# USB Device Management in GNU/Linux Systems

Edilberto Deroncelé, Allan Fuentes, Dayana Tejera Hernández, Haniel Cáceres Navarro, Abel Fírvida Donestévez, Michel Febles Parker



# USB device management in GNU/Linux Systems

Eng. Edilberto Blez Deroncelé[1], MSc. Allan Pierra Fuentes[2], MSc. Dayana Caridad Tejera Hernández[3], Eng. Haniel Cáceres Navarro[4], Eng. Abel Alfonso Fírvida Donestévez[5], Eng Michel Evaristo Febles Parker[6]

1 University of Informatic Science, Free Software Center, Road to San Antonio de los Baños, 2 ½ Km, Torrens, La Havana, Cuba,
Email: eblez@uci.cu,
WWW home page: http://www.uci.cu

2 University of Informatic Science, Free Software Center, Road to San Antonio de los Baños, 2 ½ Km, Torrens, La Havana, Cuba,
Email: apierra@uci.cu,
WWW home page: http://www.uci.cu

3 University of Informatic Science, Free Software Center, Road to San Antonio de los Baños, 2 ½ Km, Torrens, La Havana, Cuba,
Email: dtejera@uci.cu,
WWW home page: http://www.uci.cu

4 University of Informatic Science, Free Software Center, Road to San Antonio de los Baños, 2 ½ Km, Torrens, La Havana, Cuba,
Email: hcaceres@uci.cu,
WWW home page: http://www.uci.cu

5 University of Informatic Science, Free Software Center, Road to San Antonio de los Baños, 2 ½ Km, Torrens, La Havana, Cuba,
Email: aafirvida@uci.cu,
WWW home page: http://www.uci.cu

6 University of Informatic Science, Free Software Center, Road to San Antonio de los Baños, 2 ½ Km, Torrens, La Havana, Cuba,
Email: mfparker@uci.cu,
WWW home page: http://www.uci.cu

**Abstract**. Protecting the access to USB ports has the same priority for information security than firewalls and antivirus software. Nowadays there are some tools that allow us to monitor and regulate the access to USB devices, but all of them are distributed under proprietary licenses. This work presents an application that solves the mentioned problem: ¿How controlling the access to USB mass storage devices in GNU/Linux Operating Systems?

# 1 Introduction

One of the most important advantages of devices guided by USB industrial standard is the access easiness to every kind of computer that provides its usage. Theoretically this can be an advantage to the enterprises, except for the fact that concepts “security” and “access” are completely opposite in the Security Information area.

After the creation of the Universal Serial Bus, at the beginning of 90s, the development of technologies around of USB devices has been constantly evolving. The most recently versions of this kind of devices have increase its storage capacity, which implies the improvement of its performance and the decrease of its height.

So, with USB devices user can count with an easy means of transport, to store information while maintaining its availability. Anyway, the easy access of these devices to computers and the sensitivity of the information handled by them, can bring a set of disadvantages and vulnerabilities, both for end users and businesses. Deliberate or accidental users can:

- Remove critical data.
- Exhibit confidential information.
- Introduce malicious code that can affect the entire corporate network.
- Transferring inappropriate or offensive hardware company material.
- Make personal copies of company information and intellectual property.
- Connect portable devices and consequently distract workers during working hours.

In an attempt to control these threats, companies began to prohibit use in workstations of USB devices for personal use. However, practice indicates that trusting on voluntary compliance by users is not enough. The best way to monitor and regulate access, and maintain control of these devices has been by placing technological barriers.

Migration to open source technologies and open standards in the companies has found barriers in this regard. The protection of information and the USB ports of computers was made in companies using proprietary applications like "MyUSBOnly", "GFI EndPointSecurity" and others. How it has not been found a similar application in the field of free software. This research persues the goal of present a tool similar behavior but to enviroments GNU/Linux. The version we have today in GNU/Linux specifically Cuban GNU/Linux is the Unix Smart Blocker for Universal Serial Bus, USB$^2$ 1.0 tool, which is the application shown in this paper.

## 2 Contents

A technological barrier to the uncontrolled use of USB devices in a company is the solution proposed in this research. The application USB$^2$ 1.0 will allow controlling and restricting the use of USB devices on computers that use GNU operating systems.

