



Article

Distributed Architecture to Enhance Systems Protection against Unauthorized Activity via USB Devices

José Oliveira ^{1,*}, Pedro Pinto ^{1,2,*}  and Henrique Santos ^{3,†} 

¹ Instituto Politécnico de Viana do Castelo, 4900-347 Viana do Castelo, Portugal

² Instituto Universitário da Maia, 4475-690 Maia, and INESC TEC, 4200-465 Porto, Portugal

³ Department of Information Systems, Universidade do Minho, 4800-058 Guimarães, Portugal; hsantos@dsi.uminho.pt

* Correspondence: j.o.oliveira@estg.ipvc.pt (J.O.); pedropinto@estg.ipvc.pt (P.P.)

† These authors contributed equally to this work.

Abstract: Cyberattacks exploiting Universal Serial Bus (USB) interfaces may have a high impact on individual and corporate systems. The BadUSB is an attack where a USB device's firmware is spoofed and, once mounted, allows attackers to execute a set of malicious actions in a target system. The countermeasures against this type of attack can be grouped into two strategies: physical blocking of USB ports and software blocking. This paper proposes a distributed architecture that uses software blocking to enhance system protection against BadUSB attacks. This architecture is composed of multiple agents and external databases, and it is designed for personal or corporate computers using Microsoft Windows Operating System. When a USB device is connected, the agent inspects the device, provides filtered information about its functionality and presents a threat assessment to the user, based on all previous user choices stored in external databases. By providing valuable information to the user, and also threat assessments from multiple users, the proposed distributed architecture improves system protection.

Keywords: USB; threat assessment; BadUSB attack; HID; distributed architecture



Citation: Oliveira, J.; Pinto, P.; Santos, H. Distributed Architecture to Enhance Systems Protection against Unauthorized Activity via USB Devices. *J. Sens. Actuator Netw.* **2021**, *10*, 19. <https://doi.org/10.3390/jsan10010019>

Received: 31 December 2020

Accepted: 23 February 2021

Published: 2 March 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Public and private institutions are constantly dealing with new cyberthreats and cyberattacks [1] intended to disrupt infrastructure sectors such as water, power, transportation, communication and health-care systems [2–4]. At the same time, according to [5], the use of Universal Serial Bus (USB) drives to transfer data between systems is increasing.

The USB devices offer a large attack surface on computer systems, since they are affordable, accessible and require low interaction to install and use [6]. The USB-related attacks exist in multiple forms, such as USB Mass Storage devices that contain malware, smart drives that include malicious auto-run payloads or programmable Human Interface Device (HID), where malicious code is embedded in the device's firmware and asks to install a hidden USB human interface, such as keyboard, mouse or other interface devices [7]. An example of the latter attack is known as BadUSB [8,9], which exploits a vulnerability in USB firmware by reprogramming the USB device to act as a defined HID and discreetly execute commands or run malicious programs on a target.

The BadUSB attacks can be prevented by adopting specific strategies, such as disabling USB ports on computers or disabling the Plug and Play (PnP) devices' installation (i.e., disable all peripheral devices). However, these fixes permanently disable the USB capability and prevent new devices from being mounted. Additional efforts have been made to allow USB devices to be mounted only after a user verification. Previous work in [10] proposed a system protection agent that blocks USB devices' installation and enables users to check and mount each device after being informed about its category. However, this proposal had the following limitations: (1) the filtering process is based on a short and less accurate

vendor-defined Identifier (ID) and (2) the devices are presented to the user without any prior or distributed threat assessment.

This document proposes a distributed architecture to enhance system protection against BadUSB attacks. The architecture is composed of agents running in Microsoft Windows Operating System (OS) and a set external Database (DB). The architecture allows the intersecting of the installation of the device driver and presents the user with a local and remote threat assessment based on the identified functionalities of the HID. The threat assessment is carried out on the basis of information collected anonymously by other users.

This paper is organized as follows. Section 2 presents context and the related work. Section 3 details the proposed distributed architecture. Section 4 describes the implementation of a functional prototype and provides results. Section 5 draws conclusions.

2. Related Work

The USB is a communication standard [11], commonly used for connecting peripherals to computers. Introduced in the 1990s, the USB interface and protocol increased in popularity because it facilitated communication between peripherals and hosts [12]. However, according to [13], one of the major weaknesses of the USB protocol is its flexibility, and potential threats increase if the device is connected to other devices, networks or the Internet [14]. An infected computer might uncover confidential information or personal data, participate in distributed attacks against other computer systems [15], or it might sabotage computer-controlled industrial processes. As explained in [16,17], a regular user may plug a USB flash drive found on the ground into a corporate computer without knowing that it may be an infected device, ready to start an attack. USB drives may turn into a security threat when they are used to transport and spread malware such, as the worm Conficker, which impacted a set of computers in Manchester City Council [18,19], or Stuxnet, a computer worm conveyed in an infected USB thumb drive [20], intended to hit particular industry sectors [21]. An overview on USB-related attacks can be found in [22,23].

