مشاهده و استنتاج از خروجی‌های برنامه کاربردی استوار هستند، از یک غیب‌گو به نام «سیستم برون خط تشخیص آسیب پذیری PARS» (سمت راست شکل ۴) بهره می‌برد که در بخش ۳ معرفی شد. بدین ترتیب مولد نیازی به مشاهده نتایج ندارد چرا که پروکسی‌های PARS ساختار پرس‌وجوی ساخته شده را به مولد گزارش می‌دهد و مولد را در تکمیل حمله یاری می‌کند.

لازم به ذکر است که اولین قدم یک حمله موفق زمانی است که بردار حمله بتواند ساختار پرس‌وجوی ساخته شده را از شکل عادی آن خارج کند. درهم شکستن ساختار یک پرس‌وجو تنها با گروهی از عبارات زبان SQL امکان‌پذیر است. این عبارات و حروف رزرو شده که همواره حملات موفق با آن‌ها شروع می‌شوند را «عبارات شکافت» نامیده‌ایم. اگر برنامه کاربردی به صورت موثر تمامی عبارات شکافت را فیلتر یا خنثی کند، هیچ حمله موفقی به برنامه ممکن نخواهد بود.

نمونه‌ای از عبارات شکافت عبارتند از «'»، «OR»، «AND» و «UNION». این عبارات همگی از حملات واقعی و گزارش کدهای سوء استفاده آنان برگرفته شده است. یک عبارت شکافت مانند «'» در پارامتر نام کاربری مثال مذکور می‌تواند پرس‌وجو را بدین صورت درآورد:

[9] - crawler

```
statement = “SELECT * FROM users WHERE
name = ''' AND  Pass ='xyz';”
```

اگر چه این پرس‌وجو از لحاظ قواعد دستوری SQL نادرست است، اما مشاهده این پرس‌وجو برای «مولد حمله PARSgen» یک معنی اضافه خواهد داشت : « برنامه کاربردی در مقابل عبارت شکافت «'» شکسته شد و امکان تقویت بردار حمله بوسیله این عبارت شکافت تا تشکیل یک حمله کامل وجود دارد». مولد حمله اکنون می‌تواند یکی از انواع حملات ذکر شده در بخش ۲-۱ را به صورت افزایشی کامل کند و به سمت تشکیل یک پرس‌وجوی (از لحاظ قواعد دستوری) درست پیش برود و این به معنی تولید یک بردار سوء استفاده برای آسیب پذیری کشف شده است. لازم به ذکر است که دیگر ابزارهای آزمون که از غیب‌گوی PARS بی‌بهره هستند، فقط درصورت موفقیت کامل یک حمله می‌توانند به آسیب پذیر بودن یک پارامتر پی ببرند. این درصورتی است که «سیستم آزمون PARSgen» از کوچکترین تغییر در ساختار پرس‌وجو (که می‌تواند با افزایش و تکمیل به یک حمله موفق تبدیل شود) آگاه است.

مسئله دیگر در این میان این است که مولد درصورتی که با اعمال عبارات شکافت موفق به تغییر ساختار پرس‌وجو نشود، آن پارامتر را امن تشخیص می‌دهد و دیگر به امتحان الگوهای حمله نمی پردازد. به عبارت دیگر ازآن‌جا که همگی الگوهای حمله بدون استثنا با همین عبارات شکافت شروع خواهند شد، در صورت عدم توفیق در تغییر ساختار پرس‌وجو، بی اثر بودن حمله برای مولد اثبات می‌شود.

در ضمن درهنگام پیاده‌سازی سیستم PARS در درون PARSgen، پاره‌ای نقایص در ایده اولیه مشاهده شد که به منظور رفع آن، مقدار درهم شده و مقدار واقعی درکنار هم به سمت برنامه کاربردی فرستاده می‌شود. این مسئله مشکلات مذکور را برطرف کرده، نیاز به مسیر جانبی در سیستم PARS (شکل ۳) را از بین می‌برد و پیاده سازی PARSgen را آسان‌تر می‌سازد. بنابراین مقدار درهم شده از خلال برنامه کاربردی و در دل خود پرس‌وجو به پروکسی پایگاه داده می رسد که ذکر جزئیات آن در [۲] آمده است.

