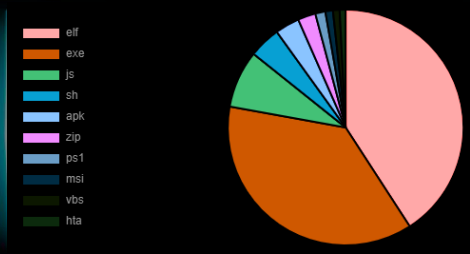

SORINTSEC
THREAT HUNTING
LAB
PowerShell P/Invoke of CredEnumerate for Credential Manager Extraction
PUBLIC

# Index

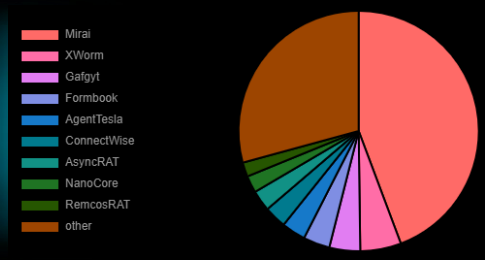
# Global Weekly Threat Overview

Threat actors are exploiting a recently disclosed critical security flaw in Ghost CMS to inject malicious JavaScript code with an aim to fuel ClickFix attacks. According to QiAnXin XLab, the activity involves the exploitation of CVE-2026-26980, an SQL injection vulnerability in Ghost's Content API that could allow an unauthenticated attacker to read arbitrary data from the database. The security flaw was addressed in February 2026 in version 6.19.1. The vulnerability was discovered by Anthropic using Claude.

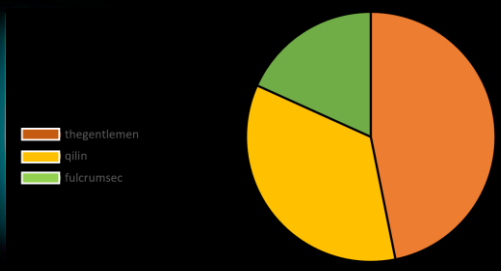
Top 10 file types




Top 10 malware family




Top 3 Ransomware Group




Cybersecurity researchers have shed light on a cross-platform malware called RemotePE that has been put to use by the North Korea-linked Lazarus Group in attacks targeting financial and cryptocurrency organizations. RemotePE is part of a multi-stage attack chain that involves two loaders tracked as DPAPILoader and RemotePELoader. DPAPILoader decrypts and loads RemotePELoader from disk using the Windows Data Protection API (DPAPI). RemotePELoader beacons to a C2 server and waits until it receives the next stage: RemotePE, a RAT executed entirely in memory and never written to disk, leaving no filesystem artifacts.

# Global Weekly Notable One

## Credential Access: Credentials from Password Stores

In one of the latest campaigns threat actors abused SEO poisoning and typosquatted domains to impersonate Gemini CLI and Claude Code, targeting developers and technically inclined users searching for legitimate AI tooling online. Instead of relying on traditional phishing attachments or obvious malware droppers, the operators used fake installer pages that instructed victims to copy and paste malicious PowerShell commands directly into their terminal, blending the attack into a workflow that many developers already consider normal and low risk.

Within that broader campaign, one of the most interesting post-exploitation behaviors is the use of PowerShell P/Invoke of CredEnumerateW to access credentials stored in the Windows Credential Manager. By using an inline C# and define DllImport wrappers the script can invoke the native Windows API directly from PowerShell and enumerate credential entries available in the current user's logon session.

