Live Troubleshooting session

Q: Can you provide some real world examples of using linked cells?

What problems do they solve?

Q: When using both Active Directory and MIT Kerberos, how does one get referrals to work between the realms?

Q: How can an OpenAFS cell be constructed to provide disaster recovery for a small subset of critical volumes if the primary cell servers were to become partitioned (or destroyed)?

Q: We're getting ready to migrate from 3 older Kerberos/OpenAFS servers to 3 new ones (CentOS 3 to Solaris 10).

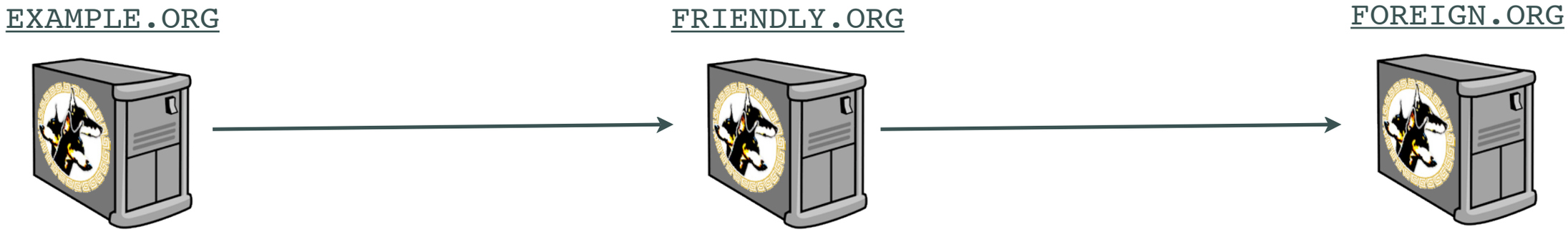
While I've read it's a matter of adding the new ones and doing 'vos moves' to copy the files over, I'd be curious if there are additional gotchas that we may run into.

Q: How can a Kerberos realm be renamed with minimal impact on end users?

The realm rename has been ordered for political reasons and there is no way out.

Q: The Keberos [capaths] section confuses me.

Can you explain how
to create a valid configuration?




```
[capaths]
    EXAMPLE.ORG = {
        FRIENDLY.ORG = .
        FOREIGN.ORG = FRIENDLY.ORG
    }

    FOREIGN.ORG = {
        FRIENDLY.ORG = .
        EXAMPLE.ORG = FRIENDLY.ORG
    }
```

Q: Would you guys talk about AFS tuning/ optimization? While I hear that AFS speed is an issue for many people, it would be nice to discuss how to appropriately tune our clients and servers for better performance.

AFS Performance

- Are the OpenAFS defaults adequate for most sites?
- What are the defaults?

AFS Performance

- Are the OpenAFS defaults adequate for most sites?
- What are the defaults?

Client: All values auto tune depending on cachesize and cachetype

File Server:

# of threads	9
# of cached directory blocks	90
# of cached large vnodes	400
# of cached small vnodes	400
Volume cache size	400
# of callbacks	60,000
# of rx packets	150

AFS Performance II

- Who should consider additional tuning? Those with large numbers of volumes? Large files? Lots of small servers? Small number of large servers (2+ TiB)?

AFS Performance II

- Who should consider additional tuning? Those with large numbers of volumes? Large files? Lots of small servers? Small number of large servers (2+ TiB)?

Everyone

Client Tuning

- Set a sensible cachesize. We can deal with really large caches now.

- Consider a larger chunksize, unless you've got lots of tiny files and not much cache

- Otherwise, let it do its own thing - the autotuning will probably do what you want.

