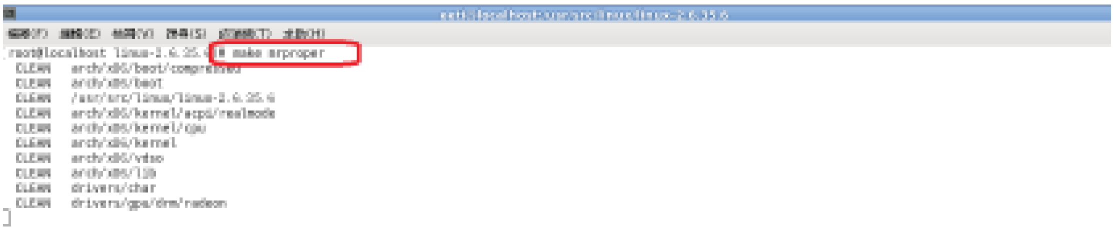
EETI Kernel Patch Guide for Linux

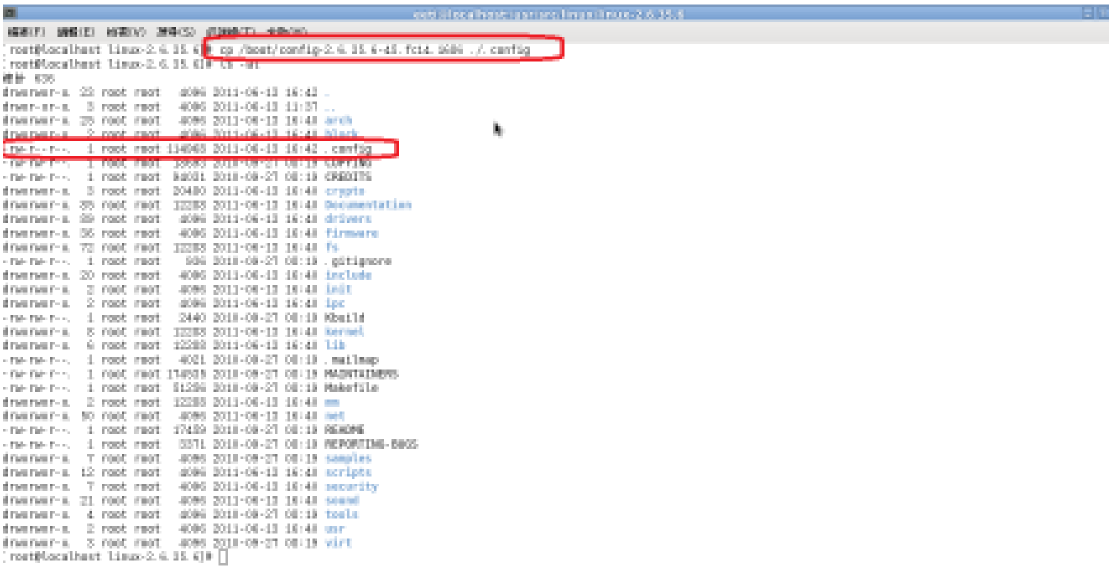
The document is used to guide users to patch the running kernel to disable inbuilt input driver support. It's highly recommended that follow the steps below to rebuild the running kernel before installing EETI Linux driver. Please note the kernel patch is for USB interface only.

1. Check the version of running kernel and download the kernel source from Linux kernel website. www.kernel.org
2. Copy the package of kernel source to /usr/scr/.
   # cp linux-<kernel-version>.tar.bz2 /usr/src/.
3. Uncompress the package of kernel source.
   # tar jxf linux-<kernel-version>.tar.bz2
4. Change working directory to the directory of kernel source.
   # cd /usr/src/linux-<kernel-version>/
5. Execute "make mrproper" to remove old compilation files.
   # make mrproper


root@localhost:/usr/src/linux/linux-2.6.35.6
檔案(F) 編輯(E) 檢視(V) 終端(S) 分頁(T) 求助(H)

```
[root@localhost linux-2.6.35.6]# make mrproper
  CLEAN   arch/x86/boot/compressed
  CLEAN   arch/x86/boot
  CLEAN   /usr/src/linux/linux-2.6.35.6
  CLEAN   arch/x86/kernel/acpi/realmode
  CLEAN   arch/x86/kernel/cpu
  CLEAN   arch/x86/kernel
  CLEAN   arch/x86/vdso
  CLEAN   arch/x86/lib
  CLEAN   drivers/char
  CLEAN   drivers/gpu/drm/radeon
```

