Discovering and Plotting Hidden Networks created with USB Devices

Francisco Ramírez Pablo González Carmen Torrano José María Alonso

CDO, Telefónica
Madrid, Spain
{pablo.gonzalez, carmen.torrano, chema}@11paths.com
franciscojose.ramirezvicente@telefonica.com

Abstract- **USB is still a dangerous vector of attack. Attacks such as Stuxtnet makes this evident. USB connections could be stablished between devices that are apparently isolated, constituting what is called a *Hidden Network*. These networks are created through the use of USB devices that allows communication between physically or logically isolated computers. Being aware of such links is essential for guarantying the security of the network, machines connected and data within them. This paper presents a tool able to detect these links automatically, both from outside the network and locally. These links are explicitly plotted in graphs, making them visible and allowing taking security measures against threats. Furthermore, this tool can be used for forensic purposes, in case of data exfiltration for example, since it represents graphically the trazability of a USB within a network, besides showing which USB devices were connected to a certain computer.**

***Index Terms*- Hidden networks, USB devices, forensics, network security, network connections.**

Tipo de contribución: *Investigación original (límite 8 páginas)*

I. INTRODUCTION

Several security incidents reveal that USB should be paid attention since it could be used for malicious intentions. Even for air-gapped networks, USB represents an attack vector. Then, having a computers network not connected through Ethernet cable or Wi-Fi to other networks is not enough and any type of external connection for computers may constitute a threat.

In network analysis, usually attention is paid to connections at link level, such as Ethernet, Wi-Fi connections, etc. However, USB connections are often ignored.

USB connections also represent a mean to spread attacks, infect with malware or steal important data. In fact, several famous attacks have been performed through USB connections. Next, a review of the most important incidents is presented.

Firstly, Stuxnet [1] is mentioned, given its importance and repercussion. This malware is designed to target supervisory control and data acquisition (SCADA) systems that monitor and control industrial processes. In fact, in 2010 it was able to infect a Nuclear Power Plant in Iran. The worm took control of a thousand of machines involved in nuclear material production and gave them instructions that leave them inoperable. Stuxnet could access the network via an infected USB device. Then it scanned the network and propagated though it, reprogramming the software that controls certain machines involved in the nuclear process. The consequence was that thousands of those machines were left out of service.

Other popular malware is Brutal Kangaroo. This is one of the tools developed by CIA and included in Vault 7. This malware has been specially designed to infect air-gapped computers, i.e., isolated computers that are not connected to Internet. The term air-gapped makes reference to the space between the system and Internet. The first step of the operation of this malware is infecting a computer connected to Internet inside the target network and installing the malware. From there, the infection to air-gapped computers is done by means of a USB device infected with some malware.

There are other threats related to USB, such the so called "USB killer" [2], that damages the computer it is connected to by accumulating part of the electric energy of the computer and lately sending that energy brusquely against it. Rubber Ducky is a keyboard included in a USB device that types in a computer as soon as it is connected. As a keyboard, it can type malicious code or launch programs located either in the victim device or in the USB itself.

All these facts denote that attention should be paid to USB connections and devices, since they represent a potential threat. These USB connections can create what is called a *Hidden Network*, i.e., networks created through the use of USB devices that allow communication between physically or logically isolated computers. This is the case of Brutal Kangaroo, where all computers infected are part of a hidden network where elements within it can communicate and exchange data. This takes major importance considering the value of data in a computer and also circulating between the nodes of a network, in both the corporative and personal spheres.

In order to help in the automation of the discovery of hidden networks, in this paper we present a tool that makes explicit those hidden links created by USB connections. Our tool can be used to collect this information from outside the network and also locally. Furthermore, since the connection with the computers in the network can be done by using different protocols, we have developed software that covers different scenarios. Our solution collects the information of the computers within the domain in a file and it even plots the hidden USB links in a graph shape. It makes possible to be

aware of such links, what is very important for protecting and securing the network and devices connected to it. The discovery of such links can be useful for example in cases of data exfiltration, to help in the forensic analysis by revealing which USB devices where connected to a computer and tracing the USB within the network. This capacity is not exclusive for USB memories but also for devices connected through USB, such an external disk.

By mapping a network, a greater level of understanding regarding the network and the threats inside it can be achieved. This is important from the security point of view in order to adequately protect and secure all elements of the architecture, having more insights of the borders inside the network to mitigate intrusions, detecting attacks, or preventively carry out the implementation of safety measures.

The rest of this paper is structured as follows: Section II gives insights about hidden links, showing an example that presents their potential risks. Section III describes the registry and the particular branch where information about USB devices connected to a computer can be collected. Section IV explains the structure of our solution, that works in different modes: from outside the network and locally.
Section V shows the implementation details of our software for discovering hidden networks, concretely of our tool and Powershell scripts. Section VI presents details about graph plotting. Section VII covers the experimental stage where we have tested our software in different environments. Mitigation measures against hidden networks are explained in section VIII and finally, the conclusions drawn as result of this study are summarized in section IX.