To classify malicious USB devices, the authors extended the categories defined in [24] and proposed the following updated list [10]:

1. USB Mass Storage devices containing malware: consists of external USB storage that is used to inject malicious code to a computer. Examples can be found in [20,25]. If these devices are shared between multiple computers, the malicious code can be widely disseminated. Microsoft Windows allowed autorun by default, allowing automatic file execution, a feature that was later abandoned [26];
2. U3 smart drives: These devices are similar to Mass Storage Devices but have a special partition that is recognized by Microsoft Windows as a CD-ROM. CD-ROMs can use the autorun feature to execute malicious code. If the U3 Thumb Drive runs a malicious autorun payload, then the intention was to harm the system by delivering malware [24];
3. USB devices in the Middle (USBiM): a USB device that installs activity loggers in the host computer, such as keyloggers. Some of these devices are available as forensic tools [27]; however, other devices with malicious purposes fit into this category, such as printer loggers, or hardware USB sniffers. In [28], a proof of concept is presented, where a device is capable of achieving the same results as hardware keyloggers, keyboard emulation and BadUSB hardware implants;
4. Denial of Service USB devices: USB devices intending to disrupt services. An example of Permanent Denial of Service (PDoS) hardware used in this type of attack is the USB killer [29] that uses the USB power lines to charge capacitors and, when fully charged, discharge high voltage (200 volts) over the data lines of the host device, permanently damaging any circuit board with no electrical surge protection. Another example would be [30] where attackers have compromised over 25,000 devices and used them to launch a Distributed Denial of Service (DDoS);

5. USB with programmable HID consists of malicious code that is embedded in device's firmware and requests a USB human interface. As pointed out in [31], this procedure provides unacknowledged and malicious functionality that lies outside the apparent purpose of the device.

The BadUSB attack can be included in the last category. As detailed in [8], this attack consists of a reprogrammed USB chip capable of emulating other devices. As explained in [16], unlike USB drives that cannot automatically execute code, HID such as keyboards do not require user confirmation, nor do they require USB autorun to be enabled in the target system. Thus, if a device is identified as a keyboard, it is automatically installed and may send a fast set of keystrokes intended to compromise the machine it is connected to. The authors in [24] presented a programmable USB device that is capable of performing the BadUSB attack by emulating a keyboard and typing out commands defined inside a script stored on the device. Another example is found in [32], where a USB device can be recognized on a computer as a keyboard and mouse. Current efforts have been made to tackle BadUSB attacks. Security software firms such as Symantec [33] proposed a set of recommendations, while they do not provide an effective solution to this type of attack. The authors of [34] proposed protecting data against BadUSB attacks in corporate environments by using the cloud as a storage method to share information through the internet between company members. The defenses against BadUSB attacks can be categorized into two types: physical blocking of USB ports and software blocking.

In physical blocking, the attacker is not able to connect any device to the target system. Strategies for the physical obstruction of the USB ports may range from the simple filling of ports with epoxy resin [35] to commercial solutions such as [36], which deposit a lockable plug into the port. In [37], a simple adapter to physically cover the USB port is presented.

In the software blocking, USB ports are blocked by software/applications. The authors of [5] propose an application to block the system when a USB drive is inserted. However, while protecting against BadUSB, blocking the OS operations prevents the regular usage of the system as well. In [31], GoodUSB is proposed as a system that blocks any USB device and waits for user authentication before proceeding with installation; however, no further information regarding threat assessment is provided to aid the user decision.

In the previous work [10], the authors propose a system based on an agent that blocks PnP device installation on *Registry* and listens to OS events, waiting for device change notifications. The solution is available in [38]. For every device that is plugged or removed, the agent captures the event and then makes a new list of devices that are already plugged. If a new device is added, the agent obtains its vendor-related IDs to identify and notify the user about the functionalities before installing. The main limitations of this proposal are that it uses a short vendor-defined ID, which makes filtering procedures less accurate and effective, and there is no remote threat assessment to help the user choose whether to install the device or not.

The current paper proposes a distributed architecture that can be included in the software blocking category, addresses the limitations of the previous proposals, and thus enhances the system's protection against BadUSB attacks. The proposed architecture is detailed in the next section.

3. Distributed Agent Architecture

The proposed architecture is presented in Figure 1 and is composed of computers, agents, an External DB, and, in case of a corporate environment, a Corporate DB. The computers have Microsoft Windows OS installed and the agents are applications installed in the computers. The External DB holds threat assessment information, which is available to all agents. In case of a corporate environment, the Corporate DB has a partial threat assessment information only related to corporate computers.

