# REGWATCH SENTINEL

# Host-Based Windows Registry Persistence Detection System

**Domain:** Cybersecurity / Blue Team / Web Security
**Type:** Practical Security Project
**Submitted by:** *Chirayu Paliwal*
**Duration:** Internship Practical Assignment
**Tools & Technologies:** Python, Windows Registry APIs, VS Code, PowerShell

## 1. INTRODUCTION

Modern cyber intrusions rarely rely on loud or destructive techniques in their early stages. Instead, attackers focus on persistence—ensuring their access survives reboots, logouts, and user activity. One of the most abused persistence mechanisms in Windows environments is the Windows Registry, particularly auto-start locations such as Run and RunOnce keys.

The Windows Registry is a powerful configuration database intended for legitimate system and application behavior. However, attackers frequently exploit this trust by registering malicious executables to launch automatically at user logon or system startup. These modifications often appear to benign and are overlooked by traditional antivirus tools that rely on static signatures.

REGWATCH SENTINEL is a lightweight, behavior-focused registry monitoring system designed to detect suspicious registry-based persistence mechanisms. The project simulates how a SOC analyst or endpoint detection tool would observe registry modifications, evaluate contextual risk, map behavior to attacker techniques, and generate an explainable detection report.

## 2. OBJECTIVE OF THE PROJECT

- To monitor critical Windows Registry locations associated with persistence.
- To detect suspicious auto-start registry entries.
- To analyze executable paths and contextual indicators of abuse.
- To apply rule-based risk scoring aligned with SOC detection logic.
- To map detected behaviors to MITRE ATT&CK persistence techniques.
- To generate structured, analyst-readable detection reports.

### 2.1. PROBLEM STATEMENT

The Windows Registry was never designed as a security boundary—it is a configuration store. Attackers take advantage of this by placing malicious entries in legitimate auto-run locations, allowing them to persist without dropping obvious malware or triggering alarms.

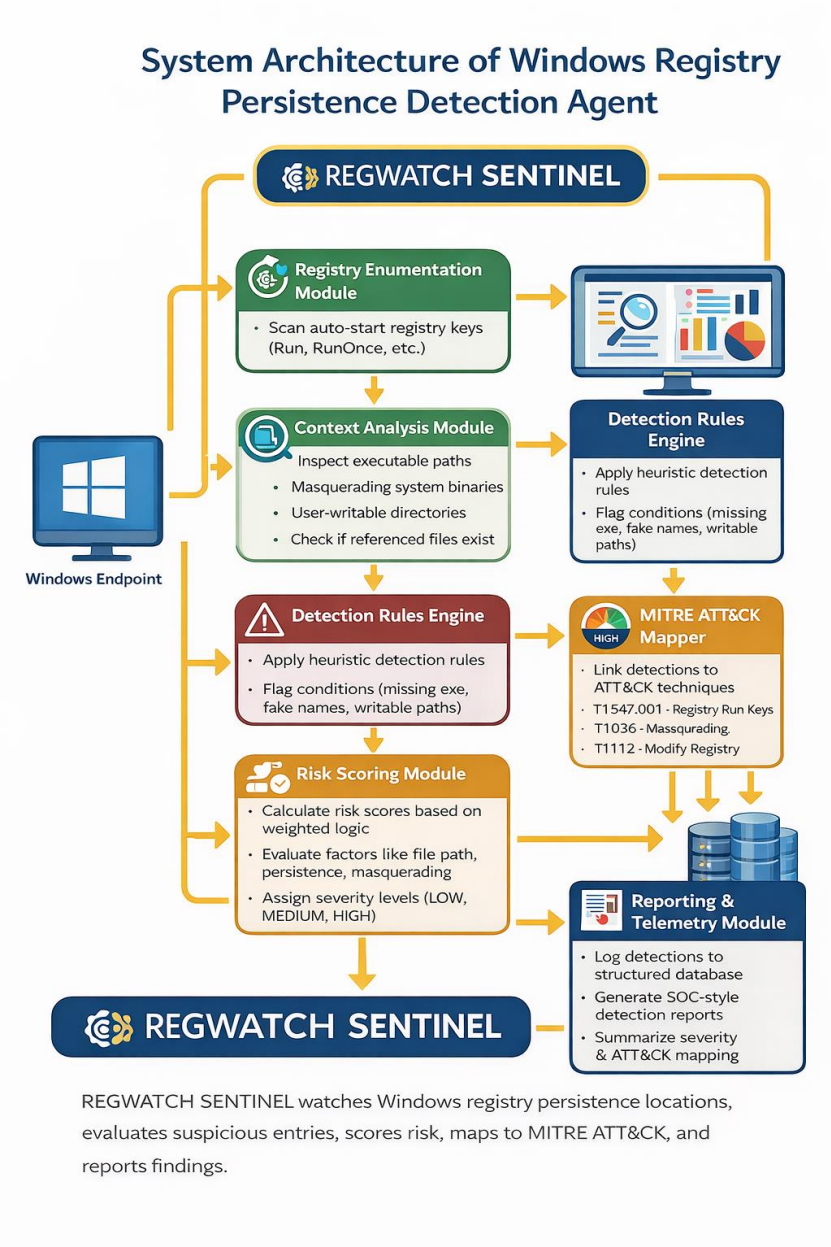
Beginner detection tools often fall into two traps:

1. They flag every registry entry as suspicious, creating noise.
2. They detect registry changes but fail to explain *why* those changes are dangerous.

