Weaponizing the Raspberry Pi Pico: Implications for Everyday Cybersecurity

Weaponizing the Raspberry Pi Pico: Implications for Everyday Cybersecurity

1. Introduction

The USB Rubber Ducky, introduced in 2010 by Hak5, popularized keystroke injection as a practical method of cyberattack by exploiting fundamental trust assumptions built into operating systems (Wilson, 2022). What began as a tool developed by founder Darren Kitchen to streamline mundane office tasks, soon became a powerful instrument capable of executing attacks requiring nothing more than a simple USB device (*Hak5/The Official USB Rubber Ducky Payload Repository*, 2025).

This project explores keystroke injection attacks, looking at how they work, how they are used, and the security weaknesses they reveal in everyday computer systems. While large-scale cyberattacks such as ransomware attacks and data breaches dominate headlines, everyday vulnerabilities like keystroke injection attacks continue to pose significant risks to individuals and organisations. This project also explores strategies to mitigate these attacks, including restricting USB access and promoting user awareness to recognise and avoid suspicious devices.

2. Background

2.1 USB Human Interface Devices

The USB Human Interface Device (HID) class is a specification for USB devices that allows them to communicate with the host system without needing specialised drivers. These devices include, but are not limited to, keyboards, mice, game controllers, and touchscreens. In essence, it covers anything a human use to send input to the host. The operating system allows HID devices to interact and communicate directly with the local machine, enabling more convenient use and plug-and-play functionality without requiring drivers to be installed beforehand (Acroname, n.d.).

2.2 Keystroke Injection Attacks and HID Exploits

Keystroke injection attacks take advantage of the HID trust model by using malicious USB devices that pretend to be legitimate keyboards (Zhao & Wang, 2019). A common example is the Hak5 Rubber Ducky, which looks like a normal flash drive but acts as a programmable keyboard to send automated keystrokes. These devices can type commands on a target system much faster than any human could react, making the attack difficult to stop once it begins. Since, they operate at the hardware level, where operating systems have few security checks, they can easily bypass defences. As a consequence, once connected, they can run system commands, install programs, change settings, or steal data with the same permissions as a regular user (Zhao & Wang, 2019).

2.3 Attack Tools for Keystroke Injection

The Hak5 Rubber Ducky retails for over A$150, however, the same functionality can be reproduced affordably using microcontrollers such as the Digispark, the Arduino or Raspberry Pi series, which costs around A$6. For this project, a Raspberry Pi Pico was used to replicate Rubber Ducky functionality for testing. These devices primarily use the DuckyScript, a scripting language that consists of simple commands that are translated into payloads and automatically executed upon connection (Hak5, 2025).

2.4 Attack Methods and Techniques

Timer Delays

Attackers often use evasion techniques to increase the effectiveness of the attacks, most of these accessible to a user with standard permissions. One of the key strengths of HID attacks is its speed. However, this increases the risk of being detected by antivirus software (Gurčinas et al., 2023). Therefore, attackers add random time delays in input to closer mimic a real human typing, successfully bypassing detection tools that monitor for automated bot-like input.

Living Off The Land Binaries

"Living Off The Land Binaries" or "LOLBins" are local system binaries and preinstalled tools that attackers frequently exploit in their attacks (Nikhil Mohanlal, 2021). These are non-malicious and are preinstalled tools local to the system, which allow for operations aimed at bypassing detection and escalate privileges (Oleksandra Rumiantseva, 2023). Some of the commonly used binaries include *Certutil,* PowerShell, and Windows Management Instrumentation Command-line (WMIC). Certutil is a Windows utility that allows users to manage certificates on the local machine, and since it comes preinstalled by Windows, can bypass most antivirus software. It also offers other utilities like encoding and decoding, which attackers use to download encoded PowerShell commands from public servers onto the victim’s machine. This is then decoded by Certutil and executed, camouflaged as normal network configurations done by the user (Carrier, 2024).