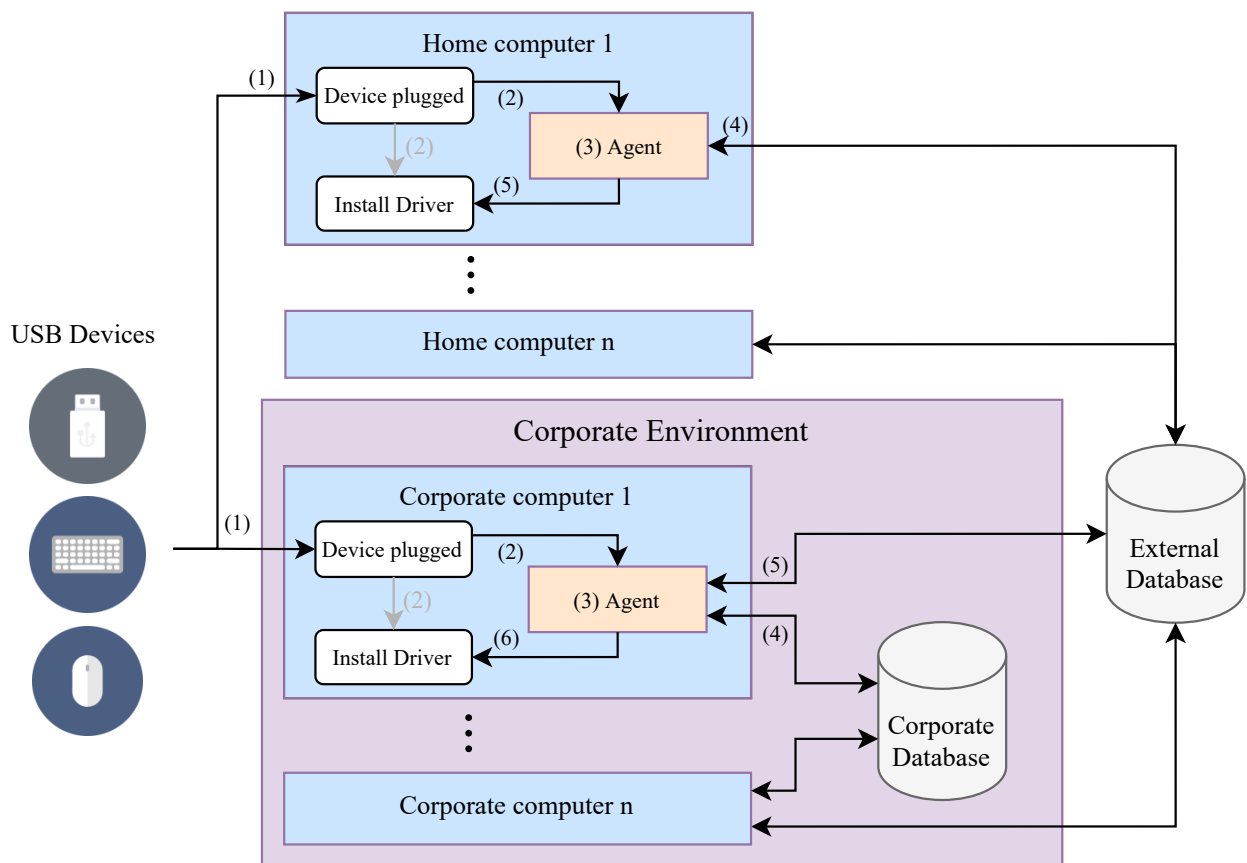


Figure 1. A basic scheme of the system distributed architecture.

In case a USB PnP device, such as a keyboard, mouse or other, is plugged into a home computer (1), OS detects the device, overrides the driver installations and forwards information (2) to the agent. The agent (3) identifies the device functionality using information on the OS driver’s files.

Then, the agent requests a threat assessment from External DB (4), presents it to the user, and requests action prior to installation (5). In a corporate environment, the procedure is similar to one for the home computer, but it may use the two Threat Assesments (TAs), one from Corporate DB (4) and the other from the External DB (5), prior to the installation on (6).

Figure 2 details the internal procedures inside each computer with an agent. First, when the agent starts, it blocks the PnP device installation on *Registry* (1), collects all the information of the devices mounted on the system, and inserts this in a list.

In step (2), the OS generates notifications for every action that occurs on USB ports. These notifications can be captured by overriding an event handler. If there is an event of device change (addition or removal), the agent collects a list of mounted devices on the system.

The mounted devices list is sent to the Agent (3), filtered and inserted in a Devices List. To allow other PnP devices by default, the agent automatically installs non-USB devices by retrieving the vendor-defined IDs (*Compatible ID* and *Hardware ID*) in a string format, and then matches this with USB and HID; if a match is found, the agent keeps the devices blocked.

