Subject: Re:  Re: [PATCH 1/2] iptables 32bit compat layer
Posted by Arnd Bergmann on Tue, 21 Feb 2006 11:56:00 GMT
View Forum Message <> Reply to Message

On Tuesday 21 February 2006 10:04, Dmitry Mishin wrote:
> On Monday 20 February 2006 18:55, Arnd Bergmann wrote:

> > Is CONFIG_COMPAT the right conditional here? If the code is only used
> > for architectures that have different aligments, it should not need be
> > compiled in for the other architectures.
> So, I'll define ARCH_HAS_FUNNY_64_ALIGNMENT in x86_64 and ia64 code and will
> check it, as Andi suggested.
>

I think nowadays, unconditionally setting CONFIG_FUNNY_64_ALIGNMENT from
arch/{ia64,x86_64}/Kconfig would be the preferred way to a #define in
include/asm.

> > > +
> > > +#ifdef CONFIG_COMPAT
> > > +#include <net/compat.h>
> > > +
> > > +struct compat_ipt_getinfo
> > > +{

> > > +};
> >
> > This structure looks like it does not need any
> > conversions. You should probably just use
> > struct ipt_getinfo then.
> I just saw compat_uint_t use in net/compat.c and thought, that it is a good
> style to use it. Does anybody know arch, where sizeof(compat_uint_t) != 4?

No, the compat layer already heavily depends on the fact that compat_uint_t
is always the same as unsigned int.

> >
> > Dito
> Disagree, ipt_entry_match and ipt_entry_target contain pointers which make
> their alignment equal 8 byte on 64bits architectures.

Ah, I see.

> > I would much rather have either an extra 'compat' argument to to
> > sock_setsockopt and proto_ops->setsockopt than to spread the use
> > of is_compat_task further.
> Another weak place in my code. is_compat_task() approach has one advantage -
> it doesn't require a lot of current code modifications.
> >
> > Is the FIXME above the only reason that the code needs to be changed?
> > What is the reason that you did not just address this in the
> > compat_sys_setsockopt implementation?
> Code above doesn't work. iptables with version >= 1.3 does alignment checks as
> well as kernel does. So, we can't simply put entries with 8 bytes alignment
> to userspace or with 4 bytes alignment to kernel - we need translate them
> entry by entry. So, I tried to do this the most correct way - that userspace
> will hide its alignment from kernel and vice versa, with not only
> SET_REPLACE, but also GET_INFO, GET_ENTRIES and SET_COUNTERS translation.
> First implementation was exactly in compat_sys_setsockopt, but David asked me
> to do this in netfilter code itself.

Ok, I see the point there. It's probably best to push down all the conversions
from compat_sys_setsockopt down to the protocol specific parts, similar to what
we do for the ioctl handlers.

I'm thinking of something like

```
int compat_sock_setsockopt(struct socket *sock, int level, int optname,
      char __user *optval, int optlen)
{
 switch (optname) {
 case SO_ATTACH_FILTER:
  return do_set_attach_filter(fd, level, optname,
         optval, optlen);
 case SO_SNDTIMEO:
  return do_set_sock_timeout(fd, level, optname,
        optval, optlen);
 default:
  break;
 }
 return sock_setsockopt(sock, level, optname, optval, optlen);
}

asmlinkage long compat_sys_setsockopt(int fd, int level, int optname,
    char __user *optval, int optlen)
{
 int err;
 struct socket *sock;

 if (optlen < 0)
```

```
  return -EINVAL;

 if ((sock = sockfd_lookup(fd, &err))!=NULL)
 {
  err = security_socket_setsockopt(sock,level,optname);
  if (err) {
   sockfd_put(sock);
   return err;
  }

  if (level == SOL_SOCKET)
   err = compat_sock_setsockopt(sock, level,
     optname, optval, optlen);
  else if (sock->ops->compat_setsockopt)
   err = sock->ops->compat_setsockopt(sock, level,
     optname, optval, optlen);
  else
   err = sock->ops->setsockopt(sock, level,
     optname, optval, optlen);
  sockfd_put(sock);
 }
 return err;
}

int tcp_setsockopt(struct sock *sk, int level, int optname, char __user *optval, int optlen)
{
 int err = 0;

 err = ip_setsockopt(sk, level, optname, optval, optlen);

#ifdef CONFIG_NETFILTER
 if (err = -ENOPROTOOPT) {
  lock_sock(sk);
  err = nf_setsockopt(sk, PF_INET, optname, optval, optlen);
  release_sock(sk);
 }
#endif
 return err;
}

int compat_tcp_setsockopt(struct sock *sk, int level, int optname, char __user *optval, int optlen)
{
 int err = 0;

 err = ip_setsockopt(sk, level, optname, optval, optlen);

#ifdef CONFIG_NETFILTER
 if (err = -ENOPROTOOPT) {
```

```
  lock_sock(sk);
  err = compat_nf_setsockopt(sk, PF_INET, optname, optval, optlen);
  release_sock(sk);
 }
#endif
 return err;
}
```

And the same for udp, raw, ipv6, decnet and each of those with getsockopt.
It is a bigger change, but it puts all the handlers where they belong
and it is more extensible to other sockopt handlers if we find more
fsckup in some of them.

Arnd <><