USB Device Authentication for the Security of Linux System

Supriya

{supriya.kundu@yahoo.com}
M.Tech From computer science & Engineering, Banasthali Vidyapith, Rajasthan

Abstract— In the world of information age, the data stored in the computer is most valuable assets to any organization. It may include most valuable data, personal informations, confidential files and folders. Due to this security is must, like anyone can use a mass storage device and copy these valuable assets. And if it happened, the result will be dangerous and irreversible. So control the USB devices, that what data they handle. So a authorized devices list is created if authorized device is plug, it is acceptable otherwise it is deny with a message that “This device is not authorized”.

Index Terms— Unauthorized Access, USB devices, Access Control, Security

1 Introduction

Authentication is the means of verifying the identity of an entity. It can also be used to verify that information and data being transmitted is the same information that was originally sent and who sent it. Closely associated with authentication is authorization, which determines the level of rights and privileges available to the authenticated entity. User authentication enables organizations to control access to their systems. User authentication is today’s way of obtaining trust in the business world. It ensures that you control exactly who accesses your system and who you ultimately do business with.

Device authentication is an security area where only authorized devices are accessed by the system. Every usb device has a unique serial number, vendor id and product id. These unique ids are extracted by us using some commands at Linux terminal.

The main contributions of this paper are summarized as follows:

- We are firstly extracting the USB device unique id’s from the devices by using some commands.
- A verify list is created when usb device is plug into the system this list is called or run.
- Another two list are also created where device serial number and their vendor and product ids are resided.

2 DESIGN

In this section, we outline the design principles we follow.

Design Principles

Our principle is using easiest engineering, to achieve reasonable security enhancement including identity authentication, connection authorization. Our design philosophies are outlined as follows:

- Prevent users from installing any device. Allow users to install only devices that are on an "approved" list. If a device is not on the list, then the user cannot install it.
- Prevent users from installing devices that are on a "prohibited" list. If a device is not on the list, then the user can install it.

- Devices are such as video camera, finger print reader, pen drive, headphone and other usb devices.

This guide describes the device installation process and introduces the identification strings that Linux uses to match a device with the device unique id's available in the system.

Following are descriptions of the scenarios presented in this paper:

Prevent installation of all devices : In Linux system we create a list of devices that are allowed to accessed by the system. All usb devices are not accessed by the system because usb ports are disable by the user and only authorized devices are accessed.

Allow users to install only authorized devices: The system wants to allow users to install only the devices included on a list of authorized devices. To complete this step, we create a list of authorized devices so that users can install only those devices that we can specify.

Control the use of USB devices: In this step system wants to prevent standard users from writing data to removable storage devices, or devices with removable media, such as a USB memory drive. To complete this step, we configure a computer policy to allow read access, but deny write access to your sample devices.

3 IMPLEMENTATION

We performed experiments in order to quantify the performances and check the authenticity of devices. In Linux we did this experiment in Udev. Udev provides a dynamic device directory containing only the files for actually present devices. It creates or removes device node files usually located in the /dev directory, or it renames network interfaces. As part of the hot plug subsystem, udev is executed if a kernel device is added or removed from the system. On device creation, udev reads the sysfs directory of the given device to collect device attributes like label, serial number or bus device number. These attributes may be used as keys to determine a unique name for the device. udev maintains a database for devices present on the system. On device removal, udev queries its database for the name of the device file to be deleted.

Configuration All udev configuration files consist of a set of lines of text. All empty lines and lines beginning with a '#' will be ignored.udev expects its main configuration file at */etc/udev/udev.conf*.

The name of the udev rules file or directory to look for files with the suffix *.rules*. All rule files are read in lexical order. The default value is */etc/udev/rules.d/*.