Invoke Expression

Another example of obfuscation is the use of the "Invoke Expression" (IEX). This command allows the user to download and execute programs entirely in memory without being written to disk, avoiding detection by many antivirus programs. There are also numerous ways to avoid detection of the IEX command itself; by splitting the string, using character substitution and other methods (*Securonix Threat Research Knowledge Sharing Series: Hiding the PowerShell Execution Flow*, 2024). As an example, this following script uses character substitution to obfuscate the code to avoid detection:

Original:

```
Set objShell = CreateObject("WScript.Shell")
objShell.Run "powershell.exe -Command & {Start-Process calc}", 0
Set objShell = Nothing
```

i: Securonix Threat Research Knowledge Sharing Series: Hiding the PowerShell Execution Flow, 2024

Obfuscated:

```
Dim s1, s2, s3, s4, s5, x, y
s1 = Chr(112) & Chr(111) & Chr(119) & Chr(101) & Chr(114) & Chr(115) &
Chr(104) & Chr(101) & Chr(108) & Chr(108)
s2 = Chr(46) & Chr(101) & Chr(120) & Chr(101)
s3 = Chr(32) & Chr(45) & Chr(67) & Chr(111) & Chr(109) & Chr(109) &
Chr(97) & Chr(110) & Chr(100) & Chr(32) & Chr(38) & Chr(32) & Chr(123) &
Chr(83) & Chr(116) & Chr(97) & Chr(114) & Chr(116) & Chr(45) & Chr(80) &
Chr(114) & Chr(111) & Chr(99) & Chr(101) & Chr(115) & Chr(115) & Chr(32) &
Chr(99) & Chr(97) & Chr(108) & Chr(99) & Chr(125)
s4 = Chr(87) & Chr(83) & Chr(99) & Chr(114) & Chr(105) & Chr(112) &
Chr(116) & Chr(46) & Chr(83) & Chr(104) & Chr(101) & Chr(108) & Chr(108)
Set x = CreateObject(s4)
y = s1 & s2 & s3
x.Run y, 0
Set x = Nothing
```

ii: Securonix Threat Research Knowledge Sharing Series: Hiding the PowerShell Execution Flow, 2024

PowerShell Flags

To make keystroke injection attacks stealthier, payloads frequently use PowerShell one-liners with specific flags designed to avoid detection. They are particularly powerful as they decrease the time required for execution, reduce the visibility of the attack and the risk of user interruption (Hak5 - YouTube, 2016). These flags are commonly used:

- **-WindowStyle Hidden / -W Hidden:** keeps the PowerShell window hidden from the user, running scripts silently in the background.
- **-NoProfile**: avoids loading user profiles for faster load-up and execution.
- **-NonInteractive**: prevents PowerShell from prompting the user for any input.
- **-ExecutionPolicy Bypass**: allows execution of untrusted scripts, suppressing warnings and prompts.
- **-ErrorAction SilentlyContinue**: suppresses all visible error messages.

Remote Triggered Payloads

Despite its many obfuscation and concealment methods, the main disadvantage of a keystroke injection attacks is its visibility. Even with hidden windows, the attacker relies on the victim's lack of response during the attack, but even so, it is often obvious that someone launched an attack on their device. To solve this, Krzysztof Witek created an open-source Rubber Ducky clone that allows the attacker to trigger the payload whenever they want, for example when the victim is distracted or away from the computer (Hackaday, 2025).

2.5 High-Profile USB-Based Attacks

While this project focuses on keystroke injection via HID-emulating devices, USB-based attacks also include a wider collection of threats that can target anything from private individuals to national organisations. In 2008, the United States Department of Defense was infected with malware (Wikipedia, n.d.), originating from a malicious USB device plugged into a laptop at a U.S. military base in the Middle East. The attack led to the creation of the U.S. Cyber Command, now responsible for the Department of Defense's cyber operations. This incident is widely regarded as the worst breach of U.S. military computers in history, and as stated by then–Deputy Secretary of Defense William J. Lynn III: *"marked a turning point in U.S. cyberdefense strategy"* (CNET, n.d.). Other espionage-focused attacks include the China-linked SOGU campaign, described by cybersecurity firm Mandiant as *"one of the most aggressive cyber espionage campaigns targeting both public and private sector organizations globally across industry verticals"* (Mandiant, n.d.). These examples show that USB-based attacks are not limited to simple keystroke injection or individual targets, they remain an ongoing threat to organisations and even government agencies.

