

# [patch] lockf(3) user-exploitable kernel panic

**Source:** <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/arch/2004-04/0032.html>

---

**From:** Devon H. O'Dell ([dodell\\_at\\_sitetronics.com](mailto:dodell_at_sitetronics.com))

**Date:** 04/14/04

Date: Wed, 14 Apr 2004 10:26:32 +0200

To: [freebsd-arch@freebsd.org](mailto:freebsd-arch@freebsd.org)

Hello arch@,

cperciva@ noted to me a few days ago that he'd received an email from a person who had found a user-exploitable panic in FreeBSD involving POSIX-type advisory-mode locks. The theory of the exploit is that one creates a significantly large file (>3 times the size of the amount of kernel memory) and lock non-contiguous chunks of the file. I've seen and tested a proof-of-concept for this exploit; it certainly panics the system with a non-privileged account. Per cperciva@'s suggestion, I'm asking if arch@ would like to give me some input / feedback on this patch.

The enclosed patches (available on <http://freebsd0.sitetronics.com/~dodell/patches/lockfix.tar.gz> as well) fix this problem by creating a new rlimit that limits the maximum number of POSIX-type advisory-mode locks a user can hold at one time.

There are a couple of immediate problems I see with the patch myself:

- 1) I have to pass a struct proc \* to change\_ruid. If a user changes his/her uid, the number of advisory-mode locks needs to be transferred to the new uid and the only way I figured to do that would be to count the number of advisory-mode locks held by a process (I didn't need to track this across a fork() since POSIX locks are not inherited between processes). This means I have to move the definition out of /sys/sys/ucrd.h and into /sys/kern/kern\_prot.c. It also means that OSF/1 compatibility becomes broken on Alpha, since the setuid() in osf1\_misc.c calls the change\_[r|e|sv]uid functions that have been implemented in /sys/kern/kern\_prot.c. Solutions to this include:
  - a) creating a SuSv3-compatible setuid for use with the OSF/1, SVR4 and Linux compat ABIs (since using the BSD setuid for these also isn't totally correct).
  - b) require sys/proc.h wherever sys/ucrd.h is included (which is very ugly)
  - c) move the check out of change\_ruid() (I don't think this is correct since the chgprocnt() function is called there as well)
  - d) re-write the OSF/1 compatibility code to use its own change\_ruid\_osf1() function (bloated)

## freebsd-arch: [patch] lockf(3) user-exploitable kernel panic

2) Terms I've used for variables are a little bit ambiguous. struct flock (BSD-style) locks, which will lock a whole file are also ``advisory-mode" locks. This being the first time I've worked in depth with the kernel and SuSv3 specifications, I didn't know that there was a difference. My sysctl variable is also rather ugly. What names would you guys suggest I use instead of ``maxadvlocks"? I was thinking something along the lines of running a s/advlock/posixlock/ on the code. This is a simple fix, but an important one nonetheless.

3) Does this work justify my going through the modified files and doing style(9) changes on them? I'm willing to do this; mux@ has encouraged it; style(9) suggests that I do it if my code comprises 50% or more of the new files (which it doesn't). Again, if this is useful, I'll certainly do this.

4) I had to change lf\_split() to return a value, since a lock is potentially allocated inside it, and I couldn't previously return from that function if the split created a lock overflowing the user's limit. Is this a problem?

5) With regards to SMPng-fu, mine is probably sub-par compared to yours; are my locks / assertions correct? Am I missing a uidinfo lock in chgadvlockent()?

6) I'm not sure that 8192 locks is enough for normal user operations, but I haven't been able to test this on a desktop or server. Someone who has been able to reported that 8192 was not enough to run Enlightenment (or anything that talked to gconfd), but I think this was before I realized that F\_POSIX locks were different.

7) As previously mentioned, a modification in /sys/sys/proc.h adds an int p\_numadvlocks; to struct proc. Is this acceptable? I've not done anything special to lock it and I've marked it with the status (\*). Is this something I need to fix?

8) Are any of the modifications I've made too intrusive to the [proc|advlock|rlimit|sysctl] subsystem(s)?

9) What (extra) suggestions would you have for my patches for relevant manpages?

10) Have I missed any userland utilities that don't use libutil to check/set classes/limits (perhaps there are some in ports that I can patch as well)?