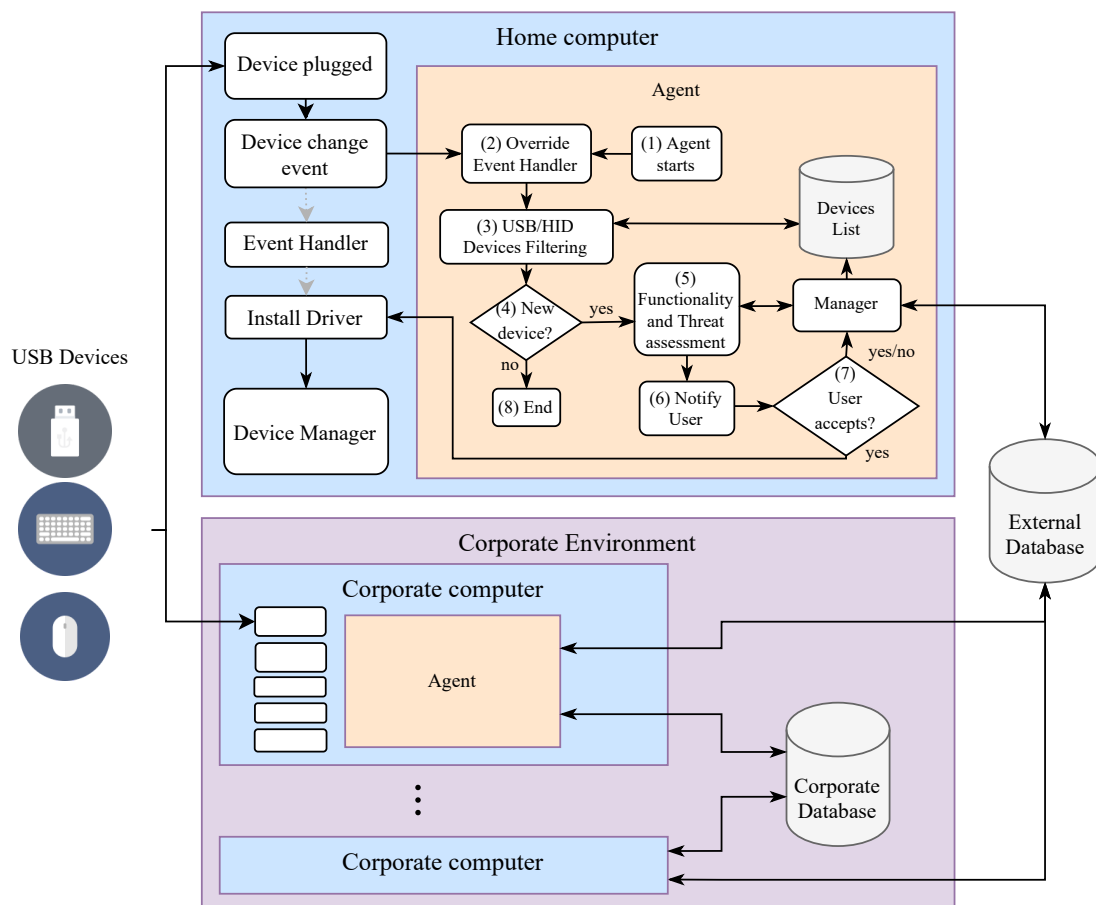


Figure 2. Detailed schema for the distributed protection agent.

In (4), the agent compares the new list of devices with the list obtained prior to the notification in step (2), to check if a new device was added. If a device is not found in the previous list from step (2), the agent requests functionality and threat assessment information (5).

The functionality information is obtained through the *INF* files, which contain information such as driver name and location, driver version information and *Registry* information [39]. Algorithm 1 presents the procedure used to collect the *INF* file path and Functionality Assessment. The agent searches for *INF* files on the system, which can be found on two separate folders in Microsoft Windows. The *INF* path is the path of the file containing the vendor-defined IDs of the device. The functionality is obtained by the string associated with the keywords class (class of the device), devicedesc (device description) and svcdesc (service description).

Algorithm 1: INF File Path/Functionality Assessment Algorithm (step 5)

```

Result: Returns compatible INF file paths and functionality for a given device
1 Procedure Find INF File
2   foreach (file in INF files) do
3     INFPath=null, deviceFunctionality=null;
4     foreach (line inside file) do
5       if (line contains vendor-defined ID) then INFPath = this file path;
6       if (line contains (class || devicedesc || svcdesc)) then deviceFunctionality = functionality;
7       if (INFPath != null && deviceFunctionality != null) then
         return (INFPath, deviceFunctionality)
     end
  end

```

Algorithm 2 presents the threat assessment procedure and user interaction. The manager performs a query regarding the External DB in order to collect a threat assessment for the device, and adds it to the list of devices. The result is obtained by searching the device using the vendor-defined IDs and retrieving the number of users that previously accepted the device, U_A , and the number of users that previously blocked the device, U_B . The threat assessment estimation is a ratio between U_A , and the sum of U_A with U_B . In step (6), the agent lists all blocked devices and their respective threat assessment. The user may choose to install the device or declare it as a threat. If the user accepts a device (7), the agent proceeds with the installation of the driver, which requires the *INF* file path and vendor-defined ID of the device. The user actions are sent to the External DB.