II. NETWORK ISOLATION AND USB CONNECTION

In order to bring clarity in the understanding of hazards in Hidden Networks created from USB devices, a simple example is presented hereafter. In this example it is assumed that an organization have a network formed by three VLANs. The first VLAN contains a computer called *A* and another one called *B*. The second VLAN contains computers *C*, *D* and *E*. And the last VLAN contains the computer *F*. Every computer in a VLAN has connectivity with other computers in the same VLAN. However, computers in different VLANs cannot communicate. The network outline of the network architecture defined in this example is represented in Fig. 1. In that figure it can be seen that computers are isolated through different VLANs.

Assuming employees of this organization exchange data by means of USB devices, there is a high probability that this information passes from one VLAN's computer to another. This scenario is represented in Fig. 2, where users of computers *F* and *E* are assumed to exchange information through a USB device and a hidden network is created between both computers. This information can be represented with two nodes, *E* and *F*, and an edge projected between the computer first introduced the USB device and the second computer.

This would represent a serious threat to security since VLANs purpose is creating independent logic networks within a network, providing fragmentation and isolation in a physical network. This link would allow a communication between those computers that did not exist before. This implies that a new communication channel is created through the USB connection between those computers, that are supposed not be able to communicate.

Therefore, adding the USB device is itself a source of threats, since a hidden network can be created inside the organization.

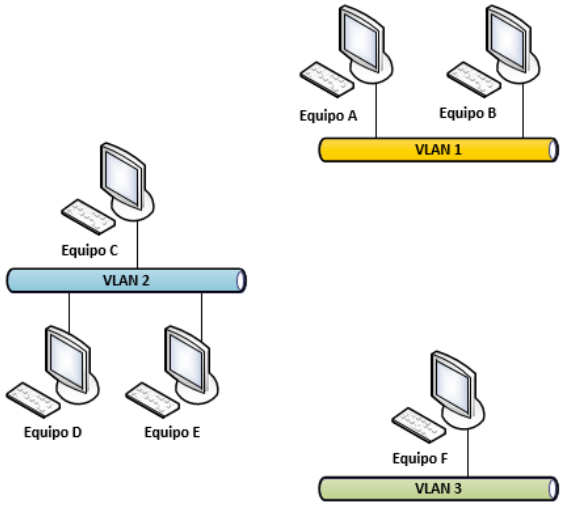

Fig. 1: Network outline with computers connected to different VLANs.

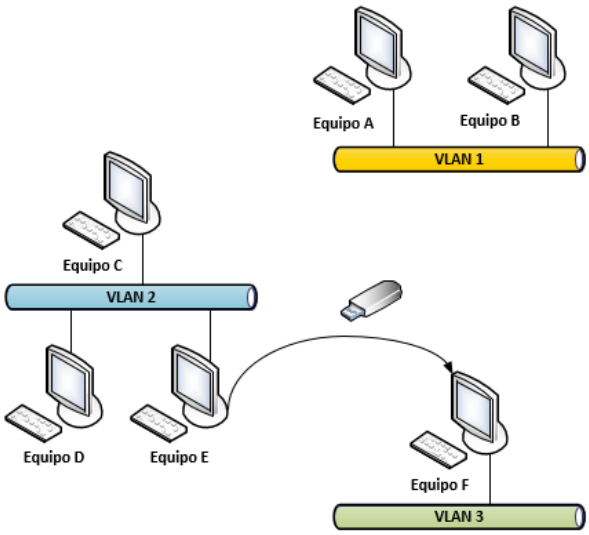

Fig. 2: Hidden link created in the previous network outline.

III. REGISTRY OF USB DEVICE CONNECTIONS

When a user connects a USB device in a Windows system, a series of entries are created in the Windows registry.
The Microsoft Computer Dictionary [3] defines the registry as: "A central hierarchical database used in Microsoft Windows 98, Windows CE, Windows NT, and Windows 2000 used to store information that is necessary to configure the system for one or more users, applications and hardware devices. The Registry contains information that Windows continually references during operation, such as profiles for each user, the applications installed on the computer and the types of documents that each can create, property sheet settings for folders and application icons, what hardware exists on the system, and the ports that are being used."

The registry controls the peripheral devices, thus, when a USB device is connected, certain information is stored there.

In particular, the USBStor key created in the Windows system registry saves information regarding all different devices inserted in the computer. Each USB connected to the computer generates a new item in the registry. Specifically, the branch of the registry where this information is stored is HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USBSTOR [4].

The following information can be collected from this branch regarding USB devices connected to one computer:

- Device name
- Serial ID (unique identifier)
- Class.
- ClassGUID.
- HardwareID.
- Service provided by the device, e.g. a hard disk.

An example of these fields of an USB device connected and stored in the USBStor key is shown in Fig. 3.

