Subject: [PATCH 2/2] iptables 32bit compat layer
Posted by dim on Mon, 20 Feb 2006 08:14:02 GMT
View Forum Message <> Reply to Message

This patch introduces compatibility functions for some ip_tables matches and
targets, using interface provided in the first patch.

> Hello,
>
> This patch set extends current iptables compatibility layer in order to get
> 32bit iptables to work on 64bit kernel. Current layer is insufficient
> due to alignment checks both in kernel and user space tools.
>

--
Thanks,
Dmitry.

```
--- ./include/linux/netfilter/xt_conntrack.h.iptcompat2 2006-02-20 10:39:53.000000000 +0300
+++ ./include/linux/netfilter/xt_conntrack.h 2006-02-20 10:14:27.000000000 +0300
@@ -5,6 +5,7 @@
 #ifndef _XT_CONNTRACK_H
 #define _XT_CONNTRACK_H

+#include <linux/config.h>
 #include <linux/netfilter/nf_conntrack_tuple_common.h>
 #include <linux/in.h>

@@ -60,4 +61,21 @@ struct xt_conntrack_info
   /* Inverse flags */
   u_int8_t invflags;
 };
+
+#ifdef CONFIG_COMPAT
+struct compat_xt_conntrack_info
+{
+ compat_uint_t statemask, statusmask;
+
+ struct ip_conntrack_tuple tuple[IP_CT_DIR_MAX];
+ struct in_addr sipmsk[IP_CT_DIR_MAX], dipmsk[IP_CT_DIR_MAX];
+
+ compat_ulong_t expires_min, expires_max;
+
+ /* Flags word */
+ u_int8_t flags;
+ /* Inverse flags */
+ u_int8_t invflags;
+};
```

```
+#endif
 #endif /*_XT_CONNTRACK_H*/
--- ./include/linux/netfilter/xt_helper.h.iptcompat2 2006-02-20 10:39:06.000000000 +0300
+++ ./include/linux/netfilter/xt_helper.h 2006-02-20 10:42:24.000000000 +0300
@@ -1,8 +1,17 @@
 #ifndef _XT_HELPER_H
 #define _XT_HELPER_H

+#include <linux/config.h>
+
 struct xt_helper_info {
  int invert;
  char name[30];
 };
+
+#ifdef CONFIG_COMPAT
+struct compat_xt_helper_info {
+ compat_int_t invert;
+ char name[30];
+};
+#endif
 #endif /* _XT_HELPER_H */
--- ./include/linux/netfilter/xt_limit.h.iptcompat2 2006-02-20 10:39:24.000000000 +0300
+++ ./include/linux/netfilter/xt_limit.h 2006-02-20 10:41:58.000000000 +0300
@@ -1,6 +1,8 @@
 #ifndef _XT_RATE_H
 #define _XT_RATE_H

+#include <linux/config.h>
+
 /* timings are in milliseconds. */
 #define XT_LIMIT_SCALE 10000

@@ -18,4 +20,19 @@ struct xt_rateinfo {
  /* Ugly, ugly fucker. */
  struct xt_rateinfo *master;
 };
+
+#ifdef CONFIG_COMPAT
+struct compat_xt_rateinfo {
+ u_int32_t avg;    /* Average secs between packets * scale */
+ u_int32_t burst;  /* Period multiplier for upper limit. */
+
+ /* Used internally by the kernel */
+ compat_ulong_t prev;
+ u_int32_t credit;
+ u_int32_t credit_cap, cost;
+
```

```
+ /* Ugly, ugly fucker. */
+ compat_uptr_t master;
+};
+#endif
 #endif /*_XT_RATE_H*/
--- ./include/linux/netfilter/xt_state.h.iptcompat2 2006-02-20 10:39:40.000000000 +0300
+++ ./include/linux/netfilter/xt_state.h 2006-02-20 10:42:13.000000000 +0300
@@ -1,6 +1,8 @@
 #ifndef _XT_STATE_H
 #define _XT_STATE_H

+#include <linux/config.h>
+
 #define XT_STATE_BIT(ctinfo) (1 << ((ctinfo)%IP_CT_IS_REPLY+1))
 #define XT_STATE_INVALID (1 << 0)

@@ -10,4 +12,11 @@ struct xt_state_info
 {
  unsigned int statemask;
 };
+
+#ifdef CONFIG_COMPAT
+struct compat_xt_state_info
+{
+ compat_uint_t statemask;
+};
+#endif
 #endif /*_XT_STATE_H*/
--- ./include/linux/netfilter_ipv4/ip_nat.h.iptcompat2 2006-01-03 06:21:10.000000000 +0300
+++ ./include/linux/netfilter_ipv4/ip_nat.h 2006-02-20 10:56:20.000000000 +0300
@@ -1,5 +1,6 @@
 #ifndef _IP_NAT_H
 #define _IP_NAT_H
+#include <linux/config.h>
 #include <linux/netfilter_ipv4.h>
 #include <linux/netfilter_ipv4/ip_conntrack_tuple.h>

