



In malware analysis, it is tempting to describe change as innovation.

New tricks, new tooling, new malware families.

What is far more revealing, however, is *how little actually changes* and **what changes anyway**.

Between late November and the end of December 2025, several Gamaredon-related VBScript loaders surfaced that are, functionally, almost identical. They all execute the same mechanism, rely on the same execution primitive, and ultimately aim for the same outcome.

And yet, something *does* change, quietly, incrementally, and very deliberately.

This article focuses strictly on **observable, concrete shifts in obfuscation**, not assumptions, not intent inferred from tooling, and not architectural leaps that are not supported by the samples themselves.

For this analysis, I used the following samples:

Date of Sample	Hash	Download
19.12.2025	6de9f436ba393744a3966b35ea2254dde2f84f5b796c6f7bee4b67ccf96ccf0a	Download
22.12.2025	9218528a40a48a3c96df3b49a7498f6ea2a961f43249081b17880709f73392c1	Download
25.12.2025	2c69fd052bfaa03cd0e956af0f638f82bc53f23ee8d0c273e688e257dac8c550	Download
30.12.2025	846748912aa6e86b9d11f6779af6aae26b7258f8610d5e28eff0083779737824	Download

25 November 2025 – Noise Without Structure

The earliest sample is dominated by volume.

Characteristics:

- Hundreds of variables that are written to once and never read again
- Repeated arithmetic mutations ($x = x + 14$) without semantic relevance
- Long linear execution flow
- No variable declarations (`Dim` entirely absent)

```

278 detectivesadn = detectivesadn + "indefiniteM98"
279 emotionalGMk = emotionalGMk + credentialsrJT
280
281
282 emotionalGMk = emotionalGMk + flightfUP
283
284
285 insultt7T = insultt7T + 22
286
287 processionueu = divideddtV
288 feedwZ1 = chainIQn
289
290
291 cadenceo0A = backedMYl
292
293
294 paragraphdQz = paragraphdQz + "K9"
295 paragraphdQz = paragraphdQz + "A0"
296 paragraphdQz = paragraphdQz + "Bg"
297 paragraphdQz = paragraphdQz + "D"
298
299 cadenceo0A = cadenceo0A + attendanceb6f
300 flexiblesFg = "expulsionEj4"
301
302 cadenceo0A = cadenceo0A + dollarsRx1
303 canenKY = canenKY + byeIOA
304
305 exclusiveF0C = exclusiveF0C + "ky"
306 exclusiveF0C = exclusiveF0C + "VzP"
307 exclusiveF0C = exclusiveF0C + "n"
308 exclusiveF0C = exclusiveF0C + "q"
309
310 cadenceo0A = cadenceo0A + captiveEqC

```

The obfuscation here serves one purpose only: **syntactic noise**.

There is no attempt to:

- Hide control flow
- Delay string resolution
- Reconstruct logic conditionally

Everything is present in the source, just buried under irrelevant assignments.

From an analyst's perspective, this sample is noisy but predictable.

Once dead code is ignored, execution logic collapses into a short, linear sequence.

19 December 2025 – Indicator overload

The mid-December sample introduces a clear and measurable change: **indicator density**.

New observations:

- A significant increase in hard-coded URLs
- URLs pointing to unrelated, legitimate, and state-adjacent domains
- No execution dependency on most of these URLs

```
88 burstih9 = "https://desolatew2k@www.unn.com.ua/pleasant/little.pdf"
89 combinationuJZ = combinationuJZ + flawHfA
90 recalledsr5 = recalledsr5 + 91
91 acquaintance81 = acquaintance81 + 93
92 combinationuJZ = combinationuJZ + aweIm4
93 counterdCG = coalTgP
94
95 combinationuJZ = combinationuJZ + somevmT
96
97
98
99 riverfTa = jobsJ4A
100 dumpG6V = "https://brainWN3@iz.net/stealing/overthrow.pdf"
101 presumebq7 = evaluationmKj
102 riverfTa = riverfTa + proposedfHC
103
104 cellel1 = cellel1 + "vQ"
105 cellel1 = cellel1 + "x"
106 cellel1 = cellel1 + "p"
107 cellel1 = cellel1 + "S"
108
109 hannahZcZ = blabauv
110 riverfTa = riverfTa + stoolVSG
111 herQ73 = displaydGY
112 fistdJ4 = conquestHOP
113 riverfTa = riverfTa + beatengk4
114
115 sophisticatedRie = sophisticatedRie + lawsLkH
116 riverfTa = riverfTa + claimedhJI
117
118
119 sightb2K = sightb2K + discretionR5e
120 accompanyingffw = 34
121
122
123 accustomedyuw = "https://tdvdnpvimnponczkizwe.supabase.co/functions/v1/clever-endpoint/Odessa_19.12.2025/ambulance.pdf"
124 compatibility510 = compatibility510 + frontd3C
```

Crucially, these URLs are **not obfuscated**. They are placed in plain sight.

