

The AI Malware Surge: Behavior, Attribution, and Defensive Readiness

arcticwolf.com/resources/blog/the-ai-malware-surge-behavior-attribution-and-defensive-readiness

Arctic Wolf Labs

March 24, 2026



Executive Summary

Over the last year, AI-assisted malware development has evolved from an experimental practice into a common part of the attacker toolkit. In a rolling window from February 2025 to February 2026, Arctic Wolf Labs observed over 22,000 distinct files triggering AI-focused YARA rules across multiple malware repositories. These files included AI-generated code, large language model (LLM)-style scaffolding, runtime AI API integration, and DeepSeek-derived artifacts.

This shift was structural, not because attackers suddenly became more sophisticated, but because AI lowered the barrier to producing functional malware. We found that AI made malware creation faster, broader, and more accessible to threat actors who previously lacked the skill to build functional tooling on their own. This pattern was observed across multiple categories of malware, including infostealers, remote-access tools (RATs), and ransomware engines.

Key Points

- AI-assisted malware development has moved from isolated experimentation to a routine part of the attacker workflow, expanding the pool of threat actors who can produce operational malware
- DeepSeek R1, released in January 2025, emerged as a prevalent AI tool in this dataset; a substantial share of reviewed samples included a DeepSeek filename prefix
- 39% of analyzed samples had zero detections by signature-based antivirus (AV) solutions at the time of collection, indicating that a significant portion of malware being created is structurally new
- Only 1.4% of AI-assisted malware was linked to known targeted attacks, known threat actors, or financially motivated cybercriminal clusters; the overwhelming majority originated from unknown or lower-skill actors
- AI amplifies the speed, scale, and reach of malware, but its behaviors remain detectable to defenders with layered visibility

The Impact of AI in the Threat Landscape

AI is reshaping the threat landscape primarily through scale: it broadens who can build malware and accelerates how quickly functional tools emerge. In our research, we observed threat actors using LLMs to produce infostealers, RATs, droppers, ransomware engines, or other malicious scripts.

We observed a consistent pattern of threat actors learning iteratively, using LLMs to scaffold code and fill in gaps, moving from broken proof-of-concept (POC) implementations into usable malware faster and more capably than they could on their own. In that sense, **AI is not simply producing more code; it is narrowing the gap between developer skill and operational capability and shortening the timeline.**

These technological developments do not imply that every piece of AI-generated malware is necessarily dangerous on the first iteration of development. Some of these AI-assisted samples were mature and operational. Others were clumsy, incomplete, or outright hallucinatory. The broader effect is that AI compresses the path from initial idea to fully realized capability. Novice malware authors who previously couldn't build functional malware without AI can now produce POC code with advanced features such as defense evasion and privilege escalation, and this changes the threat landscape structurally at scale.

Across 22,331 files we analyzed in this research, 39% were undetected by signature-based antivirus solutions at the time of our collection, suggesting active evasion or novel construction rather than code classified under existing known-bad patterns. Only 16% of files analyzed had 40 or more positive detections at the time of collection.

Another notable shift is that some malware is no longer limited to AI-assisted development; a growing subset now incorporates AI at runtime. Across the dataset, we observed roughly 8% of samples with runtime LLM API integration patterns, and approximately 7% of samples embedding

hardcoded LLM API keys. Today, much of that observed usage is still relatively primitive, such as dynamic naming or message generation, but the architectural direction matters because it points toward more adaptive malware behaviors over time.

Behavior and Attribution

AI Use During Malware Development

AI models are being used by malicious actors to generate malware architecture, templates, and loader frameworks scaffolding them. Typical artifacts of LLM use we observed included:

- Verbose or tutorial-style comments
- Numbered task breakdowns
- Markdown-like section headers
- Emoji characters
- Cited web-search references accidentally carried into code

These distinct LLM-generation artifacts informed the custom YARA rules we used to detect AI-assisted files. Runtime-focused rules, by contrast, focused on malware samples that queried provider APIs at execution time, indicating the use of AI models for dynamic text generation, naming, or ad hoc decision making.

Our analysis revealed several patterns relevant to attribution: filename conventions, citation artifacts from AI-based web searches, recurring human and programming language clusters, and family-level behaviors that make clear AI is being absorbed into distinct malware development communities. A central finding here is that AI use was clustered in recognizable ways by language, workflow, and malware family. These fingerprints provided us with a rich view into threat actor workflows.

Several patterns point to distinct threat actor communities adopting AI. These include DeepSeek-derived filenames, web-search citation markers in code, and language-based sample clustering. At the same time, hallucinated, non-executable samples suggest an iterative learning process in which threat actors refine AI-generated output until it becomes operational. Together, these findings illustrate how AI is shaping threat actor tradecraft.