۵- پیاده سازی و نتایج بدست آمده

دراین بخش، به خصوصیات نمونه پیاده‌سازی و نتایج بدست آمده از آزمون آن می‌پردازیم. جزئیات پیاده‌سازی و ذکر کامل نتایج را می‌توان در [۲] یافت.

۵-۱- خصوصیات ابزار پیاده‌سازی شده

نمونه عملی ابزار آزمون PARSgen، به صورت یک برنامه کاربردی توسعه داده شد که خصوصیات آن عبارتند از:

انعطاف در پایگاه دانش حمله: این امکان به عنوان یکی از ورودی‌های PARSgen، قابلیت انعطاف مولد را افزایش می‌دهد. این فایل شامل الگوهای حمله تزریق SQL است. برای مثال محتویات این فایل شامل مواردی مانند جدول ۱ خواهد بود. این جدول شامل چهار قسمت است و تعداد عبارات شکافت درآن ۹ مورد است. همچنین پنج مولفه شروع نیز وجود دارد که معمولا به عنوان مقادیر پیش فرض در پارامترهای تحت آزمون بکار می‌روند. در ادامه الگوهای اتصال دیده می‌شوند که همگی به منظور ایجاد فاصله در بین مولفه‌های حمله بکار می‌روند. در انتها الگوهای خاتمه هستند که برای اتمام پرس و جو بکار می‌روند. وجود چنین امکانی باعث می‌شود که تنوع حملات افزایش یابد و حملات از قالب یک سری ورودی ثابت، به صورت رشته‌ای از مولفه‌های قابل تغییر بهبود پیدا کند.

توانایی پذیرفتن کوکی: ازآن‌جا که کوکی‌ها جزئی از مکانیزم‌های احراز اصالت هستند، PARSgen این قابلیت را دارد که کوکی مورد نظر را برای آن مشخص کرد تا بتواند به صفحاتی که پس از احراز اصالت دیده می‌شوند حمله کند.

جدول ۱- مولفه‌های حمله تزریق SQL

عبارات شکافت	start patterns	join patterns	final patterns
'	null	+	;
union	1	%20	;--
and	-1	%2020	--
or	%27	/**/	/*
)	%00		#
(			%23
[			
]			
@@			

پیداکردن نقاط بالقوه آسیب‌پذیر: پیاده سازی مولد پارس در حال حاضر این امکان را فراهم می‌کند که موارد بالقوه آسیب‌پذیر در برنامه‌های کاربردی یافته شوند. موارد بالقوه آسیب‌پذیر پارامترهایی هستند که با بکارگیری عبارات شکافت در آن‌ها، پرس‌وجوی حاصل شکسته شده است. این نقاط بالقوه آسیب‌پذیر می‌بایست در یک پروسه‌ی کامل آزمون نفوذ، مورد بررسی قرار گیرند تا توان بالفعل شدن آن‌ها ثابت یا رد شود. ارائه کد سوء استفاده هنوز در PARSgen توسعه نیافته است و همچنان در حال تکمیل است.

۵-۲- ارزیابی مقایسه‌ای PARSgen بر اساس آزمون یک برنامه کاربردی آسیب‌پذیر

به منظور مقایسه PARSgen با دیگر ابزارهای موجود، یک برنامه کاربردی آسیب‌پذیر انتخاب شد. این برنامه کاربردی تحت وب، از تیم امنیت وب مرکز تخصصی آپا دانشگاه صنعتی اصفهان اخذ

شد. برنامه مذکور شامل هر چهار نوع آسیب‌پذیری ذکر شده در بخش ۲-۱ می‌باشد (امکان دور زدن مکانیزم احراز اصالت، حمله به پارامترهای عددی ، تزریق کور و حمله‌های union) می‌باشد. با توجه به ملاحظات بالا، برنامه مذکور، برای ارزیابی ابزارهای مورد نظر مناسب است. بدین ترتیب، ابزار PARSgen به همراه چهار ابزار معروف دیگر برای آزمون نفوذ به این برنامه کاربردی بکارگرفته شد.

همان‌گونه که در جدول ۲ مشاهده می‌شود، ابزارهای Secubat، Wapiti، Scrawler تنها توانسته‌اند مکانیزم احراز اصالت را دور بزنند و به پارامترهای عددی حمله کنند و توانایی کشف آسیب پذیری union و تزریق کور را نداشته‌اند.

