# 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，准备登开发完成后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。

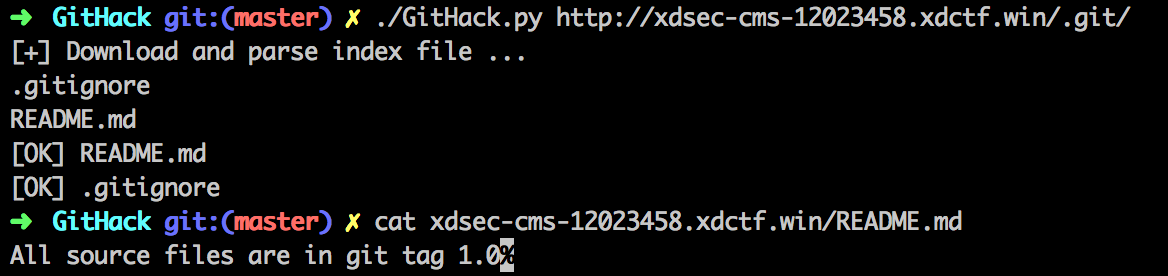
但聪明的小朋友们，你能找到时雨君的源码并发现其中的漏洞么？

”

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/>



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 .

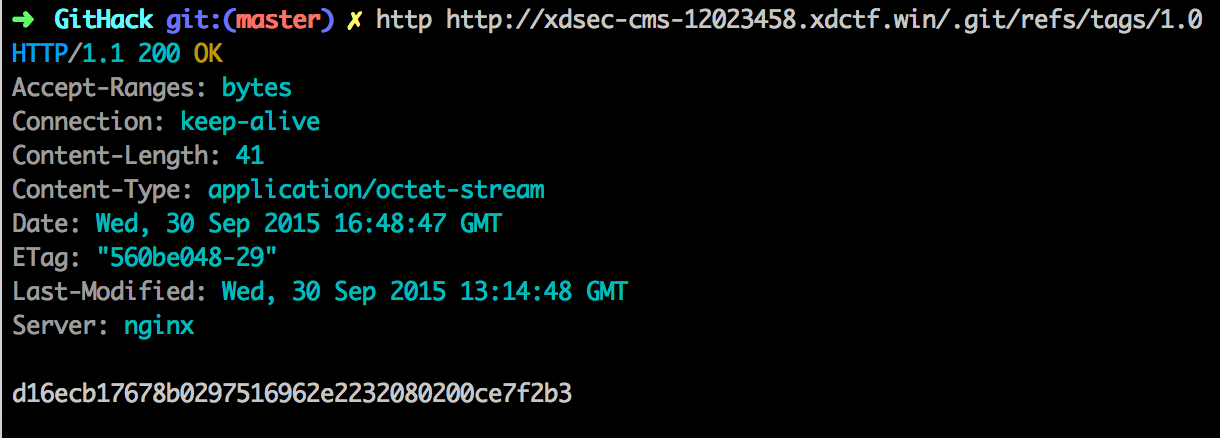
git commit

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：

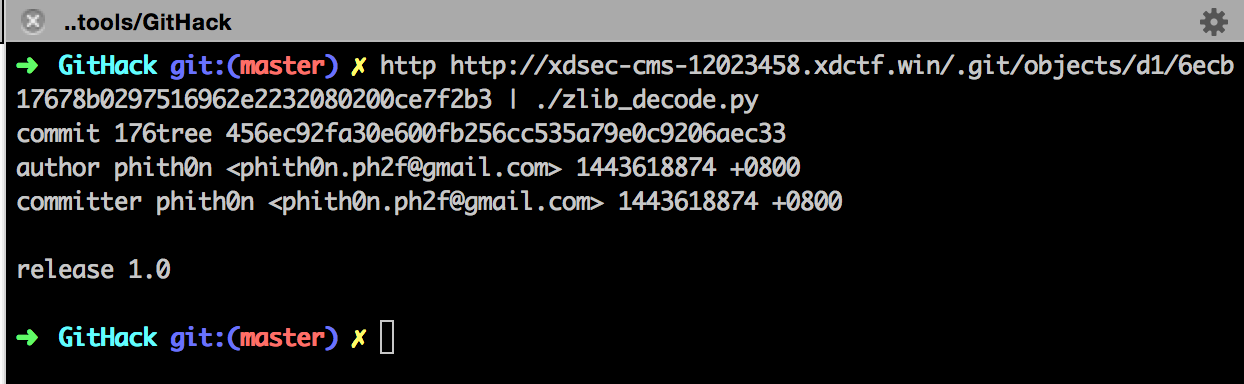


然后简单说一下git object。

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）：



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:

