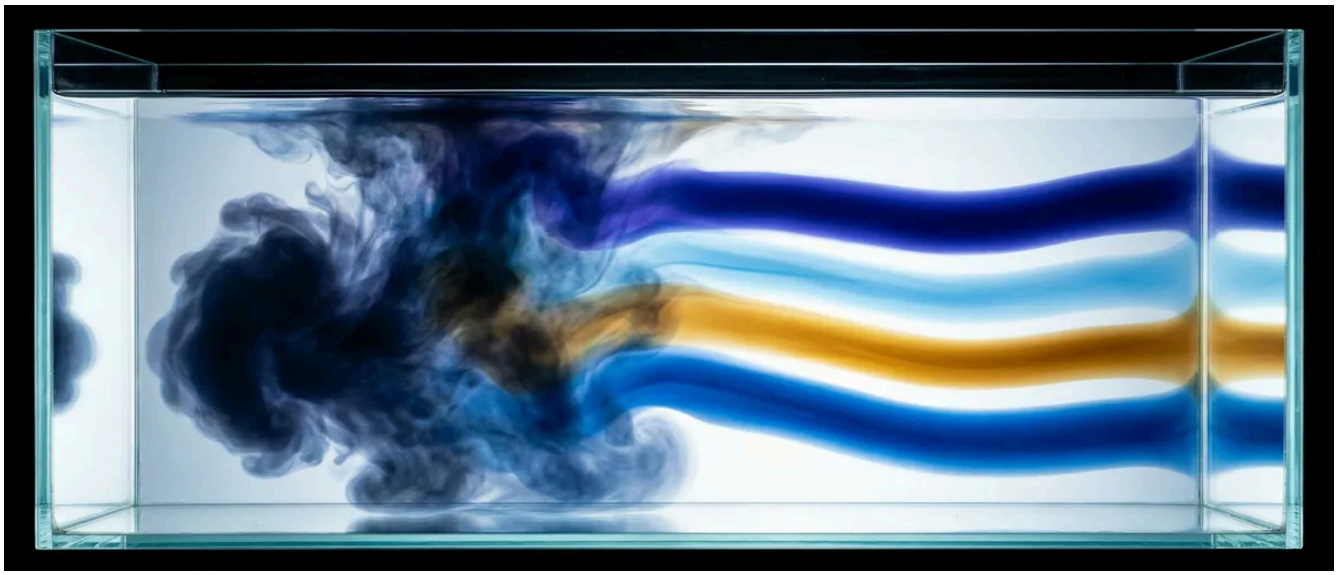
The History of Fileless Malware - Looking Beyond the Buzzword

zeltser.com/fileless-malware-beyond-buzzword

Lenny Zeltser April 20, 2017

Defenders apply "fileless malware" to various evasion techniques, far beyond its 2001 in-memory definition. Walk through the malware samples from Code Red to PyLoose so we can understand what this term means and how it evolved.

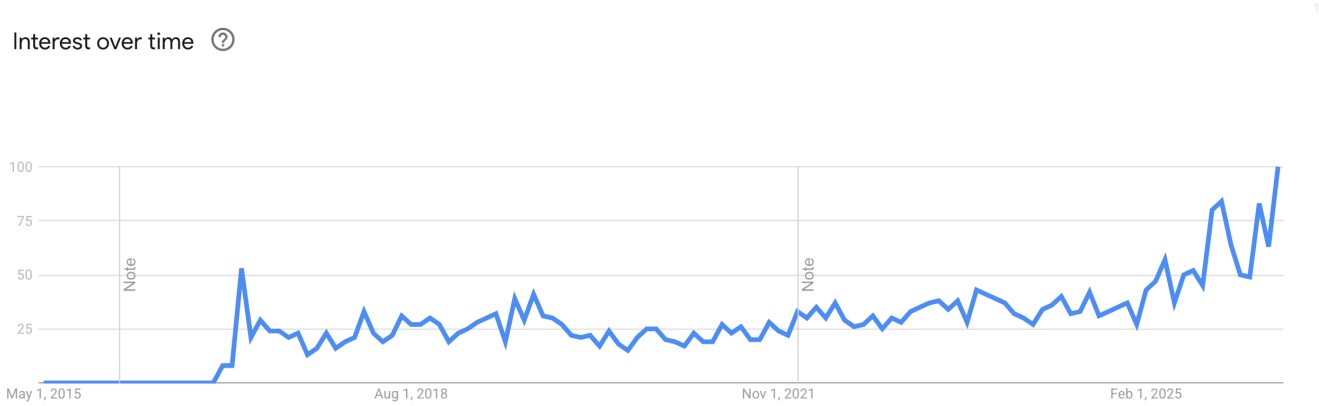


What's the deal with "fileless" malware? Though many security professionals find the term imprecise, vendors and researchers still use it when discussing threats that evade traditional defenses. Let's trace the term's evolution from Code Red in 2001 through cross-platform campaigns in the 2020s, looking beyond the buzzword.