@@ -76,6 +77,23 @@ extern int ip_nat_used_tuple(const struc
 extern u_int16_t ip_nat_cheat_check(u_int32_t oldvalinv,
         u_int32_t newval,
         u_int16_t oldcheck);
+
+#ifdef CONFIG_COMPAT
+#include <net/compat.h>
+
+struct compat_ip_nat_range
+{
+ compat_uint_t flags;
```

```
+ u_int32_t min_ip, max_ip;
+ union ip_conntrack_manip_proto min, max;
+};
+
+struct compat_ip_nat_multi_range
+{
+ compat_uint_t rangesize;
+ struct compat_ip_nat_range range[1];
+};
+#endif
 #else  /* !__KERNEL__: iptables wants this to compile. */
 #define ip_nat_multi_range ip_nat_multi_range_compat
 #endif /*__KERNEL__*/
--- ./net/ipv4/netfilter/ip_nat_rule.c.iptcompat2 2006-02-15 16:06:42.000000000 +0300
+++ ./net/ipv4/netfilter/ip_nat_rule.c 2006-02-20 10:57:25.000000000 +0300
@@ -235,6 +235,93 @@ static int ipt_dnat_checkentry(const cha
  return 1;
 }

+#ifdef CONFIG_COMPAT
+static int compat_to_user(void *target, void **dstptr,
+  int *size, int off)
+{
+ struct ipt_entry_target *pt;
+ struct ip_nat_multi_range_compat *pinfo;
+ struct compat_ip_nat_multi_range info;
+ u_int16_t tsize;
+
+ pt = (struct ipt_entry_target *)target;
+ tsize = pt->u.user.target_size;
+ if (__copy_to_user(*dstptr, pt, sizeof(struct ipt_entry_target)))
+  return -EFAULT;
+ pinfo = (struct ip_nat_multi_range_compat *)pt->data;
+ memset(&info, 0, sizeof(struct compat_ip_nat_multi_range));
+ info.rangesize = pinfo->rangesize;
+ info.range[0].flags = pinfo->range[0].flags;
+ info.range[0].min_ip = pinfo->range[0].min_ip;
+ info.range[0].max_ip = pinfo->range[0].max_ip;
+ info.range[0].min = pinfo->range[0].min;
+ info.range[0].max = pinfo->range[0].max;
+ if (__copy_to_user(*dstptr + sizeof(struct ipt_entry_target),
+     &info, sizeof(struct compat_ip_nat_multi_range)))
+  return -EFAULT;
+ tsize -= off;
+ if (put_user(tsize, (u_int16_t *)*dstptr))
+  return -EFAULT;
+ *size -= off;
+ *dstptr += tsize;
```

```
+ return 0;
+}
+
+static int compat_from_user(void *target, void **dstptr,
+  int *size, int off)
+{
+ struct compat_ipt_entry_target *pt;
+ struct ipt_entry_target *dstpt;
+ struct compat_ip_nat_multi_range *pinfo;
+ struct ip_nat_multi_range_compat info;
+ u_int16_t tsize;
+
+ pt = (struct compat_ipt_entry_target *)target;
+ dstpt = (struct ipt_entry_target *)*dstptr;
+ tsize = pt->u.user.target_size;
+ memcpy(*dstptr, pt, sizeof(struct compat_ipt_entry_target));
+ pinfo = (struct compat_ip_nat_multi_range *)pt->data;
+ memset(&info, 0, sizeof(struct ip_nat_multi_range_compat));
+ info.rangesize = pinfo->rangesize;
+ info.range[0].flags = pinfo->range[0].flags;
+ info.range[0].min_ip = pinfo->range[0].min_ip;
+ info.range[0].max_ip = pinfo->range[0].max_ip;
+ info.range[0].min = pinfo->range[0].min;
+ info.range[0].max = pinfo->range[0].max;
+ memcpy(*dstptr + sizeof(struct compat_ipt_entry_target),
+   &info, sizeof(struct ip_nat_multi_range_compat));
+ tsize += off;
+ dstpt->u.user.target_size = tsize;
+ *size += off;
+ *dstptr += tsize;
+ return 0;
+}
+
+static int compat(void *target, void **dstptr, int *size, int convert)
+{
+ int ret, off;
+
+ off = IPT_ALIGN(sizeof(struct ip_nat_multi_range_compat)) -
+  COMPAT_IPT_ALIGN(sizeof(struct compat_ip_nat_multi_range));
+ switch (convert) {
+  case COMPAT_TO_USER:
+   ret = compat_to_user(target, dstptr, size, off);
+   break;
+  case COMPAT_FROM_USER:
+   ret = compat_from_user(target, dstptr, size, off);
+   break;
+  case COMPAT_CALC_SIZE:
+   *size += off;
```

```
+   ret = 0;
+   break;
+  default:
