COSEINC LINUX ADVISORY #1

Linux Kernel Parent Process Death Signal Vulnerability

===[ABSTRACT]==========

The document describes two technically unrelated issues discovered in the Linux kernel that when exploited together may lead to local privileges escalation.

===[IMPACT]==========

Under some circumstances any local user may gain root privileges.

===[AFFECTED SOFTWARE]==========

Linux 2.6	tested on v2.6.23-rc1
Linux 2.4	not tested

===[DESCRIPTION]==========

ISSUE #1:

An unprivileged user may send arbitrary signal to a child process even if it is running with higher privileges.

When a parent process dies or exits its child processes may receive a signal. Each child process may choose and set its own "parent process death signal" using PR_SET_PDEATHSIG function of the prctl() system call.

```
PARENT                  CHILD
-------------------------------------------------------------
fork()
                        prctl(PR_SET_PDEATHSIG)
exit()'ed or killed
                        child receives the signal
```

The parent process death signal is not reset over execve() system call and is inherited by spawned process:

```
PARENT                  CHILD
-------------------------------------------------------------
fork()
                        prctl(PR_SET_PDEATHSIG)
                        execve("./a.out")
exit()'ed or killed
                        child receives the signal
```

The signal gets delivered only if parent process has sufficient privileges to send signals to child processes. Typically any child process running with higher privilege than its parent will receive no

signal.

```
PARENT                  CHILD
---------------------------------------------------------------
fork()
                        prctl(PR_SET_PDEATHSIG)
                        execve("/bin/setuid-binary")
exit()'ed or killed
                        child receives NO signal this time
```

However, above restriction may be bypassed if parent process execute setuid-root binary which dies afterwards.

```
PARENT                  CHILD
---------------------------------------------------------------
fork()
                        prctl(PR_SET_PDEATHSIG)
                        execve("/bin/setuid-binary")
execve("/bin/setuid-binary")
exit()'ed or killed
                        privileged process receives the signal
```

ISSUE #2:

The execve() system call may enable the dumpable flag for privileged process thus disabling the required protection against ptracing and dumping core.

Typically dumpable flag is cleared whenever a privileged process is started like setuid binary.

Whenever privileged process drops user privileges (i.e. changes all its UIDs and GIDs to privileged UID/GID) and then execute another binary the dumpable flag is reinitialized and set to true. The problem is illustrated below (example for typical setuid-root binary):

```
DUMPABLE    ACTION
---------------------------------------------------------------
true
                        execve("/bin/setuid-binary")
false
                        setuid(geteuid())
                        setgid(getegid())
                        setgroups()
                        execve("/bin/any-binary")
true
                        ...
```

Such a task may be forced to dump core when fatal signal gets delivered to the process (like SIGSEGV) or any other user-controllable signal raises like SIGXCPU, SIGXFSZ or SIGQUIT sent from keyboard.

===[EXPLOITATION]==

The two issues described above were successfully exploited using a proof of concept code. The exploit lets any local user to gain root privileges under specific conditions.

Successful exploitation depends on additional software that must meet specific conditions described in the advisory.

The exploit code will not be published.

===[DISCLOSURE TIMELINE]==========

27th July 2007 Vendor notification

===[AUTHOR]==========

Wojciech Purczynski <cliph@research.coseinc.com>

Wojciech Purczynski is a Security Researcher at Vulnerability Research Labs, COSEINC PTE Ltd. Wojciech Purczynski is also a member of iSEC Security Research.

===[LEGAL DISCLAIMER]==========