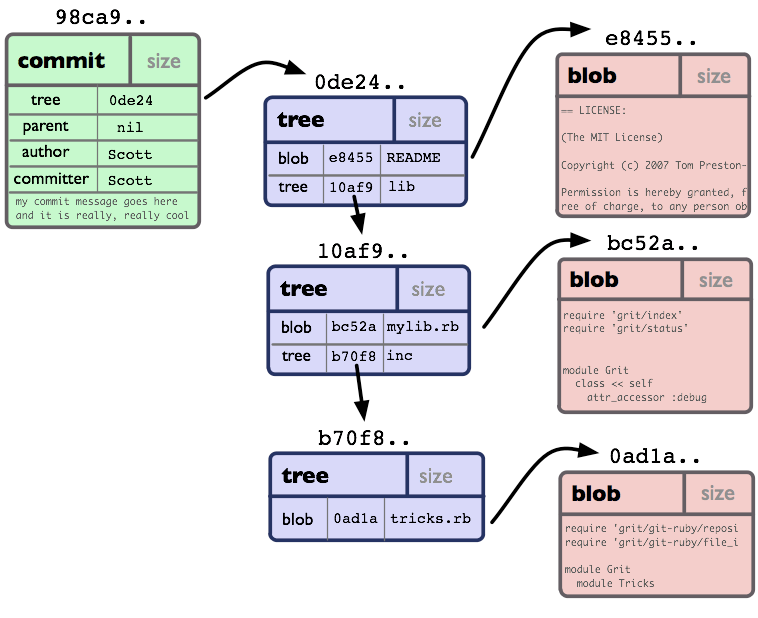


可见，得到一个二进制文件。

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>

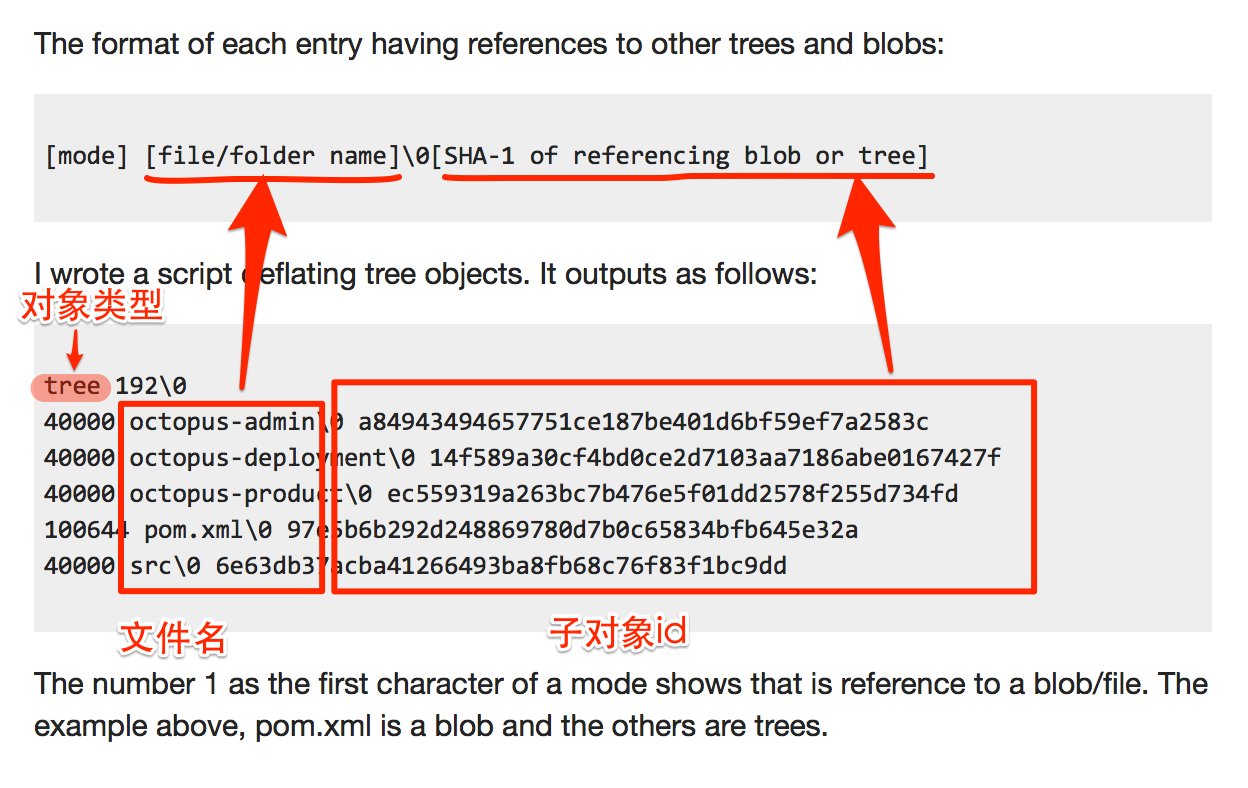
到这一步，我们接下来会接触到的对象就只有“Tree 对象”和“Blob对象”。

