RESEARCH · MALWARE ANALYSIS AND MEMORY FORENSICS

Four Files, Two Families: PureRAT and Remus on a Customer Host

One suspicious executable from a customer incident led us to four files on the same machine: two builds of the PureRAT remote-access trojan, a Remus infostealer and one we could not identify. This is how we got from the first file to the fourth, and what the evidence does and does not prove.

Or Malka, ICTBIT 25 November 2026 Customer details redacted

4
executables found in C:\ProgramData

2
malware families identified

16+ min
PureRAT active before Defender reacted

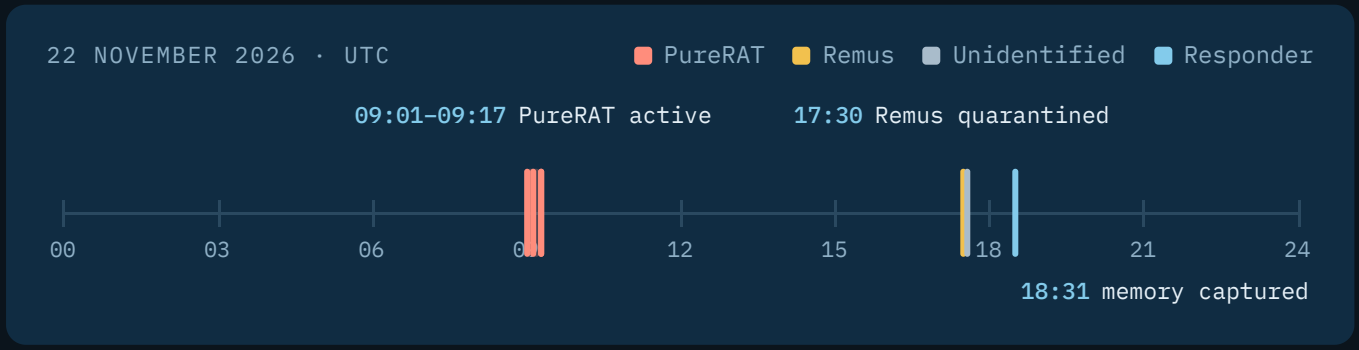
2
files with no Defender detection

> TAKEAWAY
>
> A detection is not a cleanup. Defender quarantined one copy of the RAT and left the original, a second build and an unidentified file in place, with an infostealer sitting next to them. When one commodity loader turns up, look for its neighbours.

All analysis was done in an isolated lab. We made no contact with attacker infrastructure.

The day in evidence

Every mark below is a record recovered from the host's memory image. Times are UTC, on a true 24-hour scale: PureRAT's whole visible life is the thin red band in the morning.



09:01:16

PureRAT is already installed

PureRAT

```
2026-11-22T09:01:16.287 [RTP] [MpRtp] Engine VFZ lofi/sample/expensive:
  …\Users\<user>\AppData\Roaming\pyton.exe. status=0x40070000
```

Defender's real-time engine handles the installed copy for the first time. No detection follows. Source: Defender log in memory.

09:07:36

Scheduled task `\pyton.exe` is written

PureRAT

```
2026-11-22 09:07:36 UTC   Key   …\Schedule\TaskCache\Tree   pyton.exe
```

Last write time of the task's registry key. Source: registry in memory (Volatility).

09:17:03

Defender blocks `5555.exe` writing the copy again

PureRAT

```
2026-11-22T09:17:03.438 [RTP] [Mini-filter] Blocked file(#88):
  …\AppData\Roaming\pyton.exe.
  Process: …\ProgramData\5555.exe
2026-11-22T09:17:03.681 DETECTIONEVENT MPSOURCE_REALTIME
  Trojan:Win32/Wacatac.B!ml
```

This record names the original file and where it ran from. Source: Defender log in memory.

09:17:20

The copy and its task are quarantined

PureRAT

```
2026-11-22T09:17:20.385 DETECTION_CLEANEVENT MPSOURCE_REALTIME
  MP_THREAT_ACTION_QUARANTINE 0 Trojan:Win32/Wacatac.B!ml
  file:C:\Users\<user>\AppData\Roaming\pyton.exe
```

The original `C:\ProgramData\5555.exe`, a byte-identical file, was not removed.