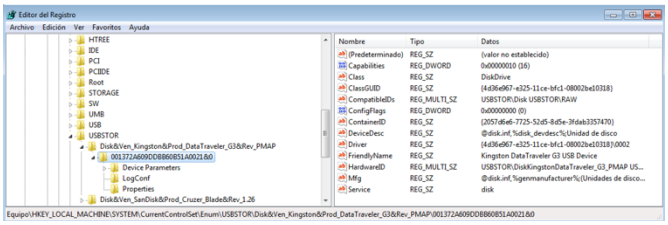

Figure 3: Registry visualization of the USB devices inserted.

From this information stored in the registry it is possible to know who is sharing a USB device and with whom, what is very useful to discover hidden links of a hidden network. As mentioned, this provides useful information in forensic investigation.

IV. SOLUTION DESCRIPTION

In this paper, we present a tool that is able to automatically discover hidden networks, what could be applied in forensic analysis. The tool can be used in two different scenarios: outside the network or locally.

- Outside the network. In this case the software collects the information of the registry regarding USB connections of computers in a given domain. It is possible to specify the set of computers of which the information will be collected. This option is designed to be used by system administrators and it is also useful in security audits.
- Local. The software collects the information of the registry of the local computer where it is run.

In the local case, it is not necessary to connect to any computer and only the information from the registry of the computer running the tool is collected. Contrarily, in the first case, the first step of the hidden network discovery process is connecting to the computers of the network that are selected to be analysed. The idea is that a central node with Active Directory runs the software and collects the information in the registry of all computers selected of the domain. The software then receives and stores information collected from the nodes.

There are different methods and protocols that can be used for stablishing the connection with nodes, depending on the architecture and technologies of the network. For this purpose, we have selected different technologies:

- WS-Management Protocol [5] and WMI [6]. Using the Windows Remote Management (Win RM) implementation [7]. We include WMI also in this section.
- SMB-PSExec [8-9].

Next, we give a brief description of these technologies.

A. WS-Management Protocol and WMI

This protocol is an open standard of the Distributed Management Task Force (DMTF) for accessing and exchanging management information with computers that implement this protocol. It is based on the Simple Object Access Protocol (SOAP) [10]. Windows Remote Management (Win RM) is the Windows implementation of this protocol. It allows to remotely run management scripts. One of the advantages of this protocol is that hardware and operating systems from different vendors can interoperate.

Related to Win RM Windows also provides an infrastructure for management of data and operations called Windows Management Instrumentation (WMI). It provides scripting languages to manage computers both locally and remotely. Additionally, it can supply management data to other components, such as Win RM.

B. SMB-PSExec

SMB stands for Sever Message Block (SMB). This is an application layer protocol with request-response nature used for sharing files, printers, serial ports and communication abstractions between nodes connected to the same network.

PsExec was born as an alternative to telnet, easier to configure and avoiding the need to install software on the remote computers to be accessed. It allows to execute processes on other systems and launching interactive command-prompts or enabling tools on remote systems.

SMB-PsExec implements remote process execution, being possible to run programs on remote computers. It can send a program to a remote computer, run it and read the result. It uses a configuration file for setting certain parameters.

Our solution covers all these scenarios. Since networks may use different technologies, we have implemented this variety of protocols so that our solution can be used in scenarios with different network architectures or configurations.

In particular, we provide a tool programmed in Python for covering the WMI and local cases. Additionally, we have programmed several scripts in Powershell for the SMB-PsExec and Win RM cases.

When it is necessary to connect remotely (outside the network flavour), all our solutions provide a file in CSV format with the information collected from the selected computers regarding the USB devices that have been connected to those machines. Additionally, the tool also provides JSON format. Furthermore, our tool is able to plot this information in a graph shape. Computers are plotted as and edges represent the hidden links between connected machines. In this way, the subtle connections are made explicit, what is very helpful for improving the security of organizations.

In the next section, we give more details about the

implementation.

V. IMPLEMENTATION

As mentioned, since the connection mode and protocols to connect with the computers in the network may vary, we have implemented different software to cover all cases. Next, we explain the details about how our software works. On the one hand the WMI and local cases are covered by our tool developed in Python. On the other hand, we explain the architecture of the Win RM and SMB-PsExec cases, implemented with scripts that use Powershell 3.0 commands. Powershell is an object-oriented command line from Microsoft, which has a simple and powerful interaction with any structure inside a Microsoft operative system.

A. WMI and local

We have developed a tool programmed in Python for covering these two cases. The architecture of this implementation can be seen in Fig. 4.

