WinHostBaseline – Quickstart Guide

1. Purpose and Overview

WinHostBaseline is a PowerShell-based host integrity toolkit that:

- Captures a **baseline snapshot** of key persistence and execution surfaces.
- Compares current state against that baseline to detect **drift** and **suspicious changes**.
- Produces a **ranked findings** list to focus analyst attention.

It targets:

- Services, drivers, processes
- Scheduled tasks, startup items
- Browser extensions
- Network sockets
- WMI persistence

All outputs are deterministic and JSON-based, suitable for automation and ingestion.

2. Prerequisites

PowerShell:

- Windows PowerShell 5.1 (supported and tested)

Permissions:

- Recommended: **Run as Administrator**
  - Required for:
    - Loading all user hives (`-AllProfiles`)
    - Full WMI coverage
    - Some registry locations

Directory Layout (defaults):

- Baselines: `C:\Temp\Scan\Ref\`
- Reports: `C:\Temp\Scan\Output\`

You can override these with `-RefDir` and `-OutputDir`.

3. Generating a Baseline

From an elevated PowerShell prompt in the script directory:

powershell

```
.\WinHostBaseline.ps1 -Mode Baseline
```

Or shorthand:

powershell

```
.\WinHostBaseline.ps1 1
```

Optional Flags

- `-AllProfiles` Load all local user hives and scan `Run/RunOnce` keys for each profile.
- `-ResolveNetworkPaths` Allow resolution of UNC/mapped paths (may hang on offline shares).
- `-Firefox` Include Firefox extension enumeration.

Example:

powershell

```
.\WinHostBaseline.ps1 -Mode Baseline -AllProfiles -Firefox
```

This writes a baseline JSON file like:

text

```
C:\Temp\Scan\Ref\Win11_10.0.22631_3880_HOSTNAME.json
```

The `Meta.SchemaVersion` is set to `2` for this build.

4. Running a Comparison

From the same directory:

powershell

```
.\WinHostBaseline.ps1 -Mode Compare
```

Or shorthand:

powershell

```
.\WinHostBaseline.ps1 2
```

You will be prompted to select a baseline (newest first). The script then:

1. Runs all collectors again.
2. Normalizes and hashes items.
3. Compares against the selected baseline.
4. Scores Added/Changed/Removed items.

Output report:

text

```
C:\Temp\Scan\Output\compare-report-HOSTNAME-YYYYMMDDTHHMMSSZ.json
```

Socket Collection Note

- Default: **LISTEN-only** TCP + all UDP endpoints (low noise).
- Use `-IncludeRemoteSockets` to include full TCP connections.

5. Understanding the JSON Report

The comparison report has three key sections:

5.1 Meta

Contains:

- `SchemaVersion` (2)
- `ComparedAtUtc`
- `BaselinePath`
- `ComputerName`
- `IncludeRemoteSockets`
- `CollectorErrors` (if any)

Use this to:

- Confirm which baseline was used.
- Check if any collectors failed or degraded.

5.2 Results

Per-category objects:

- `Category`
- `Added` – items present now but not in baseline
- `Removed` – items present in baseline but not now
- `Changed` – same key, different details
- `Collisions` – key duplication (should be rare)

Categories include:

- `Services`, `Drivers`, `Processes`
- `ScheduledTasks`, `Startup`
- `BrowserExtensions`
- `NetworkSockets`
- `WMI.Filter`, `WMI.Consumer`, `WMI.Consumer.ActiveScript`, `WMI.Binding`

5.3 RankedFindings

A flattened, sorted list of:

- `Category`
- `Type` (Added / Changed / Removed)
- `Key`
- `Score`
- `Item` (full object)

This is your triage list—start at the top and work down.

6. Recommended Operational Workflows

6.1 Build Gold Baselines

- Use **clean, known-good systems**:
  - Fresh images
  - Gold builds
  - Hardened reference hosts
- Generate baselines and store them in **version control** (e.g., Git).

6.2 Scheduled Comparisons

- Use Task Scheduler, EDR, or RMM to:
  - Run comparisons daily/weekly.
  - Collect JSON reports centrally.
  - Alert on high-scoring findings.

6.3 Version Control for Baselines

- Commit baseline JSON files:
  - Track drift over time.
  - Compare baselines across OS versions.
  - Maintain audit history for change approvals.

6.4 Re-Baselining

Re-generate baselines when:

- Major OS updates are applied.
- Large software deployments occur.
- You intentionally change startup/WMI/task configurations.

7. Troubleshooting

7.1 “No baseline files found”

- Check `RefDir` path.
- Ensure you’ve run `-Mode Baseline` at least once.
- Confirm permissions to read the directory.

7.2 CollectorErrors in Meta

Common causes:

- Access denied (run elevated).
- WMI class not available.
- Registry key missing or locked.

Action:

- Inspect `Meta.CollectorErrors` in the baseline or report.
- Use that to identify which collector and why it failed.

7.3 Long-Running or Hanging

- If network paths are slow/unreachable:
  - Avoid `-ResolveNetworkPaths` unless necessary.
- Directory hashing is capped:
  - Large extension folders are handled with limits and a `__TRUNCATED__` marker.

7.4 High Noise in Results

- Ensure you’re comparing against the **correct baseline** (same OS build + hostname).
- Re-baseline after legitimate patch cycles.
- Focus first on:
  - WMI persistence
  - Startup
  - Scheduled tasks
  - Services/drivers with signature changes