AFS Tuning - Server

- Use -L
- (Despite popular belief, -L gives you 128 threads)
- Turn up callbacks until the server screams
- Consider larger values for small and large vnodes
- Systems with more memory should look at udpsize and sendsize

AFS Performance III

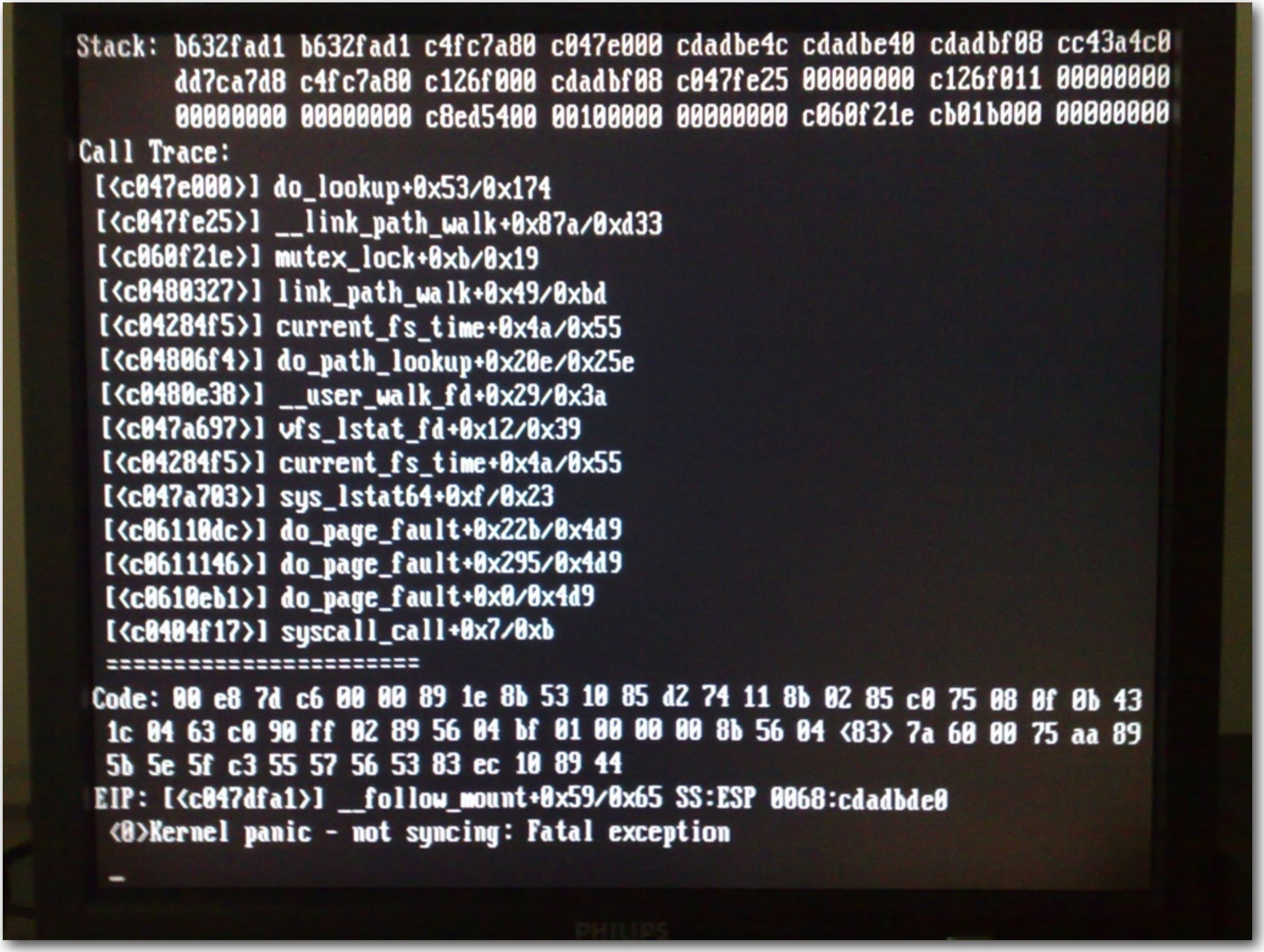
- Introduce tools and methods for determining bottlenecks.
- Step-by-step process to determine appropriate parameters.
- Other bottlenecks (network, disk I/O, etc)

