

THREAT ADVISORY

The Threat of Fileless Malware



Services Performed By:

UltraViolet Cyber TIDE Team
tide@uvcyber.com

Published Date:

October 07, 2026

TLP:GREEN



Executive Snapshot

Fileless malware refers not to a single malware family, but to an execution technique: code that runs primarily in memory, abuses legitimate system tools, and avoids writing persistent, easily scannable artifacts to disk. It matters because it exploits a structural gap in traditional security tooling. Antivirus and other static defenses are built around scanning files, and fileless techniques are specifically designed to sidestep that layer, not through some unexplainable trick, but by design. Despite the name, most fileless intrusions still leave some footprint somewhere, whether an initial dropper, a registry key, or a scheduled task; the goal is minimizing what persists and what's scannable, not eliminating every trace entirely. This technique has been used across commodity infostealers, ransomware precursor tooling, and nation-state implants alike, and it tends to extend dwell time significantly, since less forensic residue means intrusions often go undetected for longer periods.

Priorities for detecting and deterring this technique:

- Focus detection capabilities toward behavior-based and memory-aware tooling rather than static file signatures alone.
- Enable deep logging for scripting engines (PowerShell, WMI) capable of surfacing obfuscated or encoded commands.
- Monitor legitimate administrative tools for anomalous use, since the same binaries serve both normal operations and malicious execution.
- Audit registry run keys, scheduled tasks, and WMI event subscriptions regularly against known-good baselines.
- Prioritize memory capture and log preservation early in any investigation, since this evidence is the most perishable and the most critical.



TIDE Team Analysis

Fileless malware is best understood as a design philosophy rather than a category of malicious code. Rather than relying on a standalone executable that a scanner can inspect and flag, fileless techniques favor execution that happens in memory, through scripting engines, or through abuse of legitimate, already-trusted system utilities. The defining goal isn't achieving some impossible zero-footprint intrusion; it's minimizing the specific kind of footprint that traditional security tooling is built to catch; a scannable file sitting on disk.

A fileless chain often begins with a small, otherwise unremarkable trigger. This could be a malicious script embedded in a document macro, a command executed through a legitimate scripting engine like PowerShell or Windows Management Instrumentation (WMI), or a loader that injects code directly into an already-running, trusted process. From there, the actual malicious logic may never be written to disk as a standalone file. Instead, it exists only in the memory space of a legitimate process, executes its intended function, and may clear itself out or simply end when that process terminates, leaving comparatively little behind for a disk-based scan to find later.

A core feature of fileless technique is its reliance on tools that are already present and trusted on nearly every system, such as PowerShell, WMI, mshta, rundll32, and similar native utilities. These tools exist for legitimate administrative purposes, which creates a genuine detection problem. Blocking them outright isn't usually viable in an enterprise environment, and distinguishing a legitimate PowerShell command from a malicious one requires looking closely at behavior, arguments, and context, not simply the presence of the tool itself.

This technique isn't new, and its continued relevance says something about how effective it remains. *Kovter* malware, an early and well-documented example, operated almost entirely through registry-stored payloads rather than traditional disk-based files, persisting primarily through registry run keys and scripting rather than an installed binary. *Poweliks* malware followed a similar pattern, embedding its logic directly in the registry and leveraging legitimate Windows processes to execute it. More recently, post-exploitation frameworks like *Cobalt Strike's Beacon* payload have popularized reflective, in-memory loading as a standard technique, letting a payload run without ever touching disk as a separate file. Nation-state operators have used similar approaches as well; in-memory implants and legitimate-tool abuse have shown up repeatedly in long-running espionage campaigns precisely because they extend the time an intrusion can persist before discovery.

Because fileless techniques generate less forensic residue by design, intrusions using this approach tend to have meaningfully longer dwell times than those relying on traditional, file-based malware. Memory and logs become the primary sources of evidence in these cases, and both are inherently perishable; memory clears on reboot, and logs age out or get overwritten if retention isn't configured generously. This creates a narrow and shrinking window in which meaningful forensic evidence actually exists, placing real pressure on how quickly an organization can detect and respond before that evidence disappears entirely.

Precisely because fileless technique spans commodity malware, ransomware operations, and sophisticated state-sponsored tooling alike, organizations shouldn't think of it as a threat tied to any one actor or campaign. It's a recurring method that will keep appearing across very different types of intrusions, for the simple reason that it continues to work against defenses built primarily around file scanning.



