A distributed architecture for remote validation of software licenses using USB/IP protocol

Mário J. Antunes [1,2], Albano Afonso[1], Filipe Mota Pinto[1,3]

[1] School of Technology and Management, Polytechnic Institute of Leiria
Morro do Lena, Alto do Vieiro, 2411-901 Leiria, Portugal
mario.antunes@ipleiria.pt, 2091357@my.ipleiria.pt, fpinto@ipleiria.pt

[2] Center for Research in Advanced Computing Systems (CRACS)
Faculty of Science, University of Porto, R Campo Alegre, 1021/1055 4169-007 Porto Portugal
mantunes@dcc.fc.up.pt

[3] Centro Algoritmi - University of Minho; Campus Azurém, 4800-058 Guimarães, Portugal

Abstract. USB dongles have been used by a wide range of software manufacturers to store a copy-protected of their application's license. The licenses validation procedure through USB dongles faces several concerns, as the risks of theft or losing dongle. Also, in scenarios where the number of dongles is reduced, users may have to wait for dongle access, which may lead to loss of productivity. In this paper we propose a client/server distributed architecture for remote software licenses validation, through USB/IP protocol. The proposed approach aims to take advantage of USB/IP for distributed access to a set of USB dongles physically connected to a remote USB server, over a TCP/IP network. We describe the deployment and enhancements made to an existing open source USB/IP implementation and also present the results obtained with this architecture in a real world scenario, for validation of computer forensics applications licenses that uses USB dongles.

Keywords: USB/IP, dongle USB, software license, distributed systems.

1 Introduction

A dongle is a device with an USB interface, used to store a copy-protected software license. There is a wide range of such applications that use exclusively USB dongles for their license validation. Those software applications are usually very expensive and commonly used in vertical markets, in which vendors offer specific services to an industry or group of customers with particular needs. Some examples are CAD/CAM applications (e.g. AutoCAD©), some business retail software, prepress and printing software and some imaging and scanning equipments and applications used in healthcare business.

The main concerns regarding the use of USB dongles are related to the risks of theft and loss, which may grant access to the applications by unauthorized users. In dynamic organizations, with several users geographically dispersed and accessing to the same application, by having less USB dongles than the number of users that need

to use the application, may lead to a decreasing of productivity, since each user has to receive the dongle from another user that previously used it and no longer need it.

In this paper we propose an open source distributed architecture for remote access to USB dongles physically connected to a server, through a local TCP/IP network [1]. In our work we take advantage of an open source implementation of the USB/IP protocol [1], commonly used to access to USB devices, like external disk or pen devices, that are attached to a remote computer. The underpinning ideal is to physically attach the USB dongles in a server running USB/IP protocol and accepting requests from the USB/IP clients. The USB/IP client applications run on the computers where applications are installed and to whose we need to validate the corresponding license. The overall remote license validation process is transparent to the users and they did not have to carry the USB dongles with them. Moreover, this procedure keep the legal commitments previously agreed, regarding software licensing (e.g. number of allowed users), since only one user is allowed to access each USB dongle (license) in a given moment. In our work we have also developed a graphical user interface for the USB/IP client, which improved the way each USB/IP client manages the virtual connections to USB devices through USB/IP protocol. The distributed architecture proposed in this paper aims to overcome the major limitations of commercial applications, as it is open source and the client application was designed to be customized for specific companies and application domains.

The existing commercial applications for remote access to USB devices through USB/IP reveal some drawbacks. That is, their licenses are expensive and charged in a per dongle basis. As the number of attached USB dongles increase so the value of the license. Those applications also offer a general interface to access USB device contents and cannot be customized for specific applications, like licenses validation.

Computer forensics applications have been used as a case study in our work. These applications operate in a very specific field and are used to validate and analyze digital evidences collected by criminal police searches [2]. The major challenge in police departments dedicated to analyze the collected evidences is to assure the feasibility of digital evidences under investigation. In order to help police examiners on analyzing digital evidences collection, there are two well known and commercial software computers forensics applications widely used by police departments, namely Forensic Toolkit (FTK)© [3] and EnCase© [4]. For both applications, the license is stored in an USB dongle and when the user intends to gain access to the software application, attach it to an USB port and wait for the corresponding permission. If there is no valid license in USB dongle, the user will not have access to the application [5].