3. Experimental Results and Payload Demonstration

Replicating the Rubber Ducky attack using a Raspberry Pi Pico revealed just how easily an attacker can compromise a system using just a simple HID-emulating microcontroller. Immediately upon plugging it into a Windows 10 machine, the Raspberry Pi Pico ran a simple Python script that launched a terminal, executed multiple instructions, and closed all windows within seconds, completely without any user interaction. The script ran faster than a human could react, and once connected, there was no straightforward way for the user to stop the payload. Throughout the attack, no security warnings appeared, and the antivirus software remained silent. The operating system treated the Pico identically to a physical keyboard, granting it unrestricted input control. While actions like installing software might require administrator permissions, other harmful actions like file modification, browser use and script execution was still accessible with standard user permissions.

This demonstrated several critical vulnerabilities. As operating systems rarely validate HID input beyond the initial USB handshake, malicious devices like the Rubber Ducky are able to execute scripts faster than users can respond, without any interruption from the security software. This highlighted the fact that the weakness does not lie in the software but in the operating systems trust in hardware.

Payload Examples

A series of DuckyScript payloads were developed to simulate different attack types:

- **wifi.dd**: Extracts saved Wi-Fi credentials from the system using netsh commands and outputs the passwords. This showcases how an attacker can collect sensitive data and save directly to device.
- **clear_history.dd**: Clears PowerShell session history, command history, and Windows Run dialog log. Adding these commands to other payloads allows the attacker to remove traces of their activity and reduce the risk of detection during forensic analysis.
- **defender.dd**: Temporarily disables Microsoft Defender. This requires administrator privileges but shows that it is possible to turn off security software and warnings disabled without a password prompt, provided that the attacker is logged in as a user with elevated rights.
- **ducky_float.dd**: Downloads and runs a remote script using iex and DownloadFile. This payload demonstrated the ability to download remote files without writing to disk.
- **top_secret.dd**: Downloads and runs a remote script using iex and DownloadString. This payload shows how an attacker can copy files directly from the target onto own disk. With slight modification, this payload could also be used to upload files from the attacker to the target via curl or other similar tools.
- **redirect.dd**: Hosts a local HTTP server that redirects to a chosen URL. By modifying the local host file, the attacker is able to redirect any site to a locally hosted server. Even though this type of redirection is not particularly useful in a practical attack, it demonstrates the level of control an attacker can have over network behaviour, and how a browser session can be intercepted and redirected without user consent.

These payloads are short, simple, and highly effective. Many were adapted from publicly available scripts online, showing how little technical knowledge and skills are required to create or modify powerful attack tools using USB-based keystroke injection (Hak5, 2022).

4. Analysis and Mitigation

4.1 Human Factors and Social Engineering

Beyond the way the operating systems allow these devices to exploit the trust model in hardware, the real issue lies in the human factors. Research shows that social engineering is behind between 70% and 90% of cyber attacks *(KnowBe4, 2023; Nuix, 2017)*. Social engineering is the technique of manipulating or influencing victims or exploiting human errors to gather confidential information or gain unauthorized access. Physical baiting is a form of a social engineering attack, where an attacker leaves a physical device, often infected with malware, out in the open for someone to plug in (Canon, n.d.). A user may connect unknown USB devices out of habit or curiosity, unaware of the potential consequences. In 2016, Victoria Police issued a warning against unmarked infected USBs left in mailboxes around Melbourne (BBC, 2016).

As part of a study done by a group of researchers at the University of Illinois, University of Michigan, and Google, they left almost three hundred flash drives around a university campus. By placing tracking tags in files on the drive, they found that the attack had a success rate of over 50%, even showing that people were more likely to plug in an unmarked drive, compared to one with a return label (Tischer et al., 2016). This further highlights how human curiosity contributes to the effectiveness of attacks.

4.2 Payload Accessibility And Ethical Considerations

Easily available payloads available online demonstrate the accessibility of hacking tools, and with just a few searches, anyone can find pre-written DuckyScript payloads, PowerShell one-liners, or GitHub repositories ready to use. While this widespread availability assists security professionals and ethical hackers to test and discover vulnerabilities in systems, they do also pose a significant risk. These same tools can be used by malicious hackers, even with minimal technical skills, and greatly lowers the threshold to entry to cybercrime.