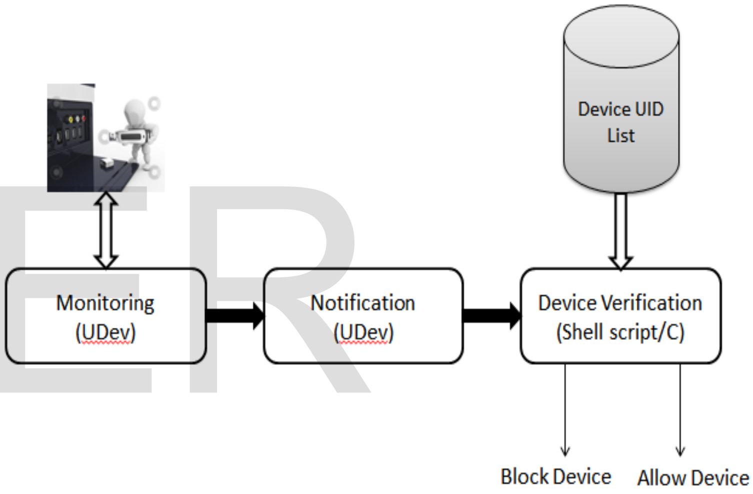


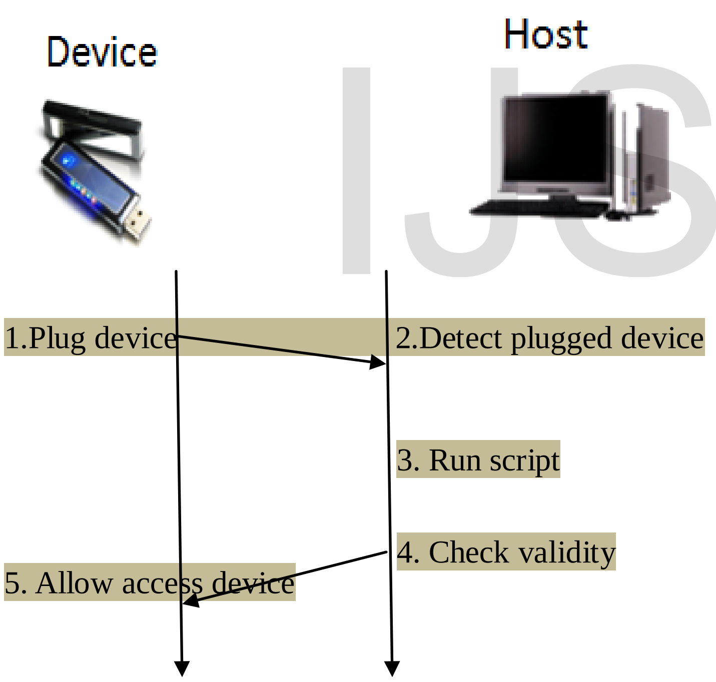
Fig1. Device Verification Modules

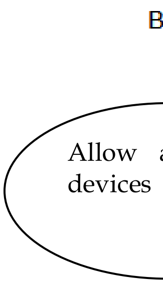
The analysis result of the data can be highlighted as follows:

- Firstly we write a script, when a usb device is plug into the system this script is run.
- After that we define udev rules in /etc/dev/udev directory. These local rules are defined by user for adding and deleting the usb devices or for blocking USB ports of system.
- Using command we fetch the serial number of usb devices and create a list of authorized devices in Linux system.