Why It Matters

Fileless technique challenges a foundational assumption behind much of traditional security tooling: that malicious activity will eventually produce a file that can be scanned, flagged, and quarantined. That gap is real, but it doesn't mean file-based scanning is obsolete or that defenders are starting from nothing. Traditional antivirus still catches a significant share of commodity threats and remains a useful baseline layer. What's changed is that it can no longer be the primary or sole line of defense against a technique specifically engineered to avoid it. The core message for stakeholders isn't that fileless malware is unstoppable; it's that stopping it requires a more mature, layered set of tools than file scanning alone, and that maturity is well within reach for organizations willing to invest in it.

Historical examples make this point concretely. Kovter and Poweliks, two early and well-documented fileless families, operated almost entirely through registry-stored payloads rather than traditional disk-based executables, persisting through registry run keys and legitimate scripting components rather than an installed binary. Neither family evaded detection indefinitely. Defenders adapted by shifting focus toward the registry locations and process behaviors these families relied on, rather than only searching for a file that may never appear. That shift, from asking "what file is this" to asking "what is this process actually doing," is the goal, and it's a shift the industry has already demonstrated it knows how to make successfully.

The reliance on legitimate, trusted utilities like PowerShell and WMI follows the same pattern. The fix isn't removing the tool; it's building the capability to tell normal administrative activity apart from malicious use of that same tool, through script logging, behavioral baselines, and attention to anomalous arguments or execution context. This is a solvable problem, and many organizations already have the pieces in place to do it well.

The extended dwell time associated with this technique is a genuine risk worth taking seriously, but it also points directly at where defensive investment pays off most: faster detection and shorter windows of undetected access. By prioritizing memory capture and increasing log retention defenders can directly address the one factor that most often determines whether this technique succeeds or gets caught early, which is time itself.

Stakeholders should treat fileless technique as an ongoing, evolving part of the threat landscape rather than a passing trend tied to any single campaign or actor. The countermeasures discussed here are real, achievable, and already proven effective against earlier fileless families like Kovter and Poweliks, demonstrating that this is a problem the security community knows how to address when it invests the right attention and resources into it. That said, meaningful protection requires sustained commitment rather than a one-time fix. Defenders will keep refining detection of fileless malware, its associated behaviors, and the signatures that distinguish malicious use of legitimate tools from normal activity, while attackers will just as persistently look for new ways to evade that same detection, whether through this technique or others that emerge in its place. This is, in effect, a continuing cycle rather than a battle with a clear endpoint. Organizations that understand this dynamic, and that treat detection maturity as an ongoing discipline rather than a project with a finish line, will be best positioned to keep pace as both sides continue to adapt.



How to Respond

- Lean on behavior-based and memory-aware detection capabilities, rather than relying primarily on static, signature-based antivirus, since fileless technique is specifically designed to evade file scanning.
- Enable deep logging for scripting engines, particularly PowerShell script block and module logging, to surface obfuscated or encoded commands that are otherwise invisible to standard event logs.
- Focus on behavioral baselines for legitimate administrative tools so that anomalous use of PowerShell, WMI, mshta, and similar utilities can be distinguished from normal operations.
- Regularly audit registry run keys, scheduled tasks, and WMI event subscriptions against known-good baselines, since these remain common persistence points even when no file is involved.
- Prioritize memory capture and extended log retention as standard practice rather than an incident-response afterthought, given how quickly this evidence can disappear once a system reboots or logs age out.

What UltraViolet Cyber is Doing

- Tracking new CVEs and high impact vulnerabilities, analyzing and deploying public Proof-Of-Concept code against custom built targets.
- Proactively enabling custom detections based on the collected artifacts, tactics, techniques, and procedures identified in this activity. UltraViolet is also maximizing detection coverage proactively through [Equinox](#).
- Performing hypothesis driven threat hunts based on threat actor behavior and artifacts. UVCyber customers will be informed of the results through secure channels.
- Parsing available victim dump data for any social, financial, business, or technical relations to UVCyber Clients and partner organizations.
- Aggregating threat intelligence from myriad sources and applying the most up-to-date knowledge to proactive threat hunting and response.

About UltraViolet Cyber

UltraViolet Cyber is a leading tech-enabled managed security services provider, delivering unparalleled cybersecurity expertise that fills technology and talent gaps across Global 2000 and Federal Government customers. Founded and operated by security practitioners from the national intelligence community, UltraViolet Cyber connects offensive security, application security, detection and response, and security engineering to deliver a differentiated approach to cybersecurity operations. Transforming customers' security programs, UltraViolet Cyber's flagship security-as-a-service solution, UV Lens, removes complex operational silos, replacing them with integrated security capabilities. UltraViolet is headquartered in McLean, Virginia with technology centers across the world.

443.351.7630 / info@uvcyber.com |  UltraViolet Cyber |   @uv_cyber
