

Linux 2.6 Kernel Capability LSM Module Local Privilege Elevation

Source: <http://www.derkeiler.com/Mailing-Lists/securityfocus/bugtraq/2004-12/0390.html>

From: flashsky fangxing (flashsky_at_xfocus.org)

Date: 12/23/04

Date: 23 Dec 2004 14:57:01 -0000

To: bugtraq@securityfocus.com

('binary' encoding is not supported, stored as-is)

Venustech AD-Lab
www.venustech.com.cn

[Security Advisory]

Advisory: [AD_LAB-04003]Linux 2.6.* Kernel Capability LSM Module Local Privilege Elevation

Authors: liangbin@venustech.com.cn

Release: 09/12/04

Class: Design Error

Remote: No, local

Vulnerable: Linux kernel 2.6.*

Linux kernel 2.5.72-lsm1

Unvulnerable: Linux kernel 2.4

Vendor: <http://www.kernel.org/>

L.INFO:

When POSIX Capability LSM module isn't compiled into kernel, after inserting Capability module into kernel, all existed normal users processes will have total Capability privileges of superuser (root).

POSIX.1e Capability is a very important component of Linux kernel. In original Linux Kernel, system security relies on it and DAC mainly. In new kernel version, Linux Security Modules (LSM) framework is introduced to provide a lightweight, general-purpose framework for access control. Some Linux security projects are 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 wants to. Under this situation, after inserting Capability module, due to error

SecurityFocus Bugtraq: Linux 2.6 Kernel Capability LSM Module Local Privilege Elevation

creds of existing processes, normal user processes will possess total privileges of root and can perform any operations (like a root process).

II.DESCRPTION:

When the privileged operations are controlled by Capability LSM module, it mediates privileged operations base on the creds of processes. The creds consists of three fields of task_struct, namely, cap_permitted, cap_inheritable and cap_effective. Before user processes carry out privileged operations (such as sethostname, override DAC, raw IO etc.), system will check cap_effective field.

(in cap_capable hooks funcation 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 Capability don't run in kernel and no other LSM security modules are loaded, kernel uses default security function ops (security/dummy.c) to mediate privilege operation. The check logic of dummy ops 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 ops do nothing about creds of processes, the creds of any process is a clone of its parent process. As results, the creds of all process, even normal user process, are as same as that of Init process. Init process is a privileged process, it is assigned total Capability privileges in the creds of it when system initiate.

Unfortunately, after Capability LSM module is loaded, it don't recomputed the creds of processes those are existing before inserting Capability module! Before inserting, only root processes can perform privileged operations controlled by dummy ops based *uids correctly. After inserting, the control of privileged operations is switch from dummy ops to Capability module based on creds. As a result, all existing processes have privileges as same as init, even they are normal user processes. A normal user (maybe a malicious user) can perform any operations through these processes!

IV.AN 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

SecurityFocus Bugtraq: Linux 2.6 Kernel Capability LSM Module Local Privilege Elevation

Capability module into kernel.

```
#modprobe capability
```

After inserting, 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 creds of vim so as to it has Capability privilege CAP_DAC_OVERRIDE and CAP_DAC_READ_SEARCH.

Let's view the creds 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 creds of vim, it has all Capability privileges besides CAP_SETPCAP.

Above test is perform in 2.6.* and 2.5.72-lsm1.

V.WORKAROUND:

In order to fix this bug, we may have two methods. One is moving the computation of creds from Capability module to kernel. The other is adding

SecurityFocus Bugtraq: Linux 2.6 Kernel Capability LSM Module Local Privilege Elevation

some code in Capability module to re-compute creds of existed process. I 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;

    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;
```

SecurityFocus Bugtraq: Linux 2.6 Kernel Capability LSM Module Local Privilege Elevation

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 creds again.

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

```
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
```

VI.CREDIT:

LiangBin(liangbin@venustech.com.cn) discovery this vuln:)
Vulnerability analysis and advisory by LiangBin and icbm.
Special thanks to "Fengshou" project members and all Venustech AD-Lab guys:P

VII.DISCLAIMS:

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,

SecurityFocus Bugtraq: Linux 2.6 Kernel Capability LSM Module Local Privilege Elevation
indirect, incidental, consequential, loss of business profits or special damages.

Copyright 1996–2004 VENUSTECH. All Rights Reserved. Terms of use.

VENUSTECH AD–Lab

VENUSTECH INFORMATION TECHNOLOGY CO.,LTD(<http://www.venustech.com.cn>)

Product
Trusted Security {Solution} Provider
Service