Windows Registry Change Monitoring System

is a project submitted in partial fulfillment of the requirements for the

Internship Role: Cyber Security Intern

Organization: Unified Mentor

Prepared By: Tanisha Sagar

Project Type: Defensive Cyber Security / Blue Team Project

Duration of Internship: 3 months

INDEX

INTRODUCTION

In Windows operating systems, the registry acts as a centralized database that stores configuration settings for the OS, applications, users, and security components. Many malware families abuse registry keys such as *Run*, *RunOnce*, and security policy paths to maintain persistence or disable protection mechanisms.

Manual monitoring of registry changes is impractical on real systems. This project demonstrates how registry monitoring can be automated using Python, allowing early detection of suspicious changes that may indicate malware behavior.

By implementing baseline creation and periodic comparison, this project simulates how real-world security tools detect unauthorized persistence mechanisms.

PROJECT OVERVIEW

This project focuses on building a Python-based system to monitor critical Windows Registry locations for suspicious or unauthorized changes. The Windows Registry plays a major role in system configuration, startup behavior, and security policy enforcement. Because of this, it is frequently targeted by malware to establish persistence or weaken system defenses.

The project uses a baseline comparison approach combined with continuous monitoring to detect registry changes related to startup persistence and Windows Defender policies. Any new, modified, or deleted registry entry is logged with a timestamp, making it easier to track potentially malicious activity.

The system is designed for defensive and monitoring purposes and follows safe practices without exploiting or modifying the system beyond controlled test values.

PROBLEM STATEMENT

Malware often modifies the Windows Registry to:

- Execute automatically on system startup
- Bypass or weaken security controls
- Maintain long-term persistence without user awareness

Traditional antivirus solutions may not always highlight *how* and *where* persistence was achieved. There is a need for a lightweight monitoring solution that can clearly identify registry changes and provide visibility into system-level modifications.

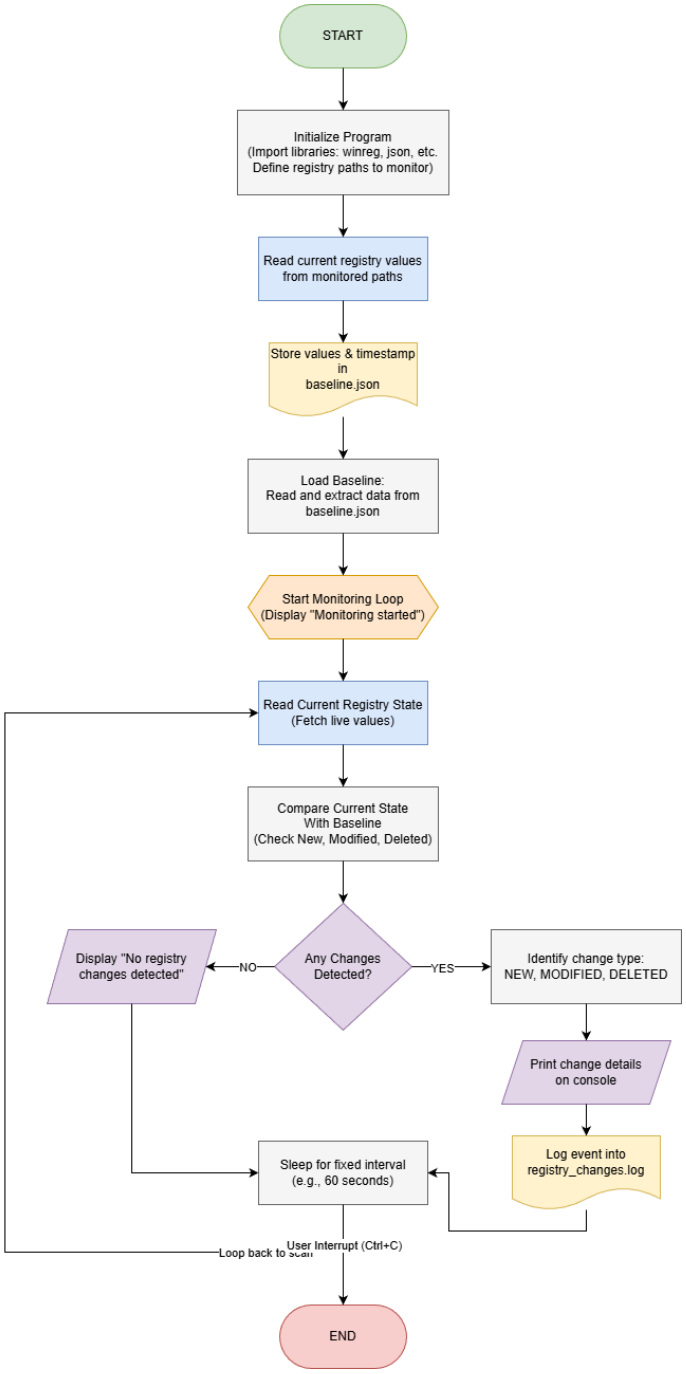
This project addresses that gap by implementing a transparent and explainable registry monitoring system.