DeepSeek-Assisted Malware Generation

DeepSeek's R1 model had an outsized influence in the malware collection that we analyzed. This AI model represented a significant uplift in capabilities compared to previous options, offering low-cost access, strong Chinese-language support, and integrated web search. These traits made DeepSeek R1 attractive to cost-sensitive, novice threat actors focused on script generation.

The global threat actor community appears to have adopted DeepSeek’s new tooling within weeks of its January 2025 release, marking a dramatic step-change in malware output rather than gradual growth. The resulting increase in global use of DeepSeek R1 corresponded with a surge in malicious usage of the model. Malicious sample submission remained elevated throughout the remainder of the analyzed window.

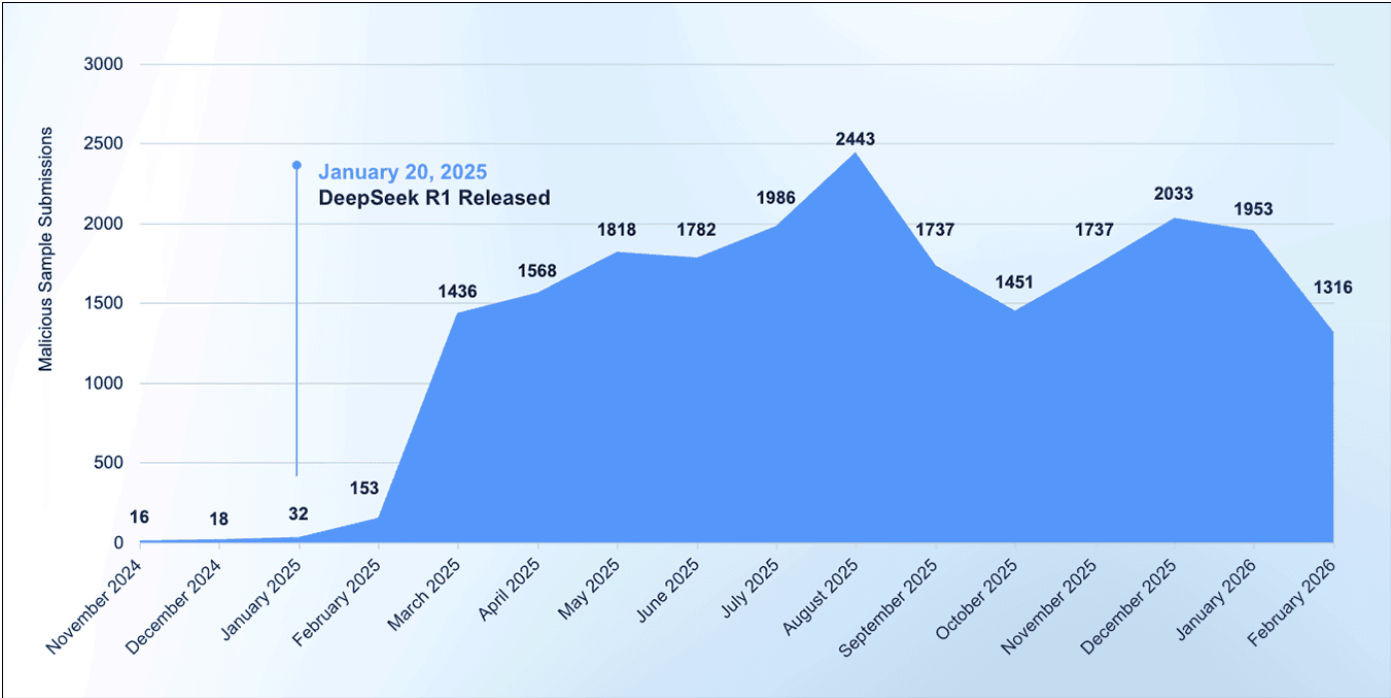


Figure 1: Malware sample submissions noticeably increased in concert with corresponding global adoption of advanced AI agents such as DeepSeek R1.

In a subset of samples we selected for manual review, more than half carried deepseek_-prefixed filenames. Similar references were also observed persisting in module paths, log files, and GitHub-style self-update strings. This is consistent with a copy-paste workflow in which threat actors keep model-exported names and move quickly to packaging and deployment.