We have made several acceptance tests in the computers forensics laboratory of Portuguese criminal Police, namely on validating the remote access to FTK© and EnCase© applications licenses. The deployed system is now being used experimentally by the computers forensics laboratory in their daily investigation routine tasks.

The rest of the paper is organized as follows. In Section 2 we introduce the necessary background to our work, namely the existing USB/IP applications and the protocol USB/IP. Section 3 presents the proposed architecture, followed by the development made and experimental results in Sections 4 and 5 respectively. Finally, Section 6 presents some conclusions and future work.

2 Background

In this section we present the background on USB/IP protocol and implementations, which constitutes the generic knowledge for understanding the approach proposed in this paper.

2.1 USB/IP applications

There are a wide range of applications that tries to overcome the physical limitation to access to USB ports, by using a remote connection through client/server USB/IP protocol. Generally, the USB devices are connected to the USB/IP server and exported for further network access by USB/IP clients.

Hirofuchi, et al. [6] firstly proposed an open source application to have a remote access to USB ports. In this first USB/IP implementation, USB devices were shared in a TCP/IP local network, in which both client and server applications ran in Linux operating systems. In 2007, Kwon et al. [7] developed a new implementation of USB/IP server with an extension for interoperability with Microsoft Windows USB/IP client. Later on, in 2009 the first USB/IP in Widows was developed, based on the commands execution in a command line interface [8]. In 2011, an enhanced version of USB/IP client for Microsoft Windows was released, with the support of ReactOS. The new features include the development of digitally signed drivers and the compatibility with 32 and 64 bits Microsoft Windows operating systems [8].

The Linux kernel versions have evolved and existing USB/IP implementations became obsolete and incompatible with early Linux versions. However, in 2011 USB/IP source code was integrated natively in Linux kernel 2.6 drivers folders and was improved in its performance and with new features [8]. At the moment there is an USB/IP implementation for Linux kernel 2.6, available in the software repositories, which is however obsolete and incompatible with more recent Linux versions. Moreover, the actual version available has one important limitation related with the fully release of the remote USB device. When the client release the virtual connection with the remote USB device, the server unbind the device and make it unavailable and not ready to be used after that by a different user [8]. There is also some implementations of USB/IB for mobile networks. In [9] the authors propose a Android mobile implementation using a mobile phone as a wireless USB storage device through USB/IP connection.

There are some implementations for Microsoft Windows based systems, which enable virtual emulation of the USB ports between two or more computers [10]. Regarding commercial applications, the licenses are increasingly expensive and are related with the number of dongles provided by the server and with the clients accessing to it. Some examples are the USB Network Gate for Windows, USB Redirector and USB over Network. USB Redirector is developed by Incentives Pro [11] and allows the use of shared USB devices remotely on the local network, wide area network or Internet as if the devices were physically connected to the remote computer. It uses a TCP/IP connection and can act both as a server and as a client. USB over Network is developed by FabulaTech [12]. This application assures the access to local USB devices by other computers on the network that have the client

application. It supports a wide range of USB devices, encrypted data communications, compatibility with 32-bit and 64-bit systems and also with Hyper-V.

2.2 USB/IP Protocol

USB/IP protocol operates in a client/server model. The server exports the USB devices physically connected that are going to be used by the USB/IP clients. Moreover, the exported USB device drivers run on the USB/IP client. Figure 1(a) illustrates the request/respond USB/IP protocol messages. The client requests to the server a list of exported USB devices, by establishing a TCP/IP connection and by sending an `OP_REQ_DEVLIST` message. The server sends back a response with an `OP_REP_DEVLIST` which contains the list of its exported USB devices. After that, USB/IP client finishes the TCP/IP connection.

After obtaining the list of exported USB devices, the client may choose to use one of them, by establishing another TCP/IP session with the server. In this new session, client sends an `OP_REQ_IMPORT` message and the server answers with a corresponding `OP_REP_IMPORT` message. If the connection was succeeded, the TCP/IP connection remains established and will be used to transfer "USB Request Block (URBs)" [13] between USB/IP client and server.