6. Copy default configuration file to the directory of kernel source.
   # cp /boot/config-<kernel-version>./.config


root@localhost:/usr/src/linux/linux-2.6.35.6
檔案(F) 編輯(E) 檢視(V) 終端(S) 分頁(T) 求助(H)

```
[root@localhost linux-2.6.35.6]# cp /boot/config-2.6.35.6-45.fc14.i686 ./.config
[root@localhost linux-2.6.35.6]# ls -al
總計 636
drwxrwxr-x. 23 root root   4096 2011-06-13 16:42 .
drwxr-xr-x.  3 root root   4096 2011-06-13 11:37 ..
drwxrwxr-x. 25 root root   4096 2011-06-13 16:48 arch
drwxrwxr-x.  2 root root   4096 2011-06-13 16:48 block
-rw-r--r--.  1 root root 114065 2011-06-13 16:42 .config
-rw-rw-r--.  1 root root  18693 2010-09-27 08:19 COPYING
-rw-rw-r--.  1 root root  94031 2010-09-27 08:19 CREDITS
drwxrwxr-x.  3 root root  20480 2011-06-13 16:48 crypto
drwxrwxr-x. 95 root root  12288 2011-06-13 16:48 Documentation
drwxrwxr-x. 99 root root   4096 2011-06-13 16:48 drivers
drwxrwxr-x. 56 root root   4096 2011-06-13 16:48 firmware
drwxrwxr-x. 72 root root  12288 2011-06-13 16:48 fs
-rw-rw-r--.  1 root root    906 2010-09-27 08:19 .gitignore
drwxrwxr-x. 20 root root   4096 2011-06-13 16:48 include
drwxrwxr-x.  2 root root   4096 2011-06-13 16:48 init
drwxrwxr-x.  2 root root   4096 2011-06-13 16:48 ipc
-rw-rw-r--.  1 root root   2440 2010-09-27 08:19 Kbuild
drwxrwxr-x.  8 root root  12288 2011-06-13 16:48 kernel
drwxrwxr-x.  6 root root  12288 2011-06-13 16:48 lib
-rw-rw-r--.  1 root root   4021 2010-09-27 08:19 .mailmap
-rw-rw-r--.  1 root root 174535 2010-09-27 08:19 MAINTAINERS
-rw-rw-r--.  1 root root  51296 2010-09-27 08:19 Makefile
drwxrwxr-x.  2 root root  12288 2011-06-13 16:48 mm
drwxrwxr-x. 50 root root   4096 2011-06-13 16:48 net
-rw-rw-r--.  1 root root  17459 2010-09-27 08:19 README
-rw-rw-r--.  1 root root   3371 2010-09-27 08:19 REPORTING-BUGS
drwxrwxr-x.  7 root root   4096 2010-09-27 08:19 samples
drwxrwxr-x. 12 root root   4096 2011-06-13 16:48 scripts
drwxrwxr-x.  7 root root   4096 2011-06-13 16:48 security
drwxrwxr-x. 21 root root   4096 2011-06-13 16:48 sound
drwxrwxr-x.  4 root root   4096 2010-09-27 08:19 tools
drwxrwxr-x.  2 root root   4096 2011-06-13 16:48 usr
drwxrwxr-x.  3 root root   4096 2010-09-27 08:19 virt
[root@localhost linux-2.6.35.6]# 
```

7. Check whether USB touch device is HID or Vendor class before patch below source code. Users can execute the command “**lsusb –v –d 0eef:**” to check interface class of USB touch device in a terminal window.

   A. Interface Descriptor: (HID compliant device)

      bLength 9
      bDescriptorType 4
      bInterfaceNumber 0
      bAlternateSetting 0
      bNumEndpoints 1
      bInterfaceClass 3 Human Interface Device
      bInterfaceSubClass 0 No SubClass
      bInterfaceProtocol 0 None
      iInterface 0

   B. Interface Descriptor: (Vendor specific device)

      bLength 9
      bDescriptorType 4
      bInterfaceNumber 0
      bAlternateSetting 0
      bNumEndpoints 1
      bInterfaceClass 255 Vendor Specific Class
      bInterfaceSubClass 255 Vendor Specific Subclass
      bInterfaceProtocol 255 Vendor Specific Protocol
      iInterface 0

8. Patch “mousedev.c”, “evdev.c” and “joydev.c” to blacklist USB touch device.

   Note:

   If USB touch device is **HID compliant device**, users should patch “mousedev.c”, “evdev.c” and “joydev.c” three files and make sure kernel USB and HID options are enabled for USB touch device.