17:30:35

`987654321.exe` is detected as Remus and quarantined

Remus

```
2026-11-22T17:30:35.227 DETECTIONEVENT MPSOURCE_REALTIME
  Trojan:Win64/Remus.JS!MTB
  file:\\localhost\C$\ProgramData\987654321.exe
2026-11-22T17:30:48.034 DETECTION_CLEANEVENT …
  MP_THREAT_ACTION_QUARANTINE
```

Detected when the file was opened over the local admin share, most likely during evidence collection. When it arrived is unknown.

17:35:33

`gfdbfs54gs5gsrfg5g45g.exe` is handled, nothing is flagged

Unidentified

```
2026-11-22T17:35:33.096 [RTP] [MpRtp] Engine VFZ lofi/sample/expensive:
  …\localhost\C$\ProgramData\gfdbfs54gs5gsrfg5g45g.exe.
  status=0x40070000 … threatid=0x80000000
```

Same access path as the Remus file, five minutes later. Source: Defender log in memory.

---

time not recorded

The responder archives four files from one folder

Responder

```
WinRAR.exe a … C:\Users\<user>\Desktop\pyton.rar
  C:\ProgramData\gfdbfs54gs5gsrfg5g45g.exe
  C:\ProgramData\987654321.exe
  C:\ProgramData\5555.exe
  C:\ProgramData\Okjlodsfe.exe
```

This command line is how we learned there were three more files. Source: process command line in memory.

18:31:24

Memory is captured

Responder

```
winpmem.exe acquire C:\IR\mem.raw
SystemTime  2026-11-22 18:31:24+00:00
```

Nine hours after the PureRAT detection: no malware process, connection or configuration was left in memory.

The four files

All four sat in `C:\ProgramData` under random names. We know when `5555.exe` ran. For the other three we know only that they were there when the responder collected them.

`5555.exe`

PureRAT, with persistence

Copy detected 09:17, original left on disk

Ran, installed itself as `pyton.exe` and created a task that relaunches it every 5 minutes.

Unknown: whether it reached its server, and when the process stopped.

`Okjlodsfe.exe`

PureRAT, second build, no persistence

No detection found

Same server, certificate and mutex as `5555.exe`. Installs nothing.

Unknown: whether it ran on the host.

`987654321.exe`

Remus infostealer

Detected and quarantined 17:30

Named by Defender. Our copy is 0 KB because it was collected after quarantine, so we did not analyse it ourselves.

Unknown: whether it ran, and for how long it was on the host.

`gfdbfs54gs5gsrfg5g45g.exe`

Unidentified

No detection found

Handled by Defender at 17:35 without a verdict. Not known to VirusTotal.

Unknown: what it is. Not analysed yet.

How we got there

1 Unpack the first file

> **WHY** The file we were given, `5555.exe`, showed nothing readable. To know what it does we had to reach the code it hides.

`5555.exe` is a .NET loader protected with .NET Reactor. The malware itself sits two layers further in and is never written to disk.

5555.exe Layer 1 · protected loader · the only file on disk

container DLL Layer 2 · no code, one encrypted resource · memory only

Tokawww.dll Layer 3 · the RAT · TripleDES, then GZip · memory only

We asked dnSpyEx which methods read the embedded resources. One of them takes a number and returns text: the string decryptor.

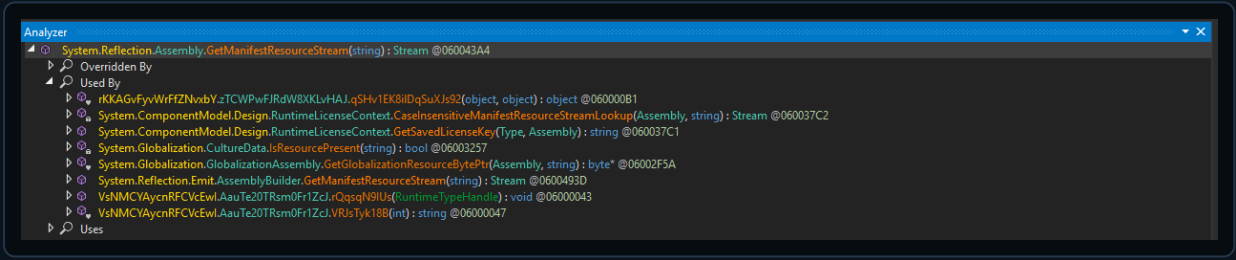


Figure 1. Who reads the resources. The method ending in `(int) : string` is the decryptor.