+   ret = -ENOPROTOOPT;
+   break;
+ }
+ return ret;
+}
+#endif
+
 inline unsigned int
 alloc_null_binding(struct ip_conntrack *conntrack,
       struct ip_nat_info *info,
@@ -300,12 +387,18 @@ static struct ipt_target ipt_snat_reg =
  .name  = "SNAT",
  .target  = ipt_snat_target,
  .checkentry = ipt_snat_checkentry,
+#ifdef CONFIG_COMPAT
+ .compat  = &compat,
+#endif
 };

 static struct ipt_target ipt_dnat_reg = {
  .name  = "DNAT",
  .target  = ipt_dnat_target,
  .checkentry = ipt_dnat_checkentry,
+#ifdef CONFIG_COMPAT
+ .compat  = &compat,
+#endif
 };

 int __init ip_nat_rule_init(void)
--- ./net/ipv4/netfilter/ipt_LOG.c.iptcompat2 2006-02-15 16:06:42.000000000 +0300
+++ ./net/ipv4/netfilter/ipt_LOG.c 2006-02-20 10:05:08.000000000 +0300
@@ -458,10 +458,25 @@ static int ipt_log_checkentry(const char
  return 1;
 }

+#ifdef CONFIG_COMPAT
+static int ipt_log_compat(void *target,
+  void **dstptr, int *size, int convert)
+{
+ int off;
+
+ off = IPT_ALIGN(sizeof(struct ipt_log_info)) -
+  COMPAT_IPT_ALIGN(sizeof(struct ipt_log_info));
+ return ipt_target_align_compat(target, dstptr, size, off, convert);
+}
```

```
+#endif
+
 static struct ipt_target ipt_log_reg = {
  .name  = "LOG",
  .target  = ipt_log_target,
  .checkentry = ipt_log_checkentry,
+#ifdef CONFIG_COMPAT
+ .compat  = ipt_log_compat,
+#endif
  .me  = THIS_MODULE,
 };

--- ./net/ipv4/netfilter/ipt_REJECT.c.iptcompat2 2006-02-15 16:06:42.000000000 +0300
+++ ./net/ipv4/netfilter/ipt_REJECT.c 2006-02-20 10:05:08.000000000 +0300
@@ -322,10 +322,25 @@ static int check(const char *tablename,
  return 1;
 }

+#ifdef CONFIG_COMPAT
+static int compat(void *target,
+  void **dstptr, int *size, int convert)
+{
+ int off;
+
+ off = IPT_ALIGN(sizeof(struct ipt_reject_info)) -
+  COMPAT_IPT_ALIGN(sizeof(struct ipt_reject_info));
+ return ipt_target_align_compat(target, dstptr, size, off, convert);
+}
+#endif
+
 static struct ipt_target ipt_reject_reg = {
  .name  = "REJECT",
  .target  = reject,
  .checkentry = check,
+#ifdef CONFIG_COMPAT
+ .compat  = compat,
+#endif
  .me  = THIS_MODULE,
 };

--- ./net/ipv4/netfilter/ipt_TCPMSS.c.iptcompat2 2006-02-15 16:06:42.000000000 +0300
+++ ./net/ipv4/netfilter/ipt_TCPMSS.c 2006-02-20 10:05:08.000000000 +0300
@@ -242,10 +242,25 @@ ipt_tcpmss_checkentry(const char *tablen
  return 0;
 }

+#ifdef CONFIG_COMPAT
+static int ipt_tcpmss_compat(void *target,
```

```
+  void **dstptr, int *size, int convert)
+{
+ int off;
+
+ off = IPT_ALIGN(sizeof(struct ipt_tcpmss_info)) -
+  COMPAT_IPT_ALIGN(sizeof(struct ipt_tcpmss_info));
+ return ipt_target_align_compat(target, dstptr, size, off, convert);
+}
+#endif
+
 static struct ipt_target ipt_tcpmss_reg = {
  .name  = "TCPMSS",
  .target  = ipt_tcpmss_target,
  .checkentry = ipt_tcpmss_checkentry,
+#ifdef CONFIG_COMPAT
+ .compat  = ipt_tcpmss_compat,
+#endif
  .me  = THIS_MODULE,
 };

--- ./net/ipv4/netfilter/ipt_TOS.c.iptcompat2 2006-02-15 16:06:42.000000000 +0300
+++ ./net/ipv4/netfilter/ipt_TOS.c 2006-02-20 10:05:08.000000000 +0300
@@ -83,10 +83,25 @@ checkentry(const char *tablename,
  return 1;
 }

+#ifdef CONFIG_COMPAT
+static int compat(void *target,
+  void **dstptr, int *size, int convert)
+{
+ int off;
+
+ off = IPT_ALIGN(sizeof(struct ipt_tos_target_info)) -
+  COMPAT_IPT_ALIGN(sizeof(struct ipt_tos_target_info));
+ return ipt_target_align_compat(target, dstptr, size, off, convert);
+}
+#endif
+
 static struct ipt_target ipt_tos_reg = {
  .name  = "TOS",
  .target  = target,
  .checkentry = checkentry,
