Wof.sys BSOD 0x139 — Full Diagnostic Report

Wof.sys BSOD — Feedback Hub Report

Bug Summary

Wof.sys (Windows Overlay Filter) causes KERNEL_SECURITY_CHECK_FAILURE (0x139) during NTFS directory enumeration when Compact OS is enabled.

Crash Details

- **Date**: 2026-03-09 16:17 (UTC+2)
- **Bugcheck**: `0x00000139` (KERNEL_SECURITY_CHECK_FAILURE)
- **Parameter 1**: `0x0a` — Indirect call guard check detected invalid control transfer (CFG violation)
- **Failure bucket**: `0x139_a_GUARD_ICALL_CHECK_FAILURE_Wof!WofPreDirectoryControlCallback+0x695`
- **Failure hash**: `{78ac0991-5393-0281-b23b-ce115373dc6d}`
- **Faulting module**: `Wof.sys` version `10.0.26100.5074`
- **Triggering process**: `claude.exe` (Anthropic Claude Code CLI)
- **Triggering syscall**: `NtQueryDirectoryFileEx` (standard directory listing)
- **Compact OS**: Enabled at time of crash

Call Stack (abbreviated)

```
nt!KeBugCheckEx
nt!guard_icall_handler+0x1e
nt!RtlpExecuteHandlerForException+0xf
nt!RtlDispatchException+0x2d2
nt!KiDispatchException+0x35f
nt!KiExceptionDispatch+0x145
nt!KiGeneralProtectionFault+0x365
nt!KscpCfgDispatchUserCallTargetEsSmep+0x11
Ntfs!FindNextIndexEntry2+0x675
Ntfs!NtfsContinueIndexEnumeration+0x10e
```

```
Ntfs!NtfsQueryDirectory+0xd32
Ntfs!NtfsCommonDirectoryControl+0x2ce
Ntfs!NtfsFsdDirectoryControl+0xcb
nt!IopfCallDriver+0x5b
FLTMGR!FltpLegacyProcessingAfterPreCallbacksCompleted+0x3fe
FLTMGR!FltPerformSynchronousIo+0x443
Wof!WofPreDirectoryControlCallback+0x695        <-- CRASH HERE
FLTMGR!FltpPerformPreCallbacksWorker+0x58f
FLTMGR!FltpPassThroughInternal+0xc0
FLTMGR!FltpPassThrough+0x29f
FLTMGR!FltpDispatch+0x264
nt!IopfCallDriver+0x5b
nt!IopSynchronousServiceTail+0x1c5
nt!NtQueryDirectoryFileEx+0xd8
nt!KiSystemServiceExitPico+0x496
```

Environment

- **OS**: Windows 11 25H2 (Build 26200.7922)
- **BuildLabEx**: 26100.1.amd64fre.ge_release.240331-1435
- **Wof.sys**: 10.0.26100.5074 (WinSxS) / 10.0.26100.1 (FileVersion metadata)
- **CPU**: 32 logical processors
- **RAM**: 192 GB
- **GPU**: NVIDIA RTX 5000 Ada Generation Laptop GPU, driver 32.0.15.9571
- **Disk**: 3.8 TB, 204 GB free
- **Compact OS**: Was enabled (now disabled as workaround)
- **Recent KBs**: KB5077241, KB5074828, KB5077869, KB5077371

Recurrence

This is the **second occurrence** of the same Wof.sys crash. First was 2026-03-06 (7 BSODs, multiple bugcheck codes). An in-place upgrade from ISO to build 26200 was performed, but Wof.sys was NOT updated (26200 shares 26100 driver binaries).

Workaround Applied

`compact /CompactOS:never` — decompressed all OS binaries to reduce Wof.sys directory control interceptions. This does not fix the bug but avoids the hot codepath.

Reproduction

Any process performing NTFS directory enumeration on a Compact OS system can trigger this crash. In this case, `claude.exe` (a Node.js-based CLI tool) was listing directory contents.