جدول ۲ نتایج بدست آمده از آزمون یک برنامه آسیب‌پذیر

	مکانیزم احراز اصالت (از ۲ مورد)	پارامترهای عددی (از ۱ مورد)	استفاده از union (از ۲ مورد)	تزریق کور (از ۲ مورد)	مثبت کاذب
Acunetix	۲	۱	۲	۲	۶
Secubat	۲	۱	۰	۰	۰
Wapiti	۲	۱	۰	۰	۰
Scrawler	۱	۱	۰	۰	۰
PARSgen	۲	۱	۲	۲	۰

در مقابل Acunetix تمام رفتارهای مشکوک برنامه کاربردی را ثبت کرده و توانسته وجود آسیب پذیری‌های مختلف را به طور کامل پوشش دهد، اما گزارش این نرم‌افزار شامل موارد بسیاری از مثبت‌های کاذب بوده است. راهکار پارس اما توانسته تمامی موارد آسیب پذیری را فارغ از نوع آن‌ها بیابد. علاوه براین به علت خاصیت ذاتی این روش که در بخش قبل ذکر شد، هیچ مورد مثبت کاذبی برای PARSgen گزارش نشده است. در مقایسه پویشگرهای دیگر با پویشگر PARSgen، می‌توان نتیجه گرفت که پویشگر PARSgen دارای تشخیص‌های نسبتا پوشا و فاقد مثبت کاذب است.

۵-۳- ارزیابی ابزار پیشنهادی بوسیله آزمون برنامه‌های معتبر

به منظور ارزیابی کارایی پویشگر PARSgen روش دیگری نیز در پیش گرفته شد. جهت کشف آسیب‌پذیری‌های واقعی، چندین برنامه کاربردی متن باز را مورد آزمون قرار دادیم. این برنامه‌های کاربردی از پرکاربردترین و معروف ترین ابزارهای مدیریت محتوا انتخاب شدند. مولد پارس موفق شد مواردی از آسیب‌پذیری‌های بالقوه در برنامه‌های کاربردی مذکور را پیدا کندکه به منظور اثبات هر مورد آن، می‌بایست کد سوء استفاده برای آن تولید گردد. برنامه‌های کاربردی مورد آزمون PHPNuke نسخه‌های ۸,۰ (آخرین نسخه منتشر شده تا زمان آزمون)، ۷,۳، و ۶,۰ و PHPBB نسخه ۳,۰ (آخرین نسخه منتشر شده تا زمان آزمون) هستند.

جدول۳- نتایج بدست آمده از آزمون برنامه‌های کاربردی

برنامه کاربردی تحت آزمون	پارامترهای بالقوه آسیب‌پذیر
PHPNuke 8.0	۲
PHPBB 3.0	۰
PHPNuke7.3	۹
PHPNuke 6.0	۱۲

پس آزمون این ابزارها، پارامترهای بالقوه آسیب‌پذیر متعددی بدست آمد که نتایج آن را در جدول ۳ می بینید.

در میان پارامترهای آسیب‌پذیر یافت شده در PHPNuke نسخه ۷,۳ و ۶,۰ چندین پارامتر وجود دارد که قبلا آسیب‌پذیر بودن آنان گزارش شده بود. برای مثال در نسخه 6.0 پارامتر pollID به حمله تزریق SQL از نوع «افشای اطلاعات با استفاده از union» آسیب‌پذیر بود و توسط PARSgen کشف شد. این آسیب‌پذیری با اعمال ورودی:

```
1 and 1=0 union  SELECT
1,2,3,4,@@version,6,7,8,9,10,11,12
```

در URL برنامه به صورت:

```
http://localhost/Nuke60/html/modules.php?nam
e=Surveys&op=results&pollID= [SQL]
```

باعث صدور پرس‌وجوی زیر شد که این پرس‌وجو باعث لو رفتن نام کاربری و مقدار درهم شده کلمه عبور از پایگاه داده می‌شود:

```
select tid, pid, pollID, date, name, email,
url, host_nam e, subject, comment, score,
reason from nuke_pollcomments where pollID=1
and 1 = 0 union SELECT 1, 2, 3, 4,
@@version, 6, 7, 8, 9, 10, 11, 12
```

برخلاف برنامه PHPNuke، در برنامه PHPBB 3.0 تمامی پارامترها به خوبی اعتبار سنجی می‌شدند و به همین علت هیچ مورد قابل تزریقی پیدا نشد.

