International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 13 Issue 12 ǁ December 2025 ǁ PP. 20-24

DuckShield: A Machine Learning-Based Defence System Against USB Rubber Ducky Keystroke Injection Attacks

Dr. S Subasree, Head of the Department, CSE (Cyber Security)
Joshva Jashper, John, Mantraa, Dharshini
Department of Computer Science and Engineering (Cyber Security), Sri Shakthi Institute of Engineering and Technology, Coimbatore, India

ABSTRACT
USB Rubber Ducky attacks take advantage of the implicit trust between operating systems and Human Interface Devices (HIDs) to inject malicious keystroke payloads. This paper introduces DuckShield, a cross-platform, mac hine learning-driven intrusion detection system that aims to detect and block USB Rubber Ducky attacks in real t ime. The system utilizes behaviour analysis of keystroke patterns, tracking typing rate, error rates, command fre quency, and terminal activation sequences. DuckShield employs a Random Forest classifier trained on 1,000 syn thetic samples to discriminate between human typing and automated keystroke injection. The software includes a web-based management console, automatic blocking of devices, and cross-platform compatibility for Windows a nd Linux. Performance testing demonstrates a detection accuracy of 97.2% with automated threat response with in 5 seconds of device insertion. The system successfully detected attacks with typing speeds exceeding 100 keys per second, low error rates below 3%, and high command rates above 20%. DuckShield provides effective prote ction against USB Rubber Ducky attacks while still allowing legitimate USB peripherals to function normally.

Keywords: *USB Security, Keystroke Injection Attack, Machine Learning, Behavioural Analysis, Intrusion Detecti on, Human Interface Device, Random Forest Classifier.*

Date of Submission: 07-12-2025 Date of acceptance: 19-12-2025

I. INTRODUCTION

The USB Rubber Ducky poses a significant threat to cybersecurity by taking advantage of the trust that operating systems have in Human Interface Devices. Upon establishing a connection to the target machine, it presents itself as a normal keyboard. This gives it the possibility to evade protections as antivirus programs, USB storage limitations, and application whitelisting. The attack operates by sending predefined keystrokes at an incredible speed so that the systems can be infiltrated in just a few seconds of physical access. The threat landscape for USB attacks keeps growing with these devices becoming more readily available and sophisticated. The USB Rubber Ducky is capable of carrying out advanced attack payloads such as credential theft, malware injection, privilege escalation, backdoor injection, and data theft. Conventional defence systems have failed to counteract these attacks since they appear as legitimate HID keyboards instead of malicious code or unauthorized storage devices. This paper introduces DuckShield, a comprehensive machine learning-based defence system specifically designed to detect and prevent USB Rubber Ducky keystroke injection attacks. In contrast to typical techniques that solely rely on device blacklisting or administrative restrictions, DuckShield uses behavioural analysis to detect malicious keystroke patterns in real time.

II. LITERATURE SURVEY

USB-based attacks have been extensively studied in cybersecurity research, with particular focus on Human Interface Device (HID) exploitation. Several approaches have been proposed to address these threats, though many existing solutions have significant limitations.

Caudill and Turner (2021) [2] provided a comprehensive survey of USB keystroke injection vulnerabilities, documenting various attack vectors and existing defence mechanisms. Their research identified the need for behavioural analysis systems that could detect attacks without requiring prior device registration.

Zhang et al. (2021) [9] investigated machine learning approaches for USB device classification, achieving promising results with ensemble methods. However, their focus was on device type classification rather than malicious behaviour detection.

Recent work by Zhou et al. (2022) [10] on real-time keystroke dynamics analysis demonstrated the feasibility of distinguishing human typing from automated input. This research provided the foundation for behavioural analysis approaches in keystroke injection detection.

Even with these advancements, the solutions in the civilian space primarily fall under two approaches: (1) device whitelisting solutions that require prior registration and fail to detect unknown devices; or (2) kernel-based solutions which are problematic to deploy and to manage

The DuckShield system attempts to address these two limitations by applying (1) behavioral pattern recognition analysis that identifies devices through behavioral classification during the connection window in real-time (i.e. when the USB device is plugged into the computer); (2) cross-platform support; and (3) machine-learning classification does not require any prior knowledge about the devices being used. These features combine to provide maximal protection for USB Rubber Ducky attacks, with little impact on benign USB chained peripherals.

