IOActive Security Advisory

Title	Mach Exception Handling Privilege Escalation
Severity	Medium
Date Discovered	January 5, 2010
Discovered by	Richard van Eeden

Affected Products

Apple Mac OS X 10.5.x and 10.6

Impact

Local users can execute arbitrary code with root privileges.

Technical Details

Mach exception handling suffers from a vulnerability that allows an attacker to gain access to the memory of a `suid` process (set user identifer). Due to a vulnerability that's similar to CVE-2006-4392 (found by Dino Dai Zovi of Matasano Security), it's possible for a `suid` process to inherit the Mach exception ports of the parent.

The `catch_exception_raise`, `catch_exception_raise_state`, and `catch_exception_raise_state_identity` callbacks have send rights to the thread that generated the exception, which means that a lesser privileged process will be able to modify the task's address space once an exception occurs. The code that is responsible for resetting the Mach exceptions ports can be bypassed by executing the `suid` binary in a `vfork()`, as shown in the following code sample:

```
File: sys/bsd/kern/kern_exec.c
2745 static int
2746 exec_handle_sugid(struct image_params *imgp)

[…]

2855                 /*
2856                  * Have mach reset the task and thread
ports.
2857                  * We don't want anyone who had the ports
before
2858                  * a setuid exec to be able to
access/control the
2859                  * task/thread after.
```

```
2860                     */
2861                    if (current_task() == p->task) {
2862                            ipc_task_reset(p->task);
2863                            ipc_thread_reset(current_thread());
2864                    }
2865
```

Proof of Concept

IOActive has developed a proof of concept that gains root privileges on Mac OS. In order to exploit this vulnerability an attacker has to:

1. Install an exception handler using `task_set_exception_ports()`.
2. Execute a `suid` binary in a `vfork()` with the RLIMIT_STACK `rlimit` set to a small value, which forces a crash.
3. Write the shellcode to the `suid` process using `vm_allocate and vm_write`.
4. Set the program counter of the thread with `thread_set_state()` to the shellcode location.

```
[richard@research ~]# ./golden_delicious

[!] Executing "/sbin/ping" ...

sh-3.2# id
uid=0(root) gid=0(wheel) groups=0(wheel)
sh-3.2#
```