This is not string hiding it is **indicator flooding**.

The obfuscation shift here is not technical complexity, but **analytical friction**:

- Automated extraction produces dozens of false leads
- IOC-based clustering becomes unreliable
- Manual triage cost increases without changing execution logic

The loader still behaves linearly. What changes is the *signal-to-noise ratio*.

22 December 2025 – Defensive Reaction at the Payload Layer

The December 22 sample is not an obfuscation milestone, but it *is* a defensive one.

From a loader perspective, almost nothing changes:

- The download URL is fully present and statically recoverable
- No additional string hiding or control-flow manipulation is introduced
- Execution remains linear and transparent

However, focusing solely on loader complexity misses the actual shift.

The real change happens at the payload layer

For the first time in this series, the loader delivers **GamaWiper instead of Pterodo** for Analysis environments.

This is not a neutral substitution.

As outlined in my earlier analysis of GamaWipers behavior, this payload is explicitly designed to:

- Disrupt analysis environments
- Destroy forensic value
- Punish execution in unwanted contexts

>> <https://blog.synapticsystems.de/gamawiper-explained-gamaredon-anti-analysis/>

In other words:

Gamaredon reacts defensively, just not in the loader yet.

Why obfuscation does not increase here

The absence of additional loader obfuscation is not a contradiction, but a signal.

At this stage:

- The defensive burden is shifted entirely onto the payload
- The loader acts as a transparent delivery mechanism
- Analysis deterrence is achieved through destructive behavior, not concealment

This suggests a deliberate sequencing:

1. **Introduce a hostile payload to counter analysis**
2. Observe detection and response
3. Only then begin hardening the delivery mechanism itself

Why this sample matters

~ December 22 marks the point where Gamaredon stops merely *being observed* and starts actively **responding**.

Not by hiding better, but by ensuring that *seeing the payload has consequences*.

The subsequent increase in loader obfuscation after this date does not replace this strategy. It complements it.

Payload hostility first.
Delivery hardening second.

Notably, this change occurs almost exactly four weeks after [my article](#) outlining practical approaches to tracking Gamaredon infrastructure went public.

Whether coincidence or feedback loop, the timing aligns remarkably well with the first observed deployment of GamaWiper as an anti-analysis response.

25 December 2025 – Control-Flow Noise Appears

The Christmas sample does not introduce new primitives, but it does introduce **execution ambiguity**.

Concrete changes:

- Multiple **.Run** invocations exist
- Not all of them result in meaningful execution
- Several objects and variables are constructed but never used
- Execution order is less visually obvious

```

382 kernelGqG = "https://strangerEoD@president.gov.ua/functions/v1/dynamic-api/maid.pdf"
383 impureX3p = impureX3p + ".DCo"
384 msD35 = "https://diJqh1HIP3mNY6217jk.supabase.co/functions/v1/fourth/can.pdf"
385 bidCGL = "https://tOK7sIiuVWqX470S.supabase.co/functions/v1/brake/cooling.pdf"
386 barnjW1 = barnjW1 + "antiquityCG3"
387
388 instinctiveSS8 = "https://n46xQYa0v15h.supabase.co/functions/v1/cage/skins.pdf"
389 completelyM5Q = refersys1
390 hypothesisl3v = "https://MiCccFoj96Uwmv9tGP.supabase.co/functions/v1/grounds/sole.pdf"
391
392
393 footprintx19 = footprintx19 + "e"
394 footprintx19 = footprintx19 + "wy9"
395 footprintx19 = footprintx19 + "T"
396 footprintx19 = footprintx19 + "d"
397
398 crookedRa1 = renderedSt1
399 accidentalvDN.Run expulsiono6L
400 circusZyD = "https://overwhelmingqQu@developer.mozilla.org/functions/v1/dynamic-api/la
401 measureYfH.Run arcb9Z
402 nestdmf = 14
403 predictionz0J.Run froqNT
404 simplifiedZv = simplifiedZv + nickwtk
405
406
407 cloakAp8 = "https://appeal-improved-including-anxious.trycloudflare.com"
408 awareKE6 = awareKE6 + fanciedV3H
409 perceivef5I = grayi7B
410 reportG1a = 19
411
412 appearanceS9A = quickerk9f
413 sandwichDDZ = "https://ridn2h@rg.com/functions/v1/dynamic-api/bout.pdf"
414
415 appearanceS9A = appearanceS9A + fannyZyd
416 sentencea5W = "https://selfishn4T@www.crimea.kp.ua/fortune/judgement.pdf"
417 dummyj56 = dummyj56 + silverYio

```

This is not branching logic, but **control-flow camouflage**.

The analyst can still reconstruct execution, but:

- Dead paths look plausible
- Execution sinks are no longer unique
- Automated heuristics struggle to identify the real one

The obfuscation no longer targets strings, it targets **execution clarity**.

30 December 2025 – Fragmented Runtime Assembly

The final sample introduces the most tangible structural changes.