USB/IP client may then send two types of messages: `USBIP_CMD_SUBMIT` to submit an URB; `USBIP_CMD_UNLINK` to unbind the session defined with the remote USB device. USB/IP server answers with an `USBIP_RET_SUBMIT` or `USBIP_RET_UNLINK` respectively. The process of importing an USB device is depicted in Figure 1(b). While the USB/IP server does not receive an `USBIP_CMD_UNLINK` message, USB messages are encapsulated in IP packets and sent by the TCP/IP network.

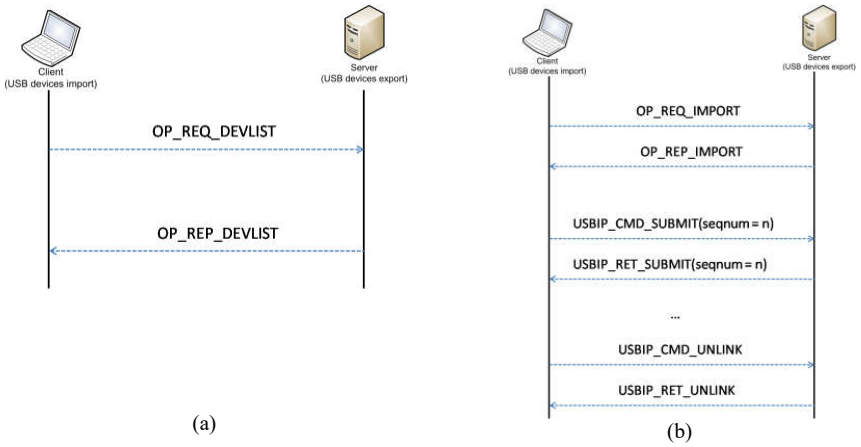
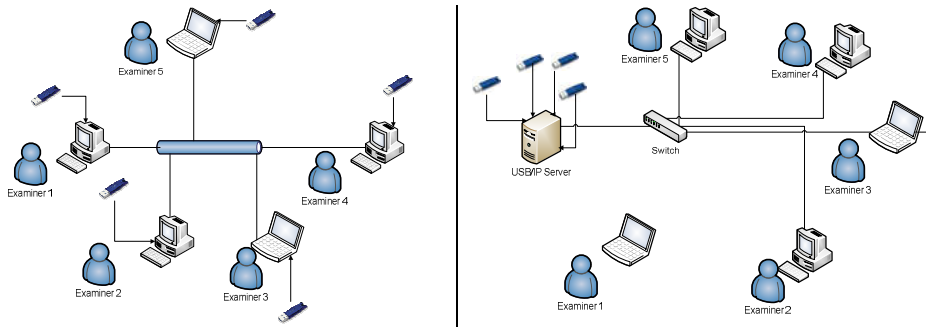


Fig. 1 (a) Request/Respond messages for obtaining a list of exported USB. (b) Messages involved on importing an USB device.

3. Proposed USB/IP distributed architecture

In this section we present the proposed open source architecture for remote validation of software license, by using USB dongles. This architecture is based on a distributed solution with remote access over USP/IP in a local TCP/IP network.

Our test scenario focused on the remote software licenses validation of computers forensics applications in use at Portuguese criminal Police. Figure 2(a) depicts the previous situation, in which the forensics applications were installed in each of examiner's computers and their license validation process was made locally through the USB dongle that stores a copy-protected license.



(a) (b)

Fig. 2 (a) Actual scenario with USB dongles physically connected to examiner's computers; (b) Proposed distributed architecture based on USB/IP client/server protocol.

In this scenario, whenever the number of required active licenses is greater than the available dongles, it is introduced an "*investigation wait-state*" or "*shares-resource*" status between examiners. The proposed and developed scenario is depicted in Figure 2(b). It has a central server that physically holds the USB bus root, accessible from elsewhere in the local network. USB/IP clients are those computers running forensics applications. Those machines use Microsoft Windows operating system with previously installed USB/IP drivers and the USB/IP client application that will interact with USB/IP server and it will manage all applications access requests throughout TCP/IP network. The USB/IP server application remains waiting for requests at TCP port 3240. The main security concern is to define a firewall rule that may allow USB/IP clients machines to have a service access to this TCP port.