+#ifdef CONFIG_COMPAT
+ .compat  = compat,
+#endif
  .me  = THIS_MODULE,
 };
```

```
--- ./net/ipv4/netfilter/ipt_multiport.c.iptcompat2 2006-02-15 16:06:42.000000000 +0300
+++ ./net/ipv4/netfilter/ipt_multiport.c 2006-02-20 10:33:20.000000000 +0300
@@ -174,11 +174,36 @@ checkentry_v1(const char *tablename,
  return (matchsize == IPT_ALIGN(sizeof(struct ipt_multiport_v1)));
 }

+#ifdef CONFIG_COMPAT
+static int compat(void *match,
+  void **dstptr, int *size, int convert)
+{
+ int off;
+
+ off = IPT_ALIGN(sizeof(struct ipt_multiport)) -
+  COMPAT_IPT_ALIGN(sizeof(struct ipt_multiport));
+ return ipt_match_align_compat(match, dstptr, size, off, convert);
+}
+
+static int compat_v1(void *match,
+  void **dstptr, int *size, int convert)
+{
+ int off;
+
+ off = IPT_ALIGN(sizeof(struct ipt_multiport_v1)) -
+  COMPAT_IPT_ALIGN(sizeof(struct ipt_multiport_v1));
+ return ipt_match_align_compat(match, dstptr, size, off, convert);
+}
+#endif
+
 static struct ipt_match multiport_match = {
  .name  = "multiport",
  .revision = 0,
  .match  = &match,
  .checkentry = &checkentry,
+#ifdef CONFIG_COMPAT
+ .compat  = &compat,
+#endif
  .me  = THIS_MODULE,
 };

@@ -187,6 +212,9 @@ static struct ipt_match multiport_match_
  .revision = 1,
  .match  = &match_v1,
  .checkentry = &checkentry_v1,
+#ifdef CONFIG_COMPAT
+ .compat  = &compat_v1,
+#endif
  .me  = THIS_MODULE,
 };
```

```
--- ./net/ipv4/netfilter/ipt_tos.c.iptcompat2 2006-02-15 16:06:42.000000000 +0300
+++ ./net/ipv4/netfilter/ipt_tos.c 2006-02-20 10:06:50.000000000 +0300
@@ -44,10 +44,25 @@ checkentry(const char *tablename,
  return 1;
 }

+#ifdef CONFIG_COMPAT
+static int compat(void *match,
+  void **dstptr, int *size, int convert)
+{
+ int off;
+
+ off = IPT_ALIGN(sizeof(struct ipt_tos_info)) -
+  COMPAT_IPT_ALIGN(sizeof(struct ipt_tos_info));
+ return ipt_match_align_compat(match, dstptr, size, off, convert);
+}
+#endif
+
 static struct ipt_match tos_match = {
  .name  = "tos",
  .match  = &match,
  .checkentry = &checkentry,
+#ifdef CONFIG_COMPAT
+ .compat  = &compat,
+#endif
  .me  = THIS_MODULE,
 };

--- ./net/ipv4/netfilter/ipt_ttl.c.iptcompat2 2006-02-15 16:06:42.000000000 +0300
+++ ./net/ipv4/netfilter/ipt_ttl.c 2006-02-20 10:06:50.000000000 +0300
@@ -57,10 +57,25 @@ static int checkentry(const char *tablen
  return 1;
 }

+#ifdef CONFIG_COMPAT
+static int compat(void *match,
+  void **dstptr, int *size, int convert)
+{
+ int off;
+
+ off = IPT_ALIGN(sizeof(struct ipt_ttl_info)) -
+  COMPAT_IPT_ALIGN(sizeof(struct ipt_ttl_info));
+ return ipt_match_align_compat(match, dstptr, size, off, convert);
+}
+#endif
+
 static struct ipt_match ttl_match = {
```

```
  .name  = "ttl",
  .match  = &match,
  .checkentry = &checkentry,
+#ifdef CONFIG_COMPAT
+ .compat  = &compat,
+#endif
  .me  = THIS_MODULE,
 };

--- ./net/netfilter/xt_conntrack.c.iptcompat2 2006-02-15 16:06:42.000000000 +0300
+++ ./net/netfilter/xt_conntrack.c 2006-02-20 10:47:27.000000000 +0300
@@ -20,6 +20,7 @@

 #include <linux/netfilter/x_tables.h>
 #include <linux/netfilter/xt_conntrack.h>
+#include <linux/netfilter_ipv4/ip_tables.h>

 MODULE_LICENSE("GPL");
 MODULE_AUTHOR("Marc Boucher <marc@mbsi.ca>");
@@ -213,10 +214,112 @@ static int check(const char *tablename,
  return 1;
 }

+#ifdef CONFIG_COMPAT
+static int compat_to_user(void *match, void **dstptr,
+  int *size, int off)
+{
+ struct ipt_entry_match *pm;
+ struct xt_conntrack_info *pinfo;
+ struct compat_xt_conntrack_info info;
+ u_int16_t msize;
+
+ pm = (struct ipt_entry_match *)match;