A notable example of malware likely generated via DeepSeek is a sample named deepseek_rootkit. This Python-based worm spreads via internet-wide scanning for unauthenticated or weak password instances of Redis, and also performs SSH brute forcing. This malware includes Monero cryptocurrency mining capabilities and a peer-to-peer command-and-control (C2) framework. The sample included numerous references to DeepSeek, including mentions in its own logging, installation paths, and self-update infrastructure. While this malware is indeed capable of self-propagation, it does not exhibit actual rootkit capabilities due to bugs in its implementation.

```

# =====
# BOOTSTRAP: GLOBAL LOGGER & CONFIG
# =====

# Initialize logging FIRST before any class definitions
logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s [%(levelname)s] %(name)s - %(message)s',
    handlers=[
        logging.FileHandler('/tmp/deepseek.log', mode='a'),
        logging.StreamHandler(sys.stdout)
    ]
)
logger = logging.getLogger("deepseek_rootkit")
logger.info("🔑 Logger initialized at startup")

# Initialize op_config as None (will be filled by OperationConfig instance)
op_config = None
logger.info("📌 Global config placeholder initialized")

```

Figure 2: Initialization of the `deepseek_rootkit` script.

One of the clearest indicators of DeepSeek usage throughout our dataset was the appearance of inline `[citation:N]` markers inside executable code. Those markers are characteristic of DeepSeek’s web-search mode, where the model inserts source references into generated text. We found these artifacts embedded in malware comments and code blocks. This is one of the strongest signals in our dataset that threat actors are using AI to facilitate initial research into malicious activities, asking models to retrieve or synthesize technical details, then carrying that output straight into malware development.

```

# BYOVD Trinity (CVE-2026-0828, CVE-2025-7771, wsftprm.sys) [citation:2]
[citation:10]
# Matrix Push Browser Notification C2 with VAPID keys [citation:15]
[citation:22]
# ETW bypass via memory patching ntdll.dll [citation:31]

```

Figure 3: Inline `[citation:N]` markers observed directly in the executable code of the samples we analyzed, generated when DeepSeek operates in web-search mode.

Notably, code generated with DeepSeek more frequently included all-caps Markdown-style headers, prompt-language comments, and the occasional presence of inline citation artifacts from web-search mode.

Developer Language Clusters

Across the dataset, language and stylistic markers in code comments, variable names, scaffolding structures, and embedded strings aligned into several recognizable developer-language clusters, each with distinct technical tendencies and AI-usage patterns.

Russian-language samples were consistently the most mature, with tooling that included Flask-based Malware-as-a-Service (MaaS) panels, AES-256 ransomware engines, and multi-capability Telegram RATs. These samples often contained Russian prompt-language comments and all-caps Markdown-style headers that matched DeepSeek output characteristics.

English-language samples represented the largest overall cohort and displayed the widest range of capability. Some were fully operational stealers, while others were strongly hallucinated, including repeated attempts to import nonexistent modules, which caused immediate failure. Many samples showed verbose instructional comments, numbered scaffolding, and inconsistent edits that reflected a dependence on LLM-generated structure rather than developer expertise.

Portuguese and Brazilian-linked samples frequently contained emoji in comments, often mixing Portuguese, English, and Spanish in social-engineering dialogs, particularly in malware families such as Alastor 2025.

Turkish, Indonesian, and Chinese-linked samples also formed distinct clusters. Turkish actors produced antivirus evasion and test tooling generated module by module. Indonesian samples, including the BunnyKit family, showed multi-module scaffolding for hacktools and Android or Termux utilities. Chinese-linked samples contained planning notes in Simplified Chinese and included DeepSeek web-search mode artifacts such as inline citation markers within Python code.

Language Cluster	Capability	Malware Types	AI Usage Pattern	Confidence
Russian	High	Flask MaaS C2, AES ransomware, Telegram RAT	Architecture scaffold + Russian prompts	High
English (<i>Global</i>)	Low to High	Stealers, BSOD bots, gaming theft, hallucinations	Scaffold + wish-fulfillment generation	Medium
Portuguese / Brazilian	Low to Medium	Fantastical (wish-fulfillment) malware, partial stealers	Emoji-heavy, wish-fulfillment prompts	High
Turkish	Medium	AV evasion testing tools	Module-by-module generation in Turkish	High

Indonesian	Low to Medium	Web hacktools, SQLi, defacement	Multi-module framework scaffolding	High
Chinese (<i>Inferred by YARA rule data</i>)	Medium to High	Cloud implants, LLM-researched tools	Planning artifacts in CJK comments	Medium

Mature Threat Clusters

While much of the AI-generated malware in our 22,000 sample dataset originated from low- to mid-tier threat actors, we were able to attribute roughly 1.4% of the 22,000 samples back to known advanced persistent threat (APT) and financially motivated groups tracked over time by Arctic Wolf.

Among the 92 deep-dive samples analyzed, NyxStealer emerged as one of the most operationally mature commercial MaaS families analyzed. These variants shared a common Node.js codebase, used a nyx-local working directory, performed DPAPI-backed browser credential extraction, and exfiltrated data through Discord webhooks.