Within this distributed architecture, the numbers of required dongles are exactly the same as in previous situation, in respect to the license agreement contract. However, in this distributed scenario we may not have an "*investigation wait-state*" or "*shares-resource*" status between the examiners.

4. Client and Server development

In this section we describe the development issues involved in the proposed distributed architecture, described in section 3. We present the major steps followed and actions taken on the development of client and server applications.

The development has been based on an already existing open source USB/IP project [6], as described on section 2. This project was initially conceived for a Linux kernel version 2.6, for both client and server applications. This approach faced some problems regarding the virtual USB connection management, which turned it in a non reliable version [8]. That is, whenever a client had ended his virtual connection to a USB device, the corresponding binding in the server was automatically removed, as well as from the exported devices list. In order to overcome this problem, it was necessary to share again the same device on the server and export it to the network. In our development we solved these issues and made available a new version of USB/IP server for Linux kernel version 3.2 [1].

The existing USB/IP client for Microsoft Windows was based on command line and thus not user friendly. Aiming to improve the application usability and user interaction with server application, we have developed a general and customizable graphical user interface (GUI) for USB/IP client, depicted in Figure 3.

Fig. 3 Generic graphical user interface (GUI) developed for USB/IP client.

The development was made in Microsoft Windows .Net 4.5 Framework. We have used C and Visual C++ programming languages. The former was used for previous existing code re-use and the later for GUI development. We have used a Dynamic-Link Library (DLL) for encapsulating the existing C language code. For the graphical user interface development we have used the Platform Invoke [14] functionalities of C++ programming language for access to encapsulated C code.

5. Experimental tests and results analysis

Figure 4 depicts the setup used in our tests made at computers forensics laboratory of the Portuguese criminal Police. It is composed by a client and a server interconnected by a switch, in the same IP network. In our experiments we have used the following USB dongles: CodeMeter [15] that stores FTK© software license and Aladdin HASP

HL [16], which has the EnCase© software license. USB/IP server runs in an Ubuntu 12.04 LTS system, while the USB/IP client is installed in a Microsoft Windows 7 system, in 64 bits. USB/IP client system has EnCase© and CodeMeter Control Center© applications, being the later responsible for FTK© users license management.

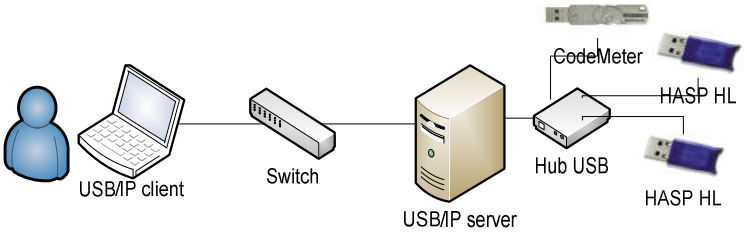


Fig. 4 Physical topology.

In this test users were able to establish a TCP/IP network connection to the USB/IP server in which the USB dongles were previously exported. The graphical user interface customized to this particular scenario allowed a simple and easy to use USB/IP dongles management, as depicted in Figure 5.

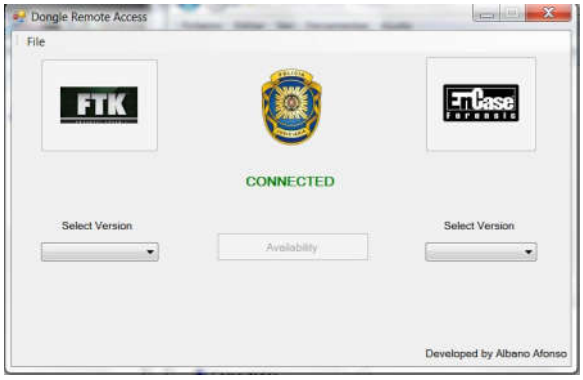
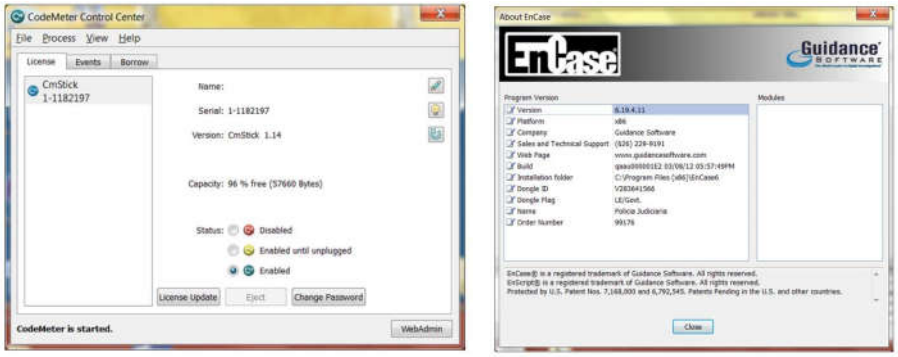


