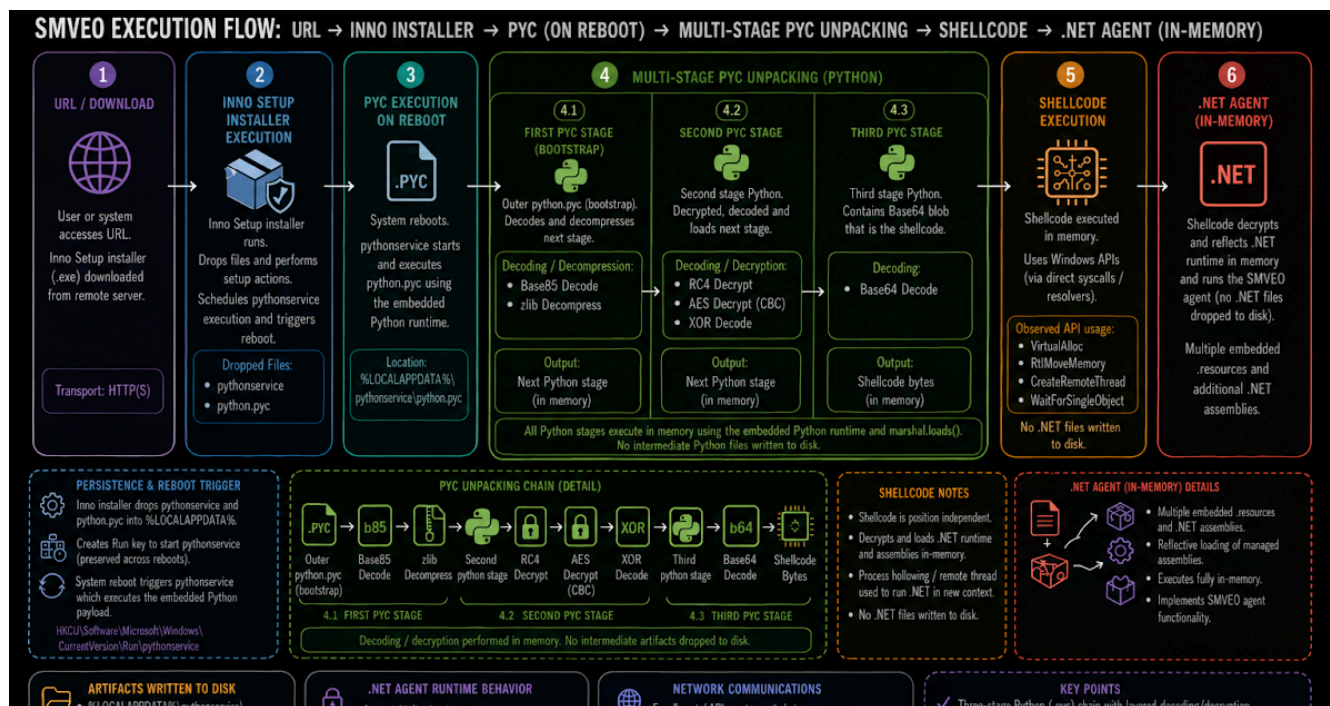


PureRAT variant observed in AI Video Player

 blog.lukeacha.com/2026/06/purerat-variant-observed-in-ai-video.html

Luke Acha

June 18, 2026



Executive Summary

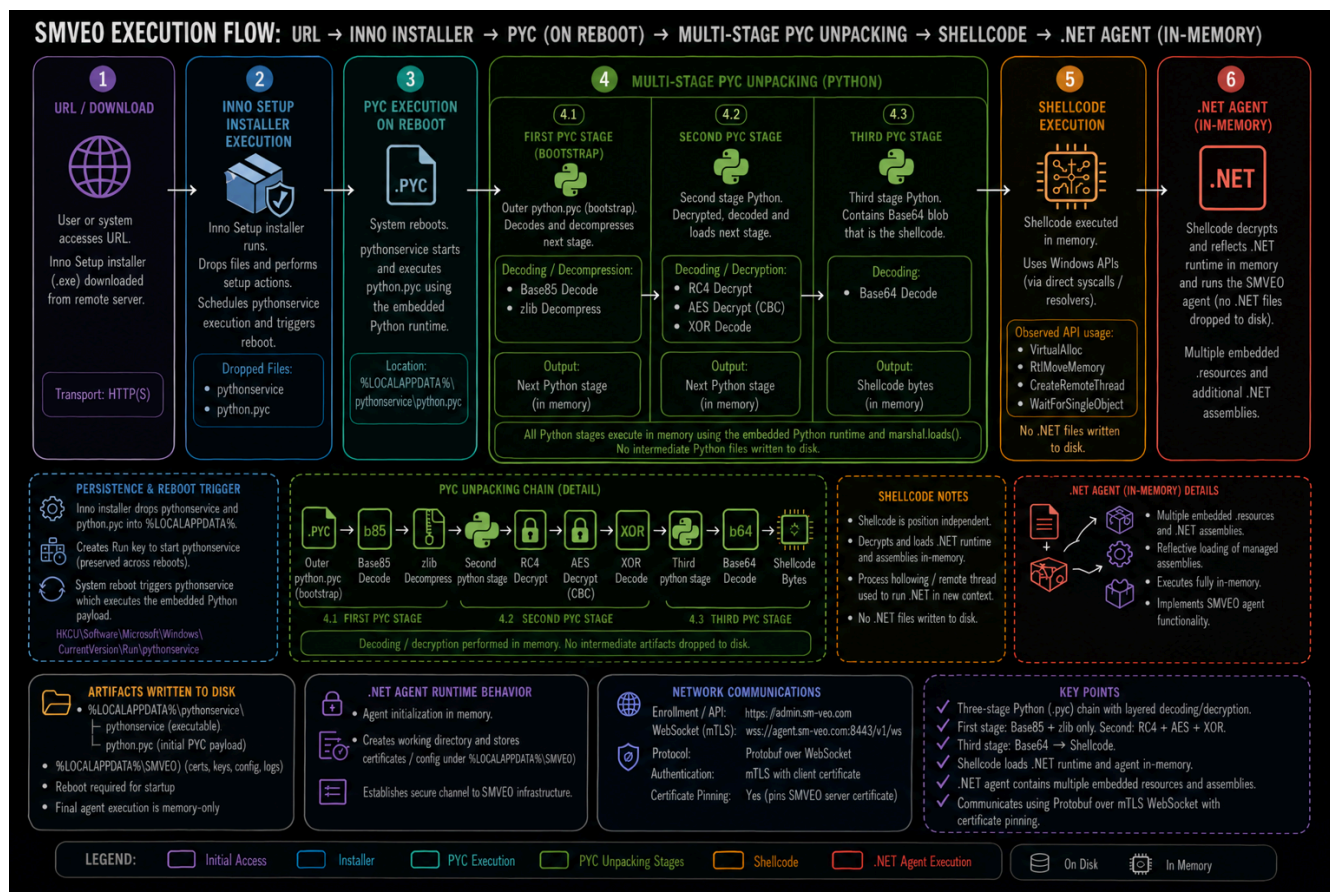
I analyzed a malware chain beginning with DriveVideoSetup-x64-0.1.0.exe, distributed through a fake Drive Video /SMVEO-themed lure. The malware uses a staged execution chain that moves from Python bytecode into shellcode, then into multiple protected .NET assemblies.

The chain includes heavy obfuscation and encryption, runtime patching, certificate generation, protobuf references, and authenticated WebSocket communications.

Dynamic analysis observed the malware enrolling with infrastructure, creating certificate material under %LOCALAPPDATA%\SMVEO\, connecting to agent.sm-veo.com, and logging agent starting 2.3.0, auto-enrolling, and ws connection.

Multiple indicators suggest this activity is [DonutLoader](#) leading to [PureRAT](#).