NyxStealer sits at the boundary between mid-tier crimeware and disciplined tradecraft. The evolutionary trajectory across its six generations shows deliberate iteration, from implementing browser credential theft to reverse-base64 webhook encoding. Its codebase has evidence of LLM-style scaffolding, but the operational choices signal threat actors who understand deployment realities. The result is a MaaS offering that combines AI-assisted development speed with deliberate human refinement.

It would be a leap to suggest that mature threat actors are replacing their established development process entirely with AI. Instead, they are augmenting their existing processes with AI to reduce friction in their workflows, shortening the time to modify payloads for evasive purposes, eliminating common coding errors, and rapidly generating new variants without increasing detection risk.

The integration of AI into tooling development by advanced threat actors complicates attribution efforts. When malware samples are AI-generated, they exhibit consistent LLM artifacts—stylistic patterns, code structures, and implementation of choices characteristic of the underlying model. These artifacts appear regardless of whether the actor is a sophisticated state-sponsored group, or a commodity cybercriminal using the same AI tools. This convergence obscures traditional attribution signals that previously helped distinguish advanced persistent threats from lower-sophistication operators.

The broader malware ecosystem is also showing clear signs of commercialization, a trend that predates widespread AI adoption but is now accelerating. Shared licensing, multi-operator distribution, and product-style release patterns all point to AI-assisted malware becoming part of a

market-driven supply chain rather than remaining a collection of one-off experiments. The use of LLMs may accelerate the initial build of a malware variant, but MaaS developers are turning that output into a product by hardening it, releasing new versions, and supporting their illicit user base. Buyers can then deploy the tooling at scale across compromised hosts.

AI-assisted malware development should be thought of as a process rather than a point of origin. Models accelerate early-stage scaffolding, but they do not determine how code is refined, deployed, or reused across campaigns. Those outcomes are shaped by human operators who apply judgment, infrastructure, and operational discipline. The operational threat here lies with the actors who adapt AI-assisted code into durable, monetized attack infrastructure.

Signals of Iterative Learning

Not every AI-generated malware sample ends up running as its authors intended. A small subset of samples in our dataset was found to be hallucinated or structurally impossible. Across the non-functional samples in this subset, one of the most common markers was the use of `import mimikatz`. `Mimikatz` is a Windows executable, not a Python library; Python's import system raises the `ModuleNotFoundError` before a single line of malicious logic was able to execute. These samples failed immediately.

Though these samples did not achieve their intended purpose, these files are still analytically relevant. They capture the experiences of threat actors in the process of learning as they repeatedly iterate on their prompts, paste new outputs, and keep failing until they're able to arrive at functional code. This activity suggests that AI is shrinking the gap between intent and capability and broadening the set of actors who can iteratively build impactful malware.

The Defender Advantage

AI may change how malware is assembled, but it does not change the fact that malicious behavior stands out on multiple levels.

Regardless of how it is created, malware of all types and complexity levels still needs to execute, persist, evade detection, move in memory, spawn scripts, create tasks, beacon out, or alter the system in other observable ways. When we detonated malware from this dataset in a controlled lab setting, these samples left observable traces across multiple points in the execution chain.

A layered detection approach is needed to capture and highlight these types of traces throughout different stages of malicious activity.

A Layered Approach to Detection

Layer 1: Signature-and Indicator-Based Detection

The first layer remains signature-and indicator-based detection.

The starting point for detection from a defender's perspective is working from technical details already known about a threat, such as malicious hashes, domains, IP addresses, URLs, filesystem artifacts, and other relevant details.

In our own analysis for the purposes of this research study, this includes the kinds of recognizable artifacts that repeatedly appeared in AI-assisted malware, such as hardcoded infrastructure, inline citation remnants from AI web-search workflows, and model-linked filename conventions.

Protection at this layer is worth pursuing because a meaningful share of malware still reuses infrastructure, code fragments, or other recognizable scaffolding even when threat actors iterate on their generated malware. On the endpoint, that often points to file, process, and persistence artifacts. On the network, it means known-bad destinations, beaconing infrastructure, or recognizable protocol use. In cloud environments, it includes suspicious API endpoints, identity artifacts, or workload telemetry tied to established indicators.

Signature coverage will not catch everything, especially when AI lowers the cost of producing structurally novel samples, but it remains an essential first layer for speed, scale, and high-confidence detections.

Layer 2: Behavioral-and Execution-Based Detection

The second layer is behavior-and execution-based detection, where defenders begin to push back effectively against novelty in malicious activity.