Anatomy of a cache manager security fix

(and other encounters with the Linux kernel)

Background

- A number of reports of panics within the Linux VFS layer
- Locally repeatable case required
  - A *slow* client writing to a file in AFS
  - Another client does an ‘ls -l’ on the file being written, then panics
- Panics suck. They’re much harder to debug than an oops


Stack: b632fad1 b632fad1 c4fc7a80 c047e000 cdadbe4c cdadbe40 cdadbf08 cc43a4c0
dd7ca7d8 c4fc7a80 c126f000 cdadbf08 c047fe25 00000000 c126f011 00000000
00000000 00000000 c8ed5400 00100000 00000000 c060f21e cb01b000 00000000
Call Trace:
[<c047e000>] do_lookup+0x53/0x174
[<c047fe25>] __link_path_walk+0x87a/0xd33
[<c060f21e>] mutex_lock+0xb/0x19
[<c0480327>] link_path_walk+0x49/0xbd
[<c04284f5>] current_fs_time+0x4a/0x55
[<c04806f4>] do_path_lookup+0x20e/0x25e
[<c0480e38>] __user_walk_fd+0x29/0x3a
[<c047a697>] vfs_lstat_fd+0x12/0x39
[<c04284f5>] current_fs_time+0x4a/0x55
[<c047a703>] sys_lstat64+0xf/0x23
[<c06110dc>] do_page_fault+0x22b/0x4d9
[<c0611146>] do_page_fault+0x295/0x4d9
[<c0610eb1>] do_page_fault+0x0/0x4d9
[<c0404f17>] syscall_call+0x7/0xb
=======================
Code: 00 e8 7d c6 00 00 89 1e 8b 53 10 85 d2 74 11 8b 02 85 c0 75 08 0f 0b 43
1c 04 63 c0 90 ff 02 89 56 04 bf 01 00 00 00 8b 56 04 <83> 7a 60 00 75 aa 89
5b 5e 5f c3 55 57 56 53 83 ec 10 89 44
EIP: [<c047dfa1>] __follow_mount+0x59/0x65 SS:ESP 0068:cdadbde0
<0>Kernel panic - not syncing: Fatal exception
_