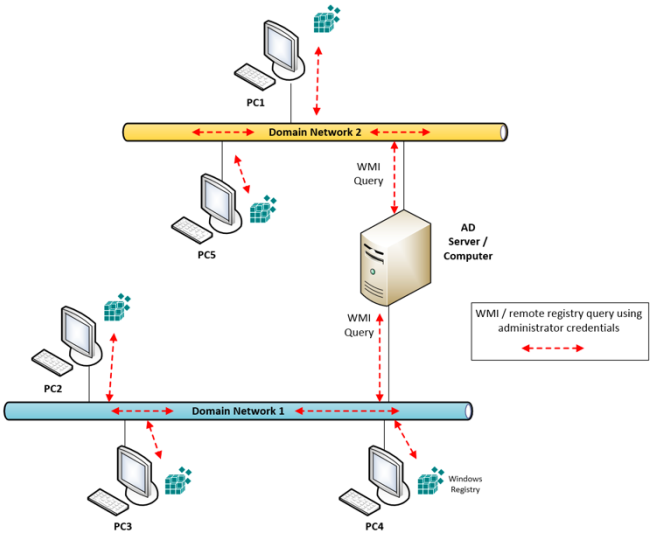

Fig. 4: Tool execution diagram for WMI and local cases.

The tool has a friendly interface that allows the user to interact with it a simple and easy way. The user interface can be seen in Fig.5. This interface makes possible to specify the path of the project and a file .hn is created, that includes the project name and paths of the resulting file in a CSV or JSON format.

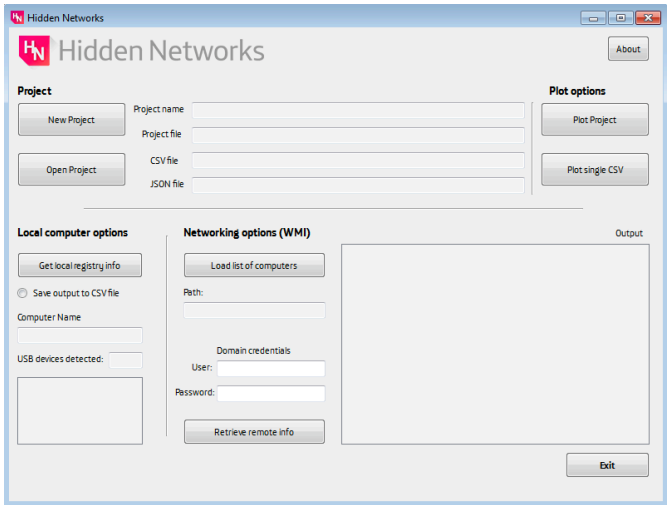

Fig. 5: User interface of the tool for the WMI and local cases

An example of CSV file can be seen in Fig. 6. About the values of the CSV file, fields “usbdevice_name” and “usbdevice_id” are collected from the registry as explained in Sec. III. Fields “computer_name” and “computer_ip”, that correspond to the computer name and its IP address, are obtained using the “socket” library in Python.

When a CSV file is opened, the tool automatically converts it to JSON format.

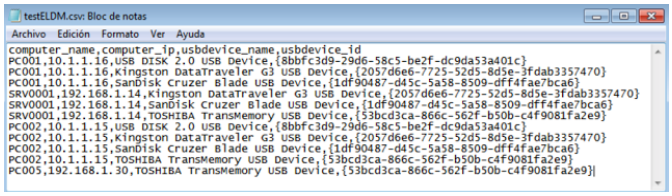

Fig. 6: Example of CSV file.

In the WMI mode, the tool connects with those remote computers selected, that are specified in a plain text file. An example of such file is shown in Fig. 7, where the IP address or FQDN of the selected devices are written down.

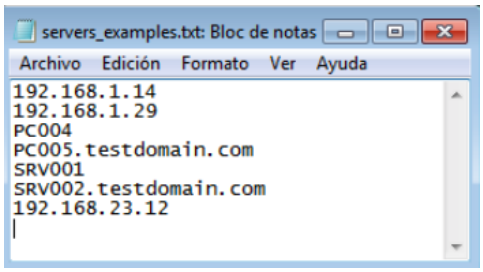

Fig. 7: Example of file with selected computers to be analysed.

After introducing credentials for the domain administrator, the information from key in the registry is read and stored. Differently, in the local mode the key is not downloaded but it is directly read from the registry, what supplies more depth in the exploration of the registry.

Results are shown in the white rectangle on the bottom right of Fig. 4. As can be seen in the graphical interface, the local analysis is located on the left and the network case appears on the right side.

If the user has information about USB connections, instead of creating a new project and collecting the information, the tool can also be used to plot such information.

For representing hidden networks, the tool draws a graph per USB device. The user can choose to either visualize all graphs or choose a specific USB.

Furthermore, our hidden networks tool offers the possibility to plot directed graphs. It indicates the order, from the oldest date to the newest one, when the USB was connected. The date the USB was inserted can be found in the file C:\Windows\inf\setupapi.dev.log [4]. An example is depicted in Fig. 8, where broader line represents an arrow.