A breakpoint inside it gave us its whole string table, which held the keys to the next layer:

```
c0mMTS6EXn6j0DWrTPvcrQ==              16-byte TripleDES key
5JDs/siCuZE=                          8-byte IV
b4ADXrgwpFU3vUe1Yd.Apxkba4OfPB8sNyuS3  name of the encrypted payload
Load                                  Assembly.Load, called by name
```

A breakpoint on `Assembly.Load` in the .NET runtime then caught layer 2 as it was loaded. We decrypted layer 3 offline with the recovered key, without running it.

---

2 Run it in an isolated lab

> **WHY** The RAT decrypts its settings only while running, so the server address cannot be read from the file.

- **Server:** connection attempts to `101.99.76[.]85`, rotating through TCP ports 56001, 56002 and 56003.

- **Persistence:** a copy at `%AppData%\Roaming\pyton.exe` and a task that restarts it every 5 minutes.
- **Mutex:** `8bd8d55a0deb`, so only one copy runs at a time.

A dump of the running process held the decrypted settings side by side:

```
101.99.76[.]85                          server
MIIE4DCCAsigAwIBAgIQANEpScf6…F0uFg==    pinned server certificate, CN=Zxqnne
pyton.exe                               install name
APPDATA                                 install folder
8bd8d55a0deb                            mutex
```

3 Read the victim's memory

> **WHY** The lab shows what the malware can do. Only the victim's memory shows what happened there.

The process list and network scan came back empty: by capture time the malware had been gone for hours. So we searched the raw memory for the indicators from step 2.

The file name `pyton.exe` hit Microsoft Defender's own logs, still cached in memory. Those records are the timeline at the top of this page. A second search, for the responder's collection command, exposed three more files in the same folder, and one of them carried a new verdict:

```
2026-11-22T17:30:35.227 DETECTIONEVENT MPSOURCE_REALTIME
  Trojan:Win64/Remus.JS!MTB
  file:\\localhost\C$\ProgramData\987654321.exe
```

4 Unpack the second build

> **WHY** `Okjlodsfe.exe` had no detection and looked like `5555.exe` in dnSpyEx. If it is the same malware, it may point to another server.

The breakpoints from step 1 unpacked it in a single run.

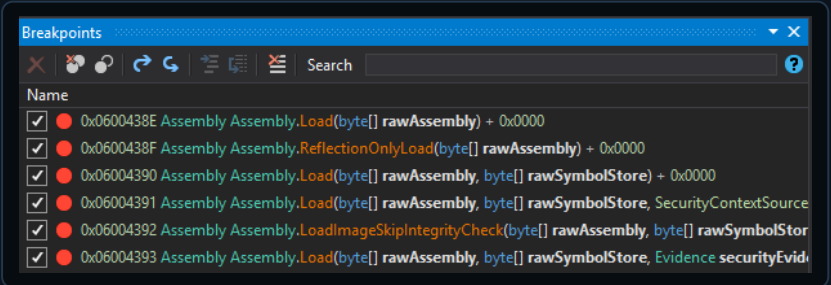


Figure 2. One breakpoint on every `Assembly.Load(byte[])` overload catches each layer as it loads.

Its payload, `Cdjouk.dll`, has the same structure and references as `Tokawww.dll`. When run, it takes the same mutex.

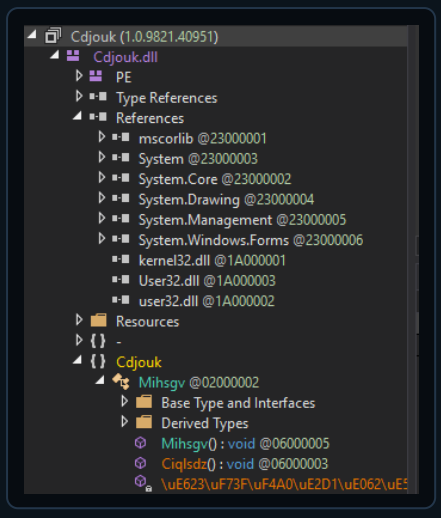


Figure 3. `Cdjouk.dll` : the same nine references as the first build.