   If USB touch device is **Vendor specific device**, users should patch “mousedev.c” and “joydev.c” two files and make sure kernel USB and event options are enabled.

   Users can execute “make menuconfig” to configure Linux kernel option.

   ```
   # make menuconfig
   ```

Note: The following is used for Linux kernel **2.6.33 and older**.

1.) Append the below RED code to patch “evdev.c”.

```
static struct input_device_id evdev_blacklist[] = {   /* Added by EETI */
 {
  .flags  = INPUT_DEVICE_ID_MATCH_BUS | INPUT_DEVICE_ID_MATCH_VENDOR,
  .bustype = BUS_USB,
  .vendor = 0x0EEF,
 },
 {},      /* Terminating entry */
};

static struct input_handler evdev_handler = {
  .event = evdev_event,
  .connect = evdev_connect,
  .disconnect = evdev_disconnect,
  .fops = &evdev_fops,
  .minor = EVDEV_MINOR_BASE,
  .name = "evdev",
  .id_table = evdev_ids,
  .blacklist = evdev_blacklist, /* Added by EETI */
};
```

2.) Append the below RED code to patch “mousedev.c”.

```
static struct input_device_id mousedev_blacklist[] = {     /* Added by EETI */
  {
  .flags = INPUT_DEVICE_ID_MATCH_BUS | INPUT_DEVICE_ID_MATCH_VENDOR,
  .bustype = BUS_USB,
  .vendor = 0x0EEF,
  },
  {
  .flags  = INPUT_DEVICE_ID_MATCH_BUS | INPUT_DEVICE_ID_MATCH_VENDOR,
  .bustype = BUS_VIRTUAL,
  .vendor = 0x0EEF,
  },
  {},      /* Terminating entry */
};
```

```
static struct input_handler mousedev_handler = {
  .event = mousedev_event,
  .connect = mousedev_connect,
  .disconnect = mousedev_disconnect,
  .fops = &mousedev_fops,
  .minor = MOUSEDEV_MINOR_BASE,
  .name = "mousedev",
  .id_table = mousedev_ids,
  .blacklist = mousedev_blacklist, /* Added by EETI */
};
```

3.) Append the below RED code to patch “joydev.c”.

```
static const struct input_device_id joydev_blacklist[] = {
  {
  .flags = INPUT_DEVICE_ID_MATCH_EVBIT | INPUT_DEVICE_ID_MATCH_KEYBIT,
  .evbit = { BIT_MASK(EV_KEY) },
  .keybit = { [BIT_WORD(BTN_TOUCH)] = BIT_MASK(BTN_TOUCH) },
  },      /* Avoid itouchpads and touchscreens */
  {
  .flags = INPUT_DEVICE_ID_MATCH_EVBIT | INPUT_DEVICE_ID_MATCH_KEYBIT,
  .evbit = { BIT_MASK(EV_KEY) },
  .keybit = { [BIT_WORD(BTN_DIGI)] = BIT_MASK(BTN_DIGI) },
  },      /* Avoid tablets, digitisers and similar devices */
  {
  .flags = INPUT_DEVICE_ID_MATCH_BUS | INPUT_DEVICE_ID_MATCH_VENDOR,
  .bustype = BUS_VIRTUAL,
  .vendor = 0x0EEF,
  },      /* Added by EETI */
  { }     /* Terminating entry */
};

static struct input_handler joydev_handler = {
  .event = joydev_event,
  .connect = joydev_connect,
  .disconnect = joydev_disconnect,
  .fops = &joydev_fops,
  .minor = JOYDEV_MINOR_BASE,
  .name = "joydev",
  .id_table = joydev_ids,
  .blacklist = joydev_blacklist,
};
```

Note: The following is used for Linux kernel **2.6.34 and upwards**.