This project addresses that gap by focusing on **context-aware detection**. Instead of treating registry activity as inherently malicious.

REGWATCH SENTINEL evaluates *how* registry entries are used, *where* executables are located, and *whether* they resemble known attacker tradecraft.

## 3. ARCHITECTURE




REGWATCH SENTINEL follows a modular detection pipeline inspired by real-world SOC and EDR tools.

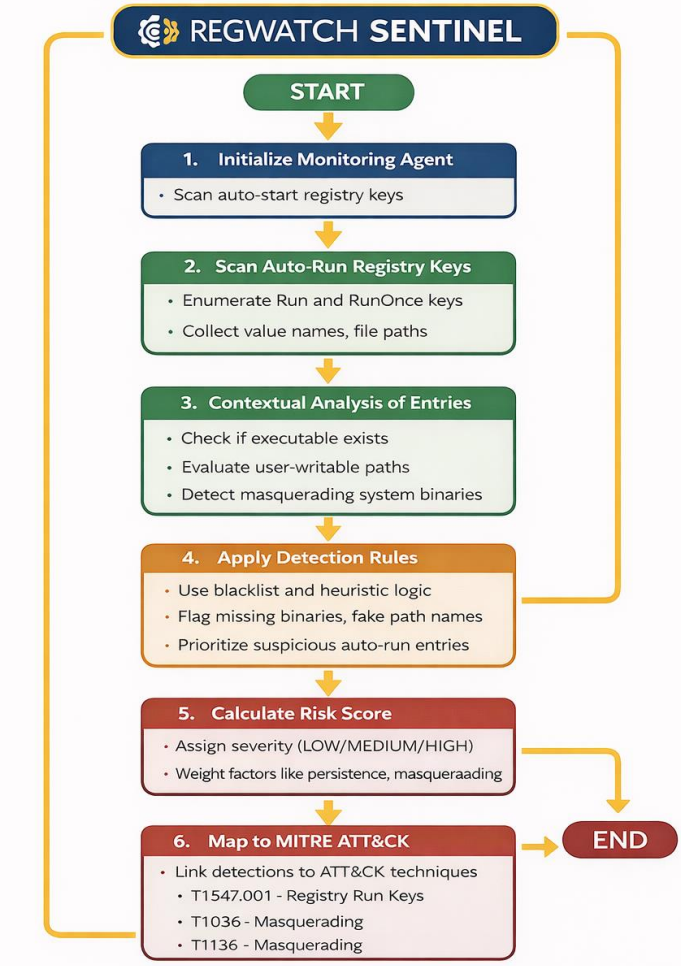
The system begins by scanning known registry locations commonly abused for persistence. These include user-level and system-level auto-run keys. Once entries are collected, the system evaluates each entry using contextual checks such as executable path legitimacy, file existence, privilege scope, and masquerading behavior.

The detection engine applies rule-based logic to assign severity and risk scores. Finally, events are logged and compiled into a structured detection report that mirrors analyst-facing alerts.

## 3.1 ARCHITECTURE COMPONENTS

- **Registry Enumeration Module**
  Scans predefined Windows Registry auto-start locations.
- **Context Analysis Module**
  Analyzes executable paths, user-writable locations, and masquerading indicators.
- **Detection Rules Engine**
  Applies heuristic and blacklist-based rules to registry entries.
- **Risk Scoring Module**
  Assigns severity and numerical risk scores based on multiple indicators.
- **MITRE ATT&CK Mapper**
  Maps detected behavior to relevant persistence and defense-evasion techniques.
- **Reporting & Telemetry Module**
  Stores events and generates a SOC-style detection report.

## Detection Workflow and Logic Flow




REGWATCH SENTINEL

REGWATCH SENTINEL scans auto-run registry keys, analyzes suspicious entries in context, scores the risk, maps to MITRE ATT&CK, and generates an analyst-readable report.

## 4. METHODOLOGY

### *Step 1: Registry Monitoring*

- Enumerated Windows Registry Run and auto-start keys.
- Collected value names, data paths, registry hives, and associated users.

### *Step 2: Executable Context Analysis*

- Checked whether referenced executables exist on disk.
- Analyzed file locations for user-writable directories.
- Identified masquerading attempts using system binary names.

### Step 3: Persistence Detection

- Flagged registry entries that enable automatic execution.
- Distinguished user-level vs system-level persistence.

### *Step 4: Rule-Based Detection*

- Applied detection rules for:
    - Missing executables
    - Suspicious file paths
    - Masquerading system binaries
    - Auto-run persistence behavior

### *Step 5: Risk Scoring*

- Assigned severity levels (LOW / MEDIUM / HIGH).
- Calculated cumulative risk scores based on multiple indicators.