۶- نتیجه‌گیری

در این مقاله ابزار PARSgen برای آزمون و کشف تزریق SQL معرفی شد. این ابزار شامل خزنده، پایگاه دانش حمله تزریق SQL و مولد حمله بود که می‌توانست با استفاده از قابلیت انعطافی که معماری PARS و پایگاه دانش تزریق SQL به او می‌دهد، حملات موثری را برروی برنامه‌های کاربردی تحت وب انجام دهد. این ابزار می‌تواند موارد بالقوه آسیب‌پذیر را در برنامه‌های کاربردی بیابد و تمام موارد یافت شده فاقد مثبت کاذب خواهد بود.

همچنین جهت انجام عملیات کشف، نیاز به تفسیر یا تغییر کد منبع نخواهد داشت و در هنگام صدور پرس‌وجوهای پویا نیز قابل استفاده خواهد بود. علاوه بر این، PARSgen می‌تواند صفحات احراز اصالت شده را نیز پویش کند. این ابزار بر روی چندین برنامه کاربردی معتبر آزموده شد و توانست موارد بالقوه آسیب-پذیری را بیابد که در میان این موارد، آسیب‌پذیری‌های قابل سوء استفاده نیز قابل رؤیت بود. در آینده ما سعی خواهیم کرد که این ابزار را به گونه‌ای توسعه دهیم که امکان تولید کد سوء استفاده[10] از آسیب‌پذیری‌های یافت شده را داشته باشد.

مراجع

[۱]- ا. اعرابی، م. برنج‌کوب، م.ع. منتظری، «تشخیص آسیب‌پذیری تزریق SQL بر اساس راهکاری مستقل از فناوری توسعه وب»، ششمین کنفرانس انجمن رمز ایران، ۱۳۸۸.

[۲] احسان اعرابی، « طراحی و پیاده‌سازی یک ابزار تحلیل پویای بدون مثبت کاذب برای آزمون آسیب‌پذیری تزریق SQL » پایان‌نامه کارشناسی ارشد، دانشگاه صنعتی اصفهان، دانشکده برق و کامپیوتر، ۱۳۸۹.

[3]- Chris Anley, "Advanced SQL Injection In SQL Server Applications", NGSSoftware Insight Security Research (NISR) Publication, 2002.

[4]- Stuart McDonald, "SQL Injection: Modes of attack, defence, and why it matters", GIAC Security Essentials Certification, 2002

[5]- ZeQ3uL && JabAv0C, "Full MSSQL Injection PWNage", Availabe at : www.milw0rm.com/author/1456 , 2009 .

[6]- Jason A. Medeiros, " Understanding mySQLUnion Piosioning", Grayscale Research, 2008.

[7]- Marezzi, "SQL Injection Tutorial", Availabe at : www.milw0rm.com, 2008

[8]- Omni, "MySQL: Secure Web Apps - SQL Injection techniques", available at : http://omni.playhack.net, 2009.

[9] - 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.

[10]- Stefan Kals, Engin Kirda, Christopher Kruegel, and Nenad Jovanovic, "SecuBat: A Web Vulnerability Scanner", International World Wide Web Conference Committee (IW3C2), 2006.

[11]- 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

[12]- MeiJunjin, An approach for SQL injection vulnerability detection, Sixth International Conference on Information Technology: New Generations, IEEE, 2009.

[13]- William G. J. Halfond , Alessandro Orso, “AMNESIA: analysis and monitoring for Neutralizing SQL-injection attacks”, Year of Publication: 2005

[14]- "Acunetix, Web Security Scanner", Available at: http://www.acunetix.com

[15]- "Wapiti, Web application vulnerability scanner / security auditor", Available at: http://wapiti.sourceforge.net/

[16]- E.Peterson , "Finding SQL Injection with Scrawlr", Available at: http://www.communities.hp.com/securitysoftware/blogs/spilabs/archive/2008/06/24/finding-sql-injection-with-scrawlr.aspx

[10] - Exploit Code