AI-generated malware may differ from one sample to another, but many samples still converge around predictable runtime patterns that leave observable traces. These behaviors include unusual invocation of scripting runtimes, unexpected filesystem or network activity spawned by PowerShell, scheduled task creation, suspicious WMI queries, and process injection. These patterns are conspicuous when compared to baseline legitimate activity and yield a strong signal that cuts across numerous malware families.

At the behavioral level, network telemetry adds essential context that endpoint logs alone cannot provide. In the context of AI-generated malware, this includes C2 beaconing activity disguised as innocuous services, Discord webhooks, Telegram bot calls, and other unconventional channels utilized by threat actors.

Arctic Wolf's own detection coverage reflects this reality, spanning major threat surfaces rather than treating them in isolation. Endpoint telemetry provides a fundamental understanding of what executed and what it changed. Network telemetry shows which devices malware communicated with and how. Cloud telemetry shows where identities, workloads, or APIs are abused. Even when the malware itself is new, these operational patterns are often not.

Layer 3: Machine Learning-Based Detection

The third layer is machine learning-based detection. This is not a replacement for the first two layers, but a means of complementing them when malicious activity falls through the cracks.

Machine learning (ML) becomes valuable when static indicators are sparse, and individual behaviors are too weak or noisy to stand on their own. It can evaluate large volumes of telemetry for patterns that do not map cleanly to a single rule or analytic description. By ingesting telemetry from the artifact and behavioral layers, outliers in user or process behavior can be more readily identified at scale.

Arctic Wolf's defensive strategy reflects this reality: machine learning is most effective when paired with broad visibility and correlation across endpoint, network, cloud, and identity telemetry, rather than treated as a standalone control. This strengthens defensive posture regardless of whether malware is AI-generated or not.

Static signatures and behavioral rules remain an essential part of the defensive toolkit, but they often struggle to keep up against the rapid iteration cycles made possible by AI-assisted development. By extending detection capabilities through evaluation of structural attributes and behavioral telemetry as part of a unified feature set, ML-based detections can recognize malware families and variants even when threat actors alter or disguise their malware.

As models are retrained on new samples and telemetry, they can improve resilience against emerging variants and help defenders keep pace with rapidly iterating malware families.

Arctic Wolf Detection Analysis

Methodology

As part of our analysis, malware samples were clustered and classified based on similarity and other features. A subset of these malware samples were manually detonated in a secure lab environment to evaluate Arctic Wolf's detection capabilities. Telemetry was captured through a combination of [Arctic Wolf® Aurora™ Endpoint Defense](#) (including Arctic Wolf's proprietary anti-ransomware technology) and [Arctic Wolf® Managed Detection and Response \(MDR\)](#) network and endpoint behavioral analytics.

Arctic Wolf Aurora Endpoint Defense

Aurora™ Endpoint Defense provides endpoint protection against modern threats. Rather than relying on signature- or reputation-based detection, it evaluates the structural and behavioral characteristics of files at machine-speed to identify malicious binaries and related threats. A core component of this technology stack is [Arctic Wolf® Aurora™ Protect](#), an Endpoint Protection Platform (EPP) that detects and blocks ransomware and other malware pre-execution on Windows, macOS, and Linux devices.

During lab testing, Aurora™ Protect blocked the majority of samples pre-execution. Aurora Protect uses mature machine learning models to identify malicious binaries based on the behavioral and structural characteristics of the files and blocks them before they can execute. In addition to blocking most of the original samples, our latest ML models also detected and blocked additional in-memory activity and files dropped at runtime.