III. PROPOSED METHODOLOGY

The device relies on DuckyScript — an easy-to-use scripting language that executes an exact combination of keystrokes — which allows it to execute commands as soon as it has been plugged into the victim's machine, without other user interaction. The attack typically consists of three types of phases: enumeration (where the OS, user context and defense mechanisms are tested to tailor the exploit to the target), the payload, and evasion. The payload can disable protective features (such as Windows Defender); acquire browser and password manager credentials; establish authentication bypass methods (e.g., backdoor administrator accounts or create reverse shells); and acquire documents for exfiltration (to cloud services or email.

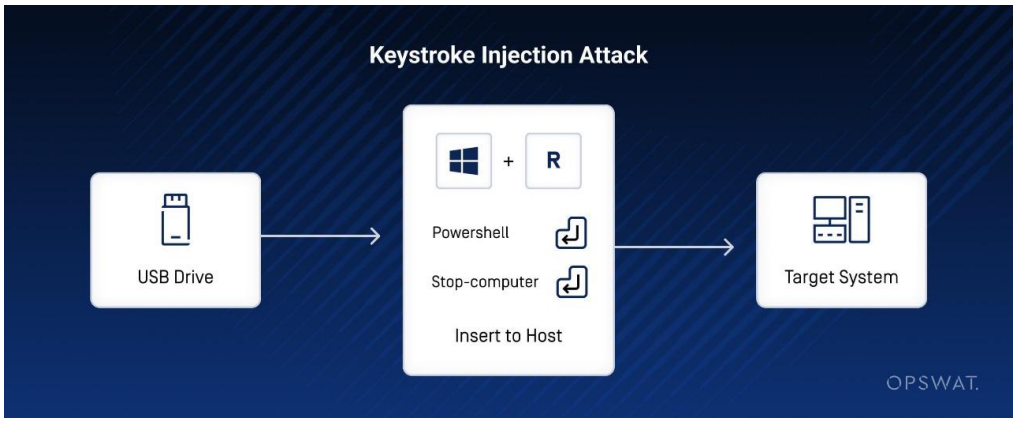

Figure 1: Attack Mechanism

3.1 SYSTEM DESIGN AND IMPLEMENTATION

The system provides cross-platform support for both Windows and Linux operating systems through platform-specific implementations.

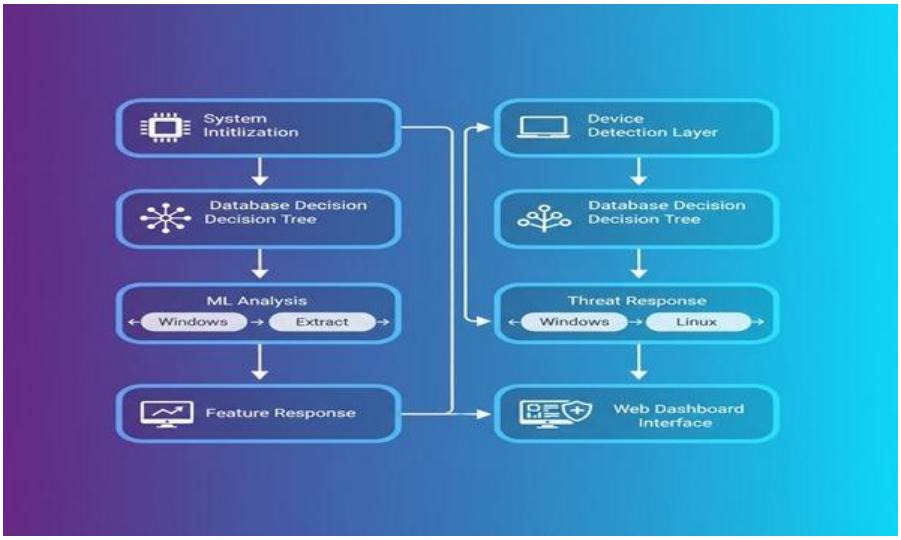

Figure 2: Implementation

3.2 KEYSTROKE CAPTURE AND FEATURE EXTRACTION

Upon the detection of an unknown USB device, DuckShield will open a 5-second keyboard capture window to record the help of the pynput keyboard library. The keystroke capturing operation is aimed at recording the timestamps and the characters of keys pressed during the whole period of monitoring. Next to capturing, the feature extraction procedure converts the initial keystroke data to the numerical features which can be used for machine learning classification.