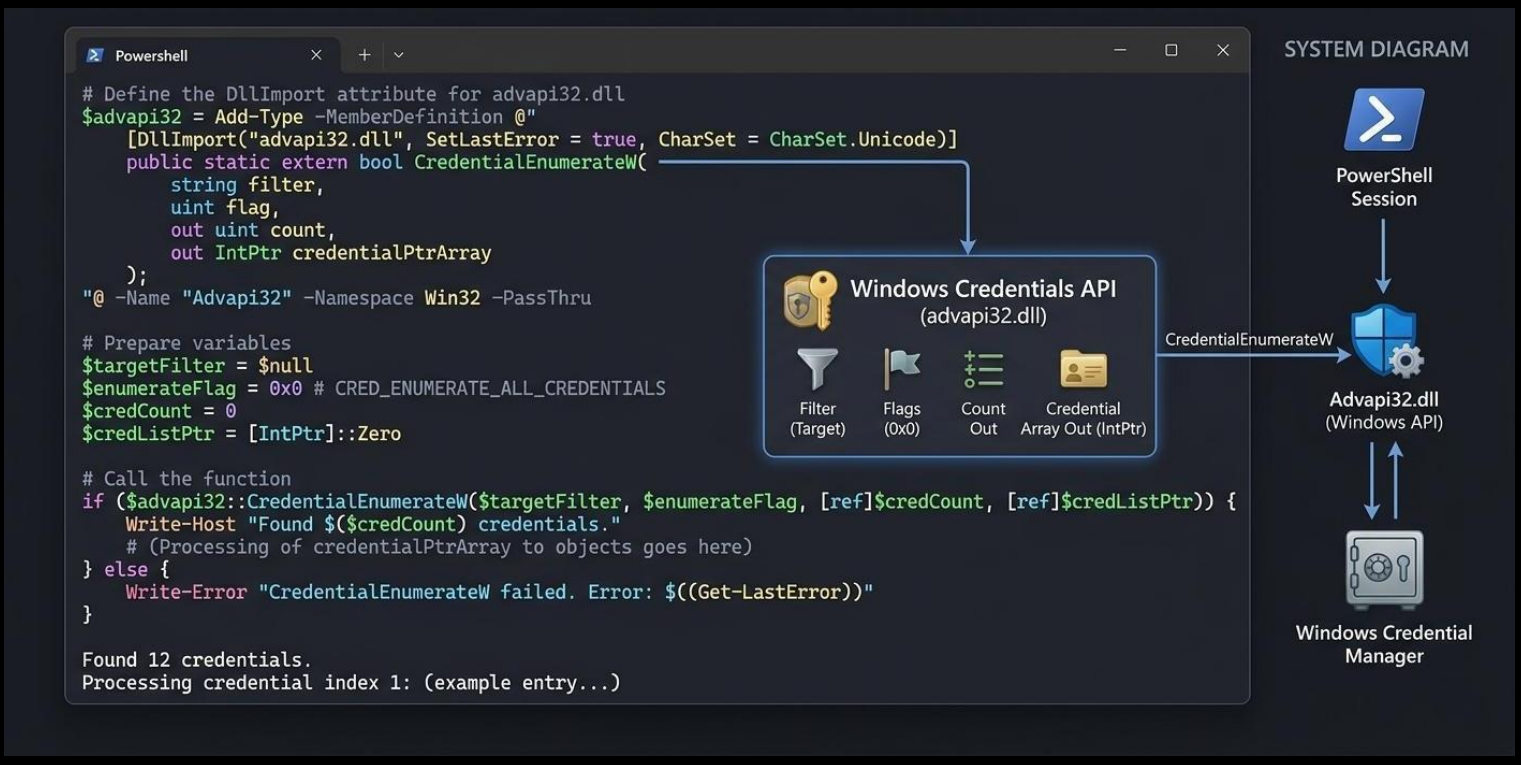
# Threat Hunting Activity

| | |
|---|---|
| **TACTIC** | Credential Access |
| **TECHNIQUES** | T1555 – Credentials from Password Stores |

Adversaries may acquire credentials from the Windows Credential Manager. The Credential Manager stores credentials for signing into websites, applications, and/or devices that request authentication through NTLM or Kerberos in Credential Lockers. The Windows Credential Manager separates website credentials from application or network credentials in two lockers. As part of Credentials from Web Browsers, Internet Explorer and Microsoft Edge website credentials are managed by the Credential Manager and are stored in the Web Credentials locker. Application and network credentials are stored in the Windows Credentials locker.

# Threat Hunting Activity

An interesting behavior of this campaign is the use of PowerShell P/Invoke to reach CredEnumerate and pull stored credentials from Windows Credential Manager. PowerShell is used not just as a scripting language, but as a runtime that can compile and execute inline C# code through Add-Type. That inline code declares native function imports with DllImport, typically targeting advapi32.dll, so the script can call CredEnumerateW directly from the Windows API. CredEnumerateW is designed to enumerate credentials stored in the Windows Credential Manager for the current logon session.



# Threat Hunting Activity

Detection can be based on Powershell script block telemetry containing P/Invoke declarations targeting advapi32.dll's CredEnumerate function.

**Script Block Text**

```
1  $advapi32 = Add-Type -MemberDefinition @"
2  [DllImport("advapi32.dll", SetLastError=true, CharSet=CharSet.Unicode)]
3  public static extern bool CredEnumerateW(string Filter,
4    int Flags, out IntPtr Count, out IntPtr CredentialPtr);
5  [DllImport("advapi32.dll")]
6  public static extern void CredFree(IntPtr Buffer);
7  [DllImport("kernel32.dll", SetLastError=true, CharSet=CharSet.Ansi)]
8  public static extern IntPtr GetProcAddress(IntPtr hModule, string procNam
9  "@ -Name WinCredentials -Namespace Win32 -PassThru
```


PUBLIC