Getting better information

- Serial console (easy, but you need the hardware)
- netdump / diskdump (RHEL4 and friends)
- kdump (RHEL5)
- If you're sending us crash dumps please remember
  - They are big. In some cases **really** big
  - They may contain keys and other information you don't want us to know.

```
Unable to handle kernel paging request at ffffffff
[<ffffffff80019621>] __follow_mount+0x6e/0x7f
PGD 203067 PUD 205063 PMD 0
Oops: 0000 [1] SMP
last sysfs file: /block/ram0/range
CPU 0
Modules linked in: openafs(PU) nfs(U) lockd(U) fscache(U) nfs_acl(U) sunrpc(U) _
Pid: 7621, comm: ls Tainted: P 2.6.18-128.1.1.el5.inf.1 #1
RIP: 0010:[<ffffffff80019621>] [<ffffffff80019621>] __follow_mount +0x6e/0x7f
RSP: 0018:ffff810014543c38 EFLAGS: 00010246
RAX: ffff81003f4520c0 RBX: ffffffffb631c00c RCX: 0000000000000001
RDX: ffff810014543cc8 RSI: ffffffffb631c00c RDI: ffff810014543cc8
RBP: ffff810014543cc8 R08: 0000000000000019 R09: ffff8100179528b8
R10: ffff8100221a8c40 R11: ffffffff8012172b R12: 0000000000000000
R13: ffff8100179528b8 R14: ffff810014543e48 R15: ffff810014543cb8
FS: 00002af1f14bdc10(0000) GS:ffffffff803ac000(0000) knlGS: 0000000000000000
CS: 0010 DS: 0000 ES: 0000 CR0: 000000008005003b
CR2: ffffffffb631c0bc CR3: 0000000016f22000 CR4: 00000000000006e0
Process ls (pid: 7621, threadinfo ffff810014542000, task ffff810033b8b7e0)
Stack: ffffffffb631c00c ffffffffb631c00c ffff810017952800 ffffffff8000cbb6
ffff8100345e4000 ffff810014543cc8 ffff81003f4520c0 00000000000041ed
ffff810014543e48 ffff810017952800 ffff81003f4520c0 0000000000000000
Call Trace:
[<ffffffff8000cbb6>] do_lookup+0x65/0x1e6
[<ffffffff80009fc7>] __link_path_walk+0xa01/0xf42
[<ffffffff8000e80a>] link_path_walk+0x5c/0xe5
[<ffffffff8002c77e>] mntput_no_expire+0x19/0x89
[<ffffffff800ebc5f>] sys_getxattr+0x51/0x62
[<ffffffff8000c9d5>] do_path_lookup+0x270/0x2e8
[<ffffffff800123ef>] getname+0x15b/0x1c1
[<ffffffff80023298>] __user_walk_fd+0x37/0x4c
[<ffffffff8003ebba>] vfs_lstat_fd+0x18/0x47
[<ffffffff8002c77e>] mntput_no_expire+0x19/0x89
[<ffffffff800ebc5f>] sys_getxattr+0x51/0x62
[<ffffffff8002a603>] sys_newlstat+0x19/0x31
[<ffffffff8005d116>] system_call+0x7e/0x83


Code: 83 be b0 00 00 00 00 75 95 5b 5d 44 89 e0 41 5c c3 41 57 41
RIP [<ffffffff80019621>] __follow_mount+0x6e/0x7f
RSP <ffff810014543c38>
CR2: ffffffffb631c0bc
<0>Kernel panic - not syncing: Fatal exception
```

Where did it all go wrong?

```
RIP [<ffffffff80019621>] __follow_mount+0x6e/0x7f
```

- 0x6e is the offset within the __follow_mount function

- Need a kernel with debugging symbols

- Use objdump -S to tell you what the code at that location is, and work back from there

- lxr.linux.no (or the OpenAFS source, as appropriate) is your friend

objdump in action

```
objdump -S /usr/lib/debug/lib/modules/2.6.18-128.1.6.el5/vmlinux
     | grep '<__follow_mount>:'

ffffffff800195b3 <__follow_mount>:
```

- You can add 0x6e in your head, can't you?

```
static __inline__ void atomic_inc(atomic_t *v)
{
        __asm__ __volatile__(
ffffffff80019610:       f0 ff 02                lock incl (%rdx)
ffffffff80019613:       48 89 55 08             mov    %rdx,0x8(%rbp)
ffffffff80019617:       41 bc 01 00 00 00       mov    $0x1,%r12d
ffffffff8001961d:       48 8b 75 08             mov    0x8(%rbp),%rsi
ffffffff80019621:       83 be b0 00 00 00 00    cmpl   $0x0,0xb0(%rsi)
```

Ooops

- Ooopses occur within modules, rather than in the kernel itself

- Easier to debug, because typically the machine doesn't crash, and the necessary information is in dmesg or syslog