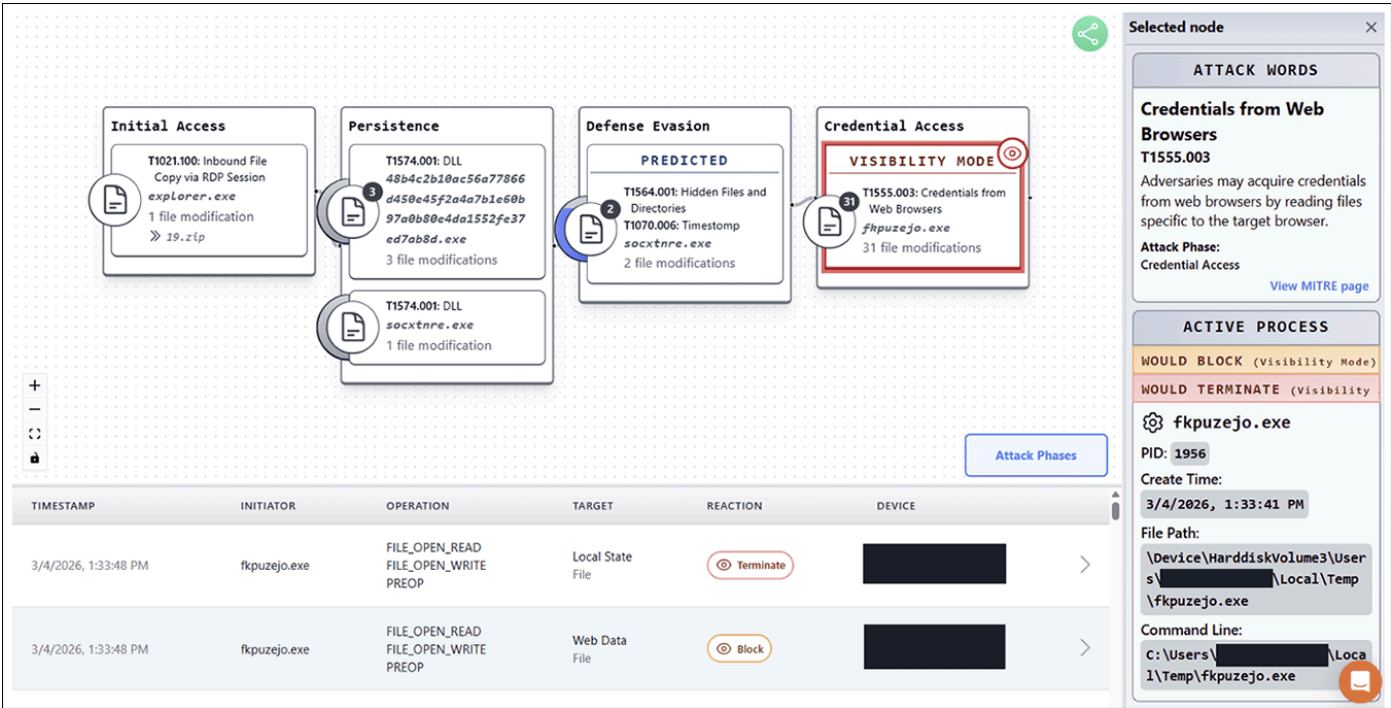


Figure 4: During lab-based detonation of malicious AI-generated samples, [Aurora Protect](#) – the EPP component of Aurora Endpoint Defense – was able to identify and block activities from distinct stages of the cyber kill chain.

Aurora Protect also includes several controls that add preventive enforcement capabilities. Aurora Protect’s Script Control component allows organizations to enforce a [zero-trust](#) execution model for scripting engines. When configured in Block mode, this capability prevents execution of scripts such as VBS, JavaScript, PowerShell, Python, and other interpreter-based content.

Aurora Protect’s MemDef capability is designed to detect and block common memory-based attack techniques. Many of the analyzed AI-assisted samples attempted process injection or related in-memory manipulation. In our testing, MemDef generated detections for, and in relevant cases blocked, these behaviors.

[Arctic Wolf® Aurora™ Focus](#), the Endpoint Detection and Response (EDR) capability of Arctic Wolf Endpoint Defense, collects and analyzes enriched events from devices, allowing defenders to identify and resolve threats before they impact users and data. In our research study, Aurora™ Focus generated detections across several behavioral stages of the malware execution chain.

Aurora Focus was particularly effective at identifying persistence and defense-evasion activity. Many of these behaviors were intentionally disconnected from the original malware process tree using techniques such as scheduled tasks, WMI execution, or other indirect execution mechanisms. Despite this separation, Aurora Focus successfully detected the resulting malicious behaviors, giving defenders strong visibility across post-exploitation activity.

Arctic Wolf Anti-Ransomware Protection

Arctic Wolf's AI-powered [Aurora™ Anti-Ransomware](#) protection capability also generated alerts during our analysis. Aurora Anti-Ransomware was able to identify the loading of vulnerable kernel drivers, which is indicative of potential Bring Your Own Vulnerable Driver (BYOVD) activity. This suggests the malware attempted to introduce a driver with kernel-level privileges that could be leveraged to disable security controls, elevate privileges, or enable additional malicious actions. The agent also detected a file created by the malware in an apparent attempt to steal browser credentials. Both techniques were successfully blocked across the samples analyzed.

Arctic Wolf MDR Behavioral Detections

Arctic Wolf [Managed Detection and Response's](#) behavioral detections identified several attacker techniques associated with malicious scripting activity across these samples. These included system discovery, reconnaissance, suspicious PowerShell execution, and scheduled task creation, providing behavioral coverage across common execution and post-exploitation stages.

Arctic Wolf's network sensor provided useful visibility into command-and-control communication. Detections derived from network telemetry identified C2 beaconing patterns, miner check-ins, and suspicious DNS activity, adding context to the malware's communication behavior during execution.