PROJECT OBJECTIVES

The main objectives of this project are:

- To monitor critical Windows Registry locations commonly abused by malware
- To create a baseline of trusted registry values
- To detect newly added, modified, or deleted registry entries
- To log detected changes with timestamps for analysis
- To understand registry-based persistence techniques used by malware

SYSTEM ARCHITECTURE



Flowchart 1: Flowchart illustrating the working of the Windows Registry Change Monitoring System

SCOPE OF THE PROJECT

The scope of this project is limited to monitoring selected registry paths related to:

- Startup persistence (`Run` and `RunOnce`)
- Windows Defender security policies

The project does not include:

- Active malware removal
- Kernel-level monitoring
- Registry restoration or rollback

The system is intended for learning, detection, and monitoring purposes only.

TOOLS AND ENVIRONMENT USED

The following tools and environment were used:

- **Python 3.x** – Core programming language
- **winreg module** – To read Windows Registry values
- **Windows 10 / 11 (64-bit)** – Testing environment
- **Command Prompt / PowerShell** – Script execution
- **GitHub** – Version control and project hosting
- **VS Code**– Visual Studio Code (VS Code) was used as the primary development environment for writing, testing, and debugging Python scripts.
- **draw.io** – Used for designing
- flowcharts and architecture diagrams.

METHODOLOGY

The project follows a structured workflow:

1. Identify registry paths commonly targeted by malware
2. Create a baseline snapshot of registry values
3. Store baseline data in a JSON file
4. Periodically read current registry values

5. Compare current values with the baseline
6. Detect and classify changes as:
    - New
    - Modified
    - Deleted
7. Log all detected changes with timestamps
8. Continuously monitor until stopped by the user

Registry Paths Monitored

The following registry paths are monitored:

- `HKCU\Software\Microsoft\Windows\CurrentVersion\Run`
- `HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce`
- `HKLM\Software\Microsoft\Windows\CurrentVersion\Run`
- `HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce`
- `HKLM\Software\Policies\Microsoft\Windows Defender`

These locations are commonly used for startup persistence and security configuration changes.

ANALYSIS EVIDENCE AND OBSERVATIONS

This section documents the practical execution of the project, along with evidence collected during testing. Screenshots are included to demonstrate baseline creation, registry changes, and detection results.

1. Baseline Creation Analysis

Initially, a baseline of selected Windows Registry keys was created to represent the normal and trusted state of the system.
The baseline acts as a reference point against which all future registry changes are compared.

The following registry locations were included in the baseline:

- Startup persistence keys (`Run` and `RunOnce`)
- Security policy keys related to Windows Defender

The baseline was generated using the `create_baseline.py` script. When executed, the script successfully scanned the configured registry paths and stored all existing values in a JSON file (`baseline.json`) along with a timestamp.