+ msize = pm->u.user.match_size;
+ if (__copy_to_user(*dstptr, pm, sizeof(struct ipt_entry_match)))
+  return -EFAULT;
+ pinfo = (struct xt_conntrack_info *)pm->data;
+ memset(&info, 0, sizeof(struct compat_xt_conntrack_info));
+ info.statemask = pinfo->statemask;
+ info.statusmask = pinfo->statusmask;
+ memcpy(info.tuple, pinfo->tuple, IP_CT_DIR_MAX *
+   sizeof(struct ip_conntrack_tuple));
+ memcpy(info.sipmsk, pinfo->sipmsk,
+   IP_CT_DIR_MAX * sizeof(struct in_addr));
+ memcpy(info.dipmsk, pinfo->dipmsk,
+   IP_CT_DIR_MAX * sizeof(struct in_addr));
+ info.expires_min = pinfo->expires_min;
+ info.expires_max = pinfo->expires_max;
```

```
+ info.flags = pinfo->flags;
+ info.invflags = pinfo->invflags;
+ if (__copy_to_user(*dstptr + sizeof(struct ipt_entry_match),
+     &info, sizeof(struct compat_xt_conntrack_info)))
+  return -EFAULT;
+ msize -= off;
+ if (put_user(msize, (u_int16_t *)*dstptr))
+  return -EFAULT;
+ *size -= off;
+ *dstptr += msize;
+ return 0;
+}
+
+static int compat_from_user(void *match, void **dstptr,
+  int *size, int off)
+{
+ struct compat_ipt_entry_match *pm;
+ struct ipt_entry_match *dstpm;
+ struct compat_xt_conntrack_info *pinfo;
+ struct xt_conntrack_info info;
+ u_int16_t msize;
+
+ pm = (struct compat_ipt_entry_match *)match;
+ dstpm = (struct ipt_entry_match *)*dstptr;
+ msize = pm->u.user.match_size;
+ memcpy(*dstptr, pm, sizeof(struct compat_ipt_entry_match));
+ pinfo = (struct compat_xt_conntrack_info *)pm->data;
+ memset(&info, 0, sizeof(struct xt_conntrack_info));
+ info.statemask = pinfo->statemask;
+ info.statusmask = pinfo->statusmask;
+ memcpy(info.tuple, pinfo->tuple, IP_CT_DIR_MAX *
+   sizeof(struct ip_conntrack_tuple));
+ memcpy(info.sipmsk, pinfo->sipmsk,
+   IP_CT_DIR_MAX * sizeof(struct in_addr));
+ memcpy(info.dipmsk, pinfo->dipmsk,
+   IP_CT_DIR_MAX * sizeof(struct in_addr));
+ info.expires_min = pinfo->expires_min;
+ info.expires_max = pinfo->expires_max;
+ info.flags = pinfo->flags;
+ info.invflags = pinfo->invflags;
+ memcpy(*dstptr + sizeof(struct compat_ipt_entry_match),
+     &info, sizeof(struct xt_conntrack_info));
+ msize += off;
+ dstpm->u.user.match_size = msize;
+ *size += off;
+ *dstptr += msize;
+ return 0;
+}
```

```
+
+static int compat(void *match, void **dstptr, int *size, int convert)
+{
+ int ret, off;
+
+ off = XT_ALIGN(sizeof(struct xt_conntrack_info)) -
+  COMPAT_XT_ALIGN(sizeof(struct compat_xt_conntrack_info));
+ switch (convert) {
+  case COMPAT_TO_USER:
+   ret = compat_to_user(match, dstptr, size, off);
+   break;
+  case COMPAT_FROM_USER:
+   ret = compat_from_user(match, dstptr, size, off);
+   break;
+  case COMPAT_CALC_SIZE:
+   *size += off;
+   ret = 0;
+   break;
+  default:
+   ret = -ENOPROTOOPT;
+   break;
+ }
+ return ret;
+}
+#endif
+
 static struct xt_match conntrack_match = {
  .name  = "conntrack",
  .match  = &match,
  .checkentry = &check,
+#ifdef CONFIG_COMPAT
+ .compat  = &compat,
+#endif
  .me  = THIS_MODULE,
 };

--- ./net/netfilter/xt_helper.c.iptcompat2 2006-02-15 16:06:42.000000000 +0300
+++ ./net/netfilter/xt_helper.c 2006-02-20 10:46:05.000000000 +0300
@@ -24,6 +24,7 @@
 #endif
 #include <linux/netfilter/x_tables.h>
 #include <linux/netfilter/xt_helper.h>
+#include <linux/netfilter_ipv4/ip_tables.h>

 MODULE_LICENSE("GPL");
 MODULE_AUTHOR("Martin Josefsson <gandalf@netfilter.org>");
@@ -148,16 +149,101 @@ static int check(const char *tablename,
  return 1;
```

```
 }

+#ifdef CONFIG_COMPAT