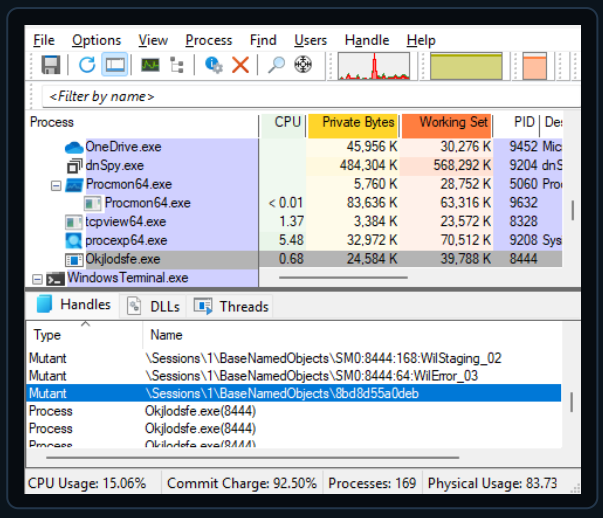


Figure 4. `Okjlodsfe.exe` holding mutex `8bd8d55a0deb` .

	5555.EXE	OKJLODSFE.EXE
Server	101.99.76[.]85	Identical
Pinned certificate	CN=Zxqnne	Identical
Mutex	8bd8d55a0deb	Identical
Payload	Tokawww.dll	Cdjouk.dll
Persistence	pyton.exe and a task	None
In the lab	Connects within seconds	No connection attempt in several minutes
Defender on the host	Copy detected	No detection found

Same operator, same server. The second build adds no new network indicator, only a second file to hunt for.

The server, three days later

> **WHY** An address seen once in a lab could already be dead. A passive lookup shows whether it is still serving, without touching it.

On 25 November, Censys still listed the address with the PureRAT port open.