Files Included

File	Contents
`030926-32171-01.dmp`	Kernel minidump from the crash
`windbg-analyze.txt`	Full `!analyze -v` output from WinDbg
`system-info.txt`	OS version, Wof.sys version, GPU, memory, hotfixes, disk
`event-logs.txt`	Bugcheck, Kernel-Power, WHEA, critical/error events (7 days)
`driver-list.txt`	All loaded drivers + filter manager minifilter instances
`bsod-triage-output.txt`	Full triage script output
`bsod-analysis.md`	Incident history and root cause analysis

BSOD Analysis

2026-03-09 — Wof.sys recurrence (1 BSOD)

Time	Bugcheck	Faulting Module	Process	Trigger
16:17	`0x00000139` (0x0a)	`Wof.sys` 10.0.26100.5074	`claude.exe`	`NtQueryDirectoryFileEx`

WinDbg bucket:
`0x139_a_GUARD_ICALL_CHECK_FAILURE_Wof!WofPreDirectoryControlCallback+0x695`

Root cause: Same Wof.sys CFG guard bug as 2026-03-06. The in-place upgrade to build 26200 did NOT update Wof.sys — it stayed at 10.0.26100.5074 (26200 shares 26100 driver binaries). Compact OS was enabled, making Wof.sys intercept every NTFS directory IRP.

Mitigation applied: `compact /CompactOS:never` — decompressed 43,020 OS binaries. Re-enable with `compact /CompactOS:always` once Microsoft patches Wof.sys.

System state post-mitigation: No WHEA errors, no RAM issues, no GPU crash, Norton clean, SFC clean, 204GB free, build 26200.7922 (25H2). KB5077241 + KB5074828 installed same day (WU reboots at 07:38-07:42 were planned).

Dump: `C:\Windows\Minidump\030926-32171-01.dmp`

2026-03-06 — Multi-BSOD storm (7 BSODs)

Summary

6+ BSODs in a single day with **6 different bugcheck codes**. Norton BehaviorShield restart log suggests possibly **10+ reboots** (shield restarted at 1:36 AM, 4:00 AM, 7:02 AM, 8:59 AM, 10:06 AM, 10:23 AM, 10:33 AM, 11:08 AM, 2:39 PM, 3:48 PM, 4:04 PM).

Crash Timeline (from Event Log)

Time	Bugcheck	Name	
~09:50	`0x00000139`	KERNEL_SECURITY_CHECK_FAILURE	Kernel da
~10:06	`0x0000003b`	SYSTEM_SERVICE_EXCEPTION	STATUS_
~10:23	`0x0000014f`	PDC_WATCHDOG_TIMEOUT	Power de hung
~11:02	`0x0000001e`	KMODE_EXCEPTION_NOT_HANDLED	STATUS_
~15:29	`0x00000050`	PAGE_FAULT_IN_NONPAGED_AREA	Invalid m
~15:48	`0x0000010e`	VIDEO_MEMORY_MANAGEMENT_INTERNAL	GPU VR

Hardware Context

- **GPU**: NVIDIA RTX 5000 Ada Generation Laptop GPU (16GB VRAM)
- **GPU Driver**: `32.0.15.9571` dated **2026-02-27** (8 days old)
- **Intel iGPU**: UHD Graphics, driver `31.0.101.5081`
- **RAM**: 192GB (Windows Memory Diagnostic: **PASSED** — but basic test only)
- **Other drivers**: Meta Virtual Monitor, LuminonCore IDDCX Adapter

Norton AV Findings

False positives — Norton quarantined these as `Win64:Evo-gen [Trj]`:

- **2025-10-18**: `AnthropicClaude\app-0.13.108\...\claude-native-binding.node` — Claude Desktop native binding
- **2026-02-08**: `JetBrains\PyCharm2025.3\aia\codex\bin\codex-acp-x64-windows.exe` — JetBrains AI Codex
- **2026-01-28**: `huggingface\gradio\frpc\frpc_windows_amd64_v0.3` — Gradio tunnel (PUP)

Today (2026-03-06):

- **No new detections/quarantines** in Norton logs
- Detection manager reinitialized 5 times (after reboots): 2:02 AM, 6:44 AM, 8:26 AM, 9:11 AM, 2:07 PM

- Norton BehaviorShield was running stably for weeks before today (normal multi-day uptimes)
- **Norton is unlikely the direct BSOD cause** — no kernel-level actions today, and prior uptime was normal

Norton kernel drivers on system:

- `nllwireguard.inf` (Norton VPN, WireGuard) — v0.10.1.70, 2025-05-02
- `nllwintun.inf` (Norton VPN, WinTun) — v0.14.1.89, 2023-11-10