+static int compat_to_user(void *match, void **dstptr,
+  int *size, int off)
+{
+ struct ipt_entry_match *pm;
+ struct xt_helper_info *pinfo;
+ struct compat_xt_helper_info info;
+ u_int16_t msize;
+
+ pm = (struct ipt_entry_match *)match;
+ msize = pm->u.user.match_size;
+ if (__copy_to_user(*dstptr, pm, sizeof(struct ipt_entry_match)))
+  return -EFAULT;
+ pinfo = (struct xt_helper_info *)pm->data;
+ memset(&info, 0, sizeof(struct compat_xt_helper_info));
+ info.invert = pinfo->invert;
+ memcpy(info.name, pinfo->name, 30);
+ if (__copy_to_user(*dstptr + sizeof(struct ipt_entry_match),
+     &info, sizeof(struct compat_xt_helper_info)))
+  return -EFAULT;
+ msize -= off;
+ if (put_user(msize, (u_int16_t *)*dstptr))
+  return -EFAULT;
+ *size -= off;
+ *dstptr += msize;
+ return 0;
+}
+
+static int compat_from_user(void *match, void **dstptr,
+  int *size, int off)
+{
+ struct compat_ipt_entry_match *pm;
+ struct ipt_entry_match *dstpm;
+ struct compat_xt_helper_info *pinfo;
+ struct xt_helper_info info;
+ u_int16_t msize;
+
+ pm = (struct compat_ipt_entry_match *)match;
+ dstpm = (struct ipt_entry_match *)*dstptr;
+ msize = pm->u.user.match_size;
+ memcpy(*dstptr, pm, sizeof(struct compat_ipt_entry_match));
+ pinfo = (struct compat_xt_helper_info *)pm->data;
+ memset(&info, 0, sizeof(struct xt_helper_info));
+ info.invert = pinfo->invert;
+ memcpy(info.name, pinfo->name, 30);
+ memcpy(*dstptr + sizeof(struct compat_ipt_entry_match),
```

```
+    &info, sizeof(struct xt_helper_info));
+ msize += off;
+ dstpm->u.user.match_size = msize;
+ *size += off;
+ *dstptr += msize;
+ return 0;
+}
+
+static int compat(void *match, void **dstptr, int *size, int convert)
+{
+ int ret, off;
+
+ off = XT_ALIGN(sizeof(struct xt_helper_info)) -
+  COMPAT_XT_ALIGN(sizeof(struct compat_xt_helper_info));
+ switch (convert) {
+  case COMPAT_TO_USER:
+   ret = compat_to_user(match, dstptr, size, off);
+   break;
+  case COMPAT_FROM_USER:
+   ret = compat_from_user(match, dstptr, size, off);
+   break;
+  case COMPAT_CALC_SIZE:
+   *size += off;
+   ret = 0;
+   break;
+  default:
+   ret = -ENOPROTOOPT;
+   break;
+ }
+ return ret;
+}
+#endif
+
 static struct xt_match helper_match = {
  .name  = "helper",
  .match  = &match,
  .checkentry = &check,
+#ifdef CONFIG_COMPAT
+ .compat  = &compat,
+#endif
  .me  = THIS_MODULE,
 };
 static struct xt_match helper6_match = {
  .name  = "helper",
  .match  = &match,
  .checkentry = &check,
+#ifdef CONFIG_COMPAT
+ .compat  = &compat,
```

```
+#endif
  .me  = THIS_MODULE,
 };

--- ./net/netfilter/xt_length.c.iptcompat2 2006-02-15 16:06:42.000000000 +0300
+++ ./net/netfilter/xt_length.c 2006-02-20 10:28:17.000000000 +0300
@@ -63,16 +63,34 @@ checkentry(const char *tablename,
  return 1;
 }

+#ifdef CONFIG_COMPAT
+static int compat(void *match,
+  void **dstptr, int *size, int convert)
+{
+ int off;
+
+ off = XT_ALIGN(sizeof(struct xt_length_info)) -
+  COMPAT_XT_ALIGN(sizeof(struct xt_length_info));
+ return ipt_match_align_compat(match, dstptr, size, off, convert);
+}
+#endif
+
 static struct xt_match length_match = {
  .name  = "length",
  .match  = &match,
  .checkentry = &checkentry,
+#ifdef CONFIG_COMPAT
+ .compat  = &compat,
+#endif
  .me  = THIS_MODULE,
 };
 static struct xt_match length6_match = {
  .name  = "length",
  .match  = &match6,
  .checkentry = &checkentry,
+#ifdef CONFIG_COMPAT
+ .compat  = &compat,
+#endif
  .me  = THIS_MODULE,
 };

--- ./net/netfilter/xt_limit.c.iptcompat2 2006-02-15 16:06:42.000000000 +0300