- Don't 'CUT HERE'. There's usually critical information on the line before

Disobeying orders

```
assertion failed: code != -EAGAIN, file: /usr/src/modules/openafs/src/libafs/MODLOAD-2.6.18-4-k7-MP/osi_vnodeops.c, line: 484
------------[ cut here ]------------
kernel BUG at /usr/src/modules/openafs/src/libafs/MODLOAD-2.6.18-4-k7-MP/osi_vnodeops.c:484!
invalid opcode: 0000 [#1]
SMP
Modules linked in: tcp_diag inet_diag openafs ppdev lp xt_state xt_tcpudp iptable_nat ip_nat ip_conntrack nfnetlink
iptable_filter ip_tables x_tables ipv6 bridge vfat fat dm_snapshot dm_mirror dm_mod it87 hwmon_vid eeprom ds1621 i2c_isa
agpgart sr_mod snd_intel8x0 snd_ac97_codec snd_ac97_bus snd_pcm_oss snd_pcm snd_mixer_oss snd_seq_dummy snd_seq_oss
snd_seq_midi snd_rawmidi snd_seq_midi_event snd_seq snd_timer snd_seq_device snd soundcore psmouse i2c_nforce2 rtc
snd_page_alloc floppy serio_raw parport_pc parport irtty_sir sir_dev i2c_core irda crc_ccitt pcspkr evdev eth1394 ext3 jbd
mbcache raid1 md_mod ide_generic ide_cd cdrom sd_mod sata_nv libata scsi_mod ohci1394 ieee1394 generic forcedeth amd74xx
ide_core ehci_hcd ohci_hcd usbcore thermal processor fan
CPU: 0
EIP: 0060:[<f8d05407>] Tainted: P VLI
EFLAGS: 00210286 (2.6.18-4-k7 #1)
EIP is at afs_linux_lock+0x190/0x2bc [openafs]
eax: 00000081 ebx: fffffff5 ecx: 00000000 edx: ffffff7f
esi: 00000006 edi: d3a2bf4c ebp: c1933efc esp: d3a2beb0
ds: 007b es: 007b ss: 0068
Process iceape-bin (pid: 15935, ti=d3a2a000 task=ce554aa0 task.ti=d3a2a000)
Stack: e31a4740 df756940 f7ff5cfc 00000000 c1933f00 c1933f00 c1933f08 c1933f08
d7d3edc0 00003e3f 00000001 c1933f1c c1933f1c e31a4740 00000101 00000000
00000000 ffffffff 7fffffff 00000000 00000000 00000000 00000000 00000003
Call Trace:
[<c01592a5>] nameidata_to_filp+0x19/0x28
[<f8d05277>] afs_linux_lock+0x0/0x2bc [openafs]
[<c016d746>] fcntl_setlk+0x100/0x20e
[<c0169717>] sys_fcntl64+0x6f/0x80
[<c0102c57>] syscall_call+0x7/0xb
Code: c7 8b 04 24 89 da 80 64 24 38 7f e8 84 70 46 c7 83 f8 f5 89 c3 75 1e b9 e4 01 00 00 ba 9f 6c d1 f8 b8 3d 6d d1 f8 e8 ac
45 ff ff <0f> 0b e4 01 9f 6c d1 f8 eb 08 85 c0 0f 84 b8 00 00 00 8d 74 24
EIP: [<f8d05407>] afs_linux_lock+0x190/0x2bc [openafs] SS:ESP 0068:d3a2beb0
```

```
May 5 08:58:57 tristero kern.emerg<0> kernel: ------------[ cut here ]------------
May 5 08:58:57 tristero kern.crit<2> kernel: kernel BUG at kernel/cred.c:360!
May 5 08:58:57 tristero kern.emerg<0> kernel: invalid opcode: 0000 [#1] SMP
May 5 08:58:57 tristero kern.emerg<0> kernel: last sysfs file: /sys/class/net/vmnet8/statistics/collisions
May 5 08:58:57 tristero kern.warn<4> kernel: Modules linked in: sg sr_mod usb_storage binfmt_misc openafs(P) ppdev lp ac battery vmnet vmblock
vmci vmmon ipv6 aes_i586 aes_generic cbc dm_crypt w83627hf hwmon_vid fuse firewire_sbp2 loop snd_mpu401 snd_wavefront snd_cs4232 snd_wss_lib
snd_opl3_lib snd_hwdep snd_mpu401_uart snd_intel8x0 snd_ac97_codec ac97_bus snd_pcm_oss snd_mixer_oss snd_pcm snd_seq_dummy snd_seq_oss
snd_seq_midi snd_rawmidi snd_seq_midi_event snd_seq snd_timer snd_seq_device rtc_cmos i2c_i801 psmouse analog ns558 snd parport_pc evdev
rtc_core rtc_lib shpchp parport pci_hotplug hid_belkin gameport serio_raw i2c_core iTCO_wdt soundcore snd_page_alloc button floppy pcspkr ext3
jbd mbcache dm_snapshot ide_generic dm_mirror dm_region_hash dm_log dm_mod ide_cd_mod cdrom sd_mod ata_generic usbhid hid ata_piix