Recommendation: Add Claude Code exclusions to Norton

```
Norton → Settings → Antivirus → Exclusions/Exceptions:
- C:\Users\tom\.claude\*
- C:\Users\tom\AppData\Local\AnthropicClaude\*
- C:\Users\tom\AppData\Roaming\npm\node_modules\@anthropic-ai\*
- C:\Users\tom\AppData\Local\JetBrains\*\aia\*
```

Diagnosis (Revised after Norton analysis)

Primary suspect: NVIDIA driver 32.0.15.9571

- `0x0000010e` (VIDEO_MEMORY_MANAGEMENT_INTERNAL) **directly implicates GPU/VRAM**
- Driver is only 8 days old — could be buggy
- The variety of other crashes (access violations, page faults) can cascade from GPU driver kernel corruption
- System was stable for weeks before today — correlates with new driver settling in

Secondary: Failing RAM

- 6 different bugcheck codes = classic intermittent RAM failure pattern
- Windows Memory Diagnostic is shallow — only catches obvious bit-flip errors
- MemTest86 (boot-level, multi-pass) is the definitive test

Tertiary: Thermal throttling / overheating

- Laptop with RTX 5000 Ada = high thermal load
- Could cause varied crashes under sustained load

Ruled out: Norton causing BSODs

- No new detections or quarantine actions today
- Norton has only VPN tunnel drivers in kernel (WireGuard/WinTun) — no filesystem minifilter
- Multi-week stable uptime before today's crash storm

Recurring: SQL Server PolyBase crashing

- `SQL Server PolyBase Data Movement (MSSQLSERVER)` terminates repeatedly after each reboot
- Consequence of the BSODs, not a cause
- Service restarts 3x then gives up (7260s cooldown)

Action Plan

Step 1: Roll back NVIDIA driver (most impactful, easiest)

```
# Check available driver versions
pnputil /enum-drivers | findstr /i "nvidia"

# Option A: Roll back via Device Manager
devmgmt.msc
# -> Display adapters -> NVIDIA RTX 5000 -> Properties -> Driver -> Roll Back
         Driver

# Option B: Download previous stable driver from NVIDIA
# Use DDU (Display Driver Uninstaller) in Safe Mode for clean removal
```

Step 2: Run thorough memory test

```
# Schedule Windows Memory Diagnostic (basic -- already ran, passed)
mdsched.exe

# Better: Download and boot MemTest86 from USB
# https://www.memtest86.com -- run 4+ passes overnight
```

Step 3: Check thermals

```
# Install HWiNFO64 and monitor temps under load
# GPU throttle temp: ~83C, CPU: ~100C
# If hitting these -> clean fans / repaste
```

Step 4: Monitor with WinDbg (if crashes continue after driver rollback)

```
# Install WinDbg
winget install Microsoft.WinDbg

# Analyze minidump
windbg -z C:\Windows\Minidump\030626-30937-01.dmp
# In WinDbg: !analyze -v
```

Step 5: Verify system files

```
# Run as Administrator
sfc /scannow
DISM /Online /Cleanup-Image /RestoreHealth
```

Minidump Files

- `C:\Windows\Minidump\030626-30343-01.dmp` – 0x50 PAGE_FAULT (15:48)
- `C:\Windows\Minidump\030626-30937-01.dmp` – 0x10e VIDEO_MEMORY (16:04)
- Other dumps from earlier today may have been rotated out (only 2 remain)

Norton Log Paths

- Detections: `C:\ProgramData\Norton\Antivirus\log\detections.log`
- Scans: `C:\ProgramData\Norton\Antivirus\log\scans.log`
- Behavior Shield: `C:\ProgramData\Norton\Antivirus\report\BehaviorShield.txt`
- FileSystem Shield: `C:\ProgramData\Norton\Antivirus\report\FileSystemShield.txt`
- AntiRansomware: `C:\ProgramData\Norton\Antivirus\report\AntiRansomwareShield.txt`
- Note: Logs are locked by NortonSvc — copy to temp before reading

BSOD Diagnostics Skill

AI-assisted BSOD recovery workflow. Claude loads this skill on “bsod” trigger, runs the decision workflow unattended through diagnostics, and escalates to the user only for repair actions that need confirmation (Rule 5). The 14 scripts + decision PUML enable full triage-to-fix without manual intervention on the diagnostic side.