Frenzy for Fileless

The notion of fileless malware was gaining significant attention at industry events, private meetings, and online discussions in the years leading up to 2017. This might have been because the threat highlighted deficiencies in old-school endpoint security methods, allowing new approaches to showcase their strengths.

Google Trends data shows people's interest in the term spiking around 2017 and climbing to a new peak by early 2026.



This pattern corresponds to the publicly discussed malware that exhibited “fileless” capabilities through the mid-2010s, and to the burst of research and marketing publications that appeared in 2017. Researchers and vendors kept writing about fileless threats long after that, documenting new techniques through the 2020s.

What is Fileless Malware?

Let’s get this out of the way: Fileless malware sometimes has files. Most people seem to be using the term fileless malware in a manner consistent with the following definition:

> Fileless malware is malware that operates without placing malicious code on the file system.

This definition accommodates situations where the infection began with a malicious script or even a benign executable on the file system. It also matches the scenarios where the sample stored artifacts in the registry, even though Windows keeps registry contents on disk. It applies regardless of how the infection occurred, whether through an exploit of a vulnerability, a social engineering trick, or the misuse of a feature.

Though initially fileless malware referred to malicious code that remained solely in memory without even implementing a persistence mechanism, the term evolved to encompass malware that relies on aspects of the file system for activation or persistence. Let’s review some of the malicious programs that influenced how we use this term today.

2001-2003: Code Red and SQL Slammer

The notion of malicious code that resides solely in memory certainly existed before the 21st century. Yet it wasn’t until the highly prolific Code Red worm left its mark on the Internet in 2001 that the term fileless malware entered general parlance. The earliest public reference I could find dates to summer 2001, when Kaspersky Labs published an announcement that quoted none other than Eugene Kaspersky:

> "We predict that in the very near future, such 'fileless' worms as Code Red will become one of the most widespread forms of malicious programs, and an anti-virus' ineffectiveness in the face of such a threat simply invites danger."

Code Red exploited a vulnerability in Microsoft IIS web servers, remaining solely in memory of the infected host, as explained by the Center for Applied Internet Data Analysis.

A year and a half later, another worm—SQL Slammer—spread like wildfire by exploiting a vulnerability in Microsoft SQL servers. Robert Vamosi, writing for ZDNet in 2003, described this malware as "file-less" and mentioned that it resided "only in memory, much as Code Red."

I came across another early mention of this term in the 2003 patent filing by the venerable Peter Szor, who worked at Symantec at the time. Titled *signature extraction system and method*, the patent defines fileless malware as:

> "Malicious code that is not file based but exists in memory only… More particularly, fileless malicious code … appends itself to an active process in memory…"

The original definition of fileless malware was close to the English meaning of the words describing malware that remains active without leaving any files.

2012: A Bot That Installed the Lurk Trojan

The next fileless malware reference I could locate appeared nearly a decade after Code Red and SQL Slammer worms. In 2012, Sergey Golovanov at Kaspersky Labs presented his analysis of a bot that didn't save any files to the hard drive, explaining that:

> "We are dealing with a very rare kind of malware — the so-called fileless malicious programs that do not exist as files on the hard drive but operate only in the infected computers RAM."

The unnamed sample exploited a client-side Java vulnerability and operated solely in memory of the affected javaw.exe process. Sergey mentioned that the bot could install the Lurk banker trojan.

Earlier that year, Amit Malik at SecurityXploded published an educational write-up, explaining how to achieve "in-memory or file-less execution" of a Windows program without saving it to disk after downloading it from the Internet.

2014: Poweliks, Angler, Phase Bot

The malicious programs outlined above remained purely memory-resident, leaving no direct footprints on the file systems. As a result of this volatility, they disappeared once the system was rebooted. In contrast, 2014 brought us Poweliks malware, which G Data's Paul Rascagnères described as "persistent malware without a file." This sample entered the system by exploiting a Microsoft Word vulnerability. It used PowerShell and JavaScript along with shellcode to jumpstart its in-memory execution. Kevin Gossett at Symantec described its persistence mechanism like this:

> "Normally, malware will place an entry in the Run subkey that points to a malicious executable which is then executed. Poweliks makes the Run subkey call rundll32.exe, a legitimate Windows executable used to load DLLs, and passes in several parameters. These parameters include JavaScript code that eventually results in Poweliks being loaded into memory and executed."