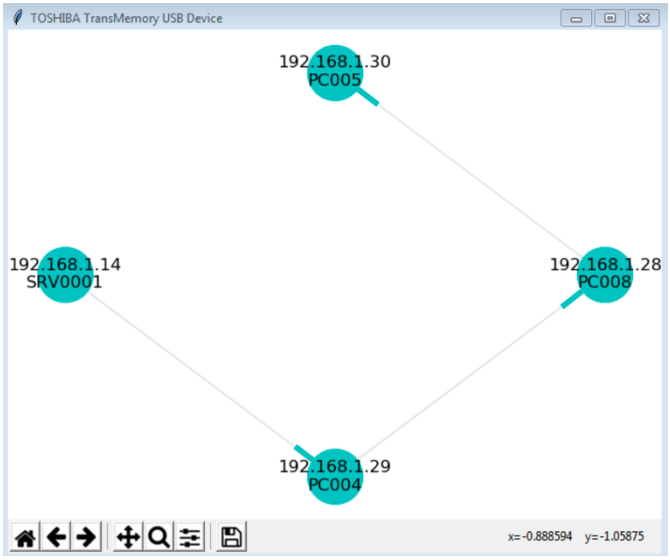

Fig. 8: Example of directed graph.

From the security point of view, it is important to mention that although our tool access the cited branch of the registry, it does not access to other files or data stored in the computer.

For interested readers, the code of our tool can be downloaded here:
https://github.com/ElevenPaths/HiddenNetworks-Python

B. *WS-Management Protocol*

This solution and the SMB-PSExec one are based on the following process:

- Connection to computers. Sending the script that collects information from the registry regarding USB connections of the computers connected.
- Receiving and storing information from the nodes.

Every computer selected in the network executes the scripts received and returns the output data to the central node, that collects the information reported by the different nodes of the network. Fig. 9. represents the architecture of the script execution in an Active Directory.

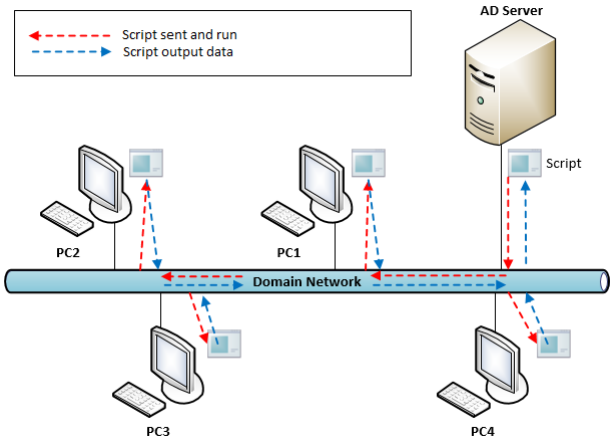

Fig. 9: Tool execution diagram for WS-Management Protocol and SMB-PsExec, using Powershell scripts.

The script WinRM version requires the activation of the Windows Remote Management (WinRM) service in each of the network computers to be audited.

Domain administrator credentials are required when executing the script to approve execution on remote computers in the local network.

The script implementation is composed of two steps:

- A *Launch* program that connects to remote computers.
- A *Recollect* program that collect information from USB devices. It is passed as parameter of the previous program to be executed by every computer selected to do it.

The execution of the "Launch" program is based on the PowerShell command "Invoke-Command". It allows to connect with a computer in the network passing the FQDN, computer name or IP address as parameters, besides executing the PoweShell script.

The outcome of the program is CSV file called "USBDATA.csv" containing the following fields: Name of the computer, IP (on IPv4 format), USB name, ID (unique identifier). An example of this file is shown in Fig. 10.

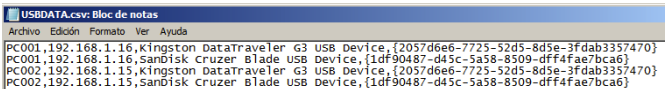

Fig.10: Example of CSV file with results of USB connections in selected remote computers.

This script *Recollect* is responsible for gathering all information referring the USB devices connected to the computer and it runs locally in the computers to be audited.

C. *SMB-PsExec*

In order to run the script through SMB, it is necessary to have PSTools previously installed, specifically to execute the PSExec command in the computers to be checked.

The operating philosophy will be practically the same of the WinRM version. It will be connected from the server to the remote computer and the script should be run from the server with domain administrator account, then the USB data collection script will be executed. The computers selected to

run the scripts are specified in a file called *servers.txt*. This could be done by specifying the FQDN name or IP address of the computer.

Since the connection protocol is different, the “Launch” script has a few modifications to fit with this new type of connection. Instead of using the "Invoke-Command” command, a shell from Powershell is opened and the script runs from it. The script is downloaded from the network location, preferably from a web server that would execute download through some HTTP protocol. In this way, subsequent problems with execution policy and permits, that might be found when accessing the local shared resource, are avoided.

Similar to the previous version of WinRM, results are stored in a CSV file. To avoid synchronization problems and allow time enough for the program to run on the remote computer, some delays have been included in the code. It should also be taken into consideration that the duration of the delay may vary depending on the environment where the script is run.

It is important to properly configure the location paths for each of the files before running the script, specifically, the path of the *Recollect* script, path of the *servers.txt* file and the path of the CSV file recollecting the information.

The script *Recollect* is almost the same than in the previous case. The generated *USBData.CSV* file will be exactly the same as the one previously shown.