Fig. 5 Graphical user interface (GUI) adapted for the USB dongles management related with computers forensics applications used in criminal Police department.

After the TCP/IP connection has been established between the USB/IP client and the remote USB dongles, the applications FTK© and EnCase© have successfully validated the user license and allowed the user access. Figure 6(a) depicts the report obtained by CodeMeter Control Center©, after enabling the user license for FTK©. In Figure 6(b) we present the details of user license obtained by EnCase©. The overall process was transparent to the police examiner, since he is able to validate its license stored in the dongle and gain access to the computers forensic application.



(a) (b)

Fig. 6 (a) CodeMeter Control Center© report after detected license stored in remote USB dongle; (b) General information about the EnCase© user license found in the remote USB *dongle*.

Table 1 summarizes the comparison between the three types of access to USB dongles: local access and distributed access with USB/IP protocol, which may be assured by commercial or open source applications.

	Local access	**Commercial applications**	**OpenSource applications**
Access Performance	USB resources sharing. Physical access per user	Multi-user environment with transparent access to different devices	
Productivity	Must wait for physical access to dongles.	Distributed access with an increase of productivity	
Risks	Theft, loss and unauthorized access.	Possible interception of USB/IP packets.	
License	Operating system license	Charged by the number of remote dongles and accesses.	GPL license for USB/IP application with free access
Client application	No specific client application required	Generic front-end USB/IP client	Customizable front-end USB/IP client

Table 1. Comparison of local, commercial and open source distributed versions.

An important advantage of using a distributed approach for remote license validation is that we may mitigate the risk of loss and theft, as the user did not carry the USB dongle with him and thus did not have to connect it in the PC running the application. There are also some scenarios where we may have a higher number of users when comparing with the number of available USB dongles. In these situations, a user has to wait for a free dongle, which will be handed to him by another user who does not need it anymore. In some companies, these constraints may cause flaws in

management and loss of productivity, which are mitigated by a distributed USB/IP implementation. The approach proposed in this paper, as it is open source, has a GPL license and thus with no associated cost. It is worth noting that in USB/IP distributed approaches, all the legal requirements to use the licenses are fully attained, as we only have one user using a license per user at each given moment.

5. Conclusions and future work

In this paper we have proposed a client/server distributed architecture based on USB/IP protocol for remote validation of copy-protected licenses stored in USB dongles. We have tested this distributed architecture in a real world scenario that uses two distinct computers forensics software applications for digital analysis of evidences by the Portuguese criminal Polices, namely FTK© and EnCase©. From the literature review, we did not find any open source USB/IP implementation to deal with remote validation of software licenses stores in USB dongles.

For that purpose we have deployed and developed a distributed USB/IP architecture for remote access to USB dongles available in an USB/IP server. The developments were made to an existing and outdated open source implementation of USB/IP protocol. In the server side we upgraded the existing USB/IP version for newly Linux 3.2 kernel version. Regarding the USB/IP client side, our development was two-fold. On one hand we have updated the existing version of the client for Linux kernel in version 3.2. On the other hand we have made available a USB/IP client version for Microsoft Windows 7 (64 bits), with a friendly and easy to use graphical user interface that allows the management of both the several USB dongles available in the server and also the virtual connections of those bounded in the PC.