Trigger: user says “bsod” or reports system instability / blue screen **Flow:** load skill -> run decision workflow -> diagnose -> propose fix -> confirm -> execute **Output:** diagnosis with root cause, recommended fix, and scripts to run

OS Structure

Recovery workflows differ by OS. Scripts and tools are organized per platform:

```
src/skill/bsod-diagnostics/
  skill-bsod-diagnostics.md       # This file (OS-agnostic overview)
  skill-bsod-diagnostics.puml     # Component diagram
  bsod-decision-workflow.puml     # Decision flowchart (source of truth)
  bsod-analysis.md                # Incident history
  win/bin/                        # PowerShell scripts (14)
  ubuntu/bin/                     # Bash scripts (planned)
  macos/bin/                      # zsh/bash scripts (planned)
```

```
agent/                          # Portable LLM agent config
  Modelfile                      # Ollama / LM Studio preset
  agent-loader.sh                # Skill context loader
```

OS	Shell	Crash type	Dump location	Repair tool
Windows	PowerShell	BSOD (bugcheck)	`C:\Windows\Minidump\*.dmp`	DISM, sfc, setup.exe
Ubuntu	Bash	Kernel panic/oops	`/var/crash/`, `/var/log/kern.log`	apt, dpkg, live USB
macOS	zsh	Kernel panic	`/Library/Logs/DiagnosticReports/`	Disk Utility, macOS Recovery

MCP Tool Integration

Each skill is an **MCP tool** exposed via `RecoveryMcpServer.kt` (JBang). Any MCP client connects and invokes recovery tools:

Mode	Transport	Client	When
Online	SSE (port 3001)	Claude Code, remote LLM	System has network
Offline	stdio	DictaLM3 1.7B GGUF via Ollama	No network, GPU may be down

```
# Online — Claude or any MCP client connects over network
jbang agent/RecoveryMcpServer.kt --sse --port 3001

# Offline — pipe to local GGUF model
jbang agent/RecoveryMcpServer.kt  # stdio, local model reads/writes
```

14 MCP tools registered: `recovery/triage`, `recovery/storage-cleanup`, `recovery/system-repair`, etc. OS auto-detected — routes to `win/bin/*.ps1`, `ubuntu/bin/*.sh`, or `macos/bin/*.sh`.

Okio KMP with Kotlin wasmJs gives full filesystem control on any platform — scripts can be invoked from the Ktor MCP server or directly by the local JBang agent.

Offline / Portable mode

The diagnostic phase (steps 1-3) requires no internet — scripts read local event logs, minidumps, and driver info. A small portable LLM (GGUF) can run the full triage using this skill's PUML + scripts, then hand off to the user for repair decisions.

Requirements: - This skill directory (md + puml + win/bin/*.ps1 or ubuntu/bin/*.sh) - Shell (PowerShell on Windows, bash on Ubuntu) - WinDbg (optional, Windows dump analysis) - No API keys, no cloud, no network needed for diagnostics - **Model:** DictaLM3 Thinking 1.7B (GGUF) — smallest known thinking model, Hebrew+English, runs on CPU if GPU is down - **Runtime:** Ollama CLI or LM Studio (GGUF loader) - **Testing:** VMware VM with snapshots — never test repair scripts on production hardware with a 1.7B model - **Snapshots:** Take a VMware snapshot before every repair test — instant rollback on failure

Rules

1. **Never tamper with CBS/WinSxS** — do not remove, rename, or replace manifests, packages, or component directories. Orphaned references cause CorruptManifest and can render the system unbootable. A reboot after such tampering = recovery environment only.
2. **Stick to vendor tools** — only use DISM, sfc, setup.exe, Device Manager, and other Microsoft-provided repair paths. If DISM fails (especially 0x800f0915), go straight to in-place upgrade repair — do not attempt manual CBS surgery.
3. **Free disk space BEFORE any repair** — in-place upgrade needs ~30GB free, DISM needs ~10GB. An upgrade that fails mid-way due to low disk = unbootable system. Run `storage-cleanup.ps1` first.
4. **In-place upgrade is the correct escalation** — when DISM /RestoreHealth fails, skip retries and manual fixes. Mount ISO, run setup.exe with “Keep everything”. This is the vendor-supported repair path that bypasses CBS entirely.
5. **Diagnostic scripts are safe, repair scripts need confirmation** — scripts that only read/report can run freely. Scripts that modify system state require explicit user approval.

