Keystroke Injection for Sending Phishing Emails from an Internal Address

James Palmer
Dept. of Computer Science and Information Systems
University of North Alabama
Florence, AL
jpalmer8@una.edu

***Abstract*—Keystroke injection attacks pose a significant threat to endpoint security by exploiting unlocked systems and emulating human input through malicious USB devices. These attacks can automate a range of malicious actions, such as sending phishing emails directly from a compromised organization's email client, effectively bypassing conventional email filtering mechanisms. This paper introduces a penetration testing tool that demonstrates this tactic by crafting and sending phishing emails via keystroke injection, highlighting a critical vulnerability in physical endpoint access. To mitigate this threat, a lightweight detection and prevention mechanism is proposed, capable of identifying keystroke behavior characteristic of injection attacks. The countermeasure successfully detected and blocked the penetration tool's activity in all test cases, with zero false negatives observed. These findings underscore the need for a comprehensive security posture that combines robust technical safeguards with strong administrative policies, such as user awareness training and enforced device locking practices.**

***Index Terms*—DuckyScript, Flipper Zero, keystroke injection, Outlook, phishing, social engineering.**

I. Introduction

The following sections define the attack vectors used in the penetration testing tool, explain how those attacks is carried out, and give information on the impact of the attacks when possible.

A. Phishing

Phishing is an ever-present threat to both individuals and organizations. The goal of a phishing email is to convince the recipient to perform some action that is against their best interest. Usually, this comes in the form of clicking a link that leads to malicious content. The method by which phishing emails convince users to comply with their requests is social engineering, which is the practice of exploiting human nature through influence, manipulation, deception, or other means [1].

Phishing emails tend to use the same overall approach to influence their recipients. The most common feature of phishing emails is verbiage that elicits a sense of urgency. In fact, "important update", "important", "urgent", and "attention" are recorded as the most frequently used words in phishing emails [2]. A sense of urgency may convince recipients to act without taking the time to scrutinize the email itself or the address that it originated from. While many phishing emails use negative emotions to spur their victims into action, some use positive emotions. These emotions are usually caused by a phishing email that details some offer or deal that is "too good to be true." The vast majority of malicious emails contain either a hyperlink or attachment that is used to deliver a payload or extract personal information. In the case of a hyperlink, they commonly lead to some site that looks identical to a page that the recipient is familiar with, and the recipient is therefore comfortable with entering in sensitive information. Attachments usually contain malware of some sort, requiring less action on the part of the recipient in order for the attack to be successful [1].

Phishing may be targeted at a very wide range of people or just a select few. Spear phishing is the term used to describe phishing emails that are specifically crafted to target either a small group or single individual. With a much smaller pool of recipients, attackers can tailor phishing emails that are much more likely to trick the recipients by including personalized information specific to those targets. Whaling is the use of phishing to target high-ranking individuals within an organization. These attacks can be particularly dangerous because compromise of highly privileged credentials is much more likely to lead to major financial losses. Before performing a whaling attack, malicious actors will usually gather a very large amount of information about the target to greatly increase the attack's chances of success.

Email has been used as an attack vector for almost as long as email services have been publicly available, with the term "phishing" being coined in 1996. The first phishing attacks utilized AOL accounts with falsified credit card information to send malicious emails anonymously. Later attacks involved the impersonation of AOL employees and asking users to "verify" their personal information like login credentials or payment information, thereby tricking victims into sending attackers sensitive information. By 2005, well over a million people had fallen victim to a phishing attack, and hundreds of millions dollars had been stolen by those performing the attacks [1]. Today, phishing emails have become incredibly common and pervasive. In fact, approximately 3.4 billion phishing emails are sent every day. In 2021, over 80% of businesses fell victim to phishing attacks. Multiple institutions, including the FBI and IBM, have labeled phishing as one of the most common attack methods for cybercrime [2]. In 2023, the FBI received almost 300,000 reports of phishing attacks. The losses included in those reports total $18.7 million, incurred by both

private individuals and businesses [3]. According to IBM, 15% of organizational data breaches began with a phishing email. Additionally, the average breach caused by phishing resulted in $4.88 million in losses. Recovery from a phishing attack also proved to be incredibly difficult, with the average time to resolve the attack being 261 days [4].