For the implementation we have focused on Windows systems so far. It is also possible to collect information regarding USB connections in other operating systems. For example, computer systems running Mac OS X or macOS have a file with a PLIST extension, which store this information over the USB devices connected to the computer. The file is named *com.apple.finder.plist*.

VI. GRAPH REPRESENTATION

The information contained in the csv and json files are plotted by the tool in a graph representation. Our tool offers the possibility to visualize the graphs independently for each USB connected, that is, for the sake of clarity the representation of the hidden links associated to each USB device are represented in separate windows.

Figure 8 shows an example of representation of the graph corresponding to the USB called “Kingston Data Traveler G3”. In the figure, computers are represented as nodes and edges are represented as links between computers when the same USB have been connected to both of them, thus, edges represent the hidden links. In Fig. 11 it can be seen that the Kingston USB device was connected to four different computers (IP: 192.168.1.14, 192.168.1.16, 192.168.1.28, 192.168.1.29) that are located on the same network.

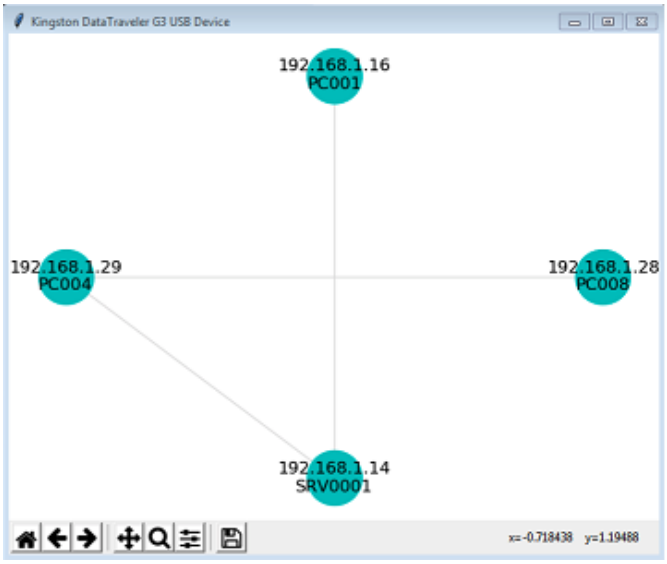

Fig. 11: Example of graph corresponding to USB “Kingston Data Traveler G3”.

Graphs have been plotted using python, concretely, the NetworkX library [11] has been employed.

VII. EXPERIMENTATION

In order to test our tool, its functionalities have been tested in different environments.

In a first approach, the auditing implementations have been tested on a single-Domain network with an Active Directory (AD) to automate the information collection as much as possible. The scenario has six virtual machines, in particular five computers and a server connected to the same network. Regarding operating systems, the computers were running Windows 7 and Windows 2008 server. In this scenario, the Powershell scripts were run. As result, the server was able to connect with the computers and collect from the registry the USB information of the computers selected for that. The result obtained is shown in Fig. 12.

```
PS C:\scripts\HiddenNetworks> .\HiddenNetworks.ps1
Computer:  PC003
No USB data found
Computer:  PC004
USB found:  SanDisk Cruzer Blade USB Device
USB ID:  {1df90487-d45c-5a58-8509-dff4fae7bca6}
Computer:  PC005
USB found:  Kingston DataTraveler G3 USB Device
USB ID:  {2057d6e6-7725-52d5-8d5e-3fdab335747d}
Computer:  PC002
USB found:  Kingston DataTraveler G3 USB Device
USB ID:  {2057d6e6-7725-52d5-8d5e-3fdab335747d}
Computer:  PC002
USB found:  SanDisk Cruzer Blade USB Device
USB ID:  {1df90487-d45c-5a58-8509-dff4fae7bca6}
```

Fig. 12: USB devices connected to computers in the network.

In this case, we used the tool Gephi [12] to plot the corresponding graph. The result can be seen in Fig. 13. Each link between two nodes (representing computers in the network) means a hidden link. The label in the links refers to the USB identifier.

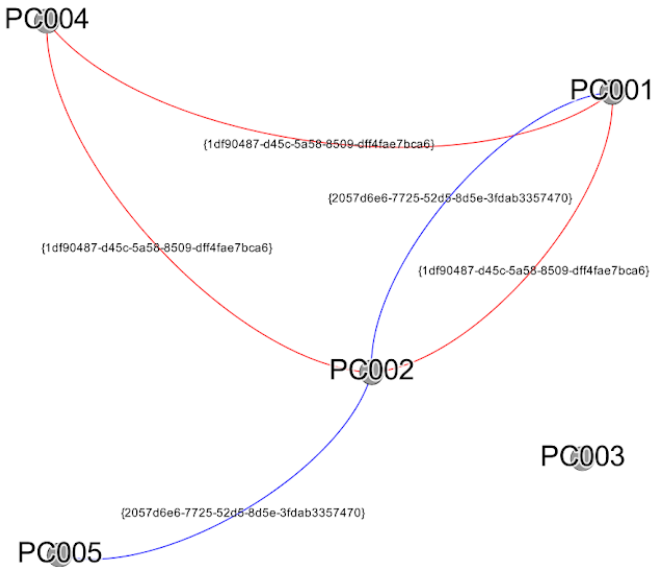

Fig. 13: Hidden networks discovered with Powershell scripts.