Observation:

- The baseline file accurately captured existing startup applications such as system services and trusted software.
- No security or startup alerts were generated at this stage, confirming that the baseline represents a clean state.

Fig 1: Command Prompt showing `python create_baseline.py` execution

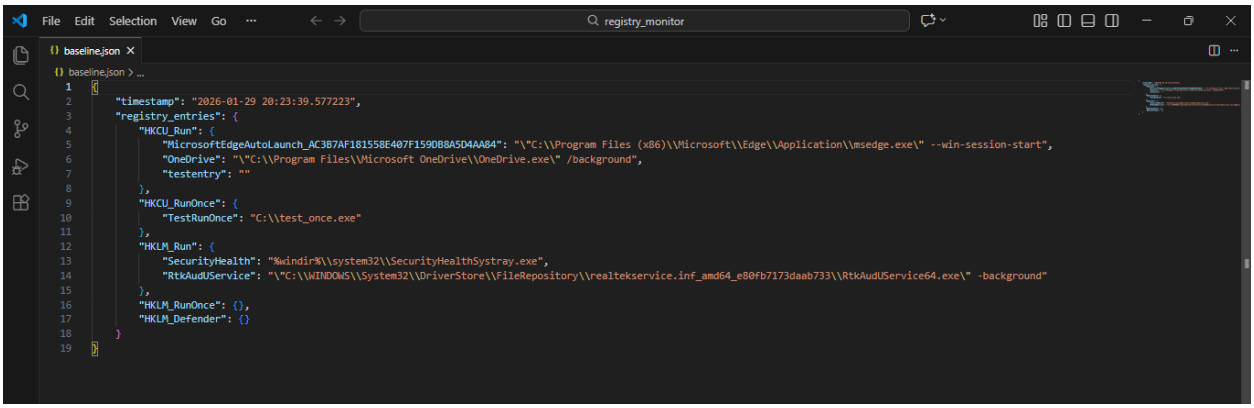


Fig 2: `baseline.json` opened in VS Code showing stored registry entries

2. Startup Persistence Detection (Run & RunOnce Keys)

To test startup persistence detection, a new registry entry was manually added to the `Run` and `RunOnce` registry locations using the Windows Registry Editor.

After creating the new registry value, the `compare_registry.py` script was executed. The script compared the current registry state with the previously stored baseline.

Observation:

- The system successfully detected newly added startup entries.
- The detection output clearly identified the registry location and entry name.
- The event was logged as a **NEW** entry in the log file.

This confirms that the system can effectively identify persistence mechanisms commonly used by malware to execute on system startup.