The experiments were made in the computers forensics laboratory of Portuguese criminal Police, both with external devices (e.g. disks, pens, cameras) and with USB dongles which store copy-protected software licenses of computer forensics applications. The results were very promising, since we have succeeded on validating the user access to computer forensics applications by accessing to a remote USB dongle with the license. In an operational point of view, the proposed architecture have had a great impact in the examiners' daily routine, as they did not have to carry the USB dongles and may easily access to the licenses. In organizational terms, this distributed architecture means more flexibility and more physical security in the management of USB dongles, avoiding the risk of loss or theft. In particular, the architecture and developments presented in this paper are now being implemented in computers forensics laboratory of Portuguese criminal police.

The distributed architecture proposed in this paper is based on open source components, available with a GPL license. This feature brings also some benefits when comparing with existing commercial solutions, as it is free of charge and easily adjustable to different business environments.

Our future work will include tests in a wider network, in which client and server belong to different IP networks. We also aim to evaluate the impact of using USB/IP protocol to extend a cloud infrastructure. That is, the USB peripherals connected to an USB/IP server can be remotely accessed in a cloud environment and benefits for

being in such environment. We are also evaluating this distributed USB/IP infrastructure on Raspberry Pi computers, in order to identify the advantages and challenges on having such an implementation on this kind of computers.

Acknowledgements: This work is financed by the ERDF – European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project FCOMP-01-0124-FEDER-037281

References

[1] Albano Afonso; "USBport Through TCP/IP - USB ports replication"; MSc thesis in Informatics and Multimedia, by Polytechnic Institute of Leiria; November 2012.

[2] E. Casey, *Handbook of Digital Forensics and Investigation*.: Academic Press, 2010.

[3] AccessData FTK, https://ad-pdf.s3.amazonaws.com/FTK_DataSheet_web.pdf (Dec 2013)

[4] Guidance Encase, http://www.guidancesoftware.com/encase-forensic (Nov 2013)

[5] U., Salvaneschi, P., Balducci, F., Jacomuzzi, P., & Moroncelli, C. Piazzalunga, "Security strength measurement for dongle-protected software," *IEEE Security and Privacy*, vol. 5(6), pp. 32-40, November 2007.

[6] Takahiro Hirofuchi, Eiji Kawai, Kazutoshi Fujikawa, and Hideki Sunahara, "USB/IP a Peripheral Bus Extension for Device Sharing over IP Network," in *FREENIX Track: 2005 USENIX Annual Technical Conference*, Anaheim, California, USA, 2005, pp. 47-60.

[7] Han Wook Cho, and Yong Ho Song Wonhong Kwon, "Design and Implementation of Peripheral Sharing Mechanism on Pervasive Computing with Heterogeneous Environment," in *SEUS'07 Proceedings of the 5th IFIP WG 10.2 International Conference on Software technologies for embedded and ubiquitous systems*, Greece, 2007, pp. 537-546.

[8] USB/IP Project, http://usbip.sourceforge.net/ (accessed November 2013)

[9] Kyuchang Kang, Dongoh Kang, Kiryong Ha, and Jeunwoo Lee, "Android phone as wireless USB storage device through USB/IP connection ," in *Consumer Electronics (ICCE), 2011 IEEE International Conference*, Las Vegas, NV , 2011, pp. 289-290.

[10] C. Y., & Lee, T. L. Ing, "USB Device Sharing Server for Office Environment," in *Asia-Pacific Services Computing Conference, 2008. APSCC'08. IEEE*, Yilan, 2008, pp. 953-956.

[11] Incentives Pro, http://www.incentivespro.com/usb-redirector.html (Nov'13)

[12] FabuaTech, http://www.usb-over-network.com/usb-over-network.html (Nov'13)

[13] Jan Axelson, *USB Complete: The Developer's Guide 4th Edition*. United States of America: Lakeview Research LLC, 2009.

[14] Andrew Troelsen, *COM and.NET Interoperability*.: Apress Media, LLC, 2002.

[15] Wibu-Systems, http://www.wibu.com/hardware-copy-protection.html (Nov 2013)

[16] SafeNet, http://www.safenet-inc.com. (Nov 2013)