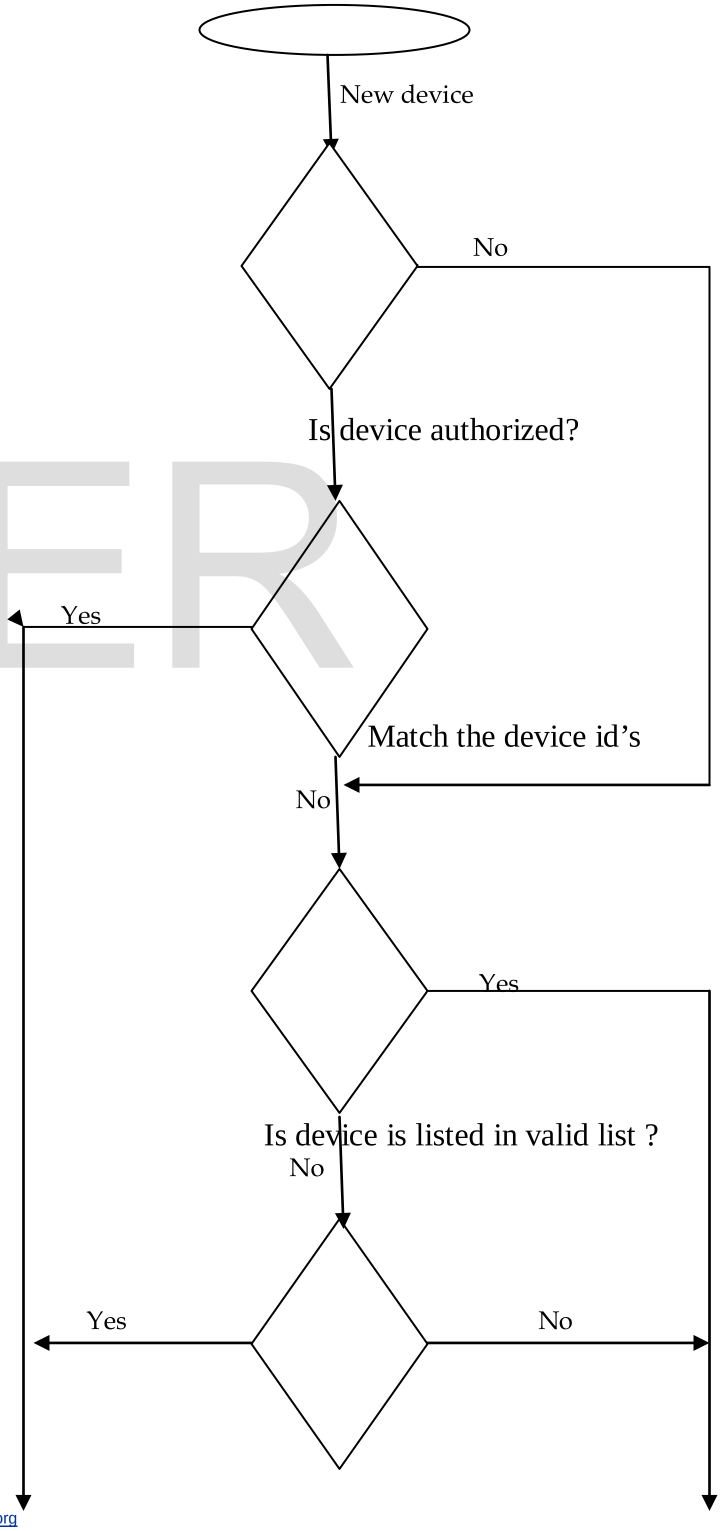
- If any devices which did not have a serial number we fetch the vendor and product id's of usb devices, and create a valid list of vendor and prduct id's .
- A log file is also creating that show the device is accepted or not.
- When usb device is plug into the system script is run. If device id's is match that are reside into the list, system will allow to access those usb device.
- If device id's is not matched by the valid list, script is run and usb device is denied with a time and date message.

Registration process for USB devices





Flow graph for process of device authentication



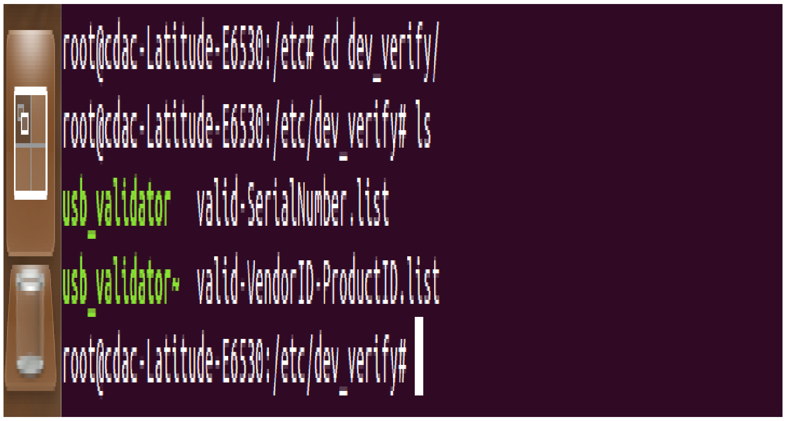
Check list of valid device id's

Prevent access of all devices

RESULTS

1) Local Rules defined for adding and removing usb devices and block all ports.