Decision Workflow

See bsod-decision-workflow.puml — source of truth.

```
0. Rescue drive     -> ventoy-setup.ps1 (prepare while OS still boots)
1. Free space       -> storage-cleanup.ps1
2. Triage (parallel)-> bsod-triage.ps1 + norton-check.ps1 + driver-check.ps1
3. Analyze codes    -> DecisionTree (GPU / Memory / Kernel / Power /
Multiple)
4. Fix attempt      -> driver rollback, WinDbg bg analysis
5. System repair    -> system-repair.ps1 (DISM + sfc, vendor tools only)
6. Escalate         -> inplace-repair.ps1 (when DISM fails)
7. Hardware tests   -> MemTest86, thermals (if software fixes don't resolve)
```

Components

Component	Purpose
Rules	5 hard rules governing all repair actions

Component	Purpose
StorageCleanup	**Prerequisite** — free disk before any repair
QuickTriage	9 parallel checks on BSOD report
DecisionTree	Route to fix based on bugcheck pattern
WinDbgAnalysis	Background task — exact faulting module from dump
NortonAV	Check AV logs (locked — copy first)
DriverActions	Rollback, DDU, safe mode disable
SystemRepair	DISM + sfc (vendor tools only)
FODDiagnostic	**Diagnostic only** — detect stale FOD blocking DISM
InplaceRepair	In-place upgrade from ISO (correct escalation when DISM fails)
HardwareTests	MemTest86, thermals
NortonExclusions	Prevent Claude/AI false positives
BugcheckRef	Code -> cause lookup

Windows Scripts

Path: `src/skill/bsod-diagnostics/win/bin/`

Script	Phase	Purpose	Usage
storage-cleanup.ps1	1	Temp, WU cache, minidump rotation	`-File storage-cleanup.ps1 [-DryRun]`
bsod-triage.ps1	2	All 9+ triage checks in one run	`-File bsod-triage.ps1`
norton-check.ps1	2	Copy locked logs, search AI false positives	`-File norton-check.ps1`
driver-check.ps1	2	GPU info, NVIDIA packages, recent drivers	`-File driver-check.ps1`
windbg-analyze.ps1	3	WinDbg !analyze -v (finds kd.exe dynamically)	`-File windbg-analyze.ps1 [-DumpFile path]`
system-repair.ps1	5	DISM RestoreHealth + sfc /scannow (admin)	`-File system-repair.ps1`
dism-restore-with-iso.ps1	5	Mount ISO + DISM /RestoreHealth /Source	`-File dism-restore-with-iso.ps1 [-IsoPath path]`

Script	Phase	Purpose	Usage
dism-monitor.ps1	5	Monitor DISM progress, CBS/DISM logs	`-File dism-monitor.ps1`
fod-fix.ps1	5	**Diagnostic only** — detect stale FOD packages	`-File fod-fix.ps1`
gpu-safe-mode.ps1	4	Disable/enable NVIDIA GPU for safe mode	`-File gpu-safe-mode.ps1 -Action disable|enable`
download-win11-iso.ps1	6	Download Win11 ISO via MCT (version-stamped)	`-File download-win11-iso.ps1 [-OutDir path]`
inplace-repair.ps1	6	In-place upgrade repair (keeps apps/data)	`-File inplace-repair.ps1 [-IsoPath path] [-Quiet]`
upgrade-monitor.ps1	6	Monitor in-place upgrade progress and logs	`-File upgrade-monitor.ps1`
ventoy-setup.ps1	0	Ventoy multi-boot USB/microSD setup + ISO copy	`-File ventoy-setup.ps1 [-DriveLetter E] [-DownloadOnly]`
wof-replace.ps1	–	**UNSAFE** — manual Wof.sys replace, prefer inplace-repair	deprecated

All scripts: `powershell -ExecutionPolicy Bypass -File <path>`

Invoking from bash (Claude Code)

Use **single quotes** around `-Command` to prevent bash from interpreting `$`:

```
powershell -NoProfile -Command 'Get-WinEvent -FilterHashtable
        @{LogName="System"; Id=1001} -MaxEvents 5'
```

Do NOT use `2>$null` — bash interprets that. Use `-ErrorAction SilentlyContinue` instead.