Algorithm 2: Threat Assessment Algorithm (step 5) and User Interaction (steps 6 and 7)

- 1 Query to External DB to obtain U_A, U_B using vendor-defined IDs;
- 2 Threat Assessment Estimation = $\frac{U_A}{U_A+U_B} (\times 100)$;
- 3 **if** User accepts the device **then**
- 4 | Install device and send to analysed devices tab;
- 5 | Manager sends an update query to the External DB with the $U_A + 1$;
- 6 **else** Manager sends an update query to the External DB with the $U_B + 1$;

4. Implementation and Results

The proposed distributed agent architecture was implemented as a functional prototype with the topology presented in Figure 3. The topology is composed of a corporate computer and a DB server. The corporate computer features an i7-6700HQ CPU, 16GB DDR4 2133MHz of RAM and 256GB M.2 SSD of storage capacity. The agent was developed using C# programming language and deployed in a Microsoft Windows 10 Pro Creators Update, version 10.0.15063. The DB server runs on a computer featuring an Intel Core 2 Duo, with 4GB DDR2 of RAM and a 125GB Kingston SSD. In the DB server, a WampServer [40] was configured with two MySQL DBs, a Corporate and an External DB.

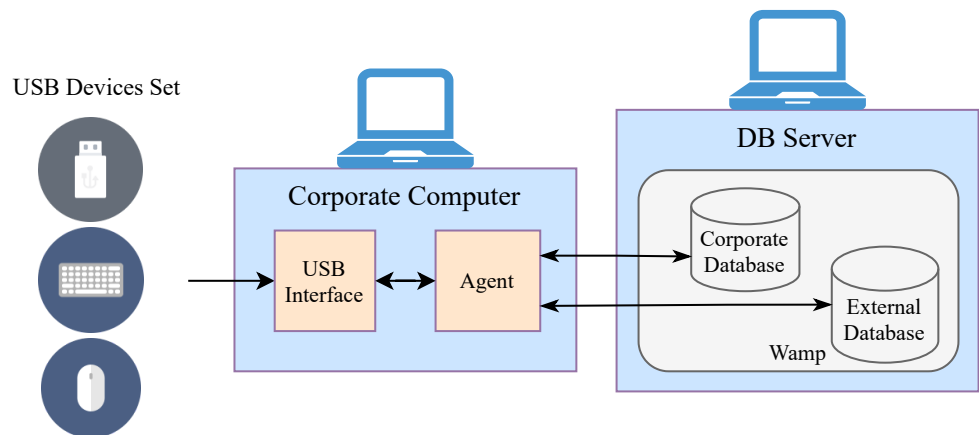


Figure 3. Test-bed topology.

When running and inserting USB devices, the agent presents an User Interface (UI) with two tabs: analysed devices and pending devices. Figure 4 presents the analysed devices tab, with the working devices on the system. Figure 5 shows the pending devices tab, with a filtered list of pending devices and their threat assessment from Corporate DBs and External DBs. In this tab, the User Experience (UX) enables users to accept devices or declare a threat, thus blocking the device.



Figure 4. Main page with the list of analysed devices.