2) List of valid device id's

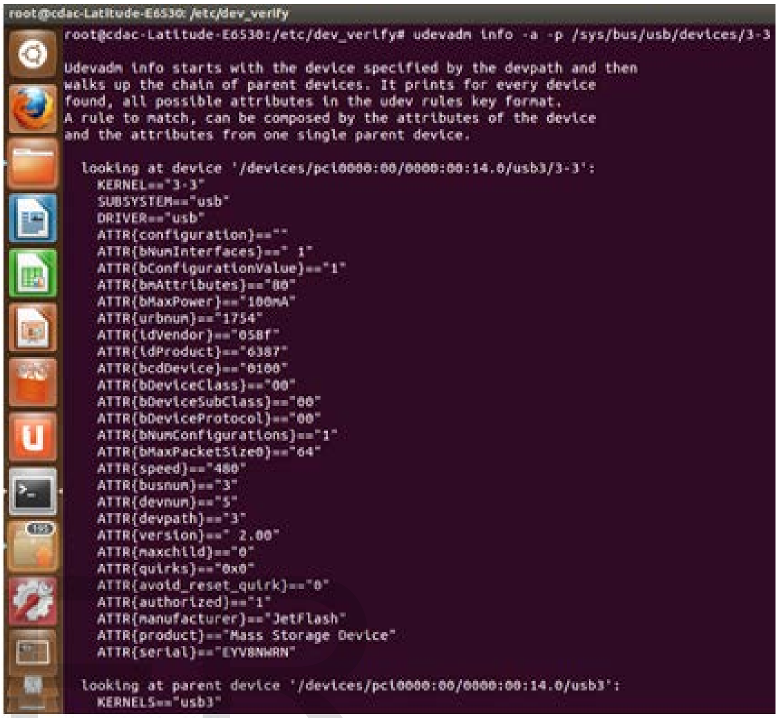


```
root@cdac-Latitude-E6530:/etc# cd dev_verify/
root@cdac-Latitude-E6530:/etc/dev_verify# ls
usb_validator   valid-SerialNumber.list
usb_validator~  valid-VendorID-ProductID.list
root@cdac-Latitude-E6530:/etc/dev_verify# 
```

3) Extra

Prevent access of all devices

b devices.



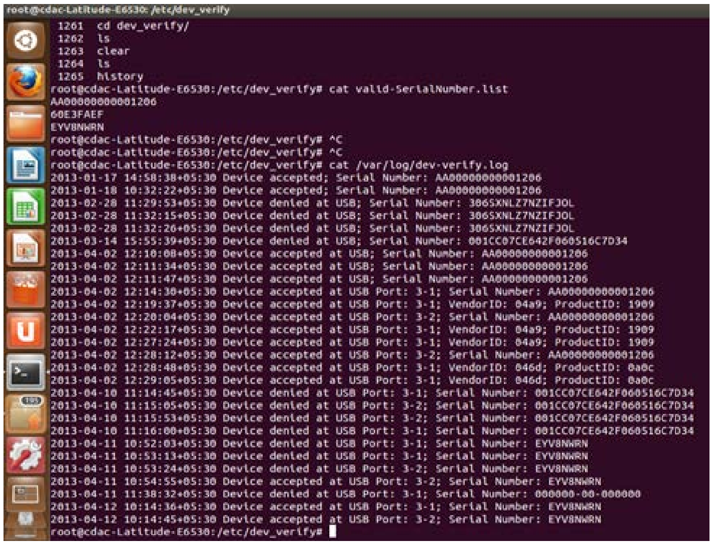
```
root@cdac-Latitude-E6530: /etc/dev_verify
root@cdac-Latitude-E6530:/etc/dev_verify# udevadm info -a -p /sys/bus/usb/devices/3-3

Udevadm info starts with the device specified by the devpath and then
walks up the chain of parent devices. It prints for every device
found, all possible attributes in the udev rules key format.
A rule to match, can be composed by the attributes of the device
and the attributes from one single parent device.

  looking at device '/devices/pci0000:00/0000:00:14.0/usb3/3-3':
    KERNEL=="3-3"
    SUBSYSTEM=="usb"
    DRIVER=="usb"
    ATTR{configuration}==""
    ATTR{bNumInterfaces}==" 1"
    ATTR{bConfigurationValue}=="1"
    ATTR{bmAttributes}=="80"
    ATTR{bMaxPower}=="100mA"
    ATTR{urbnum}=="1754"
    ATTR{idVendor}=="058f"
    ATTR{idProduct}=="6387"
    ATTR{bcdDevice}=="0100"
    ATTR{bDeviceClass}=="00"
    ATTR{bDeviceSubClass}=="00"
    ATTR{bDeviceProtocol}=="00"
    ATTR{bNumConfigurations}=="1"
    ATTR{bMaxPacketSize0}=="64"
    ATTR{speed}=="480"
    ATTR{busnum}=="3"
    ATTR{devnum}=="5"
    ATTR{devpath}=="3"
    ATTR{version}==" 2.00"
    ATTR{maxchild}=="0"
    ATTR{quirks}=="0x0"
    ATTR{avoid_reset_quirk}=="0"
    ATTR{authorized}=="1"
    ATTR{manufacturer}=="JetFlash"
    ATTR{product}=="Mass Storage Device"
    ATTR{serial}=="EYV8NWRN"

  looking at parent device '/devices/pci0000:00/0000:00:14.0/usb3':
    KERNELS=="usb3"
```