这个图可以表示对象间的关系：



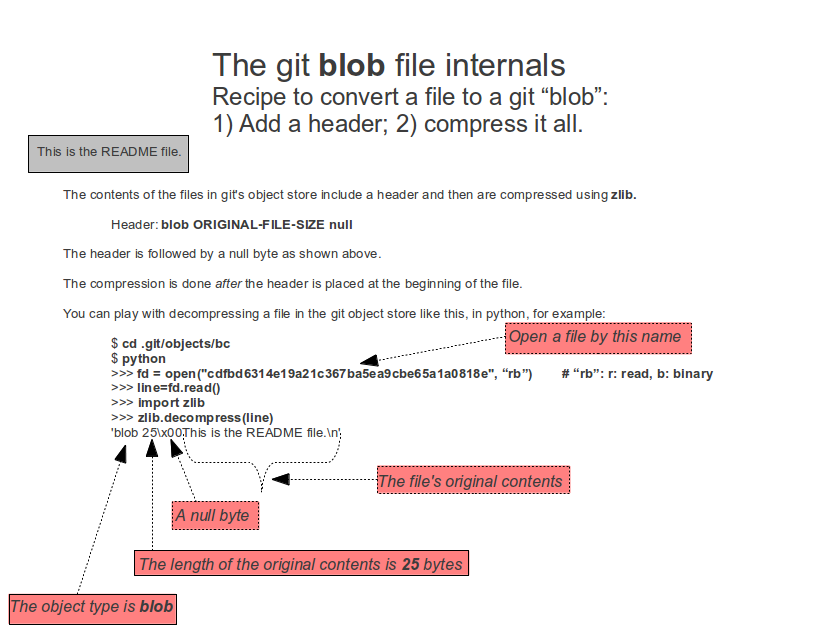
实际上我第一次获取的d16ecb17678b0297516962e2232080200ce7f2b3就是commit对象（绿色），刚才获取的456ec92fa30e600fb256cc535a79e0c9206aec33是tree对象（蓝色），真正保存文件内容的是blob对象（红色）。