B. Keystroke Injection

Keystroke injection is an attack vector whereby a specialized hardware device connects to a target computer under the guise that the device is a common Human Interface Device (HID). Upon successful connection, the malicious device will "inject" keyboard or mouse input, causing the target computer to perform some series of malicious actions autonomously. Keystroke injection itself is not malicious, and it has a wide variety of uses. However, it can be incredibly dangerous when used for malicious purposes. The vast majority of keystroke injection takes place through a USB connection. Under normal circumstances, most operating systems trust USB connections by default. Therefore, the inputs sent by a malicious device are usually under no form of scrutiny. While this trust is convenient for legitimate HID input and helpful implementations of keystroke injection, it does present a major vulnerability if an attacker has physical access to a target device.

Due to the nature of keystroke injection, it is usually utilized as the first action of an attack, with some other malicious action taking place after the injection has occurred. This may take the form of keystroke injection being used to download malicious programs that then continue the attack, or injection being used to disable security features on a target machine, making the target vulnerable to further attacks. Commonly, keystroke injection devices begin the injection automatically once they are connected to a computer. Since keystroke injection requires physical interaction with a target computer, it is rare when compared to other attack vectors that can be utilized from any remote location.

Much like phishing emails, attackers may employ social engineering in order to ensure their malicious device is connected to the target computer. One very common social engineering approach is to leave a harmless-looking USB device that has the appearance of a flash drive laying somewhere the victim is likely to find it. If the victim is a curious person and wants to examine the content on the flash drive, he may plug the device into his computer, starting the attack without the malicious actor being present. Malicious USB devices may also be given as gifts to targets in the hopes they will plug them into their computers.

II. Methods

Presented in this paper is a penetration testing tool that takes advantage of both keystroke injection and phishing emails in order to perform an attack. Additionally, a countermeasure is described that prevents successful use of the penetration testing tool.

A. Penetration Testing Tool

The penetration testing tool is a collection of programs that enable a tester to perform a keystroke injection attack on computers with active authenticated sessions. The attack can target computers that use either Windows 10 or Windows 11 and have Outlook installed. Currently, the hardware utilized is a Flipper Zero with its official firmware installed, including the Bad USB program. The Bad USB program allows for the execution of DuckyScript version 1.0 keystroke injection scripts from a Flipper Zero connected to a computer through a USB connection. The scripts are interpreted by the Flipper Zero, converting the contents of the scripts into keystrokes. These keystrokes are then sent sequentially to the target computer. Figure 1 contains an image of a Flipper Zero.

Fig. 1. Image of a Flipper Zero device

When the physical connection is made between the Flipper Zero and the target computer while the Bad USB program is active, the Flipper Zero appears as an "HID Keyboard Device" in Windows Device Manager. Figure 2 displays the hardware ID provided by the Flipper Zero upon connection. This hardware ID falsely identifies the Flipper Zero as a keyboard and mouse manufactured by Logitech, Inc. This deception is accomplished by sending a report descriptor that supplies the false information to the Windows operating system.

With the connection complete, the Bad USB program can begin to send data that the computer will accept as valid keyboard and mouse input. The basic order of operations performed by the keystroke injection are as follows:

1) Open the Windows Start Menu
2) Type "outlook" into the search bar and press ENTER, thereby opening Outlook
3) Once Outlook is open and ready to accept input, open a new message
4) Enter in some combination of recipient addresses

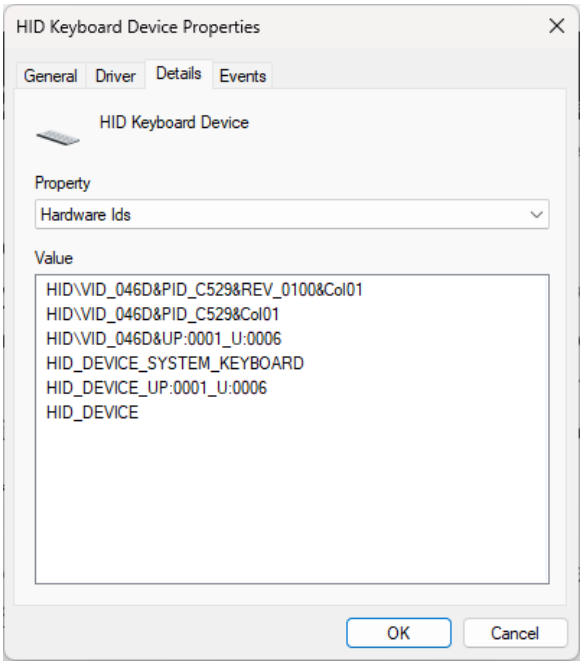

Fig. 2. Image of a Flipper Zero's HID information on a Windows 11 machine

5) Write the phishing email, including a link for the recipients to click
6) Send the message
7) Close Outlook

This attack can be incredibly dangerous because it takes advantage of the fact that there is a legitimate user logged on to the machine being used in the attack. When the phishing emails are sent, they will come from a legitimate, internal email address. Since the sender is both verified and internal, this attack bypasses the usual warnings present on Outlook that warn users of emails that originated from outside their organization. Also, mechanisms that either warn users of clicking links from outside sources or preventing the act altogether are completely bypassed. The specifics of two different approaches to this attack are detailed below.

