The Vulnerability

On 22nd of October 2013 it was reported by Kees Cook that Vasily Kulikov had discovered a flaw in Linux kernel 3.12.2 (and earlier versions). Linux kernel is a Unix-like operating system kernel used by a variety of operating systems based on it. Kernel is a computer program that manages input/output requests from software and translates them into data processing instructions for the central processing unit and other electronic components of a computer.

The flaw was due to the Linux kernel improperly using the get_dumpable() function, which allowed local users to bypass intended ptrace (by using 'process trace' one program can control another) restrictions or obtain sensitive information from IA64 scratch registers via a crafted application, related to kernel/ptrace.c and arch/ia64/include/asm/processor.h. The get_dumpable() function does not return a boolean value, but in two cases in the kernel it was misused as a boolean.

Get_dumpable() returns the core dump mode for the queried setuid or otherwise protected/tainted binary. Setuid has 3 values:

0 - (SUID_DUMP_DISABLE) - traditional behaviour. Any process which has changed privilege levels or is execute only will not be dumped

1 - (SUID_DUMP_USER) - all processes dump core when possible. The core dump is owned by the current user and no security is applied. This is intended for system debugging situations only.

2 - (SUID_DUMP_ROOT) - any binary which normally not be dumped is dumped readable by root only. This allows the end user to remove such a dump but not access it directly. For security reasons core dumps in this mode will not overwrite one another or other files. This mode is appropriate when admins are attempting to debug problems in a normal environment.

The fix

Since 0 and 2 are protected states, most users of the function should be testing for non-SUID_DUMP_USER(1) rather than SUID_DUMP_DISABLE(0). Wrong logic to be protective:

```c
if (dumpable == 0) 
or 
if (dumpable == SUID_DUMP_DISABLE) 
or 
if (!dumpable)
```

Correct logic would be:
if (dumpable != SUID_DUMP_USER)
or
if (dumpable != 1)

The wrong logic was used in two Linux files in the following way:

if (unlikely(!get_dumpable(current->mm)))

which was changed to

if (unlikely(get_dumpable(current->mm) != SUID_DUMP_USER))

and

if (!dumpable && !ptrace_has_cap(__task_cred(task)->user_ns, mode))

was changed to

if (dumpable != SUID_DUMP_USER && !ptrace_has_cap(__task_cred(task)->user_ns, mode))

**Summary**

We can see that just by wrongly assuming a return value, the programmer can leave security vulnerabilities into the code. In this case, as we saw, if the system had set the SUID_DUMPABLE = 2, the attacker would have been able to ptrace attach to processes that had had dropped privileges to that attacker.
Referenced materials

http://seclists.org/oss-sec/2014/q1/188

http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=d049f74f2dbe71354d43d393ac3a188947811348


http://www.rapid7.com/db/vulnerabilities/linuxrpm-cesa-2014-0285