This underlines the fundamental challenge in cybersecurity education, where balancing the value of openly sharing tools and methods for educational purposes, with the potential for misuse.

4.3 Defending Against HID-based Keystroke Attacks

4.3.1 Technical Limitations of USB Defences

On the technical side, current defensive measures are insufficient in preventing keystroke attacks. Tools to monitor and restrict USB device behaviour do exist, but the most successful are generally only implemented in high security environments (Nissim, Yahalom, & Elovici, 2017). As noted in the article *"USB-based attacks"* by Nissim et al. (2017), the authors conclude that *"... current detection and prevention solutions largely tend to concentrate on specific attacks or fail to provide a comprehensive and effective solution."*

Many small businesses and individual private users operate with minimal USB access control, and device whitelisting require configuration outside the scope of most casual users. Even when such measures are in place, they can be difficult to sustain, as users still need to connect legitimate devices like keyboards and mice for everyday use.

As cited in the 2023–24 Annual Cyber Threat Report, small businesses are at greater risk for attacks: *"The majority of reports were from small businesses, and the impact on business is significant. According to the Australian Bureau of Statistics, in 2022-23, 91.9% of businesses had turnover of less than $2* million" (Australian Cyber Security Centre [ACSC], 2024). This means that with a minor budget and less security protocols in place, small businesses may need to look elsewhere to protect against these kinds of attacks.

4.3.2 Behaviour-Based Detection

Another defensive approach focuses on analysing input methods and patterns to distinguish between human users and automated bots. Researchers Kadyshevitch and Madmon (2023) explore how it is possible to detect automated bots just by looking at how input their data into forms. By building user profiles based on input method, they can detect suspicious behaviour where a sudden change like pasting their password instead of typing it manually is a strong indicator of bot behaviour.

They also look at how typing speed and fluctuations vary between humans and bots: *"Humans can get distracted when typing, or have difficulty reaching one key over another on their keyboard. As a result, bots are expected to type much faster than humans and have far less fluctuation in typing speed"* (Kadyshevitch & Madmon, 2023*)*. By further breaking down data into multiple cases, typical bot behaviour, human-like bot behaviour, and human behaviour, they were successfully able to even detect bots designed to mimic human activity. While this research appears promising for future prevention, it still requires technical implementation that might not be feasible for small businesses and casual users.

4.3.3 Education and Training as a Defence Tool

In a literature review, Junger and Bulle (2020) investigated the effectiveness of different social engineering interventions. Their research show that focused interactive training was far more effective than generic warnings which are easily ignored. By looking at various delivery methods, they saw that high-intensity interventions, like game-based training or lectures were more effective compared to low intensity like reading materials. However, as high intensity interventions might be more time-consuming and more expensive, they recommend a balance: "*a simple but effective intervention is overall preferable in terms of cost-effectiveness"* (Junger & Bulle, 2020).

This training should also be repeated as the effectiveness decreases over time. Therefore, interactive educational lessons with medium intensity on USB baiting would be most effective in preventing these attacks. These preventative measures are accessible, cost effective and easy to implement even for small businesses.

5. Conclusion

This project revealed fundamental vulnerabilities in how operating systems handle USB devices. Through the development and demonstration of multiple payloads and by successfully replicating a Rubber Ducky with a A$6 Raspberry Pi Pico, it became clear that sophisticated cyber attacks are now accessible to anyone with minimal technical knowledge.

Research into attack methods and techniques, social engineering and preventative strategies highlighted the both the simplicity of keystroke attacks, as well as the challenges of defending against them. The current defensive measures prove insufficient in protecting the ones most at risk: small businesses and individual users. Due to their limited resources and low awareness, they are disproportionately at risk and require other precautions.

While solutions such as behaviour-based analysis and USB controls appear promising, they are often impractical for most. The focus should lie in education and awareness, and repeated, interactive training programs offer the most practical and cost-effective defence against these social engineering-targeted attacks.

Ultimately, this project illustrates that as big-scale cyber attacks become more prominent, small-scale threats remain a serious concern. With inexpensive microcontrollers capable of executing powerful and catastrophic attacks, there is a need for broader cybersecurity awareness, not just at corporation level, but also among individuals and small organisations, who remain highly vulnerable to low-cost, high-impact attacks.