1) Global Address List Approach: The simpler of the two approaches takes advantage of Outlook's Global Address List, which will recommend email addresses whenever characters are entered into the recipients field. The keystroke injection script iterates through the entire alphabet, accepting the first recommended address for each letter. Therefore, if every letter successfully populates a recommended address, the attack will send a phishing email to 26 different users. Unfortunately, the likelihood of this approach's success depends greatly on the size of the target organization; the smaller an organization is, the less likely it is that every letter will lead to a recommended address. Also, the same address may be recommended more than once. For example, the email address for John Doe may be the recommended address when either "j" or "d" is entered. However, this approach requires the attacker to know very little about the organization since the target addresses do not need to be picked manually.

2) Domain Lookup Approach: This approach involves using Open Source Intelligence (OSINT) to gather target email addresses before writing the keystroke injection script. These email addresses are currently gathered using the Prospeo API. A C++ program makes a call to the Prospeo API, parses the response, and writes the emails found to a file, email_list.txt. Another C++ program then reads the contents of email_list.txt to construct a DuckyScript program that will enter all of the emails found into the recipients field of the phishing email. Both of these programs are called by a Python program that acts as a pipeline, allowing for the user to call just one program to perform all of these actions. The Python program takes a single argument: the domain to search. Therefore, simply using the command "python3 pipeline.py una.edu" would produce a DuckyScript program that would send an email to every publicly available una.edu address accessible by the Prospeo API. The current email finder program is limited to finding 50 addresses because finding more would require paying for unrestricted access to Prospeo's services. Figure 3 is a flowchart depicting the entire pipeline for creating these DuckyScript programs.

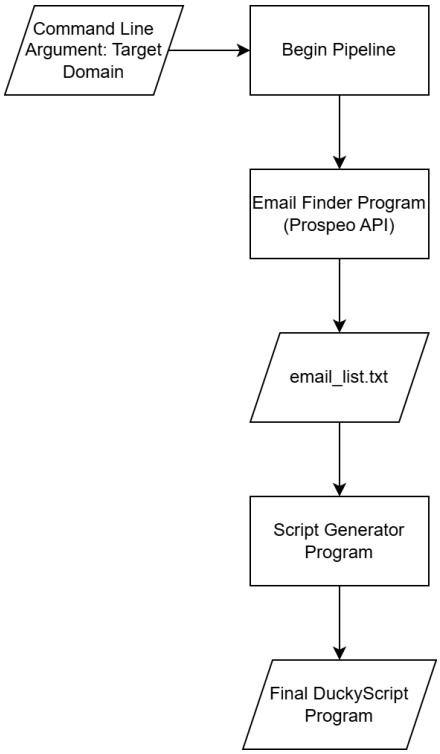

Fig. 3. Flowchart of pipeline for DuckyScript program generation

This approach allows for swift retrieval of a potentially very large pool of email addresses, but it requires the user to know the target organization's email domain ahead of time. While finding an organization's email domain is usually not a very difficult task, it does add an extra step that the previous approach avoided. The script generation could also be modified to send a separate email to every recipient rather

than putting all of the addresses into the recipients field at the same time. This would remove the risk of recipients becoming suspicious if they saw the large number of people the email was sent to. The problem with this modification, however, is that it would take an incredibly long time to complete the attack if 50 individual emails were written and sent.

Both approaches share some weaknesses. First, the use of the Outlook application means that the DuckyScript program cannot take advantage of working in a hidden window, allowing the attack to go unnoticed even if the user were still looking at his monitor. For this attack to be successful, the penetration tester would need to have access to a computer that a legitimate user was logged in to, but that user could not be present at the time of the keystroke injection. While the user's absence is not impossible, it is likely that the penetration tester will have to employ some means of convincing the user to step away from their computer. Second, since the DuckyScript program cannot use a command-line terminal, the attack takes significantly longer than those that are able to do so. No matter the approach used, the attack takes a minimum of 10 seconds, making it much more likely to be detected before the keystroke injection has completed.