A month later, security researcher Kafeine documented an Angler exploit kit infection that exhibited fileless characteristics. The attack targeted a client-side Java vulnerability and operated solely in memory of the affected javaw.exe process. In a related future occurrence, Angler began installing Bedep downloader in 2016 that, according to Palo Alto's Brad Duncan, "is installed without creating any files because it is loaded directly into memory by the exploit shellcode."

Closer to the end of 2014, security researcher Marcus Hutchins documented a fileless rootkit named Phase Bot. According to its advertisement, the sample achieved stealth "by installing on windows systems without dropping any files to disk and not having it's own process. […] Phase hids it's relocatable code encrypted in the registry and uses powershell to read and execute this position independent code into memory." Like Poweliks, this malware maintained persistence by launching rundll32.exe from an autorun registry key to execute JavaScript.

2014-2015: Duqu 2.0, Kovter

In mid-2015, Kaspersky Labs published details about an advanced adversary that operated in 2014-2015 using a sophisticated malware platform that the vendor called Duqu 2.0. The attack exploited a Windows vulnerability to install stealthy malware that remained solely in memory on the infected host. It did not implement a persistence mechanism. Instead, the

researchers explained, the attackers targeted servers with high uptime and then re-infected the systems that got “disinfected by reboots.”

Another fileless malware sample that gained attention in 2015 was Kovter. In that incarnation, the Kovter’s infection technique closely resembled that of Poweliks. Even when starting the infection with a malicious Windows executable, the sample removed that file after storing obfuscated or encrypted artifacts in the registry. At least one of its variations maintained persistence by using a shortcut file that executed JavaScript. As outlined by Andrew Dove at Airbus, this script launched PowerShell, which executed shellcode, which launched a non-malicious application after injecting malicious code into it.

2016: PowerSniff, PowerWare, August

In mid-2016, Josh Grunzweig and Brandon Levene at Palo Alto Networks documented a malicious program they dubbed PowerSniff. The infection began with a Microsoft Word document that contained a malicious macro. The in-memory mechanics of this sample resembled some aspects of Kovter and involved a PowerShell script that executed shellcode, which decoded and executed additional malicious payload, operating solely in memory. PowerSniff could temporarily save a malicious DLL to the file system.

A couple of weeks later, Mike Sconzo and Rico Valdez at Carbon Black described a ransomware sample they called PowerWare. Like PowerSniff, PowerWare began its infection with a Microsoft Office document that contained a malicious macro, which ultimately launched PowerShell that continued the infection process without placing malicious executables on the file system.

Another fileless malware sample that utilized Microsoft Word macros and PowerShell was documented later in the year by Proofpoint. It was named August. According to the researchers, the sample downloaded a portion of a payload “from the remote site as a PowerShell byte array,” executing it in memory without saving it to the file system.

2017: POSHSPY, etc.

In early 2017, Kaspersky Labs described an unnamed incident where adversaries stored Meterpreter-based malicious code solely in memory. The only file system artifacts were legitimate Windows utilities such as *sc* (to install a malicious service that ran PowerShell) and *netsh* (to tunnel malicious network traffic).

Several months later, Matthew Dunwoody at Mandiant described another sophisticated attack that involved fileless malicious code. Named POSHSPY, the sample used Windows Management Instrumentation (WMI) capabilities of the OS to maintain persistence and relied

on PowerShell for its payload. The sample could download executable files, which it would save to the file system. Matthew concluded that:

> By "living off the land," this adversary implemented "an extremely discrete backdoor that they can deploy alongside their more conventional and noisier backdoor families, in order to help ensure persistence even after remediation."

The incidents above highlighted the powerful capabilities available to intruders, even when they rely solely on built-in benign programs to execute a malicious payload on infected systems.

2019: Astaroth

In mid-2019, the Microsoft Defender Security Research Team dismantled a fileless campaign they tracked as Astaroth. It pulled the payload down through BITSAdmin and decoded it with Certutil. WMIC executed the chain via XSL scripting, and Astaroth itself ran inside the memory of the Userinit process. Microsoft noted that "at no point during the attack chain is any file run that's not a system tool."