Execution Workflow



DriveVideoSetup-x64-0.1.0.exe ↓ Persistence via CurrentVersion\Run ↓ python.pyc ↓ Base85

Delivery and Sandbox Caveat

The lure uses a fake Google Drive-style video download flow and delivers: [DriveVideoSetup-x64-0.1.0.exe](#)

The certificate signer / publisher artifact observed across related samples is: **Smartcore LLC**

Persistence is established through: **HKCU\Software\Microsoft\Windows\CurrentVersion\Run**

The Python stage may not execute immediately. In several sandbox-style environments, the installer may only stage persistence. The later Python → shellcode → .NET chain may require reboot, logon, or manual triggering of the persisted Run key.

[App Any Run Snadbox Run](#)

Note: Non-Public sandbox when run under Windows 11 has full execution chain

Python Stage

We view python disassembly using pycdas to get an understanding of the structure:

```
python.pyc (Python 3.10)
[Code]
  File Name: final_obf.py
  Object Name: <module>
  Arg Count: 0
  Pos Only Arg Count: 0
  KW Only Arg Count: 0
  Locals: 0
  Stack Size: 8
  Flags: 0x00000040 (CO_NOFREE)
  [Names]
    'exec'
    '__import__'
    'loads'
    'decompress'
    'b85decode'
  [Var Names]
  [Free Vars]
  [Cell Vars]
  [Constants]
    'marshal'
    'zlib'
    'base64'
    'c$|c}*Ru0Ub|(1m-yBz`j7p|dC8;HfMl{VD0Q5l*5+DJB003hQH%T}W9v}#M6BALIn7)~dd4tlW-e8_#F8TrX8|W1}`
    None
  [Disassembly]
    0      LOAD_NAME          0: exec
    2      LOAD_NAME          1: __import__
    4      LOAD_CONST         0: 'marshal'
    6      CALL_FUNCTION      1
    8      LOAD_METHOD        2: loads
    10     LOAD_NAME          1: __import__
    12     LOAD_CONST         1: 'zlib'
    14     CALL_FUNCTION      1
    16     LOAD_METHOD        3: decompress
    18     LOAD_NAME          1: __import__
    20     LOAD_CONST         2: 'base64'
    22     CALL_FUNCTION      1
    24     LOAD_METHOD        4: b85decode
    26     LOAD_CONST         3: 'c$|c}*Ru0Ub|(1m-yBz`j7p|dC8;HfMl{VD0Q5l*5+DJB003hQH%T}W9v}#M6BALIn7)~dd4tlW-e8_#F8TrX8|W1}`
    28     CALL_METHOD        1
```

****Note: When decoded, this loads another python file that is also encoded + encrypted.****

The persisted Python bytecode stage contains loader indicators:

marshal base64 b85decode decompress import

The deobfuscation order observed for this stage is:

base85 decode ↓ zlib decompress ↓ marshal.loads() ↓ Python loader execution

Python Stage 2:

```
base85_encoded_data
rsa_private_keyZ
compressed_data
encrypted_data
    order_len
order
rsa_encrypted_key
combined_key
rc4_key
xor_key
aes_keyZ
algor
hybrid_decrypt'
LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS
rbXYxN0NaRm9qcXR3V3AxRl daamo3VkwKT2VlM
lJZTBY0GI4aGtHZTVVZjk5Ckw0bExuejlsZ2dv
3g50TIxMW1xZwptc2NLMWcxay9SbkY0bTRMR1V
ajMKU2FWT0JJPqjlwQ0xXeTNxWctHVjloalgrVr
rVzk3V1pmRVRXODBuN0xnTmZaNmhiS2hSYlY5e
9IMVhIZG5mRk5xUXlEcnpPeY2MWFs0EZjZy90
TR0bDkxWklpVHhHczQvQUZTRThxTWJJMEZld29
QzN0REFzRU9QVytSY0NnWUFKZ2N0U3lPZ0d4ch
```

This second stage of python has 2 blobs, one is Base64 (RSA key) and the second is another xor + aes + rc4 functions.