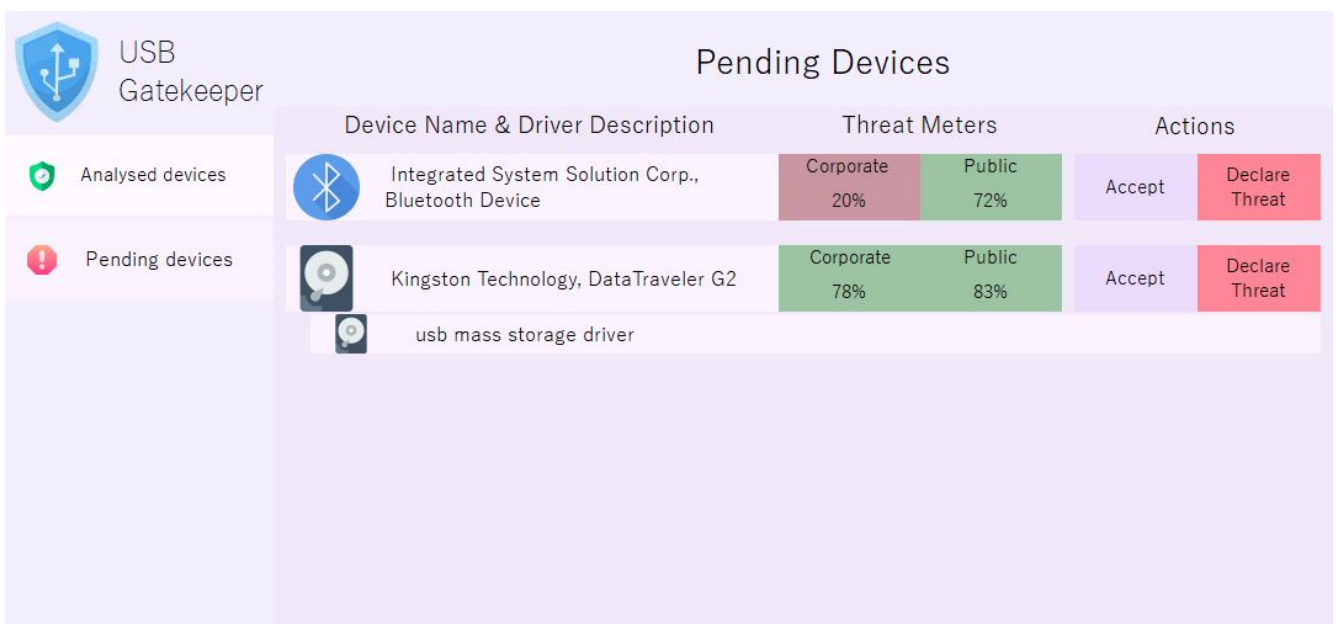


Figure 5. List of pending devices.

The threat assessments provided by the Corporate DB and the External DB (presented in Figure 5, respectively, as “Corporate” and “Public”) aid users to take action based on previous classifications by other users. Thus, we assume that the higher the number of devices already classified, the higher the quality of the threat assessments presented to the users.

Multiple USB devices, such as network and Bluetooth adapters, mass storage devices, keyboards, and mice, were connected to assess the performance of the proposed architecture regarding the time needed to find the device functionality. As an example, the set of devices presented in Table 1 was tested. For each device inserted, the device name from the OS was obtained, and the agent delay (1) and the installation delay (2) was measured in seconds. The total delay (3) was determined by the sum of both delays and a ratio between agent delay (1) and total delay (3) was calculated.

Table 1. Performance results for some devices tested with ratio included.

Description	Name Obtained from OS	Agent Delay (s)(1)	InstallationDelay (s)(2)	TotalDelay (s) (3) = (1) + (2)	DelayRatio (%) $\frac{(1)}{(3)} \times 100$
Card Reader Writer	USB 2.0-CRW	0.56	1.59	2.15	26.04
Kingston DataTraveler Mass Storage Device	DataTraveler G2	0.54	4.45	4.99	10.82
TP-Link Network Adapter	802.11n NIC	0.58	2.03	2.61	22.22
Bluetooth Adapter	ISSCEDRBTA	0.80	1.06	1.86	43.01
Kingston DataTraveler Mass Storage Device	DT 101 G2	0.75	0.67	1.42	52.81

From the set of tested devices, the agent spent between 0.56 and 0.8 s to find the functionality, corresponding to the time taken to install the device, which ranged between 10.82% and 52.81%. Thus, the additional delay introduced by the agent is considered to not affect the user experience. Further tests can be conducted to assess how much these delays vary when using other devices, operating systems or hardware.

5. Conclusions and Future Work

Security attacks using USB devices impose high risks to users and companies. In particular, the BadUSB attack allows attackers to inject a malicious set of keystrokes on the OS without the user's knowledge.

This paper proposes a distributed system protection architecture against BadUSB attacks designed for personal or corporate computers running Microsoft Windows OS. This architecture uses software blocking to enhance system protection against BadUSB attacks and it is composed of multiple agents and external DBs. When a USB device is connected, the agent inspects the device, and provides filtered information about the functionality and a threat assessment based on all previous user actions, stored in external DBs. While the agent is running, all connected USB devices must be approved by the user before use.

A prototype of the proposed architecture was developed and tested. The filtered information presented in the user interface of the agent intends to aid users in the choice to either accept or block new devices and, from a set of USB devices tested, the additional delay introduced by the agent seems not to significantly affect the user experience.

The authors consider that this architecture should be available as an open-source project, and future work can address this. The UI/UX can be improved, and, taking the results of a system usability scale tool into account, the option to uninstall devices can be added, and the overall resilience of the distributed architecture can be enhanced by using a cloud server DBs.

Author Contributions: J.O., P.P. and H.S. designed the proposed architecture; J.O. implemented and tested the proposed architecture; all authors performed the results' analysis and wrote the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank Miguel Frade for his contribution with the initial version of this work in [10].