Astaroth illustrated a pattern: Attackers running fileless campaigns commonly used legitimate Windows utilities like BITSAdmin and WMIC, called Living Off the Land Binaries (LOLBins), now tracked by the LOLBAS Project. The two ideas overlapped without being identical; fileless described whether malicious code touched the disk, while LOLBin named which legitimate tool the attacker abused.

Between 2019 and 2022, vendors reported steady fileless activity, but researchers documented few new fileless storage or execution techniques.

2022-2025: Event Logs, PyLoose, and Formal Recognition

In early 2022, Denis Legezo at Kaspersky observed a fileless campaign whose dropper hid shellcode inside Windows Event Logs. The malware wrote encrypted payloads into specific log entries, then read them back at runtime. Denis described this as the first time he had seen event logs used this way in the wild. The attackers delivered the shellcode using Cobalt Strike, a commercial post-exploitation framework whose Beacon implant ran in memory by default.

In mid-2023, Avigayil Mechtinger at Wiz documented PyLoose, a Python script that loaded XMRig into memory on Linux cloud hosts. The attackers used the memfd Linux feature to keep the binary off disk. Wiz reported PyLoose as "the first publicly documented Python-based fileless attack" targeting cloud workloads. Attackers soon ran the same technique at

scale. In early 2025, Wiz traced a campaign against exposed PostgreSQL servers, attributing it to a threat actor they called JINX-0126. The attackers compromised more than 1,500 instances and again ran XMRig-C3 through memfd.

As fileless techniques became routine, the security community formalized the vocabulary. MITRE recognized the pattern by adding a Fileless Storage sub-technique (T1027.011) to ATT&CK, citing Denis Legezo's event-log research. In February 2024, CISA, the NSA, the FBI, and partner agencies published joint guidance on living-off-the-land techniques.

Alternatives to Saying "Fileless Malware"

Sergey Golovanov's 2012 article mentioned above originally used the term fileless malware, but it was later revised to say *bodiless malware* instead. Kaspersky Labs experimented with saying bodiless malware as late as 2016, but it returned to fileless malware in 2017.

Speaking of the terms that didn't take off… The notion of *Advanced Volatile Threat* (AVT) gained some buzz in 2013 after, according to Wikipedia, being coined by John Prisco of Triumfant Inc. The short-lived AVT moniker popped up in Byron Acohido's USA Today article in 2013 in reference to a backdoored version of Apache software. According to Pierre-Marc Bureau, who was at ESET at that time, the backdoor left "no traces of compromised hosts on the hard drive other than its modified" web server binary.

In contrast, a reasonable alternative to the term fileless malware was introduced by Carbon Black in its 2016 Threat Report. The report used the phrase *non-malware attacks*. Writing on the company's blog a few months later, Michael Viscuso explained the meaning of this term like this:

> "A non-malware attack is one in which an attacker uses existing software, allowed applications and authorized protocols to carry out malicious activities. Non-malware attacks are capable of gaining control of computers without downloading any malicious files, hence the name. Non-malware attacks are also referred to as fileless, memory-based or 'living-off-the-land' attacks."

Gartner used the term "non-malware attack" in a 2017 report that highlighted Carbon Black. However, another Gartner report published a month later used "fileless attacks" instead.

The security community discarded these alternatives. Practitioners settled on *living off the land* as the sibling term for the tools attackers abuse.

Why Does It Matter?

We've stretched the term "fileless" from Code Red's pure in-memory existence to the living-off-the-land campaigns of the 2020s. Along the way, we accumulated multiple overlapping definitions and even formal recognition in MITRE ATT&CK. The term is here to stay, and writers find it useful for describing threats that evade traditional detection.

When someone says "fileless," ask what they actually mean. Are they describing in-memory execution, registry-resident persistence, script-based delivery, or living-off-the-land tactics? Understand the specific techniques behind the label.

I care about this terminology because I'm trying to avoid buzzwords and empty phrases. Perhaps you do, too.

Related Articles

Cybersecurity Advice for Political Campaigns

The Language and Nature of Fileless Attacks Over Time

About the Author

Lenny Zeltser is a cybersecurity executive with deep technical roots, product management experience, and a business mindset. He has built security products and programs from early stage to enterprise scale. He is also a Faculty Fellow at SANS Institute and the creator of REMnux, a popular Linux toolkit for malware analysis. Lenny shares his perspectives on security leadership and technology at zeltser.com.

More on
MalwareHistory
10 min to read

Published: April 20, 2017
Updated: May 26, 2026