This patch is against April 13th -CURRENT but backporting it is very simple since the main affected subsystem doesn't change much architecturally / structurally. However, this also brings into light that this problem may also affect the other BSDs (Dragonfly, Net, Open, Ekko). I cannot verify this as I do not have much experience with these other BSDs and do not know if they impose any limits on the amount of

freebsd-arch: [patch] lockf(3) user-exploitable kernel panic

kernel memory a user can have or any other limits which would disallow this to exploit to ``work". Should they be affected, what do I need to do to alert them of this?

Sorry for the somewhat ``needy" email; again, this is the first time I've developed a kernel-level patch and I'm not extremely familiar with the architectural requirements for developing one.

Thanks for your time!

Kind regards,

Devon H. O'Dell

```
diff -ur etc/login.conf etc_lockfix/login.conf
--- etc/login.conf Tue Jun 25 21:04:37 2002
+++ etc_lockfix/login.conf Tue Apr 13 12:48:10 2004
@@ -33,6 +33,7 @@
:coredumpsize=unlimited:\
:openfiles=unlimited:\
:maxproc=unlimited:\
+ :advlocks=unlimited:\
:sbsize=unlimited:\
:vmemoryuse=unlimited:\
:priority=0:\
```

```
diff -ur bin/sh/miscbltin.c bin_lockfix/sh/miscbltin.c
--- bin/sh/miscbltin.c Wed Apr 14 00:13:05 2004
+++ bin_lockfix/sh/miscbltin.c Wed Apr 14 00:12:27 2004
@@ -342,6 +342,9 @@
#ifdef RLIMIT_SBSIZE
    { "sbsize", "bytes", RLIMIT_SBSIZE, 1, 'b' },
#endif
+#ifdef RLIMIT_ADVLOCK
+ { "advlocks", (char *)0, RLIMIT_ADVLOCK, 1, 'k' },
+#endif
    { (char *) 0, (char *)0, 0, 0, '\0' }
};
```

```
@@ -358,7 +361,7 @@
    struct rlimit limit;

    what = 'f';
- while ((optc = nextopt("HSatfdsmcnuvlb")) != '\0')
+ while ((optc = nextopt("HSatfdsmcnuvlbk")) != '\0')
    switch (optc) {
        case 'H':
```

freebsd-arch: [patch] lockf(3) user-exploitable kernel panic

how = HARD;

```
diff -ur lib/libc/sys/getrlimit.2 lib_lockfix/libc/sys/getrlimit.2
--- lib/libc/sys/getrlimit.2 Tue Apr 13 23:53:52 2004
+++ lib_lockfix/libc/sys/getrlimit.2 Tue Apr 13 23:58:24 2004
@@ -98,6 +98,9 @@
The maximum size (in bytes) of socket buffer usage for this user.
This limits the amount of network memory, and hence the amount of
mbufs, that this user may hold at any time.
+.It Li RLIMIT_ADVLOCK
+The maximum number of POSIX-type (lockf(3) style) advisory-mode
+locks available to this user.
.El
.Pp
A resource limit is specified as a soft limit and a hard limit. When a
diff -ur lib/libutil/login.conf.5 lib_lockfix/libutil/login.conf.5
--- lib/libutil/login.conf.5 Wed Apr 14 00:03:21 2004
+++ lib_lockfix/libutil/login.conf.5 Wed Apr 14 00:00:28 2004
@@ -167,6 +167,7 @@
.It "sbsize size Maximum permitted socketbuffer size.
.It "vmemoryuse size Maximum permitted total VM usage per process.
.It "stacksize size Maximum stack size limit.
+.It "advlocks size Maximum number of POSIX-type advisory-mode locks.
.El
.Pp
These resource limit entries actually specify both the maximum
diff -ur lib/libutil/login_class.c lib_lockfix/libutil/login_class.c
--- lib/libutil/login_class.c Tue Apr 13 12:49:17 2004
+++ lib_lockfix/libutil/login_class.c Tue Apr 13 12:43:38 2004
@@ -59,6 +59,7 @@
{ "coredumpsize", login_getcapsize, RLIMIT_CORE },
{ "sbsize", login_getcapsize, RLIMIT_SBSIZE },
{ "vmemoryuse", login_getcapsize, RLIMIT_VMEM },
+ { "advlocks", login_getcapnum, RLIMIT_ADVLOCKS },
{ NULL, 0, 0 }
};

diff -ur sys/kern/kern_lockf.c sys_lockfix/kern/kern_lockf.c
--- sys/kern/kern_lockf.c Tue Apr 13 12:43:16 2004
+++ sys_lockfix/kern/kern_lockf.c Tue Apr 13 23:34:55 2004
@@ -50,6 +50,7 @@
#include <sys/malloc.h>
#include <sys/fcntl.h>
#include <sys/lockf.h>
+#include <sys/resourcevar.h>
```

