

JitterDropper | OALABS Research

research.openanalysis.net/jitterdropper/dropper/rust/srdi/donut/pixelrain/2026/04/13/jitterdropper.html

OALABS Research

April 13, 2026

Overview

We have observed a new Rust/MSVC Windows dropper under active development since at least 2026-03-18 with nine builds observed across two variant lines. Currently the name is unknown so we will dubbing it **JitterDropper**.

Variant I embeds the payload in **.rdata** and runs a multi-pass decryption algorithm producing a Donut shellcode loader with an embedded PE. **Variant II** ships a smaller stager that downloads a 122 byte encrypted shellcode blob from **pixelrain[.]com** and decrypts it with a single SSE-32 repeating XOR key. Every build is compiled against the same Rust 1.92.0 MSVC toolchain (**rustc/ded5c06cf21d2b93bffd5d884aa6e96934ee4234**).

Though there are multiple variants the use of a specific sleep-jitter to obfuscate API calls ties all samples to the same developer. The following specific jitter bounds are used for each API across all samples: 60 ms before InternetOpenA, 150 ms before StringFromGUID2, 100 ms before CoGetObject, 120 ms before CoUninitialize. All nine samples contain the same 32 bytes that rustc emits for **Duration::from_millis(variable)** a fixed Lemire mod 1000 reduction followed by **imul edx, eax, 0xF4240** to convert milliseconds to nanoseconds before the sleep call. An example of UnpacMe IDA plugin [UnpacMe-IDA-Byte-Search](#) identifying related samples based on this jitter implementation is shown below.

The screenshot displays the IDA View-A interface. On the left, the assembly code for `sub_140001260` is visible, showing a loop that calculates a value based on `TickCount` and `FindFileData`. On the right, the UnpacMe Search results are shown, listing 15 samples with their SHA256 hashes and classifications.

Classification	Malware Family	Labels	Threat Type	SHA256
UNKNOWN				2ef28834f30d3fd881f8c3004ab67c01e2ec283c03d57e173e43a7b30d5297b
UNKNOWN				9e714ba3e2f8fe053550bb0234fc7e3fa64ce41e601bfff8f0f4782245312fc
UNKNOWN				5605f711e9c256b09dc0f6c4f39851545dbfc05c1c5b3e5f3129b16506e7df19
UNKNOWN				4e57a67bf5a1dfa198ba9c192ed47016f1a043c0c8ae0349f32c02ed17d296d4
UNKNOWN				eede257690bdc8c0668c3bef1e394a6c1e50980ba237647fb36c26f46557a450
UNKNOWN				9957bf9bc95be77c84f83546d37ec3bd81877872f2e18e54adc014c541da3c6a
UNKNOWN				e8082e3cd0d63f83ad95af88f50e9ae26512e8911339b66b24068648363d676e
UNKNOWN				cb7264735e38f2575a354762c122a90d8994ea406f4911a42d8fa45d397fc4ff
UNKNOWN				812d2d8436985fc8531315970b2af6bcd697f9033bc12841ce467e6fda94408
UNKNOWN				75d6ab3ed1120392200c3e711e4a2464c6801e5d38db52e6efcc34fabb13baa
UNKNOWN				0776b8f1eb8251cabf811a35d83311a9a1e5013f84e9237bda5d156933a06c4b
UNKNOWN				7ed456bb2e58dbf4d2e35494b745f8ba824aefb7ed4babcc95f5484417668f
UNKNOWN				ad930a4d0bcb040fcb3f44a9b6eeac8a416fc51a7d07571ca7c514e8c7351fc
UNKNOWN				05ec21e510e17718f12f1d10de8206705ca99c6b10a2a8cccd55b2f012def3d
UNKNOWN				09e79332034fa0dcecc7d3e1668daa740a8f03bb06899b247f154815a1fd0

Samples

Variant I — embedded multi-pass decrypt, sRDI/Donut second stage

- 812d2d8436985fc8531315970b2af6bcd697f9033bc12841ce467e6fda94408 — I-8iter+sbox

- `cb7264735e38f2575a354762c122a90d8994ea406f4911a42d8fa45d397fc4ff` — I-10iter+sbox, 2026-03-23
- `e8082e3c0d63f83ad95af88f50e9ae26512e89113339b66b24068648363d676e` — I-9iter+mod24, 2026-03-27

Variant II — SSE-32 XOR, pixeldrain-fetched 122-byte hop-loader

- `9957bf9bc95be77c84f83546d37ec3bd81877872f2e18e54adc014c541da3c6a` — II baseline, 2026-04-06
- `eede257690bdc8c0668c3bef1e394a6c1e50980ba237647fb36c26f46557a450` — II + AES-256-GCM wrap, 2026-04-09
- `4e57a67bf5a1dfa198bafc192ed47016f1a043c0c8ae0349f32c02ed17d296d4` — II + `CreateRemoteThread` into `explorer.exe` + `%APPDATA%` persistence, 2026-04-09
- `9e714ba3e2f8fe053550bb0234fc7e3fa64ce41e601bff83f0f47822245312fc` — II + RWX regression, 2026-04-11
- `2ef28834f30d3fd881f8c3004ab67c01e2ec283c03d57e173e43a7b30d52f97b`
- `5605f711e9c256b09dcf06c4f39851545dbfc05c1c5b3e5f3129b16506e7df19`

Analysis

JitterDropper performs anti-analysis checks, decrypts or downloads a second-stage shellcode, and transfers control via `VirtualAlloc` -> `VirtualProtect` -> `call rbx`. All builds except 2026-04-11 use `W^X`; the regressed build allocates `RWX` directly. The stager imports Rust's standard library and `VCRUNTIME140.dll` and is built as a GUI executable whose cover window never pumps messages.