One remarkable aspect is the existence of connections between machines in different VLANs, that were connected via USB. This means the creation of a hidden network that connects machines that were otherwise isolated. This tool makes visible these links and helps in auditing networks and protecting them. This functionality is also very helpful in the incident response field.

Moreover, we also run other experiments with our tool and the WMI protocol. The first experiment was performed with four machines in the same network: three computers with Windows 7 and one with Windows 2008 server. The result plotted by the tool can be seen in Fig. 14.

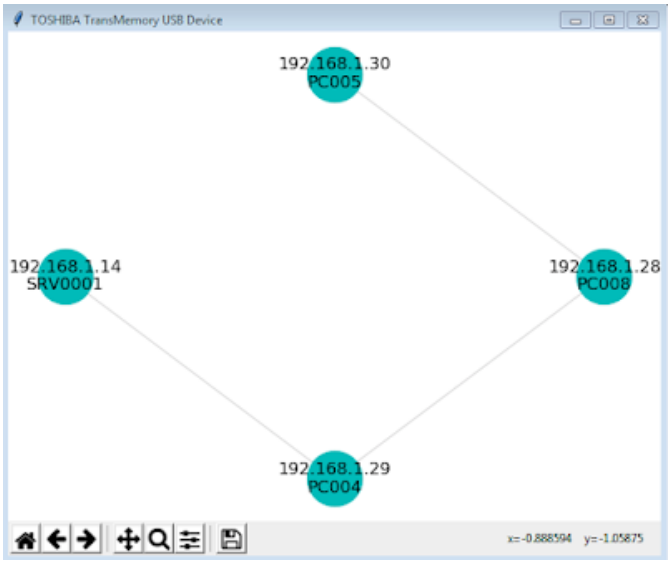

Fig. 14: Hidden networks discovered with our tool for computers in the same network. Graph corresponding to the USB "Toshiba TransMemory"

Another experiment was carried out with our tool. In this case we used four machines: two computers with Windows 7, one computer with Windows 10 and a server with Windows 2008 server. The difference with the previous experiment is that, in this case computers belonged to different subnetworks. The application is run from a machine that has visibility to both subnets. The result plotted by the tool for different USB devices can be seen in Fig. 15.

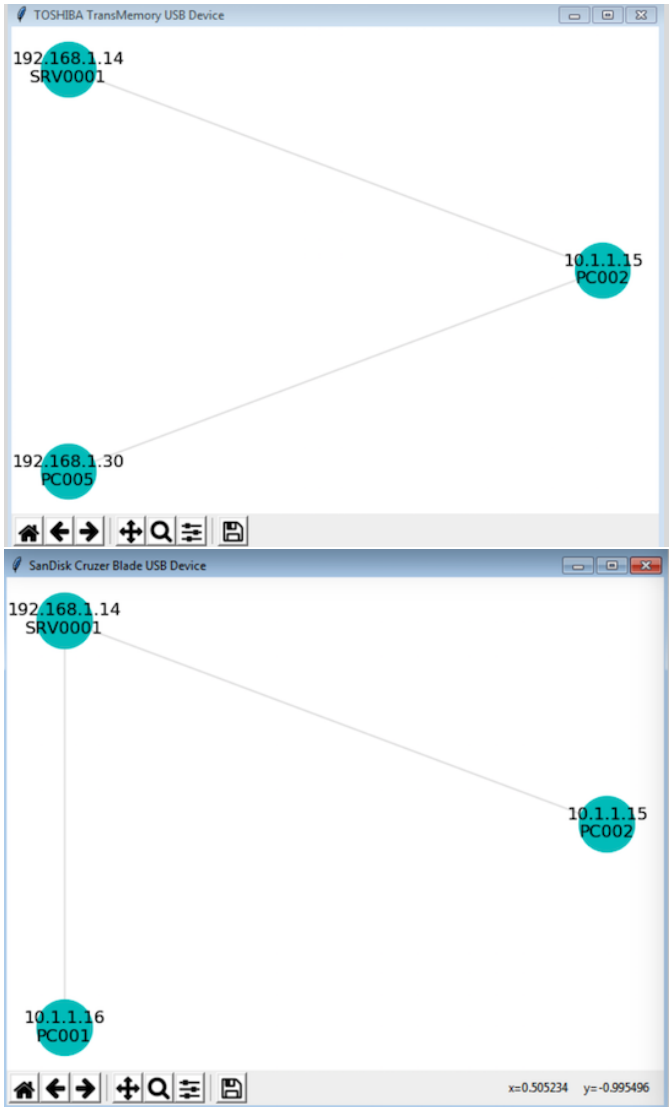

Fig. 15: Hidden networks discovered with our tool for computers in different subnetworks. Graph corresponding to the USB "Toshiba TransMemory", "Kingston Data Traveler", "SanDisk Cruzer Blade"

As can be seen in the figure, PC001 and PC002 belong to a subnetwork and SRV001 and PC005 belong to a different one. Even though, the graphic shows that it is possible to communicate computers in different subnetworks via USB. Without our tool these connections could go unnoticed, since those computers were apparently not connected.