freebsd-arch: [patch] lockf(3) user-exploitable kernel panic

```
/*
 * This variable controls the maximum number of processes that will
@@ -80,7 +81,7 @@
     lf_getblock(struct lockf *);
static int lf_getlock(struct lockf *, struct flock *);
static int lf_setlock(struct lockf *);
-static void lf_split(struct lockf *, struct lockf *);
+static int lf_split(struct lockf *, struct lockf *);
static void lf_wakelock(struct lockf *);

/*
@@ -100,6 +101,7 @@
{
    register struct flock *fl = ap->a_fl;
    register struct lockf *lock;
+ struct proc *pp = (struct proc *)0;
    off_t start, end, oadd;
    int error;

@@ -156,6 +158,19 @@
/*
 * Create the lockf structure
*/
+ if (ap->a_flags & F_POSIX) {
+ pp = (struct proc *)ap->a_id;
+ if (ap->a_op == F_SETLK) {
+ if (!chgadvlockcnt(pp, 1, lim_max(pp, RLIMIT_ADVLOCK)))
+ return (ENOLCK);
+ } else {
+ /*
+ * We are allowed this lock because we will free it
+ * no matter the outcome of the operation.
+ */
+ chgadvlockcnt(pp, 1, 0);
+ }
+ }
    MALLOC(lock, struct lockf *, sizeof *lock, M_LOCKF, M_WAITOK);
    lock->lf_start = start;
    lock->lf_end = end;
@@ -181,15 +196,21 @@

    case F_UNLCK:
        error = lf_clearlock(lock);
+ if (lock->lf_flags & F_POSIX)
+ chgadvlockcnt(pp, -1, 0);
        FREE(lock, M_LOCKF);
        return (error);

    case F_GETLK:
        error = lf_getlock(lock, fl);
+ if (lock->lf_flags & F_POSIX)
```

freebsd-arch: [patch] lockf(3) user-exploitable kernel panic

```

+ chgadvlockcnt(pp, -1, 0);
    FREE(lock, M_LOCKF);
    return (error);

    default:
+ if (lock->lf_flags & F_POSIX)
+ chgadvlockcnt(pp, -1, 0);
    free(lock, M_LOCKF);
    return (EINVAL);
}
@@ -204,11 +225,14 @@
    register struct lockf *lock;
{
    register struct lockf *block;
+ struct proc *pp = (struct proc *)0;
    struct lockf **head = lock->lf_head;
    struct lockf **prev, *overlap, *ltmp;
    static char lockstr[] = "lockf";
    int ovcase, priority, needtolink, error;
-
+
+ if (lock->lf_flags & F_POSIX)
+ pp = (struct proc *)lock->lf_id;
#ifdef LOCKF_DEBUG
    if (lockf_debug & 1)
        lf_print("lf_setlock", lock);
@@ -229,6 +253,8 @@
    * Free the structure and return if nonblocking.
    */
    if ((lock->lf_flags & F_WAIT) == 0) {
+ if (lock->lf_flags & F_POSIX)
+ chgadvlockcnt(pp, -1, 0);
        FREE(lock, M_LOCKF);
        return (EAGAIN);
    }
@@ -265,6 +291,10 @@
        wproc = (struct proc *)waitblock->lf_id;
        if (wproc == (struct proc *)lock->lf_id) {
            mtx_unlock_spin(&sched_lock);

+ if (lock->lf_flags &
+ F_POSIX)
+ chgadvlockcnt(pp,
+ -1, 0);

            free(lock, M_LOCKF);
            return (EDEADLK);
        }
@@ -309,6 +339,8 @@
        lock->lf_next = NOLOCKF;
    }
    if (error) {
+ if (lock->lf_flags & F_POSIX)