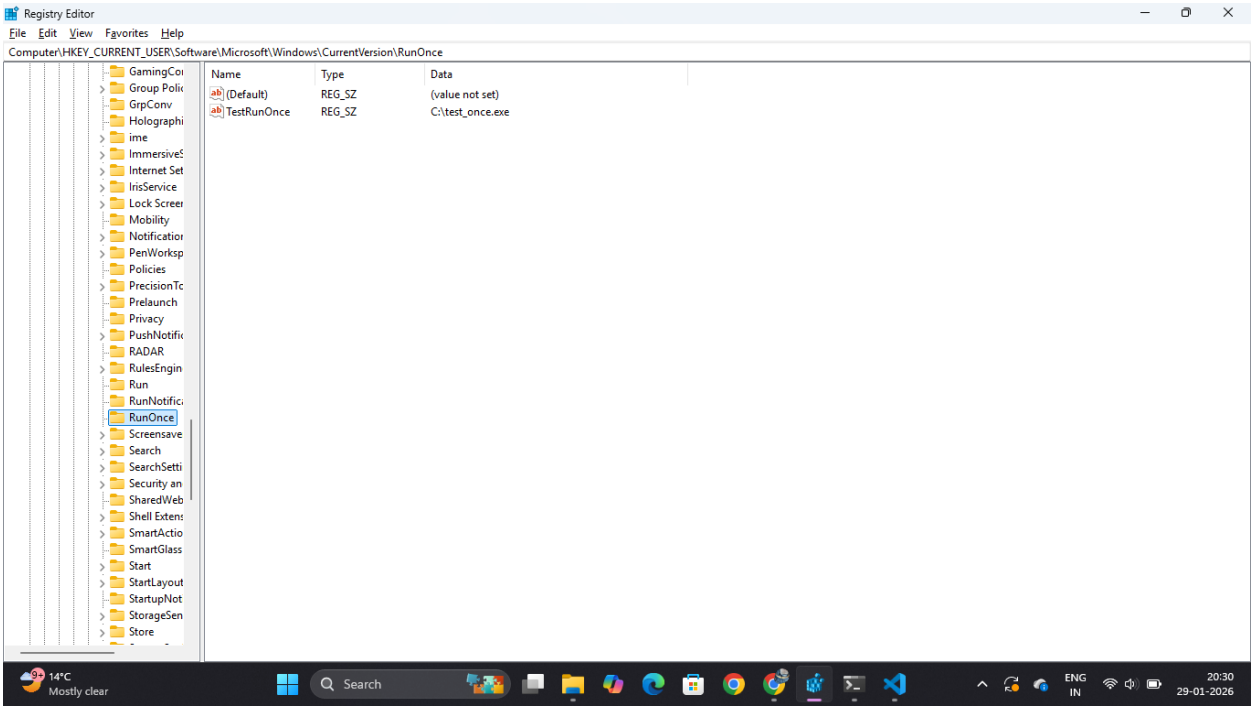


Fig 3: Registry Editor showing the newly added Run / RunOnce entry

Fig 4: Command Prompt output displaying `[NEW] HKCU_RunOnce -> TestRunOnce`

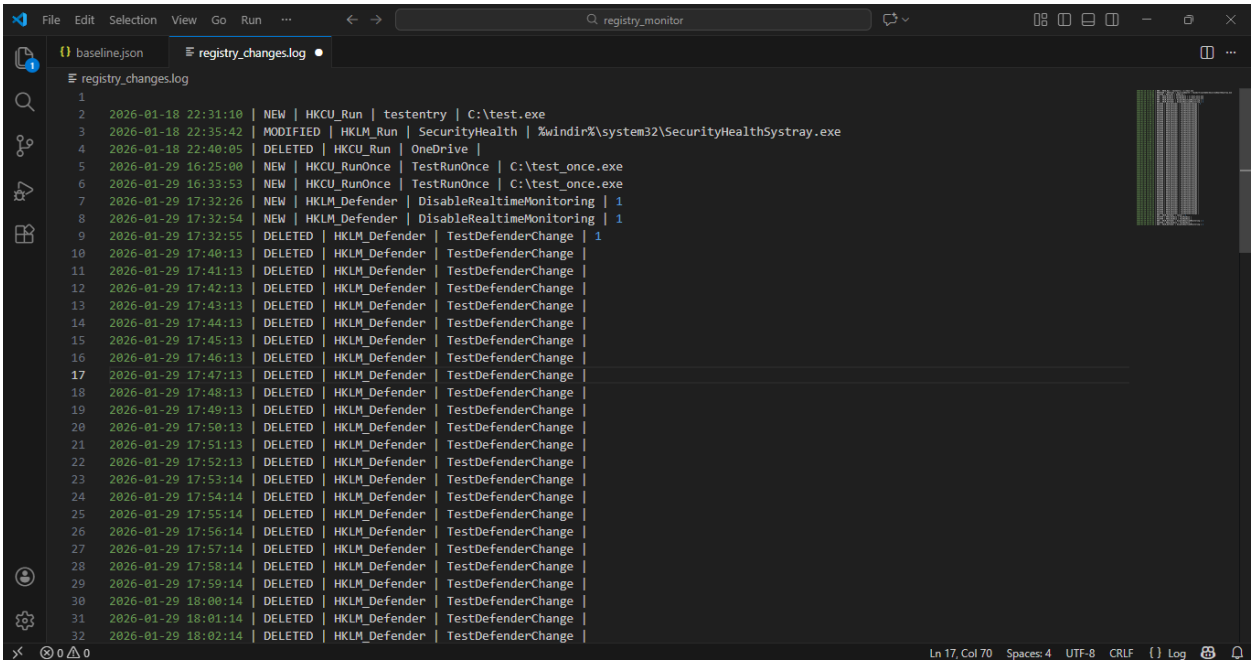


Fig 5: `registry_changes.log` *showing the logged event*

3. Security Policy Monitoring (Windows Defender Keys)

To validate security policy monitoring, a test registry value was added under the Windows Defender policy path:

`HKLM\Software\Policies\Microsoft\Windows Defender`

This simulated a change that could potentially weaken system security, such as disabling real-time protection.

Upon running the monitoring script, the system detected:

- The creation of a new Defender-related registry value
- Subsequent deletion of the same value

Observation:

- Both **NEW** and **DELETED** events were accurately detected.
- Changes under security-sensitive registry paths were logged correctly.
- This demonstrates the system's ability to monitor security configurations that are commonly targeted by malware.