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sabillon, R.; Serra-Ruiz, J.; Cavaller, V.; Cano, J. A comprehensive cybersecurity audit model to improve cybersecurity assurance: The cybersecurity audit model (CSAM). In Proceedings of the 2017 International Conference on Information Systems and Computer Science, INCISCOS 2017, Quito, Ecuador, 23–25 November 2018, Volume 2017; pp. 253–259. [CrossRef]
2. Almeida, V.A.; Doneda, D.; De Souza Abreu, J. Cyberwarfare and digital governance. *IEEE Internet Comput.* **2017**, *21*, 68–71. [CrossRef]

3. Li, X.; Zhou, C.; Tian, Y.C.; Xiong, N.; Qin, Y. Asset-Based Dynamic Impact Assessment of Cyberattacks for Risk Analysis in Industrial Control Systems. *IEEE Trans. Ind. Inform.* **2018**, *14*, 608–618. [CrossRef]
4. Floyd, T.; Grieco, M.; Reid, E.F. Mining hospital data breach records: Cyber threats to U.S. hospitals. In Proceedings of the IEEE International Conference on Intelligence and Security Informatics: Cybersecurity and Big Data, ISI 2016, Tucson, AZ, USA, 9–10 November 2016; pp. 43–48. [CrossRef]
5. Reddy, S.T.; Lakshmi, D.L.; Deepthi, C.; Sikha, O.K. USB-SEC: A secure application to manage removable media. In Proceedings of the 10th International Conference on Intelligent Systems and Control, ISCO 2016, Coimbatore, India, 7–8 January 2016; pp. 1–4. [CrossRef]
6. Jodeit, M.; Johns, M. USB Device Drivers: A Stepping Stone into Your Kernel. In Proceedings of the 2010 European Conference on Computer Network Defense, Berlin, Germany, 28–29 October 2010; pp. 46–52. [CrossRef]
7. Human Interface Device (HID). Available online: <https://www.techopedia.com/definition/19781/human-interface-device-hid> (accessed on 20 January 2021).
8. Nohl, K.; Krißler, S.; Lell, J. BadUSB—On Accessories That Turn Evil. Available online: <https://srlabs.de/wp-content/uploads/2014/07/SRLabs-BadUSB-BlackHat-v1.pdf> (accessed on 20 December 2020)
9. Shafique, U.; Zahur, S.B. *Towards Protection Against a USB Device Whose Firmware Has Been Compromised or Turned as ‘BadUSB’*; Springer: Cham, Switzerland, 2020; pp. 975–987. [CrossRef]
10. Oliveira, J.; Frade, M.; Pinto, P. System protection agent against unauthorized activities via USB devices. In Proceedings of the IoTBDS 2018—Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security, Funchal, Portugal, 19–21 March 2018; Volume 2018. [CrossRef]
11. Pham, D.V.; Syed, A.; Halgamuge, M.N. Universal Serial Bus Based Software Attacks and Protection Solutions. *Digit. Investig.* **2011**. [CrossRef]
12. Tian, J.; Scaife, N.; Kumar, D.; Bailey, M.; Bates, A.; Butler, K. SoK: “Plug & Pray” Today—Understanding USB Insecurity in Versions 1 Through C. In Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 20–24 May 2018; pp. 1032–1047. [CrossRef]
13. Benadjila, R.; Renard, M.; Trebuchet, P.; Thierry, P.; Michelizza, A.; Lefaire, J. WooKey: USB Devices Strike Back. In Proceedings of SSTIC 2018, Rennes, France, 13–15 June 2018.
14. Mertz, L. Cyberattacks on Devices Threaten Data and Patients: Cybersecurity Risks Come with the Territory. Three Experts Explain What You Need to Know. *IEEE Pulse* **2018**, *9*, 25–28. [CrossRef] [PubMed]
15. Poeplau, S.; Gassen, J.; Gerhards-Padilla, E. A honeypot for arbitrary malware on USB storage devices. In Proceedings of the 7th International Conference on Risks and Security of Internet and Systems, CRiSIS 2012, Cork, Ireland, 10–12 October 2012; pp. 1–8. [CrossRef]
16. Tischer, M.; Durumeric, Z.; Bursztein, E.; Bailey, M. The Danger of USB Drives. 2017. Available online: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7891503> (accessed on 21 December 2018).
17. Tischer, M.; Durumeric, Z.; Foster, S.; Duan, S.; Mori, A.; Bursztein, E.; Bailey, M. Users Really Do Plug in USB Drives They Find. In Proceedings of the 2016 IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, 22–26 May 2016; pp. 306–319. [CrossRef]
18. Andreasson, K.J. *Cybersecurity: Public Sector Threats and Responses*; CRC Press: Boca Raton, FL, USA, 2012.
19. Greene, T. Conficker Worm Takes Manchester Police Offline for Three Days | Security | Computerworld UK. 2010. Available online: <https://www.computerworlduk.com/security/conficker-worm-takes-manchester-police-offline-for-three-days-18640/> (accessed on 27 December 2018).
20. Terdiman, D. Stuxnet Delivered to Iranian Nuclear Plant on Thumb Drive. 2012. Available online: <https://www.cnet.com/news/stuxnet-delivered-to-iranian-nuclear-plant-on-thumb-drive/> (accessed on 30 December 2020).
21. Complex, I. Stuxnet could Hijack Power Plants, Refineries. 2010. Available online: <https://www.cnet.com/news/stuxnet-could-hijack-power-plants-refineries/> (accessed on 30 December 2020).
22. Nissim, N.; Yahalom, R.; Elovici, Y. USB-based attacks. *Comput. Secur.* **2017**, *70*, 675–688. [CrossRef]
23. Cimpanu, C. Here’s a List of 29 Different Types of USB Attacks. 2018. Available online: <https://www.bleepingcomputer.com/news/security/heres-a-list-of-29-different-types-of-usb-attacks/> (accessed on 30 December 2020).
24. Crenshaw, A. Plug and Prey: Malicious USB Devices. In Proceedings of the Shmoocon, Washington, DC, USA, 28–30 January 2011; pp. 1–41.
25. Walters, P. *The Risks of Using Portable Devices What Are the Risks?* Carnegie Mellon University: Pittsburgh, PA, USA, 2012.
26. Microsoft Releases Security Update for Autorun Vulnerability—Redmondmag.com. Available online: <https://redmondmag.com/articles/2011/02/10/update-for-autorun-vulnerability.aspx> (accessed on 30 December 2020).
27. Reviews of the Best USB Keyloggers for 2018–2019—Nerd Techy. 2017. Available online: <https://nerdtechy.com/reviews-best-usb-keyloggers/> (accessed on 21 December 2018).
28. Say hello to BadUSB 2.0: A USB Man-In-The-Middle Attack Proof of Concept | CSO Online. Available online: <https://www.csoonline.com/article/3087484/say-hello-to-badusb-20-usb-man-in-the-middle-attack-proof-of-concept.html> (accessed on 30 December 2020).
29. USB Killer v3—USBKill. Available online: <https://usbkill.com/products/usb-killer-v3> (accessed on 30 December 2020).