```

freebsd-arch: [patch] lockf(3) user-exploitable kernel panic

```

+ chgadvlockcnt(pp, -1, 0);
    free(lock, M_LOCKF);
    return (error);
}
@@ -324,6 +356,7 @@
    prev = head;
    block = *head;
    needtolink = 1;
+
    for (;;) {
        ovcase = lf_findoverlap(block, lock, SELF, &prev, &overlap);
        if (ovcase)
@@ -350,10 +383,13 @@
            * If downgrading lock, others may be
            * able to acquire it.
            */
+ /* No new locks are created; no need to check rlim */
        if (lock->lf_type == F_RDLCK &&
            overlap->lf_type == F_WRLCK)
            lf_wakelock(overlap);
        overlap->lf_type = lock->lf_type;
+ if (lock->lf_flags & F_POSIX)
+ chgadvlockcnt(pp, -1, 0);
        FREE(lock, M_LOCKF);
        lock = overlap; /* for debug output below */
        break;
@@ -363,6 +399,8 @@
        * Check for common starting point and different types.
        */
        if (overlap->lf_type == lock->lf_type) {
+ if (lock->lf_flags & F_POSIX)
+ chgadvlockcnt(pp, -1, 0);
            free(lock, M_LOCKF);
            lock = overlap; /* for debug output below */
            break;
@@ -371,8 +409,9 @@
            *prev = lock;
            lock->lf_next = overlap;
            overlap->lf_start = lock->lf_end + 1;
- } else
- lf_split(overlap, lock);
+ } else
+ if (lf_split(overlap, lock) == ENOLCK)
+ return (ENOLCK);
            lf_wakelock(overlap);
            break;

@@ -381,6 +420,7 @@
        * If downgrading lock, others may be able to
        * acquire it, otherwise take the list.
        */

```

freebsd-arch: [patch] lockf(3) user-exploitable kernel panic

```

+ /* No new locks */
    if (lock->lf_type == F_RDLCK &&
        overlap->lf_type == F_WRLCK) {
        lf_wakelock(overlap);
@@ -404,6 +444,8 @@
        needtolink = 0;
    } else
        *prev = overlap->lf_next;
+ if (overlap->lf_flags & F_POSIX)
+ chgadvlockcnt(pp, -1, 0);
    free(overlap, M_LOCKF);
    continue;

@@ -453,9 +495,10 @@
    register struct lockf *unlock;
{
    struct lockf **head = unlock->lf_head;
+ struct proc *pp = (struct proc *)0;
    register struct lockf *lf = *head;
    struct lockf *overlap, **prev;
- int ovcase;
+ int ovcase;

    if (lf == NOLOCKF)
        return (0);
@@ -466,6 +509,8 @@
    lf_print("lf_clearlock", unlock);
#endif /* LOCKF_DEBUG */
    prev = head;
+ if (unlock->lf_flags & F_POSIX)
+ pp = (struct proc *)unlock->lf_id;
    while ((ovcase = lf_findoverlap(lf, unlock, SELF, &prev, &overlap))) {
        /*
        * Wakeup the list of locks to be retried.
@@ -476,6 +521,8 @@

        case 1: /* overlap == lock */
            *prev = overlap->lf_next;
+ if (overlap->lf_flags & F_POSIX)
+ chgadvlockcnt(pp, -1, 0);
            FREE(overlap, M_LOCKF);
            break;

@@ -484,13 +531,16 @@
            overlap->lf_start = unlock->lf_end + 1;
            break;
        }
- lf_split(overlap, unlock);
+ if (lf_split(overlap, unlock) == ENOLCK)
+ return (ENOLCK);
        overlap->lf_next = unlock->lf_next;