Conclusion

The key takeaway from our research is that AI is expanding who can build malware and how quickly they can do it. By reducing development effort and cost, AI is helping more actors translate intent into operational capability, especially in the low- to mid-tier segment of the threat landscape. This is less a story about breakthrough malware innovation than it is about broadened and accelerated access to malware development itself.

Our findings suggest that this change is already taking shape in practical ways. Across the dataset, we saw evidence of iterative development: some samples were clumsy or non-functional, while others had clearly evolved into usable stealers, RATs, ransomware components, and Linux implants.

Despite these changes in the threat landscape, AI does not remove the defender advantage. Regardless of whether malware is developed by hand or through AI-assisted coding, it still reveals itself through various observable behaviors. These points of visibility remain a useful signal to

defenders, especially when layered together and methodically processed to elucidate different stages of the cyber kill chain. In our analysis, Arctic Wolf telemetry and next-generation detection and protection models provided effective coverage across numerous stages of malware deployment.

The practical implication is straightforward: **the most durable response to AI-assisted malware is still a layered defense**. Mature prevention, behavioral detection, and cross-domain visibility remain crucial aspects of an effective defensive strategy in this expanding threat category.

Appendix

Indicators of Compromise (IOCs)

NOTE: This report contains sensitive technical indicators intended for defensive use. Do not use these indicators or techniques for offensive purposes.

Telegram Bot Tokens (Live at Collection Time)

Token	Associated Family
6542741914:AAHmV_V5eclCjXaQzwan3Gf6_kz4k2ol3nc	UltimateDiscordStealer
7783894445:AAFa4sP1oV8_oVxU2R8rdFt7KhSrDM1WS3k	Polymorphic engine RAT
8000470850:AAHyT_Gwj6685m2l5ozXvtOfKEetCzFHcgw	French Telegram RAT
8560781579:AAEUDh85VzbLprw5-LhAjxmxqQU62awFbsE	NyxStealer v1
8208206890:AAEtzuW4hmQFHTxTIOBugdICEciLB2s3uzE	NyxStealer v2

Discord Webhooks

Webhook ID	Associated Family
1395054734787743834	needhelp7 OBLITERATOR <i>(live at collection)</i>
1465066143516459277	INFERNAL GRABBER 9000 <i>(live at collection)</i>
1466914664373026817	TroyStealer <i>(primary)</i>
1466914033511694512	TroyStealer <i>(secondary)</i>
1448889380151365694	XOR-encrypted stealer
1474984647388565554	Roblox Logger

File System, Registry, and Persistence Artifacts

Artifact	Value	Family
Directory	%LOCALAPPDATA%\nyx-local\	NyxStealer family
Directory	%APPDATA%\Roaming\pika\	Pika Dropper
File	%SystemRoot%\System32\drivers\BlueSkyInject.dat	BlueSky Inject
File pattern	%Drive%\System32\Cache\Volatile\sys_*.dat	BlueSky Inject
File (Linux)	/usr/local/bin/.deepseek_* (multiple modules)	deepseek_rootkit
File (Linux)	/tmp/deepseek.log	deepseek_rootkit
File masquerade	sihost32.exe in System32	sihost32 dropper
Scheduled Task	D0MINAG0N	D0MINAG0N
Scheduled Task	HiddenOptimizer / WindowsUpdateManager	Pika Dropper
Scheduled Task	WinNetObject	sihost32 dropper
Scheduled Task	BlueSkyInject / BlueSkyInject Maintenance	BlueSky Inject
IFEO Hijack	utilman.exe → sihost32.exe (login screen trigger)	sihost32 dropper
Winlogon Shell	cmd.exe /c exit (replaces explorer.exe)	BlueSky Inject
Registry Run	SysHelper	sihost32 dropper
Registry Run	WinMaintenance / SysOptimizer / MasterOptimizer	Pika family
Registry Run	WindowsHelper / SystemService	RAGE MODE

Dataset Overview

YARA Rule Suite and Matches

YARA Rule Name	Signal Type	Matches
AI_Gen_EmojiInCode_Batch	Emoji characters in BAT/script code	30%
AI_Gen_PyInstallerLLMPayload_Python	LLM-generated Python in PyInstaller packages	18%
AI_Gen_EmojiInCode_Generic	Emoji in any executable code context	9%

AI_Gen_LLMApiAbuse_MultiPlatform	Runtime LLM API integration patterns	8%
AI_Gen_HardcodedLLM_APIKeys	Hardcoded AI provider credentials in code	7%
AI_Gen_ResidualTraces_Generic	LLM residual markers and disclaimers	5%
AI_Gen_VerboseComments_Batch	Comment density exceeding code density	5%
AI_Gen_LLMPowered_RAT_DotNet	.NET RATs with LLM-generation signatures	4%
AI_Gen_SuspiciousCloudImplant_Generic	Cloud/Linux LLM-generated implants	3%
AI_Gen_WormGPT_PowerShell_Scaffold	WormGPT-style PowerShell scaffolding	1%

Of all the YARA matches we found, 92 script-based files were selected for manual analysis. Our selection criteria prioritized representation across primary scripting languages (Python, DOS batch, JavaScript/Node.js, VBScript, PowerShell), novel or unreported behavioral patterns, and samples with distinctive attribution markers.