sata_promise piix firewire_ohci firewire_core crc_itu_t libata scsi_mod ide_core ehci_hcd uhci_hcd e1000 usbcore thermal processor fan
thermal_sys intel_agp agpgart
May 5 08:58:57 tristero kern.warn<4> kernel:
May 5 08:58:57 tristero kern.warn<4> kernel: Pid: 18181, comm: apache2 Tainted: P (2.6.29.1.benp01 #2) Canterwood+ICH5
May 5 08:58:57 tristero kern.warn<4> kernel: EIP: 0060:[<c0135637>] EFLAGS: 00210283 CPU: 0
May 5 08:58:57 tristero kern.warn<4> kernel: EIP is at commit_creds+0x5c/0x181
May 5 08:58:57 tristero kern.warn<4> kernel: EAX: f6736980 EBX: ed8c2ce0 ECX: 000000d0 EDX: f6704400
May 5 08:58:57 tristero kern.warn<4> kernel: ESI: f6736980 EDI: ec872410 EBP: 00000001 ESP: ed8a3db0
May 5 08:58:57 tristero kern.warn<4> kernel: DS: 007b ES: 007b FS: 00d8 GS: 0033 SS: 0068
May 5 08:58:57 tristero kern.emerg<0> kernel: Process apache2 (pid: 18181, ti=ed8a2000 task=ec872410 task.ti=ed8a2000)
May 5 08:58:57 tristero kern.emerg<0> kernel: Stack:
May 5 08:58:57 tristero kern.warn<4> kernel: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
May 5 08:58:57 tristero kern.emerg<0> kernel: ed8c2ce0 f6736980 edd9f380 f8885e2f edd9f380 edd9f480 00000001 f888614f
May 5 08:58:57 tristero kern.emerg<0> kernel: ed8a3e1c ed8a3e18 00000001 00000021 41ff134d ed024b00 00000040 00000000
May 5 08:58:57 tristero kern.emerg<0> kernel: Call Trace:
May 5 08:58:57 tristero kern.emerg<0> kernel: [<f8885e2f>] crset+0x56/0x7b [openafs]
May 5 08:58:57 tristero kern.emerg<0> kernel: [<f888614f>] __setpag+0x173/0x18e [openafs]
May 5 08:58:57 tristero kern.emerg<0> kernel: [<f8856c3e>] PagInCred+0x73/0x8c [openafs]
May 5 08:58:57 tristero kern.emerg<0> kernel: [<f8856c8d>] afs_InitReq+0x36/0x49 [openafs]
May 5 08:58:57 tristero kern.emerg<0> kernel: [<f8860a2b>] afs_access+0x5e/0x337 [openafs]
May 5 08:58:57 tristero kern.emerg<0> kernel: [<f88891c0>] afs_linux_permission+0x6b/0xbf [openafs]
May 5 08:58:57 tristero kern.emerg<0> kernel: [<c0173d83>] inode_permission+0x58/0x7f
May 5 08:58:57 tristero kern.emerg<0> kernel: [<c0175580>] __link_path_walk+0x117/0xa5b
May 5 08:58:57 tristero kern.emerg<0> kernel: [<c0151d4a>] filemap_fault+0x93/0x351
May 5 08:58:57 tristero kern.emerg<0> kernel: [<c0175efb>] path_walk+0x37/0x70
May 5 08:58:57 tristero kern.emerg<0> kernel: [<c0176040>] do_path_lookup+0xd8/0xf1
May 5 08:58:57 tristero kern.emerg<0> kernel: [<c0176847>] user_path_at+0x37/0x64
May 5 08:58:57 tristero kern.emerg<0> kernel: [<c018e1b9>] inotify_inode_queue_event+0x45/0xc6
May 5 08:58:57 tristero kern.emerg<0> kernel: [<c016767c>] free_pages_and_swap_cache+0x77/0x8b
May 5 08:58:57 tristero kern.emerg<0> kernel: [<c018e72e>] inotify_dentry_parent_queue_event+0x5a/0x73
May 5 08:58:57 tristero kern.emerg<0> kernel: [<c01b4edd>] security_prepare_creds+0xd/0xf
May 5 08:58:57 tristero kern.emerg<0> kernel: [<c016cb36>] sys_faccessat+0x92/0x154
May 5 08:58:57 tristero kern.emerg<0> kernel: [<c016cc07>] sys_access+0xf/0x13
May 5 08:58:57 tristero kern.emerg<0> kernel: [<c0102b65>] sysenter_do_call+0x12/0x25
May 5 08:58:57 tristero kern.emerg<0> kernel: Code: 44 24 0c 00 00 00 00 c7 04 24 00 00 00 00 c7 44 24 04 00 00 00 00 64 8b 3d 00 60 3e c0 8b
97 b0 01 00 00 3b 97 ac 01 00 00 74 04 <0f> 0b eb fe 8b 02 48 7f 04 0f 0b eb fe 8b 06 85 c0 7f 04 0f 0b
May 5 08:58:57 tristero kern.emerg<0> kernel: EIP: [<c0135637>] commit_creds+0x5c/0x181 SS:ESP 0068:ed8a3db0
May 5 08:58:57 tristero kern.warn<4> kernel: ---[ end trace b7da60cd839372a2 ]---
```