+++ ./net/netfilter/xt_limit.c 2006-02-20 10:46:41.000000000 +0300
@@ -20,6 +20,7 @@

 #include <linux/netfilter/x_tables.h>
 #include <linux/netfilter/xt_limit.h>
+#include <linux/netfilter_ipv4/ip_tables.h>
```

```
 MODULE_LICENSE("GPL");
 MODULE_AUTHOR("Herve Eychenne <rv@wallfire.org>");
@@ -137,16 +138,104 @@ ipt_limit_checkentry(const char *tablena
  return 1;
 }

+#ifdef CONFIG_COMPAT
+static int ipt_limit_compat_to_user(void *match, void **dstptr,
+  int *size, int off)
+{
+ struct ipt_entry_match *pm;
+ struct xt_rateinfo *pinfo;
+ struct compat_xt_rateinfo rinfo;
+ u_int16_t msize;
+
+ pm = (struct ipt_entry_match *)match;
+ msize = pm->u.user.match_size;
+ if (__copy_to_user(*dstptr, pm, sizeof(struct ipt_entry_match)))
+  return -EFAULT;
+ pinfo = (struct xt_rateinfo *)pm->data;
+ memset(&rinfo, 0, sizeof(struct compat_xt_rateinfo));
+ rinfo.avg = pinfo->avg;
+ rinfo.burst = pinfo->burst;
+ if (__copy_to_user(*dstptr + sizeof(struct ipt_entry_match),
+     &rinfo, sizeof(struct compat_xt_rateinfo)))
+  return -EFAULT;
+ msize -= off;
+ if (put_user(msize, (u_int16_t *)*dstptr))
+  return -EFAULT;
+ *size -= off;
+ *dstptr += msize;
+ return 0;
+}
+
+static int ipt_limit_compat_from_user(void *match, void **dstptr,
+  int *size, int off)
+{
+ struct compat_ipt_entry_match *pm;
+ struct ipt_entry_match *dstpm;
+ struct compat_xt_rateinfo *pinfo;
+ struct xt_rateinfo rinfo;
+ u_int16_t msize;
+
+ pm = (struct compat_ipt_entry_match *)match;
+ dstpm = (struct ipt_entry_match *)*dstptr;
+ msize = pm->u.user.match_size;
+ memcpy(*dstptr, pm, sizeof(struct compat_ipt_entry_match));
```

```
+ pinfo = (struct compat_xt_rateinfo *)pm->data;
+ memset(&rinfo, 0, sizeof(struct xt_rateinfo));
+ rinfo.avg = pinfo->avg;
+ rinfo.burst = pinfo->burst;
+ memcpy(*dstptr + sizeof(struct compat_ipt_entry_match),
+    &rinfo, sizeof(struct xt_rateinfo));
+ msize += off;
+ dstpm->u.user.match_size = msize;
+ *size += off;
+ *dstptr += msize;
+ return 0;
+}
+
+static int ipt_limit_compat(void *match, void **dstptr,
+  int *size, int convert)
+{
+ int ret, off;
+
+ off = XT_ALIGN(sizeof(struct xt_rateinfo)) -
+  COMPAT_XT_ALIGN(sizeof(struct compat_xt_rateinfo));
+ switch (convert) {
+  case COMPAT_TO_USER:
+   ret = ipt_limit_compat_to_user(match,
+     dstptr, size, off);
+   break;
+  case COMPAT_FROM_USER:
+   ret = ipt_limit_compat_from_user(match,
+     dstptr, size, off);
+   break;
+  case COMPAT_CALC_SIZE:
+   *size += off;
+   ret = 0;
+   break;
+  default:
+   ret = -ENOPROTOOPT;
+   break;
+ }
+ return ret;
+}
+#endif
+
 static struct xt_match ipt_limit_reg = {
  .name  = "limit",
  .match  = ipt_limit_match,
 .checkentry = ipt_limit_checkentry,
+#ifdef CONFIG_COMPAT
+ .compat  = ipt_limit_compat,
+#endif
```

```
  .me  = THIS_MODULE,
 };
 static struct xt_match limit6_reg = {
  .name  = "limit",
  .match  = ipt_limit_match,
  .checkentry = ipt_limit_checkentry,
+#ifdef CONFIG_COMPAT
+ .compat  = ipt_limit_compat,
+#endif
  .me  = THIS_MODULE,
 };

--- ./net/netfilter/xt_state.c.iptcompat2 2006-02-15 16:06:42.000000000 +0300
+++ ./net/netfilter/xt_state.c 2006-02-20 10:45:05.000000000 +0300
@@ -13,6 +13,7 @@
 #include <net/netfilter/nf_conntrack_compat.h>
 #include <linux/netfilter/x_tables.h>
 #include <linux/netfilter/xt_state.h>
+#include <linux/netfilter_ipv4/ip_tables.h>

 MODULE_LICENSE("GPL");
 MODULE_AUTHOR("Rusty Russell <rusty@rustcorp.com.au>");
@@ -55,10 +56,90 @@ static int check(const char *tablename,
  return 1;
 }

+#ifdef CONFIG_COMPAT