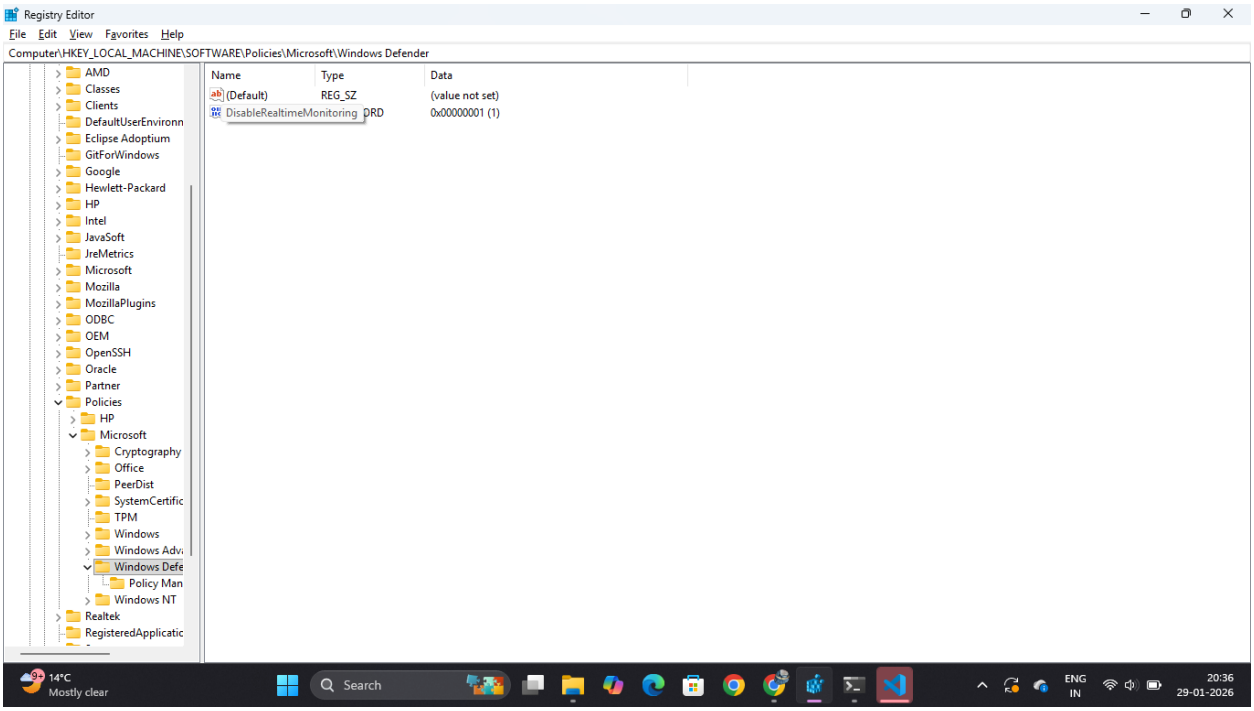


Fig 6: Registry Editor showing the Defender test value

```
D:\Cybersecurity_Projects\registry_monitor>python compare_registry.py
[*] Starting continuous registry monitoring (Ctrl+C to stop)

Checking HKCU_Run...

Checking HKCU_RunOnce...
[DELETED] HKCU_RunOnce -> TestRunOnce

Checking HKLM_Run...

Checking HKLM_RunOnce...

Checking HKLM_Defender...
[NEW] HKLM_Defender -> DisableRealtimeMonitoring
```

Fig 7: Command Prompt output showing [NEW] HKLM_Defender -> DisableRealTimeMonitoring and [DELETED] HKLM_Defender -> DisableRealTimeMonitoring

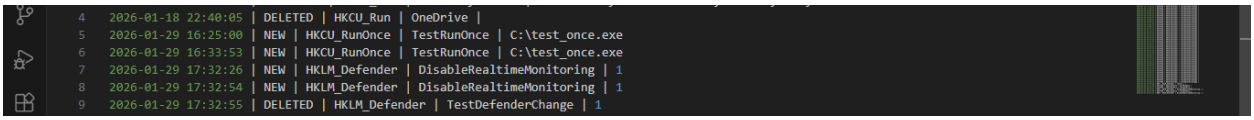


Fig 8: Corresponding log entries in registry_changes.log

4. Continuous Monitoring Behavior

The project also supports continuous registry monitoring using a loop-based mechanism. Once started, the script repeatedly checks the registry at fixed intervals and reports any detected changes in real time.

The monitoring process continues until manually stopped using `Ctrl + C`.

Observation:

- Continuous monitoring worked reliably without crashes.
- Pressing `Ctrl + C` raised a `KeyboardInterrupt`, which is expected behavior and confirms controlled termination.
- The script resumed correct operation when restarted.

```
D:\Cybersecurity_Projects\registry_monitor>python compare_registry.py
[*] Starting continuous registry monitoring (Ctrl+C to stop)
```

Fig 9: Command Prompt showing continuous monitoring running and termination using `Ctrl+C`

5. Log File Analysis

All detected registry changes were recorded in the `registry_changes.log` file. Each entry contains:

- Timestamp
- Type of change (NEW / MODIFIED / DELETED)
- Registry location
- Registry value name
- Associated data (if applicable)

Observation:

- Log entries were consistent with console output.
- The log file provides a reliable audit trail for forensic or investigation purposes.