Observed differences:

- Systematic use of `Dim` declarations
- Extensive use of short, non-semantic string fragments
- Assembly of execution-relevant strings via repeated concatenation across distant code sections
- No complete execution string exists statically
- Domains are just random invalid Domains

```

221 detrimental4c = detrimental4c + particularE0S
222
223 detrimental4c = detrimental4c + relationshipEoQ
224
225 hiA6x = hiA6x & squadeK3
226 concealWRG = "https://op5ol4pn.zn"
227 archedEj8 = "lDlYygtm9qnhT1nusoK"
228 clockvtI = "combatBV3"
229 descendedjb7 = "http://o021s.Gx"
230 remarkIst = "EIPRH0CqS18W0JMybjBHBmUgD80VpQp"
231 hiA6x = hiA6x & agonyBmJ
232 mentionYT6 = changesmvc
233 childrenFhF = childrenFhF + "onwardT6e"
234 eveningsH0x = eveningsH0x + "lessIAX"
235 duskxK = configurationop9
236 journeyNgr = journeyNgr + sleektTe
237 hiA6x = hiA6x & spokeU3Q
238 consentix4 = "geneuZi"
239
240
241 administrationsSp = "http://W352X3.AH"
242
243 developmentd41 = "/expressionz2R/lumberjp0"
244 instructafo = scheduleu8V
245 elementsCDA = "/costh1G/inevitablees7/ridiculousv1y"
246 stewdvB = poundsm1J
247
248
249
250 speedW05 = speedW05 + "Ww"
251 speedW05 = speedW05 + "qL"
252 speedW05 = speedW05 + "Vfx9"
253 speedW05 = speedW05 + "uQ"
254
255 instructafo = instructafo & darneC1
256 researchdVn = researchdVn + "Pi7034p97dpl"

```

At no point does the full execution command exist as a contiguous value in the source.

Instead:

- Fragments are combined
- Recombined
- Passed through intermediate variables

- Finalized immediately before execution

This directly degrades:

- Static string extraction
- Signature-based detection
- Regex-driven tooling

No encryption is added.

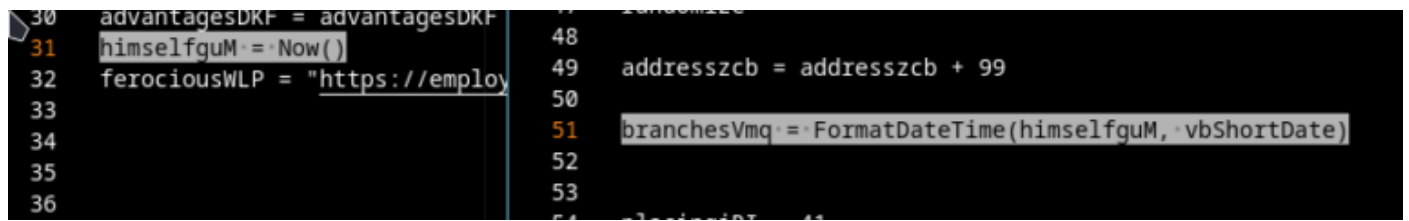
The shift is purely architectural.

05 January 2026 – Added Datetime Parameter to URL

EDIT 07.01.2026: I added this part as new findings appeared

Since early January, another small but relevant change appeared in the loader logic.

The scripts now generate a date value at runtime:



```
30 advantagesDKF = advantagesDKF
31 himselfguM = Now()
32 ferociousWLP = "https://employ
33
34
35
36
48
49 addressszcb = addressszcb + 99
50
51 branchesVmq = FormatDateTime(himselfguM, vbShortDate)
52
53
```

This value is then embedded directly into the download path, resulting in URLs like:

.../UkrNet_02.01.2026/cutting/02.01.2026/hannah8342.pdf

From a detection standpoint, this is subtle but effective

This means:

- payload paths change daily
- static URL signatures age out immediately
- and IOC reuse across campaigns becomes unreliable

Relation to Prior Observations

This behavior aligns closely with patterns discussed in my earlier article on **GamaWiper and Gamaredon's anti-analysis strategies**, where delivery behavior adapts based on perceived execution context.

<https://blog.synapticsystems.de/gamawiper-explained-gamaredon-anti-analysis/>

What Actually Changed and What Did Not

What did **not** change:

- Execution primitive
- Loader purpose
- Overall delivery mechanism

What **did** change:

- When execution-relevant strings become complete
- How many false execution paths exist
- How much irrelevant context surrounds the real logic

This is not a rewrite.

It is **iterative hardening**.

Conclusion

These samples do not demonstrate innovation.

They demonstrate **attention**.

Each iteration removes one assumption analysts rely on:

- “The string will exist somewhere”
- “The execution path is obvious”
- “Dead code looks dead”

Gamaredon did not add complexity for its own sake.

They added just enough friction to slow analysis and then stopped.

And that restraint is, in itself, the most telling signal.