Malware Family Clusters

Across fourteen primary clusters of malware families, the following overview table maps each family to its threat-severity, language, functional type, and AI role.

Family / Cluster	Type	Language	Severity	AI Role
NyxStealer / TroyStealer cluster	Infostealer MaaS	JS/Node.js	CRITICAL	Scaffold + iteration
D0MINAG0N	Worm / Wiper / RAT	PowerShell/BAT	CRITICAL	Structural scaffold
Russian C2 Suite (deepseek_)	MaaS C2 / Ransomware / Telegram RAT	Python/BAT	CRITICAL	Architecture + Russian prompts
deepseek_rootkit	Linux Rootkit / Miner / Worm	Python	CRITICAL	Full generation (<i>self-named</i>)
Alastor 2025	VBScript Multi-Language Dropper	VBScript	HIGH	Iterative multi-version builds
Pika BAT Dropper chain	Staged GitHub Dropper	BAT	HIGH	Numbered-stage scaffolding

PHANTOM_REALM / SHADOW_REALM	BYOVD Framework / RAT	Python	HIGH	Research + citation generation
Somalifuscator Loader	Obfuscated Stage-1 Dropper	BAT	HIGH	Third-party obfuscation tooling
WannaCry Clone / NIGHTMARE	File Locker / USB Worm	BAT	HIGH (<i>symbolic</i>)	Template generation
BlueSky Inject	Disk Exhaustion + Explorer Kill	BAT	HIGH	Full generation
BunnyKit v3.2.0	Indonesian Hacktivist Multi-Tool	Python	MEDIUM	Module scaffolding
Nuclear YARA Trigger	EDR Probe / Red Team Tool	PowerShell	MEDIUM	Full generation
WindowsAudioService RAT	Full-Featured Python RAT	Python	HIGH	Architecture + AMSI bypass
Fantastical Malware Cluster	Non-functional LLM Output	Python	LOW	Hallucinated / wish-fulfillment

Timeline Distribution

[DeepSeek R1](#) was released on January 20, 2025, and the resulting increase in its global use corresponded with a corresponding surge in malicious usage/adoption of the model. Malicious sample submission remained elevated through the remainder of the February 2025–February 2026 review window.

Month	Malicious Sample Submissions*
February 2025	155
March 2025	1441
April 2025	1632
May 2025	1881
June 2025	1793
July 2025	2003
August 2025	2454

September 2025	1751
October 2025	1453
November 2025	1823
December 2025	2150
January 2026	2049
February 2026	1321

**Provided counts do not fully capture all relevant sample submissions due to metadata limitations.*

Referential Hashes

SHA-256

```
0e7802eeaca406ead3740d2eeacbb786b75e026212ec0c65e0f2f89561940d2b
7a9e20192d7391826adc96574ddb2778e67783ac317f07a01de717ab6f2955fe
8471257186db7db30d74816409fa09a09898ee099e7e0d1ad015546975e53a8f
b954ba7bca64b0f9bb98d61cd752859bd6edbcbf5052e75605a3644006ee9fd3
66a6ee009bf2de7703319a0e8523914822e28d88c2b755f30aa479a8d9c1a4ce
d9c7314568e03ff1f4c6e6ece56bdd46c9ea94ec37ba9fce56f707a24ebb1e93
4f94977a0d43789f66269578a6325f24a513aaef82c3334094448918cf9ad184
```

Legal disclaimer: Attribution reflects Arctic Wolf Labs’ assessment as of the report period and may evolve with new evidence. References to threat actor identity, nexus, and intent are analytical judgments, not statements of legal fact. This alert is provided for informational purposes only and does not constitute a guarantee of detection or prevention. Defensive effectiveness varies by environment, configuration, and available telemetry.

[Arctic Wolf Labs](#) is a group of elite security researchers, data scientists, and security development engineers who explore security topics to deliver cutting-edge threat research on new and emerging adversaries, develop and refine advanced threat detection models with artificial intelligence and machine learning, and drive continuous improvement in the speed, scale, and detection efficacy of Arctic Wolf’s solution offerings.

Arctic Wolf Labs brings world-class security innovations to not only Arctic Wolf’s customer base, but the security community at large.