6. References

Acroname. (n.d.). *How USB HID makes plug and play devices work*. https://acroname.com/blog/how-usb-hid-makes-plug-and-play-devices-work

Australian Cyber Security Centre. (2024). *Annual cyber threat report 2023–24*. https://www.cyber.gov.au/about-us/view-all-content/reports-and-statistics/annual-cyber-threat-report-2023-2024

BBC. (2016). *Malicious USB sticks left in Australian letterboxes*. https://www.bbc.com/news/technology-37431335

Canon. (n.d.). *What is baiting in cyber security?*. https://business.canon.com.au/insights/what-is-baiting-in-cyber-security

CNET. (n.d.). *Bad flash drive caused worst U.S. military breach*. https://www.cnet.com/news/privacy/bad-flash-drive-caused-worst-u-s-military-breach/

Colin Wilson - Lotus Labs. (n.d.). *USB Rubber Ducky by Hak5*. Hak5 - USB Rubber Ducky. https://docs.hak5.org/hak5-usb-rubber-ducky/usb-rubber-ducky-by-hak5/

Egycondor. (n.d.). *Code obfuscation detection and mitigation*. https://medium.com/@egycondor/code-obfuscation-detection-and-mitigation-27c7480ef5ec

Feedzai. (n.d.). *What is bot detection?*. https://www.feedzai.com/blog/what-is-bot-detection/

Hackaday. (2023). *Keystroke injection*. https://hackaday.com/tag/keystroke-injection/

Hak5. (n.d.). *DuckyScript 3.0*. https://shop.hak5.org/pages/duckyscript-3-0

Junger, M., & Bulle, J. (2020). How effective are social engineering interventions? *Information and Computer Security, 28*(5), 801–819. https://doi.org/10.1108/ICS-07-2019-0078

Kadyshevitch, & Madmon. (n.d.). *Bot detection based on input method analysis*. https://transmitsecurity.com/blog/bot-detection-based-on-input-method-analysis

KnowBe4. (2023). *Social engineering accounts for 90% of attacks – why is it ignored?*. https://blog.knowbe4.com/social-engineering-accounts-for-90-of-attacks-why-is-it-ignored

Mandiant. (n.d.). *Infected USB devices steal secrets in aggressive espionage campaign*. https://www.mandiant.com/resources/blog/infected-usb-steal-secrets

Martinez, A. (2019). *Rubber Ducky: Learning about keystroke injection*. https://medium.com/trabe/rubber-ducky-learning-about-keystroke-injection-324f462f80fa

Nissim, N., Yahalom, R., & Elovici, Y. (2017). USB-based attacks. *Computers & Security, 70*, 675–692. https://www.sciencedirect.com/science/article/pii/S0167404817301578

Nuix. (2017). *Black Report 2017*. https://www.nuix.com/resources/black-report-2017

OPSWAT. (2021). *The danger of a USB device and keystroke injection attack*. https://www.opswat.com/blog/the-danger-of-a-usb-device-and-keystroke-injection-attack

PortSwigger. (2023). *Google develops Linux tool that tackles USB keystroke injection attacks*. https://portswigger.net/daily-swig/google-develops-linux-tool-that-tackles-usb-keystroke-injection-attacks

Oleksandra Rumiantseva. (2023). *What Are LOLBins?* SOC Prime. https://socprime.com/blog/what-are-lolbins/

Zhao, S., Wang, X., (2019). *USB-based attacks and defenses*. https://link.springer.com/chapter/10.1007/978-3-030-33506-9_71

Tischer, M., Durumeric, Z., Foster, S., Duan, S., Mori, A., Bursztein, E., & Bailey, M. (2016). Users really do plug in USB drives they find. *2016 IEEE Symposium on Security and Privacy (SP)*, 306–319. https://doi.org/10.1109/sp.2016.26

Wikipedia. (n.d.). *2008 malware infection of the United States Department of Defense*. https://en.wikipedia.org/wiki/2008_malware_infection_of_the_United_States_Department_of_Defense

Wired. (n.d.). *China's USB malware campaign exposed*. https://www.wired.com/story/china-usb-sogu-malware/

ztwAdmin. (n.d.). *ZTW-2025 GitHub Repository*. https://github.com/ztwAdmin/ZTW-2025/tree/main