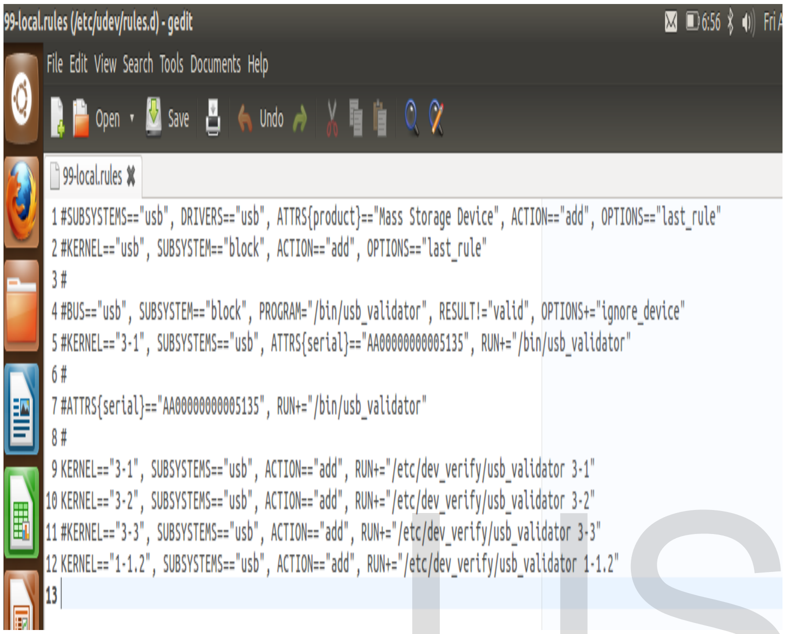
4) Show which device is accepted or which device is denied with their unique id's



```
root@cdac-Latitude-E6530: /etc/dev_verify
 1261  cd dev_verify/
 1262  ls
 1263  clear
 1264  ls
 1265  history
root@cdac-Latitude-E6530:/etc/dev_verify# cat valid-SerialNumber.list
AA00000000001206
60E3FAEF
EYV8NWRN
root@cdac-Latitude-E6530:/etc/dev_verify# ^C
root@cdac-Latitude-E6530:/etc/dev_verify# ^C
root@cdac-Latitude-E6530:/etc/dev_verify# cat /var/log/dev-verify.log
2013-01-17 14:58:38+05:30 Device accepted; Serial Number: AA00000000001206
2013-01-18 10:32:22+05:30 Device accepted; Serial Number: AA00000000001206
2013-02-28 11:29:53+05:30 Device denied at USB; Serial Number: 306SXNLZ7NZIFJOL
2013-02-28 11:32:15+05:30 Device denied at USB; Serial Number: 306SXNLZ7NZIFJOL
2013-02-28 11:32:26+05:30 Device denied at USB; Serial Number: 306SXNLZ7NZIFJOL
2013-03-14 15:55:39+05:30 Device denied at USB; Serial Number: 001CC07CE642F060516C7D34
2013-04-02 12:10:08+05:30 Device accepted at USB; Serial Number: AA00000000001206
2013-04-02 12:11:34+05:30 Device accepted at USB; Serial Number: AA00000000001206
2013-04-02 12:11:47+05:30 Device accepted at USB; Serial Number: AA00000000001206
2013-04-02 12:14:30+05:30 Device accepted at USB Port: 3-1; Serial Number: AA00000000001206
2013-04-02 12:19:37+05:30 Device accepted at USB Port: 3-1; VendorID: 04a9; ProductID: 1909
2013-04-02 12:20:04+05:30 Device accepted at USB Port: 3-2; Serial Number: AA00000000001206
2013-04-02 12:22:17+05:30 Device accepted at USB Port: 3-1; VendorID: 04a9; ProductID: 1909
2013-04-02 12:27:24+05:30 Device accepted at USB Port: 3-1; VendorID: 04a9; ProductID: 1909
2013-04-02 12:28:12+05:30 Device accepted at USB Port: 3-2; Serial Number: AA00000000001206
2013-04-02 12:28:48+05:30 Device accepted at USB Port: 3-1; VendorID: 046d; ProductID: 0a0c
2013-04-02 12:29:05+05:30 Device accepted at USB Port: 3-1; VendorID: 046d; ProductID: 0a0c
2013-04-10 11:14:45+05:30 Device denied at USB Port: 3-1; Serial Number: 001CC07CE642F060516C7D34
2013-04-10 11:15:05+05:30 Device denied at USB Port: 3-2; Serial Number: 001CC07CE642F060516C7D34
2013-04-10 11:15:53+05:30 Device denied at USB Port: 3-2; Serial Number: 001CC07CE642F060516C7D34
2013-04-10 11:16:00+05:30 Device denied at USB Port: 3-1; Serial Number: 001CC07CE642F060516C7D34
2013-04-11 10:52:03+05:30 Device denied at USB Port: 3-1; Serial Number: EYV8NWRN
2013-04-11 10:53:13+05:30 Device denied at USB Port: 3-1; Serial Number: EYV8NWRN
2013-04-11 10:53:24+05:30 Device denied at USB Port: 3-2; Serial Number: EYV8NWRN
2013-04-11 10:54:55+05:30 Device accepted at USB Port: 3-2; Serial Number: EYV8NWRN
2013-04-11 11:38:32+05:30 Device denied at USB Port: 3-1; Serial Number: 000000-00-000000
2013-04-12 10:14:36+05:30 Device accepted at USB Port: 3-1; Serial Number: EYV8NWRN
2013-04-12 10:14:45+05:30 Device accepted at USB Port: 3-2; Serial Number: EYV8NWRN
root@cdac-Latitude-E6530:/etc/dev_verify# 
```