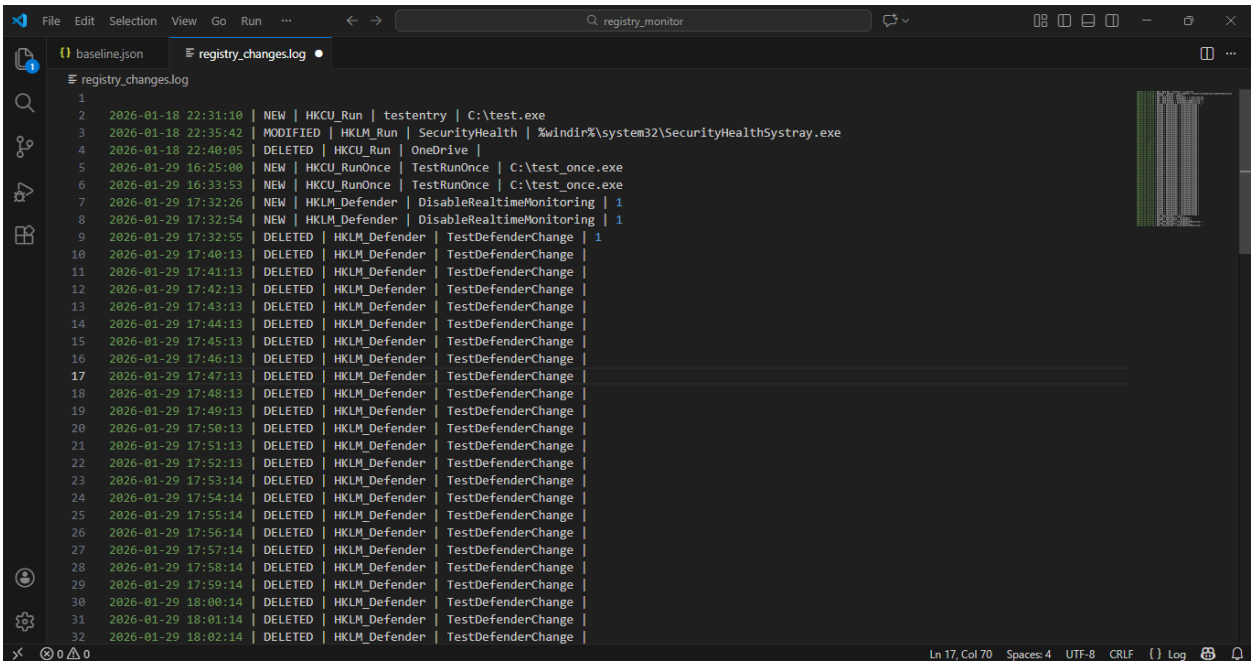


Fig 10: registry_changes.log opened in VS Code showing multiple entries

Overall Observations

- The system correctly detects startup persistence attempts.
- Security-related registry modifications are accurately identified.
- Baseline comparison reduces false positives.
- Continuous monitoring allows real-time detection of suspicious activity.

These results confirm that the implemented system meets the objectives of detecting registry-based persistence and security policy changes on a Windows system.

Risk Assessment

Registry-based persistence poses a **high security risk** because:

- Changes can survive system reboots
- Startup entries execute without user interaction
- Security policies can be silently modified

Early detection of such changes is critical in preventing long-term compromise.

Challenges Faced

Some challenges encountered during the project include:

- Understanding correct registry paths for Windows Defender policies
- Handling permission-related access issues
- Debugging baseline comparison logic
- Managing GitHub repository structure correctly

These challenges helped improve troubleshooting and system-level understanding.

Ethical and Safety Considerations

- No real malware was used in this project
- Registry modifications were limited to controlled test values
- No system security features were permanently altered
- The project was developed strictly for educational purposes

LEARNING OUTCOMES

Through this project, I gained:

- Practical understanding of Windows Registry structure
- Knowledge of registry-based malware persistence techniques
- Hands-on experience with Python system programming
- Experience with baseline comparison detection logic
- Better understanding of defensive security concepts

Future Scope

Possible future enhancements include:

- Real-time event-based monitoring instead of polling
- Email or alert notifications for detected changes
- Registry rollback functionality

- Integration with threat intelligence feeds
- GUI-based monitoring dashboard

CONCLUSION

This project demonstrates how registry monitoring can be used as an effective defensive security technique to detect suspicious system modifications. By implementing baseline comparison and continuous monitoring, the system successfully identifies changes that could indicate malware persistence or security policy tampering.

The project highlights the importance of visibility and monitoring in cybersecurity and provides a strong foundation for more advanced endpoint detection mechanisms.

Bibliography and References

1. Microsoft Windows Registry Documentation
2. Python `winreg` Module Documentation
3. MITRE ATT&CK – Persistence Techniques
4. Malware Persistence Research Articles
5. draw.io – Online diagramming tool used for system flowchart design