1.) Append the below RED code to patch “evdev.c”.

```
static bool evdev_match(struct input_handler *handler, struct input_dev *dev)
{
    /* Avoid EETI USB touchscreens */
    #define VID_EETI 0x0EEF
    if ((BUS_USB == dev->id.bustype) && (VID_EETI == dev->id.vendor))
            return false;
    return true;
}

static struct input_handler evdev_handler = {
    .event = evdev_event,
    .match = evdev_match,      /* Added by EETI*/
    .connect = evdev_connect,
    .disconnect = evdev_disconnect,
    .fops = &evdev_fops,
    .minor = EVDEV_MINOR_BASE,
    .name = "evdev",
    .id_table = evdev_ids,
};
```

2.) Append the below RED code to patch “mousedev.c”.

```
static bool mousedev_match(struct input_handler *handler, struct input_dev *dev)
{
    /* Avoid EETI USB touchscreens */
    #define VID_EETI 0x0EEF
    if ((BUS_USB == dev->id.bustype) && (VID_EETI == dev->id.vendor))
            return false;
    /* Avoid EETI virtual devices */
    if ((BUS_VIRTUAL == dev->id.bustype) && (VID_EETI == dev->id.vendor))
            return false;
    return true;
}
```

```
static struct input_handler mousedev_handler = {
    .event = mousedev_event,
    .match = mousedev_match,       /* Added by EETI */
    .connect = mousedev_connect,
    .disconnect = ousedev_disconnect,
    .fops = &mousedev_fops,
    .minor = MOUSEDEV_MINOR_BASE,
    .name = "mousedev",
    .id_table = mousedev_ids,
};
```

3.) Append the below RED code to patch “joydev.c”.

```
static bool joydev_match(struct input_handler *handler, struct input_dev *dev)
{
    /* Avoid touchpads and touchscreens */
    if (test_bit(EV_KEY, dev->evbit) && test_bit(BTN_TOUCH, dev->keybit))
            return false;
    /* Avoid tablets, digitisers and similar devices */
    if (test_bit(EV_KEY, dev->evbit) && test_bit(BTN_DIGI, dev->keybit))
            return false;
    /* Avoid EETI virtual devices */
    #define VID_EETI 0x0EEF
    if (( BUS_VIRTUAL == dev->id.bustype) && (VID_EETI == dev->id.vendor))
            return false;
    return true;
}

static struct input_handler joydev_handler = {
    .event = joydev_event,
    .match = joydev_match,
    .connect = joydev_connect,
    .disconnect = joydev_disconnect,
    .fops = &joydev_fops,
    .minor = JOYDEV_MINOR_BASE,
    .name = "joydev",
    .id_table = joydev_ids,
};
```

9. Follow the steps below to build new kernel and reboot for some changes to take effect.

    1.) # make // build kernel source code

    2.) # make modules_install // install new modules

    3.) # make install // install new kernel

    4.) # update-initramfs -k <kernel-version> -c // create init.d

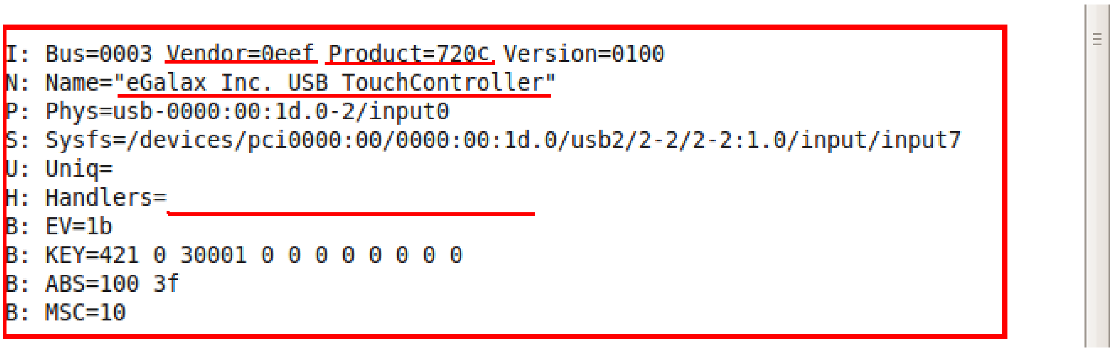
    5.) # update-grub // update grub

    6.) # reboot // restart the system

10. Users can connect an EETI USB touch device to the system and then check the below item after rebooting.

    1.) Check USB touch device handlers:

    The handler should be empty as figure below. It means there is no inbuilt input driver support for the USB touch device. The information could be found in /proc/bus/input/devices file.



```
I: Bus=0003 Vendor=0eef Product=720c Version=0100
N: Name="eGalax Inc. USB TouchController"
P: Phys=usb-0000:00:1d.0-2/input0
S: Sysfs=/devices/pci0000:00/0000:00:1d.0/usb2/2-2/2-2:1.0/input/input7
U: Uniq=
H: Handlers=
B: EV=1b
B: KEY=421 0 30001 0 0 0 0 0 0 0 0
B: ABS=100 3f
B: MSC=10
```