The code of the application currently in development can be located https://github.com/editoblezd/usbsecurity-dev

### 2.1 Methodology

During this research were used scientific methods: analysis-synthesis, historical analysis and experimental one, which allow us the foundations for the development of the application.

By Analysis-synthesis method were studied the most used technologies nowadays to identify its main features; which were the basis for the design and implementation of the first version of USB$^2$ 1.0. Some of them were: control of USB ports via a database with authorized devices and alerting users to the authorization or not of new devices.

Historical analysis method allows us to study the evolution of this kind of application and as a result, to identify the need that existed in the distributions of GNU/Linux for a tool to control access to USB storage devices. University of Informatics Sciences (UCI) users and the free software community were the main samples for this research.

The experimental method allowed testing the correction of the functionalities implemented with main samples.

The development of the application was useful to the user community GNU/Linux, because is an alternative to protect users from data loss, data robbery, or other negative consequences of the use of USB mass storage devices. Once developed the tool, safety levels in environments GNU increased and hence the users' trust in free operating systems.

### 2.2 Results and discussion

Some of the tools used to control USB devices are: USB Blocker, Device Lock, GFI EndPoint, MyUSBOnly, Endpoint Protector 4 and USB Over Network Server; but most used are the first five.

The freeware software USB Blocker created by NetWrix Corporation, is an interesting centralized locker that allows controlling access to the computers on a network using enabled USB devices. It locks different kinds of mass storage devices, such as removable hard disks, iPods and others [1].

USB Blocker can control the access of authorized or unauthorized devices, functioning as a mean of protecting security control malware, viruses or loss of sensitive in-

formation for the corporate network where is used. Unlike other solutions, does not require installation on each computer on the network, functioning as an integral network control system. Its main disadvantage is that in addition to be proprietary is conceived only for Windows systems.

Device Lock allows defining which users can access to specific ports and devices on a computer. Its management can be made using group policies in an Active Directory domain, creating a console for administration [2]. But, like USB Blocker, is available only for Windows platforms NT/2000/XP/Vista/7 and Windows Server 2003/ 2008, and is proprietary. Among the main features that can be managed with Device Lock are: users or groups access control, a whitelist of devices regardless of where they connect, ports auditing and integration of TrueCrypt and PGP Whole Disk encryption.

GFI EndPoint is another powerful tool that has among its main functions: group-based protection control, granular access control , support for various kinds of portable devices, logging user activity in SQL Server , the agent protection with increased security, control of portable storage with Unicode support and the event log [3].

MyUSBOnly provides the least amount of functionalities, but it is the most used by companies and end users. Allows quick and easy way to protect USB ports and set a password to access them. Requires Microsoft Windows 2000, XP, 2003, Vista, 7 or 2008 [4] as operating system.

All these applications are developed for Microsoft Windows operating systems which makes slower the migration process of companies to free software.

Currently, the tool Endpoint Protector 4, brand development some guidelines for tool described in this paper. Although this tool solves the problem posed in this investigation, but it is not distributed under GPL license, which becomes a obstacle for GNU environments.

The Endpoint Protector 4 Web administration and reporting console offers a complete overview of the device activity on your computers, whether you work with Windows, Mac or Linux platforms. The enterprise will be able to define access policies per user/computer/device and authorize devices for certain user or user groups. Thus, the company will stay productive while maintaining control over the device fleet use.[5]

The tool presented in this paper aims to be fast, efficient and simple. It has good acceptance by Cuban users and is compatible with GNU/Linux systems and is distributed under GPLv3.0 License. The application libraries are distributed under the LGPL v3.0 license. USB$^2$ 1.0 increases safety levels required in Cuba and its usage can be included in the safety regulations must have any cuban institution under Resolution No.127/2007 [6 ].

### 2.3 Development Technology

For the development of the application was used C, as the programming language, using glibc libraries and gtk +-2.0 for interfaces. Was used vim as Integrated Development Environment and to store information in the database is used squlite3 library.

USB[2] 1.0 1.0 has incorporated a default graphical user interface in text mode. The tool, developed as a library, provides the necessary tools so you can build a graphical interface for different environments.

