

[UNIX] Linux 2.6 Kernel Capability LSM Module Local Privilege Elevation

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2004-12/0104.html>

From: SecuriTeam (support_at_securiteam.com)

Date: 12/27/04

To: list@securiteam.com

Date: 27 Dec 2004 16:16:05 +0200

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

The SecuriTeam alerts list – Free, Accurate, Independent.

Get your security news from a reliable source.

<http://www.securiteam.com/maillinglist.html>

Linux 2.6 Kernel Capability LSM Module Local Privilege Elevation

SUMMARY

POSIX.1e Capability is a very important component of Linux kernel. In the original Linux kernel, system security relied on it and DAC. In new the kernel version, Linux Security Modules (LSM) framework was introduced to provide a lightweight, general-purpose framework for access control. Some Linux security projects were ported to LSM and accepted by kernel source, such as POSIX.1e Capability and SE-Linux. Users can compile Capability as a Linux Loadable Kernel Module, and insert it into kernel at any time he or she wants to. Under this situation, after inserting Capability module, due to an error in handling of credentials existing processes (normal user processes) will posses root privileges.

DETAILS

When privileged operations are controlled by the Capability LSM module, it mediates privileged operations based on the credentials of processes. The credentials consists of three fields of task_struct, namely, cap_permitted, cap_inheritable and cap_effective.

Before a user processes carries out a privileged operation (such as

Securiteam: [UNIX] Linux 2.6 Kernel Capability LSM Module Local Privilege Elevation

sethostname, override DAC, raw IO etc.), the operating system will check cap_effective field (in cap_capable hooks function of security/commoncap.c (2.6.*) or security/capability.c (2.5.72-lsm1)):

```
if (cap_raised (tsk->cap_effective, cap)))
    return 0;
else
    return -EPERM;
```

In general, only root processes can possess Capability privileges.

When the Capability is enabled in the kernel and no other LSM security modules are loaded, the kernel uses default security function operations (security/dummy.c) to mediate privilege operation.

The logic of dummy operations is very simple: if a process wants to perform a privileged operation, the euid property of it must be zero (root), or fsuid property must be zero when this privileged operation involved with file system.

However, dummy operations do nothing about credentials of processes, the credentials of any process is a clone of its parent process. As results, the credentials of all process, even normal user process, are the same as that of the Init process. As the Init process is a privileged process, it is assigned total Capability privileges when the system starts.

Unfortunately, after Capability LSM module is loaded, it does not recomputed the credentials of existing processes (before the insertion of Capability module).

Before the insertion, only root processes could have performed privileged operations controlled, as it is defined by the dummy operations. After insertion, the control of privileged operations is switched from the dummy operations to the Capability module based on the process's credentials.

As a result, all existing processes gain the same privileges as that set to the Init process, even if they are normal user processes.

Example:

Before loading Capability module, run a vim editor as a normal user. In vim, enter command ":r /etc/shadow" vim response "can't open file /etc/shadow" the request to access a root file is denied.

Don't end vim, switch to another console and login as root, insert Capability module into kernel.
#modprobe capability

After inserting, go back to vim and try to open file /etc/shadow again, you will find you can read, edit and save(w!) this file as a normal user! The reason for this wrong access control is error credentials of vim so as to it has Capability privilege CAP_DAC_OVERRIDE and CAP_DAC_READ_SEARCH.

Securiteam: [UNIX] Linux 2.6 Kernel Capability LSM Module Local Privilege Elevation

Let's view the credentials with a shell command.

```
$cat /proc/2454/status (2454 is the pid of vim)
```

```
Name: vim
State: S (sleeping)
SleepAVG: 91%
Tgid: 2454
Pid: 2454
PPid: 1552
TracerPid: 0
Uid: 500 500 500 500
Gid: 500 500 500 500
FDSize: 256
Groups: 500
VmSize: 9356 kB
VmLck: 0 kB
VmRSS: 2728 kB
VmData: 856 kB
VmStk: 16 kB
VmExe: 1676 kB
VmLib: 3256 kB
Threads: 1
SigPnd: 0000000000000000
ShdPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 8000000000003000
SigCgt: 00000000ef824eff
CapInh: 0000000000000000
CapPrm: 00000000ffffff
CapEff: 00000000ffffeff
```

The last three lines are the credentials of vim, it has all Capability privileges besides CAP_SETPCAP.

Above test was performed under 2.6.* and 2.5.72-lsm1.

Workaround:

In order to fix this bug, we may have two methods. One is moving the computation of credentials from Capability module to kernel. The other is adding some code in Capability module to re-compute credentials of existed process. flashsky choose the second method, add following code in security/capability.c:

```
static void recompute_capability_creds(struct task_struct *task)
{
    if(task->pid <= 1)
        return;
```

Securiteam: [UNIX] Linux 2.6 Kernel Capability LSM Module Local Privilege Elevation

```
task_lock(task);
task->keep_capabilities = 0;

if ((task->uid && task->euid && task->suid) &&
!task->keep_capabilities)
    cap_clear (task->cap_permitted);
else
    task->cap_permitted = CAP_INIT_EFF_SET;

if (task->euid != 0){
    cap_clear (task->cap_effective);
}
else{
    task->cap_effective = CAP_INIT_EFF_SET;
}

if(task->fsuid)
    task->cap_effective &= ~CAP_FS_MASK;
else
    task->cap_effective |= CAP_FS_MASK;

task_unlock(task);

return;
}
```

And add following code in Capability init function `capability_init` before it return:

```
struct task_struct *task;

read_lock(&tasklist_lock);
for_each_process(task){
    recompute_capability_creds(task);
}
read_unlock(&tasklist_lock);

return 0;
```

After modifying `capability.c`, we need "make" and "make modules_install" again. Unload Capability module (`rmmod capability; rmmod commoncap`) and retry the above example, all the accesses to a root file by normal user existed process are always denied before and after inserting Capability module.

Test and view the credentials again.

```
$cat /proc/(pid of vim)/status
```

Securiteam: [UNIX] Linux 2.6 Kernel Capability LSM Module Local Privilege Elevation

Name: vim
State: S (sleeping)
SleepAVG: 91%
Tgid: 2864
Pid: 2864
PPid: 1552
TracerPid: 0
Uid: 500 500 500 500
Gid: 500 500 500 500
FDSize: 256
Groups: 500
VmSize: 9360 kB
VmLck: 0 kB
VmRSS: 2816 kB
VmData: 860 kB
VmStk: 16 kB
VmExe: 1676 kB
VmLib: 3256 kB
Threads: 1
SigPnd: 0000000000000000
ShdPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 8000000000003000
SigCgt: 00000000ef824eff
CapInh: 0000000000000000
CapPrm: 0000000000000000
CapEff: 0000000000000000

ADDITIONAL INFORMATION

The information has been provided by <<mailto:liangbin@venustech.com.cn>>
LiangBin.

=====

This bulletin is sent to members of the SecuriTeam mailing list.
To unsubscribe from the list, send mail with an empty subject line and body to:
list-unsubscribe@securiteam.com
In order to subscribe to the mailing list, simply forward this email to: list-subscribe@securiteam.com

=====
=====

DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.
In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.