When it makes no sense

```
EIP is at put_user_size [kernel] 0x4d (2.4.21-27.0.1.TLhugemem/i686)
eax: 0dd62000 ebx: 00001000 ecx: 00000400 edx: 00001000
esi: 1fa94000 edi: feff9600 ebp: 036f3adc esp: 0dd63d24
ds: 0068 es: 0068 ss: 0068
Process cp (pid: 1715, stackpage=0dd63000)
Stack: 00000001 00000001 00000006 184f7d64 0dd63dcc 00001000 02149dc8 00001000
1fa94000 feff9600 00000000 036f3adc 19a921c4 037e0dc8 00000000 0214943b
0dd63dcc 036f3adc 00000000 00001000 00000000 00001000 00000001 00000000
Call Trace: [<02149dc8>] file_read_actor [kernel] 0x68 (0x0dd63d3c)
[<0214943b>] do_generic_file_read [kernel] 0x2eb (0x0dd63d60)
[<02149f15>] generic_file_new_read [kernel] 0xc5 (0x0dd63da0)
[<02149d60>] file_read_actor [kernel] 0x0 (0x0dd63db0)
[<0214a03f>] generic_file_read [kernel] 0x2f (0x0dd63dec)
[<22a3387f>] osi_rdwr [openafs] 0xff (0x0dd63e04)
[<22a63780>] afs_global_lock [openafs] 0x0 (0x0dd63e24)
[<22a0c2a3>] afs_UFSRead [openafs] 0x3c3 (0x0dd63e34)
[<22a37017>] afs_linux_read [openafs] 0x2b7 (0x0dd63ef4)
[<2281fe89>] ext3_file_write [ext3] 0x39 (0x0dd63f74)
[<02164fc3>] sys_read [kernel] 0xa3 (0x0dd63f94)
```

- Rebuild your module/kernel (delete as applicable) with the `-fno-omit-frame-pointer option`