Python Stage 3 (Shellcode Loader):

```
Stack Size: 2
Flags: 0x00000040 (CO_NOFREE)
[Names]
'real_entry'
'sub_ACIIIVTS'
'sub_FFVL3LGFJ'
[Var Names]
[Free Vars]
[Cell Vars]
[Constants]
[Code]
File Name: <string>
Object Name: real_entry
Arg Count: 0
Pos Only Arg Count: 0
KW Only Arg Count: 0
Locals: 10
Stack Size: 12
Flags: 0x00000043 (CO_OPTIMIZED | CO_NEWLOCALS | CO_NOFREE)
[Names]
'sys','ctypes','time','sleep','base64','ctypes.wintypes','wintypes','windll','kernel32','LPVOID','c_size_t','DWORD','VirtualAlloc','argtypes','restype','HANDLE','CreateRemoteThread','CreateThread','RtlMoveMemory','WaitForSingleObject','b64decode','len','c_int'
'c_void_p'
'pointer'
'KeyboardInterrupt'
'exit'
'Exception'
[Var Names]
'sys'
'ctypes'
'sleep'
'base64'
'wt'
'kernel32'
'buf'
'memAddr'
'th'
'e'
[Free Vars]
[Cell Vars]
[Constants]
None
0
(
'sleep'
)
'6MDdCgDA3QoAB9eEtzt48h6OENG6VMISQVfo5FWjQMg0ZM3RkgzMAAAAGZp+fn7BWT0IDmCcleqjshbTlxYRzfbZlyde941zfx0px5q1N+DjyrbLaQl+hGDb1DI6vdGDxiRmzMbml1k5Syj1By1n0kZjELCSZU/INFz/9H6y4I3m0UQrGLkQjXP1tAAMX284GMu3l5q2'
```

dst16 >0x09

key16 0x19

encrypted blob @ >0x241

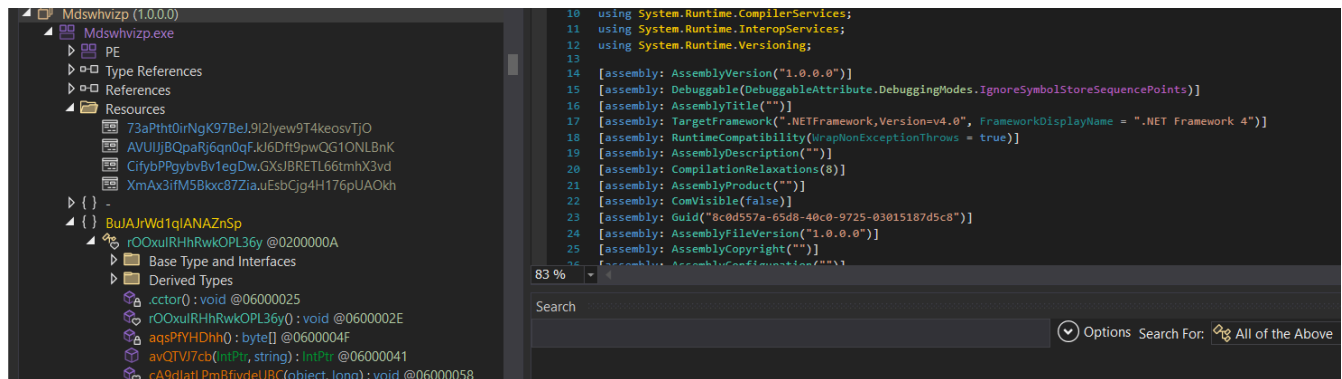
custom 16-round PRNG/stream cipher resulting PE starts at offset 0x104C after decryption