30. Thousands of Hacked CCTV Devices Used in DDoS Attacks | PCWorld. Available online: <https://www.pcworld.com/article/3089346/thousands-of-hacked-cctv-devices-used-in-ddos-attacks.html> (accessed on 30 December 2020).
31. Tian, D.; Bates, A.; Butler, K. Defending Against Malicious USB Firmware with GoodUSB. In Proceedings of the 31st Annual Computer Security Applications Conference on—ACSAC 2015, Los Angeles, CA, USA, 7–11 December 2015; [CrossRef]
32. USBKey. Available online: <http://www.meet-electronics.com/products/usbkey> (accessed on 30 December 2020).
33. Ulanoff Lance. How You Can Avoid a BadUSB Attack. 2014. Available online: <https://mashable.com/2014/10/03/how-can-you-avoid-badusb/?europe=true#DhMbENGvxiqh> (accessed on 30 December 2020).
34. Natalie, C. Protect Against BadUSB with Mobile Cloud Storage | ChaiOne. 2016. Available online: <https://chaione.com/blog/protect-badusb-cloud-storage/> (accessed on 30 December 2020).
35. Port Blocking: Why Corporate Computers Need Disabled USB Ports—The Network Hub. Available online: <https://searchnetworking.techtarget.com/blog/The-Network-Hub/Port-blocking-Why-corporate-computers-need-disabled-USB-ports> (accessed on 30 December 2020).
36. Poppe, C.H. USB Port Locking and Blocking Device. U.S. Patent 7,635,272, 22 December 2009.
37. Block Access to USB Ports—SecuPlus Shop. Available online: <https://www.secuplus-shop.nl/en/secumate-usb-port-locks.html> (accessed on 30 December 2020).
38. Oliveira, J.; Frade, M.; Pinto, P. JoseOliveira01/usb-Whitelisting: USB-Whitelisting. 2017. Available online: https://zenodo.org/record/1009691#.XB0Ya8_7TKM (accessed on 30 December 2020).
39. Overview of INF Files—Windows Drivers | Microsoft Docs. Available online: <https://docs.microsoft.com/en-us/windows-hardware/drivers/install/overview-of-inf-files> (accessed on 30 December 2020).
40. Bourdon, R. WampServer, La Plate-Forme De Développement Web Sous Windows—Apache, MySQL, PHP. 2020. Available online: <https://www.wampserver.com/en/> (accessed on 21 February 2021).