那么这个tree对象具体的文件结构是：



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：

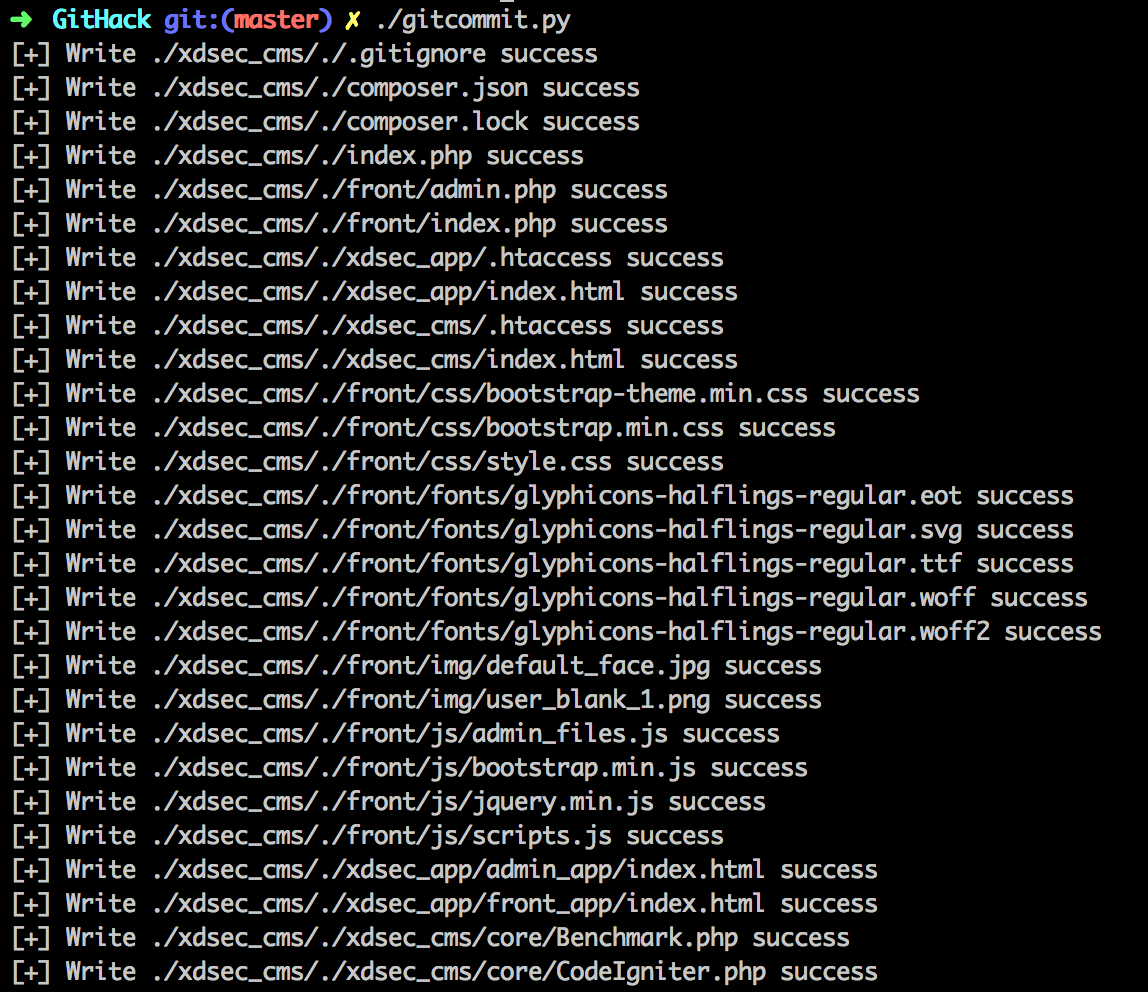


简单说就是：

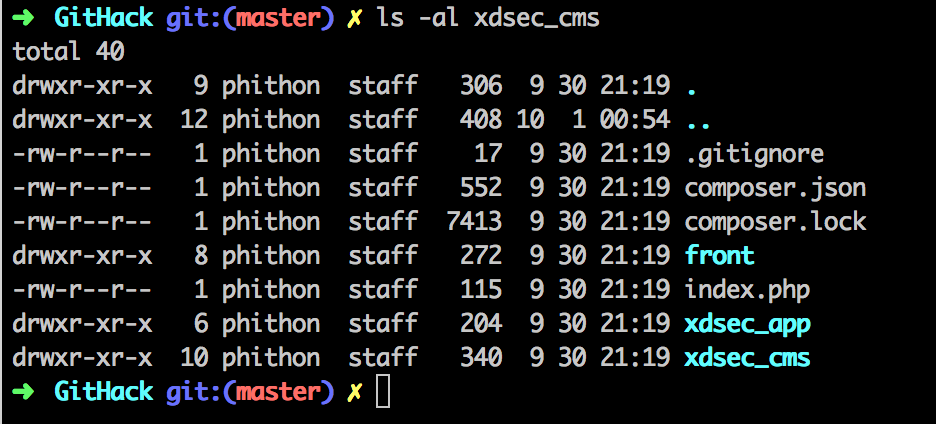
“blob [文件大小]\x00[文件内容]”