This can be extracted using [Donut-Decrypt](#) by Volexity

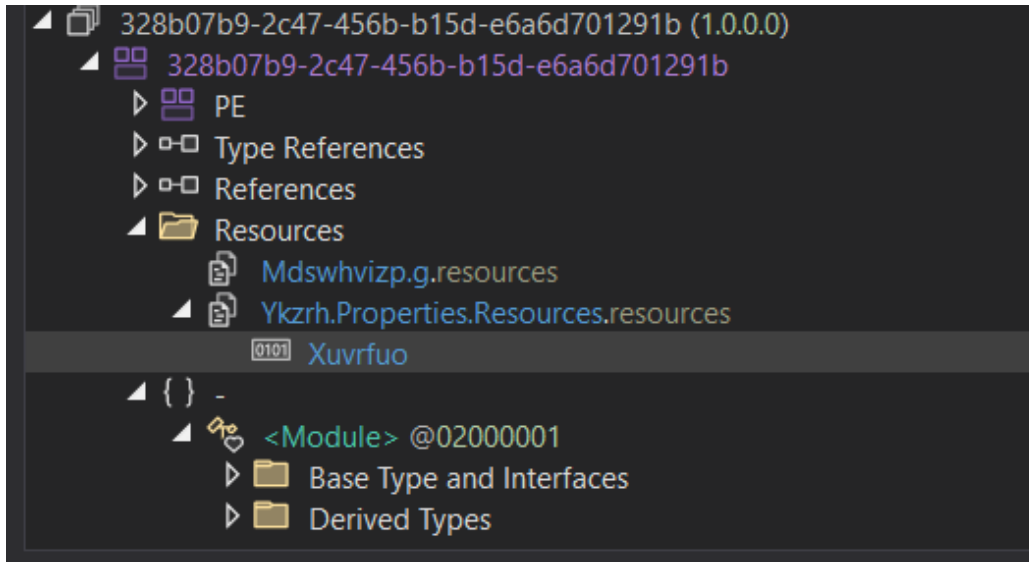
```
remnux@remnux:~/Downloads/1$ donut-decryptor shellcode3.bin
2026-06-20 15:21:04,477 - donut_decryptor.decryptor - INFO - Parsing donut from file: shellcode3.bin
2026-06-20 15:21:04,477 - donut_decryptor.decryptor - INFO - Using 1.0.64, and instance version: 1.0
2026-06-20 15:21:04,477 - donut_decryptor.decryptor - INFO - Locating instance in file: shellcode3.bin
2026-06-20 15:21:04,492 - donut_decryptor.decryptor - INFO - Found instance at: 0x5
2026-06-20 15:21:05,899 - donut_decryptor.decryptor - INFO - Writing module to: /home/remnux/Downloads/1/mod_shellcode3.bin
2026-06-20 15:21:05,900 - donut_decryptor.decryptor - INFO - Writing instance meta data to: /home/remnux/Downloads/1/inst_shellcode3.bin
2026-06-20 15:21:05,900 - donut_decryptor.cli - INFO - Parsed: 1 of 1 attempted files
```

.NET Loader and Virtualization

Follow-up analysis of extracted .NET loader stage, internally named [Ykzrh.exe](#) (smveo-csharp-agent.exe) in one recovered artifact, revealed a substantial virtualization and runtime reconstruction layer. This helps explain why the .NET portions of the chain initially appeared incomplete or nonsensical in decompilers.



The loader reconstructs method references dynamically using encrypted token maps, delegate initialization, reflection, and a custom bytecode interpreter. This appears to be protection infrastructure around the malware rather than the malware's core RAT functionality.