+static int compat_to_user(void *match, void **dstptr,
+  int *size, int off)
+{
+ struct ipt_entry_match *pm;
+ struct xt_state_info *pinfo;
+ struct compat_xt_state_info info;
+ u_int16_t msize;
+
+ pm = (struct ipt_entry_match *)match;
+ msize = pm->u.user.match_size;
+ if (__copy_to_user(*dstptr, pm, sizeof(struct ipt_entry_match)))
+  return -EFAULT;
+ pinfo = (struct xt_state_info *)pm->data;
+ memset(&info, 0, sizeof(struct compat_xt_state_info));
+ info.statemask = pinfo->statemask;
+ if (__copy_to_user(*dstptr + sizeof(struct ipt_entry_match),
+     &info, sizeof(struct compat_xt_state_info)))
+  return -EFAULT;
+ msize -= off;
+ if (put_user(msize, (u_int16_t *)*dstptr))
+  return -EFAULT;
```

```
+ *size -= off;
+ *dstptr += msize;
+ return 0;
+}
+
+static int compat_from_user(void *match, void **dstptr,
+  int *size, int off)
+{
+ struct compat_ipt_entry_match *pm;
+ struct ipt_entry_match *dstpm;
+ struct compat_xt_state_info *pinfo;
+ struct xt_state_info info;
+ u_int16_t msize;
+
+ pm = (struct compat_ipt_entry_match *)match;
+ dstpm = (struct ipt_entry_match *)*dstptr;
+ msize = pm->u.user.match_size;
+ memcpy(*dstptr, pm, sizeof(struct compat_ipt_entry_match));
+ pinfo = (struct compat_xt_state_info *)pm->data;
+ memset(&info, 0, sizeof(struct xt_state_info));
+ info.statemask = pinfo->statemask;
+ memcpy(*dstptr + sizeof(struct compat_ipt_entry_match),
+    &info, sizeof(struct xt_state_info));
+ msize += off;
+ dstpm->u.user.match_size = msize;
+ *size += off;
+ *dstptr += msize;
+ return 0;
+}
+
+static int compat(void *match, void **dstptr, int *size, int convert)
+{
+ int ret, off;
+
+ off = XT_ALIGN(sizeof(struct xt_state_info)) -
+  COMPAT_XT_ALIGN(sizeof(struct compat_xt_state_info));
+ switch (convert) {
+  case COMPAT_TO_USER:
+   ret = compat_to_user(match, dstptr, size, off);
+   break;
+  case COMPAT_FROM_USER:
+   ret = compat_from_user(match, dstptr, size, off);
+   break;
+  case COMPAT_CALC_SIZE:
+   *size += off;
+   ret = 0;
+   break;
+  default:
```

```
+   ret = -ENOPROTOOPT;
+   break;
+ }
+ return ret;
+}
+#endif
+
 static struct xt_match state_match = {
  .name  = "state",
  .match  = &match,
  .checkentry = &check,
+#ifdef CONFIG_COMPAT
+ .compat  = &compat,
+#endif
  .me  = THIS_MODULE,
 };

@@ -66,6 +147,9 @@ static struct xt_match state6_match = {
  .name  = "state",
  .match  = &match,
  .checkentry = &check,
+#ifdef CONFIG_COMPAT
+ .compat  = &compat,
+#endif
  .me  = THIS_MODULE,
 };

--- ./net/netfilter/xt_tcpmss.c.iptcompat2 2006-02-15 16:06:42.000000000 +0300
+++ ./net/netfilter/xt_tcpmss.c 2006-02-20 10:36:34.000000000 +0300
@@ -133,10 +133,25 @@ checkentry6(const char *tablename,
  return 1;
 }

+#ifdef CONFIG_COMPAT
+static int compat(void *match,
+  void **dstptr, int *size, int convert)
+{
+ int off;
+
+ off = XT_ALIGN(sizeof(struct xt_tcpmss_match_info)) -
+  COMPAT_XT_ALIGN(sizeof(struct xt_tcpmss_match_info));
+ return ipt_match_align_compat(match, dstptr, size, off, convert);
+}
+#endif
+
 static struct xt_match tcpmss_match = {
  .name  = "tcpmss",
  .match  = &match,
```

```
  .checkentry = &checkentry,
+#ifdef CONFIG_COMPAT
+ .compat  = &compat,
+#endif
  .me  = THIS_MODULE,
 };

@@ -144,6 +159,9 @@ static struct xt_match tcpmss6_match = {
  .name  = "tcpmss",
  .match  = &match,
  .checkentry = &checkentry6,
+#ifdef CONFIG_COMPAT
+ .compat  = &compat,
+#endif
  .me  = THIS_MODULE,
 };
```

File Attachments

1) diff-ms-ipt-compat2-20060217, downloaded 559 times