```

freebsd-arch: [patch] lockf(3) user-exploitable kernel panic

```

        break;

        case 3: /* lock contains overlap */
            *prev = overlap->lf_next;
            lf = overlap->lf_next;
+ if (overlap->lf_flags & F_POSIX)
+ chgadvlockcnt(pp, -1, 0);
            free(overlap, M_LOCKF);
            continue;

@@ -691,13 +741,16 @@
 * Split a lock and a contained region into
 * two or three locks as necessary.
 */
-static void
+static int
lf_split(lock1, lock2)
    register struct lockf *lock1;
    register struct lockf *lock2;
{
    register struct lockf *splitlock;
+ struct proc *pp = (struct proc *)0;

+ if (lock1->lf_flags & F_POSIX)
+ pp = (struct proc *)lock1->lf_id;
#ifdef LOCKF_DEBUG
    if (lockf_debug & 2) {
        lf_print("lf_split", lock1);
@@ -710,14 +763,19 @@
        if (lock1->lf_start == lock2->lf_start) {
            lock1->lf_start = lock2->lf_end + 1;
            lock2->lf_next = lock1;
- return;
+ return (1);
        }
        if (lock1->lf_end == lock2->lf_end) {
            lock1->lf_end = lock2->lf_start - 1;
            lock2->lf_next = lock1->lf_next;
            lock1->lf_next = lock2;
- return;
+ return (1);
        }
+
+ if (lock1->lf_flags & F_POSIX)
+ if (!chgadvlockcnt(pp, 1, lim_max(pp, RLIMIT_ADVLOCK)))
+ return (ENOLCK);
+
    /*
     * Make a new lock consisting of the last part of
     * the encompassing lock
@@ -733,6 +791,7 @@

```

freebsd-arch: [patch] lockf(3) user-exploitable kernel panic

```

splitlock->lf_next = lock1->lf_next;
lock2->lf_next = splitlock;
lock1->lf_next = lock2;
+ return (1);
}

/*
diff -ur sys/kern/kern_mib.c sys_lockfix/kern/kern_mib.c
--- sys/kern/kern_mib.c Tue Apr 13 12:43:16 2004
+++ sys_lockfix/kern/kern_mib.c Tue Apr 13 12:54:43 2004
@@ -45,6 +45,7 @@
#include <sys/system.h>
#include <sys/sysctl.h>
#include <sys/proc.h>
+#include <sys/fcntl.h>
#include <sys/lock.h>
#include <sys/mutex.h>
#include <sys/jail.h>
@@ -111,6 +112,9 @@

SYSCTL_INT(_kern, KERN_MAXPROCPERUID, maxprocperruid, CTLFLAG_RW,
    &maxprocperruid, 0, "Maximum processes allowed per userid");
+
+SYSCTL_INT(_kern, KERN_MAXADVLOCKSPERUID, maxadvlocksperuid, CTLFLAG_RW,
+ &maxadvlocksperuid, 0, "Maximum number of advisory-mode locks per userid");

SYSCTL_INT(_kern, OID_AUTO, maxusers, CTLFLAG_RDTUN,
    &maxusers, 0, "Hint for kernel tuning");
diff -ur sys/kern/kern_prot.c sys_lockfix/kern/kern_prot.c
--- sys/kern/kern_prot.c Tue Apr 13 12:43:16 2004
+++ sys_lockfix/kern/kern_prot.c Tue Apr 13 23:50:43 2004
@@ -63,6 +63,9 @@
#include <sys/socketvar.h>
#include <sys/sysctl.h>

+void change_ruid(struct ucred *newcred, struct uidinfo *ruip,
+ struct proc *pp);
+
+static MALLOC_DEFINE(M_CRED, "cred", "credentials");

SYSCTL_DECL(_security);
@@ -550,7 +553,7 @@
    * Set the real uid and transfer proc count to new user.
    */
    if (uid != oldcred->cr_ruid) {
- change_ruid(newcred, uip);
+ change_ruid(newcred, uip, p);
        setsugid(p);
    }
}
/*
@@ -865,7 +868,7 @@