Raw inflate after a small header

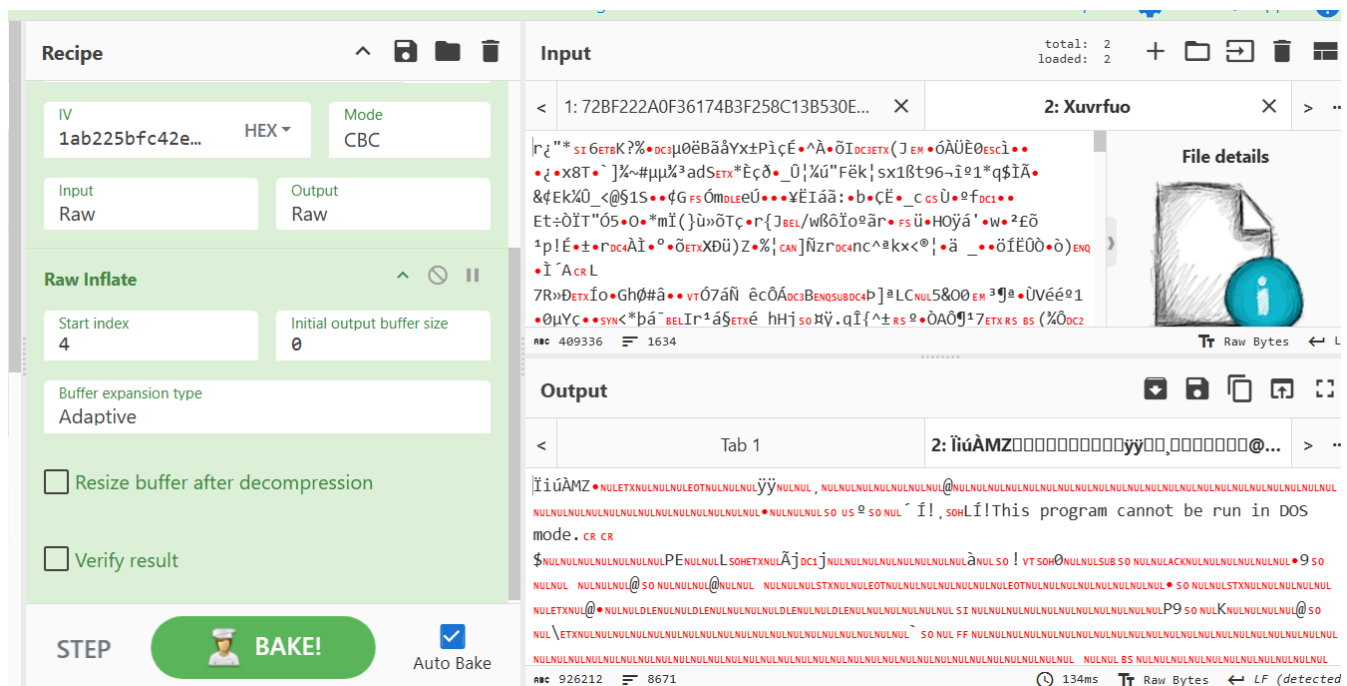
Resource: [Xuvrfuo](#)