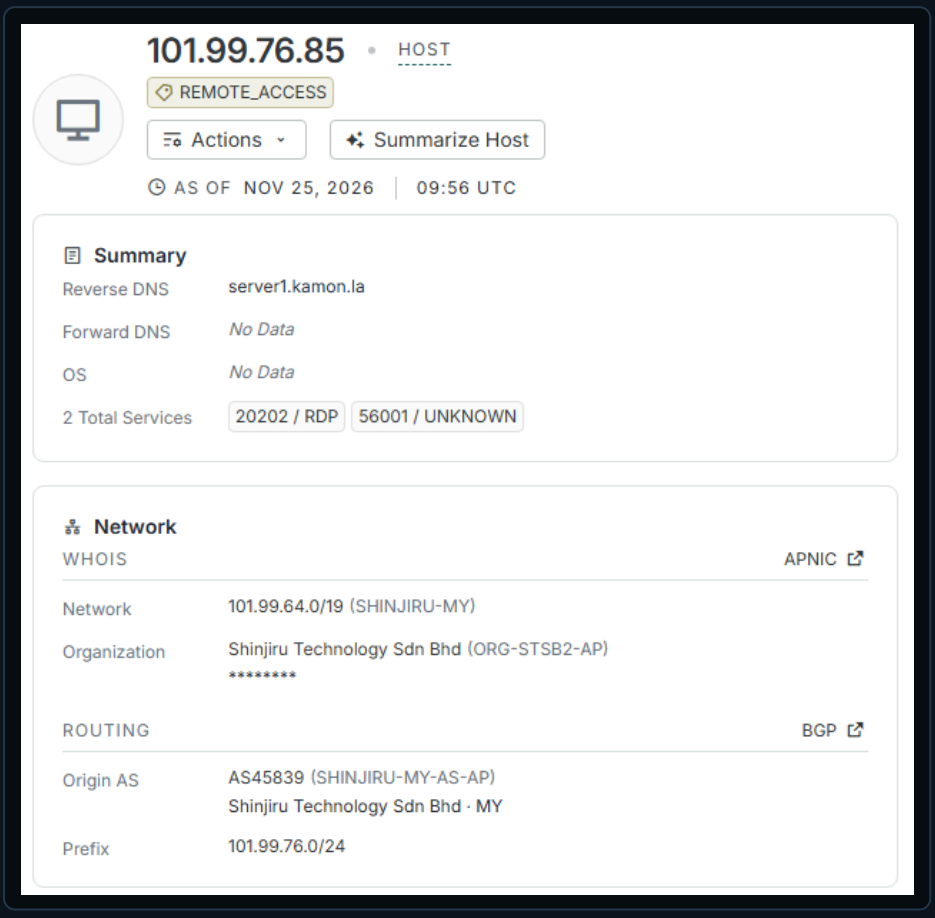


Figure 5. The server as Censys saw it on 25 November 2026, 09:56 UTC.

OBSERVED	VALUE	MEANING
Open services	`56001` unidentified, `20202` RDP	56001 is the first port the malware dials. RDP on an unusual port is a common way to administer such a server.
Reverse DNS	`server1.kamon[.]la`	A name set on the hosting side. Context, not proof of ownership.
Network	`101.99.76.0/24`, AS45839	Shinjiru Technology Sdn Bhd, a Malaysian hosting provider
Location	Zwolle, Netherlands	Geolocation of the address, per Censys

We did not connect to the server. Everything here comes from a third-party scan.

What PureRAT can do

The payload's code is virtualized, so we could not read it. Windows function names cannot be hidden, though, and the ones it imports outline the basics. These are inferred, not observed.

1C ImplMap (8)

RID	Token	Offset	MappingFlags	MemberForward	ImportName	ImportScope	Info
1	0x1C000001	0x00025F22	0x140	0xD	0x8C1	1	SetThreadExecutionState
2	0x1C000002	0x00025F2A	0x146	0xF	0x1CD7	2	GetForegroundWindow
3	0x1C000003	0x00025F32	0x146	0x11	0x1CC1	2	GetWindowText
4	0x1C000004	0x00025F3A	0x146	0x13	0xD14	2	GetWindowTextLength
5	0x1C000005	0x00025F42	0x140	0x15	0x832	2	SetProcessDPIAware
6	0x1C000006	0x00025F4A	0x100	0x17	0x135A	3	GetLastInputInfo
7	0x1C000007	0x00025F52	0x100	0x3CB	0x1DFE	1	LoadLibrary
8	0x1C000008	0x00025F5A	0x100	0x3CD	0x1A57	1	GetProcAddress

Figure 6. The eight Windows functions the payload imports.

IMPORT	LIKELY USE
`GetForegroundWindow`, `GetWindowText`	Report which window the user is looking at
`GetLastInputInfo`	Report whether the user is at the keyboard
`SetProcessDPIAware`, `System.Drawing`	Screen capture
`SetThreadExecutionState`	Keep the machine awake

Public reporting describes PureRAT as a tool for live control of a victim, including hidden remote desktop, with most features sent from the server on demand [1].

Attribution

- **PureRAT, high confidence.** TCP 56001 to 56003 are the family's default ports [1]. A published PureRAT configuration holds the same three items we recovered: a server IP, that port list and a pinned certificate [3]. The loader matches the documented PureCrypter pattern of a code-less container decrypted with 3DES [4].
- **Remus, by label only.** The name is Defender's. Public reporting describes Remus as a successor of the Lumma stealer that takes saved passwords, browser data and crypto wallets [5].
- **One operator for both PureRAT builds.** They share a server, a certificate created on 15 November 2026 and a mutex.

Limits of this analysis

Did Remus run?

Unknown. It was caught only when someone opened the file. Disk artefacts such as Prefetch would tell; memory does not.

When did PureRAT stop?

Unknown. Defender removed the copy, not the original, and the task's registry key was still present at collection.

Did PureRAT reach its server?

Unknown. Nine hours later, memory held no trace of a connection.

What is the fourth file?

Not analysed. No detection, and not on VirusTotal.

How did the files arrive?

Unknown. Memory held one weak lead, a script fragment pointing at a subdomain of an unrelated legitimate website. It is unverified, so we do not name the site.

When were the payloads built?

Their embedded dates point to 21 and 22 November, but one is later than the infection itself, so we do not rely on them.

