DragonFlyBSD - Bug #2019

panic: file desc: malloc limit exceeded

03/05/2011 05:28 PM - smag

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	vsrinivas	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

I have a crontab entry set to run pkgsrc-update and src-update. Besides that, some WSGI process running with Gunicorn (http://gunicorn.org) proxied by nginx with access to pgsql. These are not very demanded, mainly testing webapps. It also hosts a SnapLogic (http://snaplogic.org/) server.

The core.txt file is attached.

info shows:

Dump header from device /dev/serno/0606J1FW203856.s1b
Architecture: i386
Architecture Version: 2
Dump Length: 141434880B (134 MB)
Blocksize: 512
Dumptime: Thu Mar 3 19:21:54 2011
Hostname: dragon.sebasmagri.com
Magic: DragonFly Kern Dump
Version String: DragonFly v2.9.1.747.gf7b29d-DEVELOPMENT #0: Mon Feb 21 15:27:59 VET 2011
smag@dragon.sebasmagri.com:/usr/obj/usr/src/sys/GENERIC
Panic String: file desc: malloc limit exceeded
Dump Parity: 3394002437
Bounds: 0
Dump Status: good

My bandwidth is not very good, but I'm going to put the dumps available to fetch if it's needed. I'm also willing to help testing fixes.

History

#1 - 03/05/2011 10:49 PM - vsrinivas

Started working on it...

#2 - 03/06/2011 04:27 AM - vsrinivas

This is the start of a patch to handle overflows of this malloc zone. It is not enough, internal to fdcopy() there is a loop around the allocation of fd arrays; the loop allocations have not yet been corrected.

```
diff --git a/sys/kern/kern_descrip.c b/sys/kern/kern_descrip.c
--- a/sys/kern/kern_descrip.c
+++ b/sys/kern/kern_descrip.c
@@ -1813,8 +1813,8 @@ fdshare(struct proc *p)
 *
 * MPSAFE
 */
-struct filedesc *
-fdcopy(struct proc *p)
+int
+fdcopy(struct proc *p, struct filedesc **fpp)
{
struct filedesc *fdp = p->p_fd;
struct filedesc *newfdp;
@@ -1826,14 +1826,19 @@ fdcopy(struct proc *p)
```

```
* Certain daemons might not have file descriptors.
*/
if (fdp == NULL)
-               return (NULL);
+               return (0);

/*
* Allocate the new filedesc and fd_files[] array.  This can race
* with operations by other threads on the fdp so we have to be
* careful.
*/
-       newfdp = kmalloc(sizeof(struct filedesc), M_FILEDESC, M_WAITOK |
M_ZERO);
+       newfdp = kmalloc(sizeof(struct filedesc),
+                        M_FILEDESC, M_WAITOK | M_ZERO | M_NULLOK);
+       if (newfdp == NULL) {
+               *fpp = NULL;
+               return (-1);
+       }
again:
spin_lock(&fdp->fd_spin);
if (fdp->fd_lastfile < NDFILE) {
@@ -1925,7 +1930,8 @@ again:
}
}
spin_unlock(&fdp->fd_spin);
-       return (newfdp);
+       *fpp = newfdp;
+       return (0);
}

/*
diff --git a/sys/kern/kern_exec.c b/sys/kern/kern_exec.c
--- a/sys/kern/kern_exec.c
+++ b/sys/kern/kern_exec.c
@@ -338,7 +338,9 @@ interpret:
if (p->p_fd->fd_refcnt > 1) {
struct filedesc *tmp;

-               tmp = fdcopy(p);
+               error = fdcopy(p, &tmp);
+               if (error != 0)
+                       goto exec_fail;
fdfree(p, tmp);
}

diff --git a/sys/kern/kern_fork.c b/sys/kern/kern_fork.c
--- a/sys/kern/kern_fork.c
+++ b/sys/kern/kern_fork.c
@@ -283,7 +283,11 @@ fork1(struct lwp *lp1, int flags, struct
if (flags & RFFDG) {
if (p1->p_fd->fd_refcnt > 1) {
struct filedesc *newfd;
-                               newfd = fdcopy(p1);
+                               error = fdcopy(p1, &newfd);
+                               if (error != 0) {
+                                       error = ENOMEM;
+                                       goto done;
+                               }
fdfree(p1, newfd);
}
}
diff --git a/sys/sys/filedesc.h b/sys/sys/filedesc.h
--- a/sys/sys/filedesc.h
+++ b/sys/sys/filedesc.h
@@ -164,7 +164,7 @@ void        fsetcred (struct file *fp, struct u
void   fdinit_bootstrap(struct proc *p0, struct filedesc *fdp0, int cmask);
struct filedesc *fdinit (struct proc *p);
struct filedesc *fdshare (struct proc *p);
-struct filedesc *fdcopy (struct proc *p);
+int    fdcopy (struct proc *p, struct filedesc *fpp);
void   fdfree (struct proc *p, struct filedesc *repl);
int    fdrevoke(void *f_data, short f_type, struct ucred *cred);
int    closef (struct file *fp, struct proc *p);
```

#3 - 03/06/2011 01:22 PM - vsrinivas

A sample program to exhaust this limit; bump the fork parameter as you see fit.

```
main() {
int i, j;
dup2(0, 3500);
for (i = 0 ; i < 1280; i++) {
j = fork();
if (j == 0)
pause();
}
pause();
}
```

#4 - 03/06/2011 01:24 PM - vsrinivas

Commit 2994659f1e6c1ef260241491bceca91c9d2553b3 is a partial fix to the problem;
it does not handle overflows in the spinlock loop path in fdcopy and it is still
possible to make the system unusable with the sample program posted below.

Perhaps we should also raise the malloc zone limit to maxproc * MAX_FDS_PER_PROC?

#5 - 03/06/2011 04:56 PM - dillon

:Venkatesh Srinivas <vsrinivas@dragonflybsd.org> added the comment:
:
:Commit 2994659f1e6c1ef260241491bceca91c9d2553b3 is a partial fix to the problem;
:it does not handle overflows in the spinlock loop path in fdcopy and it is still
:possible to make the system unusable with the sample program posted below.
:
:Perhaps we should also raise the malloc zone limit to maxproc * MAX_FDS_PER_PROC?

No, won't work, the maximum will baloon well past any reasonable limit
when you try to do that.

We have a kern.maxfilesperuser that's supposed to handle that sort of
attack, is it not working? It might not be applicable to root though.

-Matt
Matthew Dillon
<dillon@backplane.com>

#6 - 03/06/2011 05:02 PM - dillon

Hmm. Clearly kern.maxfilesperuser isn't going to help for the
sparse file descriptor table attack. The defaults on an i386
box seem to be on the order of 6000 processes x 25000 descriptors
per process, which winds up being significant greater than a gigabyte
of ram (let alone kvm)... so it goes boom.

I think we do have to apply the maxfilesperuser limit to this situation
counted based on the size of the fd table instead of based on the number
of actual descriptors. That would handle the situation.

-Matt
Matthew Dillon
<dillon@backplane.com>

Files

core.txt	112 KB	03/06/2011	smag