### 2.4 Recognition of the device in the computer

Udev is the device manager for the Linux kernel. Essentially this device handles devices nodes that are regularly can be found in /dev directory. This is the successor of devfs and hotplug, so it handles /dev directory and all actions taken in user space when devices are added and subtracted in the computer.

How udev by default is active for all systems that use the version 2.6.x of the kernel or upper, is used to recognize devices in the system. In the process of connecting or disconnecting a device to the system, udevd demon executed by udev receives a uevent released by kernel indicating that an action of adding or disconnecting a device has been captured.

Sysfs is a virtual file system that provides the Linux kernel v2.6. Exports information about devices and drivers from the kernel device model to userspace and also allows to set its parameters. When a device is attached to the system, in addition to the information stored by udev in its database, the kernel enclosed information necessary for its work in the virtual/sys directory.

Among the possibilities of udev are its rules. These rules have a syntax that allows both read attributes of a device and change others, depending on the type of attribute. A set of rules decide what action needs to be done with obtained data. What rule will do, probably, will be to name the device, to create the appropriate device file, and run the program that has been set to activate the device.

Rules can associate a unique name to a device, but can also call an external program to give more information about the device; so that can obtained a more specific name. Most of the rules that come by default in the system are stored in /lib/udev/rules.d/.

By using these rules is performed a device recognition and are selected only storage devices to manage access to the system. The attributes that are granted to allow uniquely identify devices and thus can control them.

The device recognition is an important phase in its access control process system to the system. If this action is performed successfully can be achieved the authorization of only those devices that have been previously authorized by the administrator or superuser.

The rule would look like this:

*#Regla para usbsecurity1.0*
*ACTION=="remove",GOTO="disk_usb_end"*
*SUBSYSTEM!="block", GOTO="disk_usb_end"*
*# Para importar información de los padres.*
*ENV{DEVTYPE}=="partition",IMPORT{parent}="ID_*"*
*#Ignorar discos floppy*
*KERNEL=="fd*",ENV{UDISKS_PRESENTATION_HIDE}="1"*
*KERNEL=="sd*[!0-9]|sr*",ENV{ID_SERIAL}!="?*",SUBSYSTEMS=="usb",IMPORT{program}="usb_id --export %p"*
*#Ejecutar un programa al conectar dispositivos USB.*
*KERNEL=="sd?",ACTION=="add",ENV{ID_BUS}=="usb",SUBSYSTEM=="block",SUBSYSTEMS=="usb",PROGRAM!="/usr/sbin/usbsecurity /dev/%k", ENV{UDISKS_PRESENTATION_HIDE}="1"*
*ENV{DEVTYPE}=="disk", KERNEL!="sd*|sr*", ATTR{removable}=="1", GOTO="disk_usb_end"*
*LABEL="disk_usb_end"*

As can be seen when a storage device is connected, this is the action is ACTION = "add", the "usbclient" program is executed passing the name of the device node recognized by the kernel as argument. This program performs device access check tasks and returns true or false depending on whether the device is not authorized to access the computer or if it is respectively. In case of been authorized, the rule does not reach its end and the device goes through the normal connection and removes process. Otherwise does not allow accessing the computer by removing the device node so that can not do access device to the system even with the root password.

**2.5 Verification of access permission.**

For verification of system access, is necessary to know several device attributes such as serial number. For getting such attributes are used programs already offered by udev as usb_id which exports device information to the udev environment variables, in this case specifically are exported the attributes with udev format and the order:

*IMPORT{program}="usb_id --export %p", escrita en la regla de udev si es que otro programa no la han importado ya. La sección que comprende esta orden ejecuta de forma general:*
*# USB devices use their own serial number*
*KERNEL=="sd*[!0-9]|sr*",*
*ENV{ID_SERIAL}!="?*",SUBSYSTEMS=="usb",*
*IMPORT{program}="usb_id --export %p"*

Once imported to this information can access this in the written program usbsecurity.c using the function: g_getenv(ENV_SERIAL), which returns the serial number of the device. In addition is needed to know what type of device it is, this is if it is a partition or a disk. That is obtained through the g_getenv( ENV_DEVTYPE) function which reads this environment variable.