```

freebsd-arch: [patch] lockf(3) user-exploitable kernel panic

```

        setsugid(p);
    }
    if (ruid != (uid_t)-1 && oldcred->cr_ruid != ruid) {
- change_ruid(newcred, ruip);
+ change_ruid(newcred, ruip, p);
        setsugid(p);
    }
    if ((ruid != (uid_t)-1 || newcred->cr_uid != newcred->cr_ruid) &&
@@ -990,7 +993,7 @@
        setsugid(p);
    }
    if (ruid != (uid_t)-1 && oldcred->cr_ruid != ruid) {
- change_ruid(newcred, ruip);
+ change_ruid(newcred, ruip, p);
        setsugid(p);
    }
    if (suid != (uid_t)-1 && oldcred->cr_svuid != suid) {
@@ -1979,15 +1982,22 @@
    * duration of the call.
    */
void
- change_ruid(struct ucred *newcred, struct uidinfo *ruip)
+ change_ruid(struct ucred *newcred, struct uidinfo *ruip, struct proc *pp)
{

+ /*
+ * We don't want the number of advisory-mode locks to change
+ * while we are performing this operation.
+ */
+ PROC_LOCK_ASSERT(pp, MA_OWNED);
    (void)chgprocnt(newcred->cr_ruidinfo, -1, 0);
+ (void)chgadvlockcnt(pp, -(pp->p_numadvlocks), 0);
    newcred->cr_ruid = ruip->ui_uid;
    uihold(ruip);
    uifree(newcred->cr_ruidinfo);
    newcred->cr_ruidinfo = ruip;
    (void)chgprocnt(newcred->cr_ruidinfo, 1, 0);
+ (void)chgadvlockcnt(pp, pp->p_numadvlocks, 0);
}

/*_
diff -ur sys/kern/kern_resource.c sys_lockfix/kern/kern_resource.c
--- sys/kern/kern_resource.c Tue Apr 13 12:43:16 2004
+++ sys_lockfix/kern/kern_resource.c Tue Apr 13 12:54:43 2004
@@ -43,6 +43,7 @@
#include <sys/system.h>
#include <sys/sysproto.h>
#include <sys/file.h>
+#include <sys/fcntl.h>
#include <sys/kernel.h>
#include <sys/lock.h>

```

freebsd-arch: [patch] lockf(3) user-exploitable kernel panic

```

#include <sys/malloc.h>
@@ -605,6 +606,12 @@
        if (limp->rlim_max < 1)
            limp->rlim_max = 1;
        break;
+ case RLIMIT_ADVLOCK:
+ if (limp->rlim_cur > maxadvlocksperuid)
+ limp->rlim_cur = maxadvlocksperuid;
+ if (limp->rlim_max > maxadvlocksperuid)
+ limp->rlim_max = maxadvlocksperuid;
+ break;
    }
    *alimp = *limp;
    p->p_limit = newlim;
@@ -1080,6 +1087,70 @@
    if (uip->ui_proccnt < 0)
        printf("negative proccnt for uid = %d\n", uip->ui_uid);
    UIDINFO_UNLOCK(uip);
+ return (1);
+ }
+
+ /*
+ * Change the count of the number of advisory-mode locks in
+ * use by a user at any given time.
+ */
+int
+chgadvlockcnt(pp, diff, max)
+ register struct proc *pp;
+ int diff;
+ int max;
+ {
+ struct uidinfo *uip;
+ PROC_LOCK(pp);
+ uip = pp->p_ucred->cr_uidinfo;
+
+ /* Root is not affected by the lock limit, however,
+ * it is entirely possible that root will setuid()
+ * to another user, for whom the locks will need to be
+ * transferred. This brings up an interesting situation:
+ * root may hold more locks than the user may have. In
+ * this situation, we simply fail to upgrade the lock
+ * count at the setuid() call. However, to be able to
+ * do this, we still need to keep track of the amount of
+ * locks a process holds, even if root is the owner of the
+ * process.
+ */
+ if (pp->p_ucred->cr_uid == 0) {
+ uip->ui_advlocks += diff;
+ pp->p_numadvlocks += diff;
+ printf("Root has now acquired %d locks.\n", uip->ui_advlocks);
+ printf("Process %d has %d locks.\n", pp->p_pid, pp->p_numadvlocks);