4 Conclusion

This paper introduces an access control model for USB devices. It has device authentication and operate by a authorized user. Only authorized user and devices are accessed by the system. Main function of this utility is to secure the system from unauthorized devices. Features of this utility are user authentication and device control. Through this utility security of the system is increase. We have implemented a fully working prototype of device authentication based on USB implementation in Linux kernel and our experimental results demonstrate its practicality and effectiveness.


99-local.rules (/etc/udev/rules.d) - gedit
6:56
File Edit View Search Tools Documents Help
Open
Save
Undo
99-local.rules

```
#SUBSYSTEMS=="usb", DRIVERS=="usb", ATTRS{product}=="Mass Storage Device", ACTION=="add", OPTIONS=="last_rule"
#KERNEL=="usb", SUBSYSTEM=="block", ACTION=="add", OPTIONS=="last_rule"
#
#BUS=="usb", SUBSYSTEM=="block", PROGRAM="/bin/usb_validator", RESULT!="valid", OPTIONS+="ignore_device"
#KERNEL=="3-1", SUBSYSTEMS=="usb", ATTRS{serial}=="AA00000000005135", RUN+="/bin/usb_validator"
#
#ATTRS{serial}=="AA00000000005135", RUN+="/bin/usb_validator"
#
KERNEL=="3-1", SUBSYSTEMS=="usb", ACTION=="add", RUN+="/etc/dev_verify/usb_validator 3-1"
KERNEL=="3-2", SUBSYSTEMS=="usb", ACTION=="add", RUN+="/etc/dev_verify/usb_validator 3-2"
#KERNEL=="3-3", SUBSYSTEMS=="usb", ACTION=="add", RUN+="/etc/dev_verify/usb_validator 3-3"
KERNEL=="1-1.2", SUBSYSTEMS=="usb", ACTION=="add", RUN+="/etc/dev_verify/usb_validator 1-1.2"
```

Acknowledgment

I would like to thank my Brother and my friends who guide me and motivate me to write this paper.

References

[1] Hanjae Jeong, ”Vulnerability Analysis of Secure USB Flash Drives,” Journal of the KIISC, vol. 17, No. 6, pp.99-118, December 2007

[2] Kangbin Yim, ”A fix to the HCI specification to evade ID and password exposure by USB sniff,” Proceedings of APIC-IST 2008, pp.191-194, December 2008

[3] Kangbin Yim, ”A new noise mingling approach to protect the authentication password,” IEEE, proceedings of CISIS 2010 Conference, pp. 839-842, February 2010

[4] Microsoft MSDN Library –WM_DEVICECHANGE Message,http://msdn.microsoft.com/en-us/library/aa363480(VS.85).aspx, 2009.02.23

[5]www.securitymanagement.com_archive_library_Identity_management1003.pdf