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：



如下：



查看index.php，获取到第一个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？

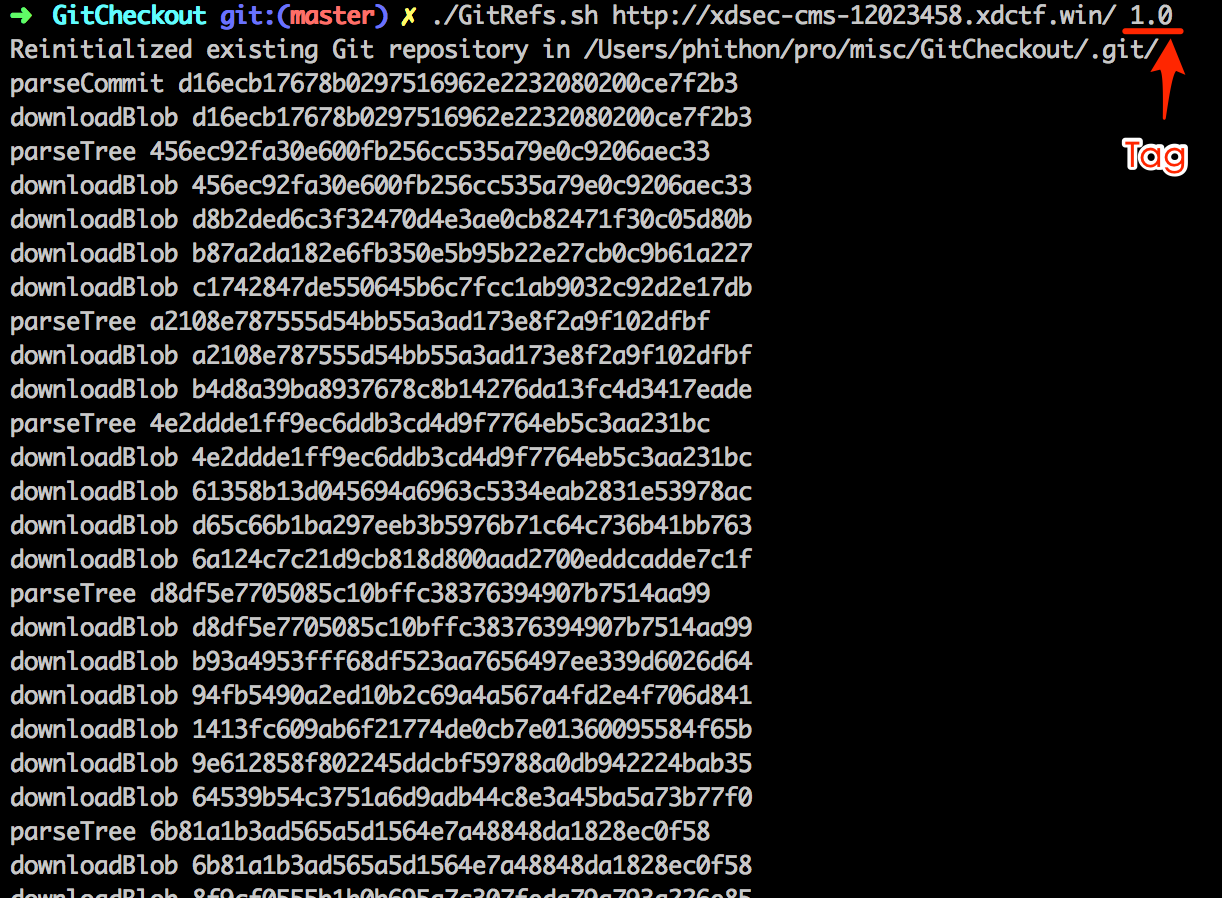
说一下“工具法”。

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就说用来解析类型为”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：



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>