```

freebsd-arch: [patch] lockf(3) user-exploitable kernel panic

```
+ PROC_UNLOCK(pp);
+ return (1);
+ }
+
+ /*
+ * Zero represents no limit on the number of locks,
+ * as opposed to no locks.
+ */
+ if (max == 0) {
+ pp->p_numadvlocks += diff;
+ uip->ui_advlocks += diff;
+ printf("User %d has now acquired %d locks but that doesn't matter because max is 0.\n",
pp->p_ucred->cr_uid, uip->ui_advlocks);
+ printf("Process %d has %d locks.\n", pp->p_pid, pp->p_numadvlocks);
+ PROC_UNLOCK(pp);
+ return (1);
+ }
+
+ /* Don't allow them to exceed max */
+ if (diff > 0 && uip->ui_advlocks + diff > max) {
+ printf("User %d has now acquired %d locks but they have exceeded max: %d.\n", pp->p_ucred->cr_uid,
uip->ui_advlocks, max);
+ PROC_UNLOCK(pp);
+ return (0);
+ }
+
+ printf("Hell, let's give user %d and process %d %d locks!\n", pp->p_ucred->cr_uid, pp->p_pid, diff);
+ uip->ui_advlocks += diff;
+ pp->p_numadvlocks += diff;
+ KASSERT(uip->ui_advlocks < 0, ("negative number of advisory-mode locks, user-count"));
+ KASSERT(pp->p_numadvlocks < 0, ("negative number of advisory-mode locks, process-count"));
+
+ PROC_UNLOCK(pp);
+     return (1);
+ }
```

```
diff -ur sys/kern/subr_param.c sys_lockfix/kern/subr_param.c
--- sys/kern/subr_param.c Tue Apr 13 12:43:16 2004
+++ sys_lockfix/kern/subr_param.c Tue Apr 13 12:54:43 2004
@@ -64,6 +64,9 @@
#ifdef MAXFILES
#define MAXFILES (maxproc * 2)
#endif
+#ifndef MAXADVLOCKSPERUID
+#define MAXADVLOCKSPERUID 8192
+#endif
```

```
int hz;
int tick;
@@ -72,6 +75,7 @@
int maxprocpid; /* max # of procs per user */
```

[patch] lockf(3) user-exploitable kernel panic

freebsd-arch: [patch] lockf(3) user-exploitable kernel panic

```

int maxfiles; /* sys. wide open files limit */
int maxfilesperproc; /* per-proc open files limit */
+int maxadvlocksperuid; /* max # of advisory-mode locks per uid */
int ncallout; /* maximum # of timer events */
int nbuf;
int nsdbuf;
@@ -111,6 +115,9 @@
    maxbcache = VM_BCACHE_SIZE_MAX;
#endif
    TUNABLE_INT_FETCH("kern.maxbcache", &maxbcache);
+
+ maxadvlocksperuid = MAXADVLOCKSPERUID;
+ TUNABLE_INT_FETCH("kern.maxadvlocksperuid", &maxadvlocksperuid);

    maxtsiz = MAXTSIZ;
    TUNABLE_QUAD_FETCH("kern.maxtsiz", &maxtsiz);
diff -ur sys/sys/fcntl.h sys_lockfix/sys/fcntl.h
--- sys/sys/fcntl.h Tue Apr 13 12:43:16 2004
+++ sys_lockfix/sys/fcntl.h Tue Apr 13 12:54:43 2004
@@ -226,4 +226,8 @@
__END_DECLS
#endif

+#ifdef _KERNEL
+extern int maxadvlocksperuid;
+#endif
+
#endif /* !_SYS_FCNTL_H */
diff -ur sys/sys/proc.h sys_lockfix/sys/proc.h
--- sys/sys/proc.h Tue Apr 13 12:43:16 2004
+++ sys_lockfix/sys/proc.h Tue Apr 13 12:54:43 2004
@@ -608,6 +608,7 @@
    void *p_emuldata; /* (c) Emulator state data. */
    struct label *p_label; /* (*) Proc (not subject) MAC label. */
    struct p_sched *p_sched; /* (*) Scheduler-specific data. */
+ int p_numadvlocks; /* (*) Number of advisory-mode locks */
};

#define p_session p_pgrp->pg_session
diff -ur sys/sys/resource.h sys_lockfix/sys/resource.h
--- sys/sys/resource.h Tue Apr 13 12:43:16 2004
+++ sys_lockfix/sys/resource.h Tue Apr 13 12:54:43 2004
@@ -85,8 +85,9 @@
#define RLIMIT_NOFILE 8 /* number of open files */
#define RLIMIT_SBSIZE 9 /* maximum size of all socket buffers */
#define RLIMIT_VMEM 10 /* virtual process size (inclusive of mmap) */
+#define RLIMIT_ADVLOCK 11 /* maximum number of advisory-mode locks per user */

-#define RLIM_NLIMITS 11 /* number of resource limits */
+#define RLIM_NLIMITS 12 /* number of resource limits */