VIII. MITIGATION

One way to prevent hidden networks is to restrict the use of USB devices in computers. Mitigation or prevention can be done through the forced use due to Active Directory policies, which restrict connection in a user computer to devices only approved by one user. The implementation of the safety policy along with a white list of approved devices for each user allows to avoid this kind of hidden links, but it is complex and expensive to maintain.

IX. CONCLUSIONS

Having networks disconnected from the Internet provides this false security feeling regarding a higher level of protection before any incident, which increases the system vulnerability. The USB is an attack vector that frequently goes unnoticed, however, the malware infection through USB devices is a real and underlying problem. USB are also potentially dangerous for cases of data exfiltration and leakage. Furthermore, USB can connect computers that are apparently isolated physically or logically. Since being aware of such connections is essential for guaranteeing the security of our computers and networks, in this paper we present a solution able to automatically discover hidden links and visualize them. It can be done both from outside the network or locally. Furthermore, we have included several mechanisms for connecting with nodes and extracting information about USB connected to them. On the one hand, we have implemented Powershell scripts that connect through WinRM and SMB-PsExec. On the other hand, we have implemented a tool in Python able to connect using the WMI protocol and that also covers the local case. We have conducted several experiments, with computers with diverse operating systems, in different VLANs and different subnets and the result of all of them probe that it is possible to connect through USB computers that are apparently isolated. This is a main issue in network and computer security, therefore, we would like to raise awareness about that. Moreover, our tools are not only able to discover this hidden links but can be very helpful in cases of incident response, for example, if a data exfiltration case takes place, it could help in the forensic tasks of tracing a certain USB within a network or informing about the USB connections that were done in a particular computer. We would like to stress that the results of this paper are not restricted to USB memories, but it can also be applied to other devices connected through USB, such as external hard disks, Wi-Fi or Bluetooth dongle, etc. The aim of this solution is to reinforce computer and network security by helping in the prevention of incidents, facilitating audits and providing utilities useful for forensic analysis cases.

ACKOWLEDGEMENT

This is a work developed by the CDO unit of Telefonica.

REFERENCES

[1] S. Karnouskos, "Stuxnet worm impact on industrial cyber-physical system security," in *IECON 2011 - 37th Annual Conference of the IEEE Industrial Electronics Society*, Melbourne, VIC, pp. 4490-4494, 2011.

[2] Microsoft Computer Dictionary. Windows Registry. https://support.microsoft.com/en-us/help/256986/windows-registry-information-for-advanced-users. Last Updated: Jan 7, 2017.

[3] Nir Nissim, Ran Yahalom, Yuval Elovici, "USB-based attacks", in *Computers & Security*, vol. 70, pp. 675-688, 2017.

[4] Abhijeet Ramani, Somesh Kumar Dewangan: "Auditing Windows 7 Registry Keys to track the traces left out in copying files from system to external USB Device" in *International Journal of Computer Science and Information Technologies*, vol. 5 ,2, pp.1045-1052, 2014.

[5] Microsoft, "WS Management Protocol", 2018. https://msdn.microsoft.com/en-us/library/aa384470(v=vs.85).aspx

[6] Microsoft, "Windows Management Instrumentation", 2018. https://msdn.microsoft.com/en-us/library/aa394582(v=vs.85).aspx

[7] Microsoft, "Windows Remote Management", 2018. https://msdn.microsoft.com/en-us/library/aa384426(v=vs.85).aspx

[8] Microsoft, "PsExec v2.2", 2016. https://docs.microsoft.com/en-us/sysinternals/downloads/psexec

[9] Richard Sharpe, "Just What is SMB?", 2002. https://docentes.uaa.mx/guido/wp-ontent/uploads/sites/2/2015/04/What-is-SMB_.pdf

[10] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, Dave Winer, "Simple Object Access Protocol (SOAP) 1.1", 2000. https://www.researchgate.net/profile/Satish_Thatte/publication/239553871_Simple_object_access_protocol_SOAP_11/links/54489e4e0cf2f14fb8142a59/Simple-object-access-protocol-SOAP-11.pdf

[11] A Hagberg, P Swart, DS Chult, "Exploring network structure, dynamics, and function using NetworkX", in *SCIPY*, 2008. http://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-08-05495

[12] Bastian M., Heymann S., Jacomy M. "Gephi: an open source software for exploring and manipulating network" in *International AAAI Conference on Weblogs and Social Media*, 2009.