Once imported this information, application checks if the serial device is already registered in its database, thus deciding the return value. This return value depends on whether the device is finally shown to the user or not.

### 2.6 Managing devices

The application has a graphical user interface with which the devices are managed, a new device can be added to the database, can be deleted and modified its identifier too. By default when a device is connected and it is not authorized is recorded in the database but is not authorized. In the ID field is set by default Not allowed, which means that although this is not registered can not access the system until you can not change the ID.

#### Adding a device

To add a device must be connected to the computer and if it is not authorized, will show the user a notification to let him know the current state of the device. To authorize a system, administrator must access the management interface for the device approval process.

In this interface the administrator must change the default identifier that takes the device, to authorize it. If, however, you will not change this ID after log all devices that still have the original identifier will be deleted from the database. Once the device is authorized it will be mounted automatically in the file system of the operating system. Once the device is authorized, is launched to the user a notification informing the changes.

#### Sorting an authorized device identifier

The process of modifying the device ID, performs in the same way that of adding a device to the system. The difference lies in that the device is already authorized in the system and as a first step the administrator or super user had to change the default identifier.

#### Removing a Device

To remove the device, simply must be selected in the administration interface and click the button to the elimination. After acceptance of the confirmation message, the device will be removed from the database and will automatically unauthorized use in the system.

**Activity Logging**

Internally the system, in addition to the information displayed in the Admin GUI, store other attributes in system logs. When any activity, both connection authorization and de-authorization is performed, is stored information relating to this as time, date, and user who performed the action and the serial number of the device. This helps finding events if an abnormality arises, to find the root of the problem. It is important to note that how this valuable information is stored, it can be known information about any device that attempts performing any action, which ensures an extra level of security, in this case the detection.

## 3 Conclusions

USB Unix Smart Blocker 1.0 provides another point of support for the growing process of migration to open source platforms. With this version of the product the administrators of workstations will be able to control the access of USB devices, increasing levels of security for enterprises.

From the economic point of view, for a country like Cuba, paying for software licenses over 600,000 workstations migration process is a rather high figure. USB Unix Smart Blocker 1.0 is distributed under GPL license, besides being a value-added product to the Cuban distribution of GNU / Linux Nova, used as the main distribution in the migration process of the country.

The architecture used in the development of the application allows for future expansion and scalability of it. What provides companies with a tool adaptable to the conditions they need.

$USB^2$ 1.0 has arrived as a free alternative in GNU/Linux systems to monitor and control USB devices.

## 4 Recommendations

It is recommended:

1. To expand the number of devices that are supported by this tool, to have control of these.
2. To use this tool to protect your USB ports, measure as important as installing a firewall and antivirus program maintenance. $USB^2$ 1.0 makes easy the protection of this critical part of your PC or laptop.

## 5 References

1. JOSÉ ALEJANDRO. USB Blocker, eficiente bloqueador de acceso USB para red [Online], 2009. [Accessed: October 29th, 2012]. Available on: http://www.acercadeinternet.com/usb-blocker-eficiente-bloqueador-de-acceso-usb-para-red/.

2. PYME. DeviceLock, control total sobre los USB en la empresa [Online]. 2010. [Accessed: October 29th, 2012]. Available on: http://www.tecnologiapyme.com/software/devicelock-control-total-sobre-los-usb-en-la-empresa.

3. GFI ENDPOINTSECURITY. Feature [Online]. 2006. [Accessed: October 29th, 2012]. Available on: http://www.gfi.com/usb-device-control#features.

4. MYUSBONLY. SYSTEM REQUIREMENTS [Online]. 2012. [Accessed: October 29th, 2012]. Available: http://www.myusbonly.com/usb-security-control-device-protection/.

5. ENDPOINT PROTECOR 4 [Online], 2013. [Accessed: December 26th, 2013]. Available on: http://www.endpointprotector.com/products/endpoint_protector#1

6. MINISTERIO DE LA INFORMÁTICA Y LAS COMUNICACIONES. RESOLUCION No. 127 /2007. [PDF] 2007. [Accessed: October 29th, 2012].