```

freebsd-arch: [patch] lockf(3) user-exploitable kernel panic

```
#define RLIM_INFINITY ((rlim_t)((u_quad_t)1 << 63) - 1)

@@ -108,6 +109,7 @@
    "nofile",
    "sbsize",
    "vmem",
+ "advlock",
};
#endif

diff -ur sys/sys/resourcevar.h sys_lockfix/sys/resourcevar.h
--- sys/sys/resourcevar.h Tue Apr 13 12:43:16 2004
+++ sys_lockfix/sys/resourcevar.h Tue Apr 13 12:54:43 2004
@@ -90,6 +90,7 @@
    long ui_procct; /* number of processes */
    uid_t ui_uid; /* uid */
    u_int ui_ref; /* reference count */
+ int ui_advlocks; /* number of advisory-mode locks */
    struct mtx *ui_mtxp; /* protect all counts/limits */
};

@@ -104,6 +105,7 @@
void calcru(struct proc *p, struct timeval *up, struct timeval *sp,
    struct timeval *ip);
int chgprocct(struct uidinfo *uip, int diff, int max);
+int chgadvlockct(register struct proc *pp, int diff, int max);
int chgsbsize(struct uidinfo *uip, u_int *hiwat, u_int to,
    rlim_t max);
int fuswintr(void *base);
diff -ur sys/sys/sysctl.h sys_lockfix/sys/sysctl.h
--- sys/sys/sysctl.h Tue Apr 13 12:43:16 2004
+++ sys_lockfix/sys/sysctl.h Tue Apr 13 12:54:43 2004
@@ -359,6 +359,7 @@
#define KERN_LOGSIGEXIT 34 /* int: do we log sigexit procs? */
#define KERN_IOV_MAX 35 /* int: value of UIO_MAXIOV */
#define KERN_MAXID 36 /* number of valid kern ids */
+#define KERN_MAXADVLOCKSPERUID 37 /* int: number of max advisory-mode locks */

#define CTL_KERN_NAMES { \
    { 0, 0 }, \
@@ -390,6 +391,7 @@
    { "bootfile", CTLTYPE_STRING }, \
    { "maxfilesperproc", CTLTYPE_INT }, \
    { "maxprocperuid", CTLTYPE_INT }, \
+ { "maxadvlocksperuid", CTLTYPE_INT }, \
    { "ipc", CTLTYPE_NODE }, \
    { "dummy", CTLTYPE_INT }, \
    { "ps_strings", CTLTYPE_INT }, \
diff -ur sys/sys/ucred.h sys_lockfix/sys/ucred.h
--- sys/sys/ucred.h Tue Apr 13 12:43:16 2004
+++ sys_lockfix/sys/ucred.h Tue Apr 13 12:54:43 2004
```

freebsd-arch: [patch] lockf(3) user-exploitable kernel panic

```
@@ -82,7 +82,10 @@
void change_egid(struct ucred *newcred, gid_t egid);
void change_euid(struct ucred *newcred, struct uidinfo *euid);
void change_rgid(struct ucred *newcred, gid_t rgid);
-void change_ruid(struct ucred *newcred, struct uidinfo *ruip);
+/*
+ * Removed change_ruid; placed definition in kern/kern_prot.c due to
+ * struct proc dependency.
+ */
void change_svgid(struct ucred *newcred, gid_t svgid);
void change_svuid(struct ucred *newcred, uid_t svuid);
void crcopy(struct ucred *dest, struct ucred *src);
```

```
diff -ur usr.bin/limits/limits.1 usr.bin_lockfix/limits/limits.1
--- usr.bin/limits/limits.1 Thu Dec 12 09:26:00 2002
+++ usr.bin_lockfix/limits/limits.1 Wed Apr 14 00:06:10 2004
@@ -357,6 +357,7 @@
```

When run in command mode and execution of the command succeeds, the exit status will be whatever the executed program returns.

.Sh SEE ALSO

+.Xr builtin 1 ,  
.Xr csh 1 ,  
.Xr env 1 ,  
.Xr limit 1 ,

---

freebsd-arch@freebsd.org mailing list

<http://lists.freebsd.org/mailman/listinfo/freebsd-arch>

To unsubscribe, send any mail to "freebsd-arch-unsubscribe@freebsd.org"