Anti-Analysis

Every build carries the same authored anti-analysis features:

- **Inline anti-debug pair** — `CheckRemoteDebuggerPresent` immediately followed by `IsDebuggerPresent`. A detection causes a silent exit.
- **GUI cover window** — `RegisterClassExA` + `CreateWindowExA` with randomised class/title strings (observed: `ZiEXzryWwge` / `QLUTPX`, `TYiYiIqIRv`). No message pump.
- **EnumWindows stall loop** — 21–27 iterations of `EnumWindows` whose callback only calls `GetClassNameA` into a stack buffer and discards the result. A wall-clock padder with no semantic payload.
- **GetTickCount -> randomised Sleep -> GetTickCount -> elapsed-check gates** at each stager checkpoint. When the elapsed delta exceeds the author's threshold (e.g. 1153 ms in `e8082e3c`), the stager exits. This defeats sandboxes that time-compress or skip `Sleep` calls. On Windows the Rust `thread::sleep` call lowers to `CreateWaitableTimerExW` + `SetWaitableTimer` rather than `Sleep`.

The jitter-budget-per-API fingerprint

The 32-byte Lemire-x1 000 000 byte window that scales milliseconds to nanoseconds for `Duration::from_millis(var)` is rustc-emitted and appears in unrelated Rust programs that call `thread::sleep` on a variable millisecond argument. It is not on its own a developer fingerprint. What is unique to JitterDropper is the author's per-API choice of **Lemire divisor** (the upper bound of each randomised sleep in milliseconds) which is unchanged across every build where that API appears.

Gated API	Divisor (ms)	Coverage
<code>InternetOpenA</code>	60	6 of 6 network-bearing builds
<code>StringFromGUID2</code>	150	4 of 4 COM-bearing builds
<code>CoGetObject</code>	100	4 of 4 COM-bearing builds
<code>CoUninitialize</code>	120	4 of 4 COM-bearing builds
<code>memcpy</code> / <code>memset</code> (pre-payload copy)	1000	6 of 6 sites
<code>GetCurrentProcess</code> (pre anti-debug)	1000	5 of 5 sites
<code>EnumWindows</code> (stall-loop pad)	1000	5 of 5 sites
<code>VirtualProtect</code> (RW -> RX transition)	49–99	7 of 7 sites (narrow cleanup band)
<code>QueryPerformanceCounter</code> (elapsed-check gate)	521–770	9 of 9 samples

The `GetTickCount` before jitter pairing rate is 80 to 100% per sample across the corpus. Unrelated Rust samples that happen to share the rustc-emitted x1 000 000 byte window show 0% `GetTickCount` pairing and a single uniform `/1000` budget. The per-API divisor table is what distinguishes JitterDropper from compiler-output coincidences.

Decryption

Variant I decrypts an embedded `.rdata` ciphertext in three passes: a byte-XOR chain, 8–10 iterations of an SSE permutation driven by an XMM constant table (with buffer rotation between iterations — in-place would alias and corrupt), and a final repeating-key XOR using either a 16-byte table (`cb7264`), a 24-byte key with `i mod 24` indexing (`e8082e`), or a similar variant. The output is a ~675 KB raw shellcode.

Variant II decrypts a 122-byte blob downloaded from `pixeldrain[.]com` with a single SSE 32-byte repeating XOR key (`xorps xmm2, xmm0 / xorps xmm3, xmm1 / movups [r13+rcx], xmm2 / movups [r13+rcx+0x10], xmm3 / add rcx, 0x20 / cmp rax, rcx / jne`).

The AES-GCM fork (`eede2576`) wraps this primitive with a `BCryptDecrypt` call using the literal UTF-16 string `ChainingModeGCM` and a key supplied in a separate 122-byte `pixeldrain[.]com` download.

Second stage

All three retrievable Variant I shellcodes share a identical **222 byte sRDI/Donut reflective loader** at the call-target of the `call rel32` at offset 0, followed by an encrypted inner PE. Two distinct inner PE families have been observed in the decrypted shellcodes, an unknown .NET stealer, and Vidar.

Variant II second stages are 122-byte shellcodes hosted on `pixeldrain[.]com/api/file/<id>`; these links were dead at the time of sample collection so no shellcode was retrieved for analysis.

Evolution

Timestamps and feature deltas are consistent with a single developer or small team iterating over five weeks. The Variant I decrypt iteration count drops 10 to 9 to 8 across builds and the s-box substitution pass is added and removed; Variant II is introduced on 2026-04-06 as a simpler network-fetching fork; on 2026-04-09 two parallel forks appear. One adds AES-256-GCM wrapping with dynamically-resolved `BCrypt`, the other adds `CreateRemoteThread` injection into `explorer.exe` and `%APPDATA%` self-copy persistence. The 2026-04-11 build regresses to `RWX`, suggesting a branch off the tree before `W^X` was merged in.