Ubuntu Scripts (planned)

Path: `src/skill/bsod-diagnostics/ubuntu/bin/`

Script	Phase	Purpose
kern-triage.sh	2	Parse kern.log, dmesg, /var/crash
driver-check.sh	2	GPU driver (nvidia-smi), kernel modules

Script	Phase	Purpose
disk-cleanup.sh	1	apt clean, journal vacuum, snap cache
system-repair.sh	5	apt –fix-broken, dpkg –configure, fsck
live-usb-repair.sh	6	Chroot repair from live USB

Bugcheck Reference

Group	Codes	Root cause	Action
Driver/RAM	0x0A, 0x1E, 0x50, 0xD1	bad driver or failing RAM	WinDbg -> driver rollback or MemTest86
GPU/VRAM	0x10E, 0x116, 0x119	GPU driver or VRAM	rollback GPU driver, DDU
Kernel	0x139	code bug or memory corruption	WinDbg -> identify module -> vendor fix
Power	0x9F, 0x14F	power management	driver or firmware update
DWM	dwm.exe crash in app log	blocks login	safe mode -> gpu-safe-mode.ps1
WoF	Wof.sys in WinDbg stack	system file bug	in-place upgrade (not DISM)

Norton AV

Known false positives (quarantined as `Win64:Evo-gen [Trj]`): - `claude-native-binding.node` — Claude Desktop - `codex-acp-x64-windows.exe` — JetBrains AI - `frpc_windows_amd64` — Gradio tunnel (PUP)

Exclusions to add:

```
Norton -> Settings -> Antivirus -> Exclusions:
- %USERPROFILE%\.claude\*
- %LOCALAPPDATA%\AnthropicClaude\*
- %APPDATA%\npm\node_modules\@anthropic-ai\*
- %LOCALAPPDATA%\JetBrains\*\aia\*
```

Log paths (locked — copy to %TEMP% first): - `C:\ProgramData\Norton\Antivirus\log\detections.log` - `C:\ProgramData\Norton\Antivirus\report\BehaviorShield.txt` - `C:\ProgramData\Norton\Antivirus\report\FileSystemShield.txt`

Incident History

Date	Analysis	Root causes	Resolution
2026-03-09	bsod-analysis.md	1 BSOD: Wof.sys 10.0.26100.5074 CFG guard bug (0x139), triggered by claude.exe dir listing	Disabled Compact OS (`compact /CompactOS:never`)
2026-03-06	bsod-analysis.md	7 BSODs: Wof.sys RTM bug (0x139), NVIDIA VRAM (0x10E), FOD corruption blocking DISM	In-place upgrade from ISO (26200)
earlier	session 30a95246	ntfs.sys paged error	DISM stuck at 62.3% (RPC error)
earlier	session 84f4722c	reboot during defrag	defrag triggered crash
earlier	session e119dd12	system rebooted	discussed MemTest86
earlier	session 0098dfb7	post-fix	updated several drivers

Lessons Learned

Lesson	Context
DISM 0x800f0915 -> go straight to in-place upgrade	Tried 4 DISM approaches, all failed on same FOD metadata
Never remove CBS packages/manifests	Removing FOD .1742 package left orphaned manifest -> CBS marked store CorruptManifest
Wof.sys crash != corrupt file	SHA256 matched ISO — it's a code bug in RTM, not file corruption
Free disk space first	35GB temp cleanup was essential prerequisite
ISO version != running OS version	ISO was build 26200, OS was 26100 — WIM install.wim version differs from DISM tool version
`$` in paths breaks bash->PowerShell	Use single quotes for -Command, avoid `2>$null`
Test on VMware first	Never run repair scripts from a 1.7B model on production hardware

Agent Files

File	Purpose
`agent/RecoveryMcpServer.kt`	JBang MCP server — exposes all recovery scripts as MCP tools

File	Purpose
`agent/Modelfile`	Ollama model config with skill system prompt
`agent/agent-loader.sh`	Launcher script (Ollama or LM Studio)

Diagrams

- skill-bsod-diagnostics.puml — component/class diagram
- bsod-decision-workflow.puml — decision flowchart (source of truth)