Indicators

KIND	INDICATOR	VALUE
NETWORK	PureRAT server	`101.99.76[.]85`
NETWORK	Server ports	`TCP 56001, 56002, 56003`
NETWORK	Server certificate SHA1	`3b43f69c339610ad4e97f927066b2c47d71704b9`
NETWORK	Certificate subject and serial	`CN=Zxqnne, D12949C7FA0D76856D8F93E5A80371`
NETWORK	Server reverse DNS, per Censys	`server1.kamon[.]la`
NETWORK	Other open service on the server	`TCP 20202 (RDP)`
HOST	Mutex	`8bd8d55a0deb`
HOST	Persistence file	`%AppData%\Roaming\pyton.exe`
HOST	Scheduled task	`\pyton.exe`
HOST	PureRAT loader	`C:\ProgramData\5555.exe`
HOST	PureRAT loader, second build	`C:\ProgramData\Okjlodsfe.exe`
HOST	Remus infostealer	`C:\ProgramData\987654321.exe`
HOST	Unidentified file	`C:\ProgramData\gfdbfs54gs5gsrfg5g45g.exe`
HOST	Defender verdicts	`Trojan:Win32/Wacatac.B!ml, Trojan:Win64/Remus.JS!MTB`
HASH	`5555.exe` and `pyton.exe`, SHA256	`7085BF9C71258AFB969D4A8448BE0DDA1B916F03C2ADC6969A18CEC9E1F93F74`
HASH	`5555.exe`, MD5	`65AE0ABC18E5399DBE5AB90E1188B81A`
HASH	Layer 2 container, SHA256	`DD1CF0EC2254D157F43A0F769A737FD33EC49651437D286C23F69CD7A46C5468`
HASH	`Tokawww.dll`, SHA256	`341614ED3983413B7A4088F76539107B3AB10EF71E4262E7CEE522C48E0655F0`

Hashes of `Okjlodsfe.exe`, `Cdjouk.dll` and the unidentified file were not recorded.

ATT&CK TECHNIQUE	ID	EVIDENCE
Scheduled Task	T1053.005	Task `\pyton.exe`, every 5 minutes
Masquerading	T1036.005	`pyton.exe`, a misspelling of `python.exe`
Obfuscated Files or Information	T1027	.NET Reactor, TripleDES and GZip layers
Reflective Code Loading	T1620	Payloads loaded with `Assembly.Load(byte[])`
Non-Standard Port	T1571	TCP 56001 to 56003

Detection and hunting

1 **Look for the neighbours**

A randomly named executable directly in `C:\ProgramData` rarely comes alone. List the folder before closing the alert.

2 **Block and alert on the network side**

Block `101.99.76[.]85`. Outbound TCP 56001 to 56003 is worth an alert on its own, since these are the family's default ports.

3 **Hunt the host indicators**

The mutex, the task and the dropped files are stable for this build.

```
# tasks that run something from the user's AppData
Get-ScheduledTask | ? { $_.Actions.Execute -like "*\AppData\Roaming\*" } |
  Select TaskName, @{n='Run';e={$_.Actions.Execute}}

# executables sitting directly in ProgramData
Get-ChildItem C:\ProgramData -File -Filter *.exe | Select Name, Length, CreationTime

# the mutex (Sysinternals)
handle.exe -a 8bd8d55a0deb
```

4 **Keep one behavioural rule**

Alert on user-level scheduled tasks that relaunch an executable from `%AppData%` every few minutes. Hashes and addresses change with the next build; this behaviour does not.

5 **Treat a stealer next to a RAT as credential theft**

If an infostealer was on the host, assume browser-saved passwords and sessions are gone, whether or not it was quarantined.

Key terms

TERM	MEANING
Loader	A wrapper that hides the real malware and runs it from memory
RAT	Remote-access trojan: gives an operator live control of the machine
Infostealer	Malware that collects saved passwords, cookies and wallets, then leaves
Mutex	A named system object, used here to keep a single running copy
Certificate pinning	The malware accepts only one specific server certificate

References

1. Netresec, PureLogs, PureRAT and misleading zgRAT
2. Netresec, PureRAT = ResolverRAT = PureHVNC
3. Simply Secure Group, From infostealer to full RAT: dissecting the PureRAT attack chain
4. ANY.RUN, Pure malware family analysis
5. PCrisk, Remus information stealer