B. Countermeasure for Developed Tool

The countermeasure is a program written in C++ that examines keystroke behavior. Two metrics are used to distinguish between human and automated keyboard input: keystroke speed and keypress duration. In order to detect all keystrokes sent to the system, the program uses a low-level keyboard hook provided by the Windows Win32 API, which is referenced as"windows.h" in C++. Since the keyboard hook used is low-level, it receives the input before it is sent to its intended application [5]. Since the keystrokes are first received by the countermeasure program, information about the time of receipt can be stored for use in keystroke injection detection.

The first metric, keystroke speed, is calculated using the differences in time between individual key strokes. These times between keystrokes are stored in a double-ended queue; as more keystrokes are detected, the oldest value will be removed, and the newly calculated value will be added. The values in the double-ended queue are used to calculate an average time between keystrokes. Finally, this average time is converted into a words per minute (WPM) value which is compared against a predetermined flag value to determine if the typing speed is inhuman. WPM values above the flag value will be printed to a console along with a warning message.

The second metric, keypress duration, is calculated using the difference in time between when a key is pressed and subsequently released. Each keystroke, including the press and corresponding release, is assigned a unique key value, and that value is stored along with the timestamp of the keypress in an unordered map. Once the key is released, the key value is used to retrieve the timestamp of the keypress and compare it to the timestamp of the key's release. If the difference between the two time stamps is below a predetermined threshold, the keystroke is considered inhumanly fast. Durations below the threshold will be printed to the console along with a warning message.

When either metric detects that a keystroke injection has occurred, the program prevents any keyboard input from reaching its destination for a set period of time (e.g. 30 seconds). This complete blocking of keyboard input is possible because of the program's use of a low-level keyboard hook. Since the low-level hook receives keyboard input first, input can be ignored instead of allowing it to travel to its intended application by forcing the program that contains the hook to sleep. Since keystroke injection is an automated process that is very unlikely to detect the input block, the injection will continue execution with no effect on the target machine.

III. Results

The penetration testing tool was tested using two different computers, one using Windows 10 and one using Windows 11. The attack was successful on both machines, leading to emails with working links to be sent to every email listed as a recipient, whether that recipient be recommended by Outlook or explicitly named using the domain lookup approach. The only difficulty presented was the inconsistent time for Outlook to make a suggestion based on the Global Address List, which led to the first approach including long wait times to ensure a suggestion had been made.

The keystroke injection detection program was able to successfully detect and prevent successful execution of both keystroke injection approaches. The detection method that measures keypress duration was able to detect both approaches very early on, meaning that neither approach was able to open Outlook. Both penetration testing approaches were tested 10 times on a Windows 10 and a Windows 11 machine, and the countermeasure detected the attacks 100% of the time. Unfortunately, it is possible for a user to accidentally trigger the keystroke injection detection if the user presses as many keys as possible on his keyboard as quickly as possible, exceeding the WPM value deemed inhuman. During normal use, this false positive is very unlikely, but it does pose an issue for user convenience if false positives do occur. Additionally, pre-programmed keyboard macros may cause the detection program to prevent keyboard input if a macro does not also emulate keypress durations. While some false positives were encountered during testing, no false negatives were observed.

IV. Discussion

The penetration testing tool presented in this paper is a product of a tradeoff between effectiveness and compatibility. While the tool takes a significant amount of time to complete its tasks, its method of attack circumvents many security features present in the newest version of Outlook available on Windows 11 machines. As of April 2025, "new Outlook," as opposed to "classic Outlook," does not support the use of macros written in Visual Basic for Applications (VBA). If macros were supported, the approach to sending emails would be very different. In fact, the attack's approach could have written a .eml file, placed it in the outbox, and then written

a macro to automatically send any emails still present in the outbox when Outlook is opened. This approach would have avoided the need to use Outlook's GUI at all, significantly decreasing the time needed to perform the attack. Additionally, the malicious emails would be sent the next time Outlook is opened, which may be minutes or hours after the penetration tester had left the area. While this approach could be much more effective, it relies on an assumption that the target computer does not use "new Outlook," which is the default Outlook application for Windows 11 distributions. While this approach that utilizes macros would have been incredibly effective, the number of machines that are vulnerable to this attack is much smaller than those vulnerable to the implemented approach.

The penetration testing tool takes advantage of a very common endpoint security issue: employees leaving computers unlocked while unattended. This oversight is especially dangerous in customer-facing roles, where systems are much more exposed to the public. Employees should be trained to lock their computers when they step away, even briefly, to help mitigate threats like keystroke injection and disclosure of sensitive information.