### *Step 6: MITRE ATT&CK Mapping*

Mapped detections to techniques such as:

- **T1547.001 –** Boot or Logon AutoStart Execution: Registry Run Keys
- **T1036 –** Masquerading
- **T1112 –** Modify Registry
- **T1105 –** Ingress Tool Transfer (contextual correlation)

### *Step 7: Reporting*

Generated a structured final detection report summarizing:

- Events
- Severity distribution
- Risk explanations
- MITRE ATT&CK coverage

## 5. TOOLS & TECHNOLOGIES USED

| Tool / Technology | Purpose |
|---|---|
| **Python 3.14** | Core detection and analysis logic |
| **Windows OS** | Target environment |
| **Windows Registry APIs** | Registry enumeration |
| **JSON** | Rule definitions and telemetry storage |
| **MITRE ATT&CK** | Technique mapping |
| **Visual Studio Code & POWERSHELL** | Development environment<br>Testing and validation |

## 6. Threat Model and Assumptions

### Attacker Assumptions

- Attacker has user or limited admin access.
- Uses registry-based persistence instead of obvious malware.
- Places payloads in user-writable directories.
- Attempts to masquerade as legitimate system binaries.

### Defender Assumptions

- User-space visibility only.
- No kernel or memory inspection.
- Focused on detection and analysis, not prevention.

## 7. Detection Logic Rationale

REGWATCH SENTINEL does not assume registry entries are malicious by default. Instead, it evaluates **the context**.

- Registry persistence increases risk but is not sufficient alone.
- Executables in user-writable paths increase suspicion.
- Masquerading system binaries significantly elevate risk.
- Missing executables strongly indicates malicious or broken persistence.

This mirrors detection logic used in enterprise-grade EDR platforms.

## 8. Risk Scoring Philosophy

Risk scoring is based on a weighted model considering:

- Presence of registry-based persistence
- Executable legitimacy
- File path trust level
- Masquerading indicators
- Combined behavioral signals

This approach avoids binary decisions and helps analysts prioritize alerts.

## 9. Observations

```
=== REGWATCH SENTINEL FINAL DETECTION REPORT ===

Total Events Logged: 2

Severity Distribution:
  HIGH: 2

Top High-Risk Events:

-------------------------------------------
Registry Path : HKCU\Software\Microsoft\Windows\CurrentVersion\Run
Command       : C:\Users\chira\AppData\Roaming\svchost.exe
Value Name    : TestPersistence
User          : chira
Host          : BARBELL
OS            : Windows-11-10.0.26200-SP0
Risk Score    : 100
Reasons:
  - Referenced executable does not exist on disk
  - Autorun registry persistence
  - Executable in user-writable directory
  - Masquerading as Windows system binary
MITRE        : T1036, T1105, T1112, T1547.001

-------------------------------------------
Registry Path : HKLM\Software\Microsoft\Windows\CurrentVersion\Run
Command       : C:\Users\chira\AppData\Roaming\svchost.exe
Value Name    : TestPersistence
User          : chira
Host          : BARBELL
OS            : Windows-11-10.0.26200-SP0
Risk Score    : 100
Reasons:
  - Referenced executable does not exist on disk
  - Autorun registry persistence
  - Executable in user-writable directory
  - Masquerading as Windows system binary
  - System-wide persistence
MITRE        : T1036, T1105, T1112, T1547.001
```

During testing, REGWATCH SENTINEL successfully detected:

- Registry entries pointing to non-existent executables.
- Persistence mechanisms using user-writable directories.
- Masquerading binaries named after Windows system processes.
- High-confidence registry-based persistence attempts.

## 10. RESULTS

The system:

- Identified high-risk registry persistence events.
- Generated structured, explainable detection reports.
- Provided clear reasoning behind each alert.
- Successfully mapped behaviors to MITRE ATT&CK techniques.

## 11. False Positives and Analyst Triage

### Expected False Positives

- Legitimate startup programs.
- Custom user automation scripts.
- Developer or admin auto-run tools.

### How REGWATCH SENTINEL Handles This

- Context-aware scoring instead of hard blocking.
- Detailed explanations included in reports.
- MITRE mapping to support analyst judgment.

## 12. CONCLUSION

Through the development of REGWATCH SENTINEL, I gained a deep understanding of how attackers abuse the Windows Registry to maintain persistence in real-world environments.

This project taught me that registry-based persistence is rarely obvious and often hides behind legitimate-looking entries. I learned how to analyze registry data in context, rather than treating every auto-run key as malicious. Evaluating executable paths, file existence, and masquerading behavior helped me understand how attackers blend into normal system activity.

Implementing rule-based detection and weighted risk scoring strengthened my ability to think like a SOC analyst—prioritizing alerts instead of making binary decisions. Mapping detections to the MITRE ATT&CK framework improved my understanding of how low-level registry changes relate to broader attacker tactics and techniques.

Overall, REGWATCH SENTINEL significantly enhanced my blue team skills. I learned how to design explainable persistence detections, analyze Windows internals from a defender's perspective, and generate reports that support analyst decision-making. This project closely mirrors real-world endpoint security monitoring and SOC operations.