Algorithm: TripleDES-CBC

Padding: PKCS7

TripleDES Key: 1E4760505B92C0AA0DE15226D001EE47

TripleDES IV: 1AB225BFC42E570D



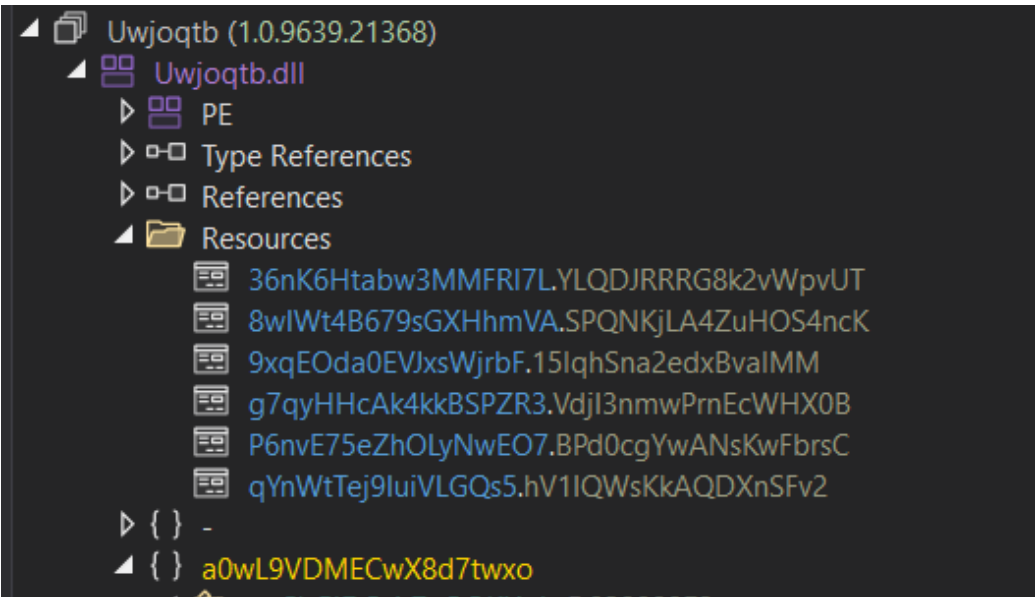
.NET Stage: Uwjoqtb

The decompressed payload from Xuvrfuo produced the protected .NET assembly:

[Uwjoqtb](#) Version Framework.NET Framework 4.0 Timestamp2026-05-23 Microsoft

DetectionTrojan:MSIL/PureRat.ABA!MTB

The assembly is heavily protected and contains multiple encrypted resources. Static decompilation shows many stubs, proxy methods, and incomplete code paths. Runtime debugging and resource decryption show that important functionality is activated dynamically.



Certificates and C2

The malware creates a working directory under:

%LOCALAPPDATA%\SMVEO\

Artifact	Purpose
client.crt	Generated client certificate.
client.key	Generated client private key.
ca.crt	CA certificate material used by the agent.

Infrastructure	Context
smveo.com	Lure / infrastructure domain.
admin.sm-veo.com	Enrollment / administration endpoint.
agent.sm-veo.com	Agent communication endpoint.
wss://agent.sm-veo.com:8443/v1/ws	Observed WebSocket C2 endpoint.

Infrastructure	Context
103.183.115.177:56001	Observed in lab.
103.183.115.177:56002	Observed in lab.
103.183.115.177:56003	Observed in lab.

The certificate workflow suggests mutual TLS-style authentication and custom certificate validation. This would make casual interception with Burp, Fiddler, or mitmproxy more difficult unless the certificate validation path is bypassed.

Detection and Hunting

Certificate signer pivot: Smartcore LLC

Runtime strings: agent starting 2.3.0, auto-enrolling, registered successfully, ws connected, running script

YARA: [GitHub YARA Rules](#)

Samples

Sample	SHA256
DriveVideoSetup-x64-0.1.0.exe	f3284e498890139de2f94be50b260ec7547a91a63d72b926b71ae343106edc02
drivevideo.pyc	e70cc6dddce8717ea5b4d3597d1543d43cfa9a0c2eac6c52f309adfe879b7352
python.pyc	ca36b7438f248e01a20808fa0bbd30938eb3fac792b5ca59bf3ade01afa48992
Ykzrh.exe (PureCrypter)	6ef97cd831f3cd77ee2dcf98b100b31672887ede7c34ed2017039b9ae434ff9d
Xuvrfuo	628adb8b1814491d74b1cbf9d1ef25e36e93f58f8bc2d24b536bcd1627232a34
Uwjoqtb.dll (PureRAT)	8e751724a30386948628c786e6cf4c894aebec8aa36d979b13d6b61a6b5e29c7
Extracted Shellcode	6805394cfec5a680860c5828ca003216fdc626e91ac649a177d2c64ee1ed8127

Key Findings

- Three-stage Python loader chain.
- Stage 3 decodes Base64 shellcode.
- Shellcode uses custom 16-round stream cipher.
- Recovered PE matches operational .NET agent.
- Agent generates certificates and enrolls with SMVEO infrastructure.
- Authenticated WebSocket communications observed.
- Strong overlap with publicly documented PureRAT activity.

[**Beware of Fake 7zip Installer: upStage Proxy**](#)

[**Microsoft Store Apps May Deliver Go Backconnect Proxy Malware**](#)

[**TamperedChef: Suspicious Recipe App is really Malware**](#)