The keystroke injection detection system presented in this paper is an effective countermeasure against all but the most sophisticated of keystroke injection attacks. Some keystroke injection attacks exist that use data from a keylogging attack to replicate a user's behavior and thereby avoid detection by normal means [6]. Detecting these more sophisticated keystroke injection attacks often requires complex behavioral analysis techniques. However, as detection methods become more complex, they also demand greater computing resources, which could potentially divert power away from core user functions. A balanced approach that effectively identifies most injection attempts while minimizing user disruption may offer the best trade-off, depending on the organization's security priorities.

V. Related Works

Since the introduction of keystroke injection, a wide range of countermeasures have been researched, documented, and developed. The simplest mitigation strategies require very little technical work in order to implement. Firstly, employees of any organization, regardless of size or function, should receive cybersecurity training, including guidance on refraining from inserting unknown USB devices into work computers. Secondly, unused USB ports should be disabled, preventing a malicious actor from taking advantage of exposed USB ports [7].

Multiple security measures have been developed that require software packages to be deployed on systems to protect them against keystroke injection. The simplest and most lightweight of these software packages implement detection based on keystroke speed. If keystrokes are entered at a rate that is beyond human ability, the software may alert the user of the threat or block the inputs from reaching their desired destination [6]. More advanced implementations use machine learning to distinguish human-generated keystrokes from machine-generated keystrokes. Some implementations require all users to provide a sample of their typing behavior. This sample will then be used as a baseline for keystroke behavior, and deviations from that baseline will be flagged as potentially malicious [8]. Other implementations use a large, general dataset of human typing behavior, thereby removing the need for every user to provide typing samples. While this approach is more convenient, it prevents the precise analysis afforded by user-specific datasets [9].

VI. Conclusion

In summary, keystroke injection attacks represent a serious vulnerability in endpoint security, particularly when systems are left unlocked and physically accessible. The penetration testing tool developed in this study demonstrates how such attacks can be leveraged to automate phishing attempts from within a trusted environment, bypassing conventional email filters and increasing the likelihood of successful social engineering. The proposed countermeasure effectively detected and blocked the tool's activity in all test cases, demonstrating its viability as a lightweight defense against this class of attack. However, technical controls alone are not sufficient. These findings highlight the importance of adopting a holistic security approach that combines effective detection mechanisms with administrative safeguards, including user training and enforced screen-locking policies, to minimize the risk of compromise.

References

[1] "History of Phishing," KnowBe4 Inc. [Online]. Available: https://www.phishing.org/history-of-phishing

[2] A. Saxena,"100+ phishing attack statistics you must know," Sprinto, Jan. 2025. [Online]. Available: https://sprinto.com/blog/phishing-statistics/

[3] "Federal Bureau of Investigation Internet Crime Report 2023," Internet Crime Complaint Center. [Online]. Available: https://www.ic3.gov/annualreport/reports/2023_ic3report.pdf

[4] "Cost of a Data Breach Report 2024," IBM. [Online]. Available: https://www.ibm.com/reports/data-breach

[5] "Hooks Overview - Win32 apps," learn.microsoft.com. https://learn.microsoft.com/en-us/windows/win32/winmsg/about-hooks

[6] V. Gurčinas, J. Dautartas, J. Janulevičius, N. Goranin, and A. Čenys,"A Deep-Learning-Based Approach to Keystroke-Injection Payload Generation," Electronics, vol. 12, no. 13, p. 2894, Jan. 2023, doi: 10.3390/electronics12132894.

[7] A. D. Ramadhanty, A. Budiono and A. Almaarif, "Implementation and Analysis of Keyboard Injection Attack using USB Devices in Windows Operating System," in 2020 3rd International Conference on Computer and Informatics Engineering (IC2IE), 2020, pp. 449-454, doi: 10.1109/IC2IE50715.2020.9274631.

[8] N. T. Arun Jothi, S. Anu, K. Harsha and R. Devi Priya, "USB Rubber Ducky Hunter A Proactive Defense Against Malicious USB Attacks Domain: Cybersecurity," in 2024 International Conference on Intelligent Systems for Cybersecurity (ISCS), 2024, pp. 1-6, doi: 10.1109/ISCS61804.2024.10581045.

[9] C. Borges, J. Araujo, R. Couto, and A. Almeida. "Keyblock: a software architecture to prevent keystroke injection attacks", in Proceedings of the XVII Brazilian Symposium on Information and Computer Systems Security , 2017, pp. 518-524, doi: 10.5753/sbseg.2017.19526.