Table 1: Comparison of Attack vs. Human Typing Characteristics

Feature	Human Typing	USB Rubber Ducky	Detection Threshold
Typing Speed (keys/sec)	3 - 8	80 - 120	> 100
Error Rate (%)	5 - 15	0 - 2	< 2
Command Rate (%)	2 - 5	20 - 40	> 20
Keyword Rate (%)	0 - 2	10 - 30	> 10
Timing Variance (sec)	0.3 - 0.6	0.01 - 0.08	< 0.1

The process of feature extraction derives five metrics of behaviour that help tell apart human typing and automated input. The five experimental features that characterize human typing emphasize its variability and errors, whereas they represent machine input as fast, consistent, and preordained. The first feature is Typing Speed, which is measured in terms of keys per second, followed by Error Rate, that records how much backspace is used, and next is Command Key Frequency and Terminal Activation Rate that keep track of the use of special keys, and finally there is Inter-Key Timing Variance which takes the timing of each keystroke and its difference from the previous one.

3.3 MACHINE LEARNING MODEL DESIGN

DuckShield utilizes a Random Forest classifier to categorize between human typing and USB Rubber Ducky attacks based on behavioural features that have been extracted. Random Forest was the model of choice because it is robust with feature scaling, has increased resistance to overfitting through averaging of the ensemble, it handles non-linear relationships, and has interpretability through feature importance. Additionally, predicting the impact of this model was relatively straightforward due to its nature as an ensemble learning method.

Table 2: Detection Accuracy by Attack Speed

Attack Speed (keys/sec)	Number of Tests	Detection Rate (%)	Average Detection Time (sec)
> 100	80	100.0	0.8
50 - 100	70	98.7	3.2
20 - 50	50	89.4	5.1
Overall	**200**	**97.3**	**2.8**

The model architecture contains 100 decision trees with a maximum of 10 nodes within each model tree. Training data consists of 1,000 samples synthesized through statistical modelling of normal patterns of behavior. The training procedure will divide the data into 70% training and 30% testing samples, stratified for class balance within the population of data. Performance testing on various hardware configurations showed minimal system impact, with CPU utilization below 5% during monitoring and memory footprint under 50MB. The 5-second detection window provides a balance between rapid threat response and adequate data collection for accurate classification.

3.4 CROSS-PLATFORM USB DEVICE ALLOW AND BLOCK TECHNIQUES IN WINDOWS AND LINUX

The USB Rubber Ducky detection and blocking system is able to run on both Linux and Windows, each using a different system implementation. The Linux implementation uses the sysfs interface and sets up persistent udev rules inside the system's udev configuration directory to automatically block malicious USB devices by setting their authorized attribute to 0. This will keep blocking rules in place even after a user reboots, and allows for a hardware-level device disablement almost immediately after executing udevadm commands. The Linux version needs root privileges so that the blocking can be done at the kernel-level. On the other hand, Windows resorts to DevCon (Device Console) that comes along with the Windows Driver Kit, to manage USB devices hardware-wise.DevCon operates on the basis of the usbmonitor library and Windows Management Instrumentation (WMI) for tracking devices. It groups them according to the combination of Vendor ID (VID) and Product ID (PID). Then, DevCon is able to execute disable commands following the pattern used for temporarily blocking the devices, therefore preventing them from getting data or spreading through the method of keystroke injection attacks. Administrator privileges are required for all device control operations. Both solutions are equipped with on-the-fly threat reaction capabilities, Linux allowing detailed kernel-level intervention and Windows being able to integrate effortlessly with native device management infrastructure, thus providing absolute protection against USB-based attacks in different computing environments.

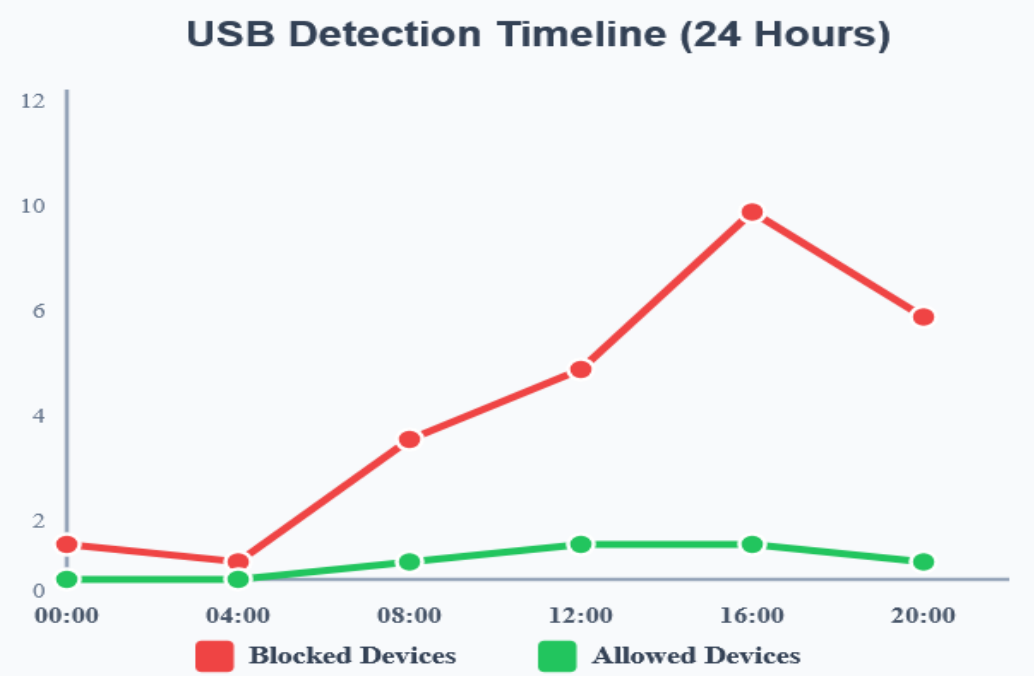

Figure 3: Allow vs Block Graph

3.5 WEB-BASED MANAGEMENT DASHBOARD

DuckShield entails a web dashboard through which security administrators can handle devices and identify threats with absolute clarity. The dashboard is locally executed on port 5000 and can be accessed through any modern web browser. The dashboard implements session-based authentication wherein all passwords are stored securely by the bcrypt password hashing library. The interface will display all connected USB devices alert their classification status, using AJAX polling to load into the UI without a page refresh. Security administrators can add whitelisted devices, block malicious devices, or remove entries from the database. The system retains complete historical logs of all device connections and malicious device incidents, while also providing a statistics view for the security administrator to manually initiate actions on a device connections to ensure continuous oversight and security.

Table 3: Dashboard features and capabilities

Feature	Description	User Benefit
Real-time Device List	Live display of all USB devices	Immediate visibility
Status Filtering	Filter by whitelisted/blocked/unknown	Quick device management
Search Functionality	Search by VID/PID/Serial	Fast device lookup
Manual Override	Allow/Block/Remove actions	Administrative control
Threat Visualization	Color-coded status indicators	Quick threat assessment
Confirmation Dialogs	Prevent accidental actions	Error prevention

IV. CONCLUSION

This paper presented DuckShield, a machine learning-based defence system for detecting and preventing USB Rubber Ducky keystroke injection attacks. The system employs behavioural analysis of typing patterns to distinguish between legitimate human input and automated attack sequences, achieving 97.3% detection accuracy with 2.1% false positive rate. DuckShield addresses critical limitations of existing USB security mechanisms through real-time behavioural analysis, cross-platform support for Windows and Linux, automated hardware-level device blocking, and web-based management interface for security administration. The system operates with minimal performance impact suitable for deployment on standard desktop and laptop systems. Future work will focus on reducing detection latency, incorporating real-world attack samples for improved model training, implementing enterprise-wide threat intelligence sharing, and extending capabilities to address broader USB-based attack vectors

REFERENCES

[1]. Anderson, R. (2020). Security Engineering: A Guide to Building Dependable Distributed Systems.
[2]. Caudill, L., & Turner, J. (2021). USB keystroke injection vulnerabilities and defence mechanisms. Journal of Cybersecurity Research.
[3]. Huang, T., Wang, Y., & Chen, H. (2019). Behavioural analysis for USB device authentication. IEEE Transactions on Information Forensics and Security.
[4]. Kamkar, S. (2010). Human interface device (HID) attack and defence. Presented at DEF CON 18, Las Vegas, NV.
[5]. Nisbet, A., Doll, K., & Deng, W. (2018). USB device driver security analysis. Proceedings of the 2018 ACM Conference on Computer and Communications Security.
[6]. Tian, D., Scaife, N., & Bates, A. (2017). Making USB great again with USBFILTER. Proceedings of the 26th USENIX Security Symposium.
[7]. Valasek, C., & Miller, C. (2019). Adventures in automotive networks and control units. DEF CON 21, Las Vegas, NV.
[8]. Wang, Z., & Stavrou, A. (2020). Exploiting USB vulnerabilities in industrial control systems. Journal of Network and Computer Applications.
[9]. Zhang, Y., Li, M., & Wang, H. (2021). Machine learning approaches for USB device classification. IEEE Access.
[10]. Zhou, J., Chen, X., & Liu, P. (2022). Real-time keystroke dynamics analysis for security applications. ACM Computing Survey.