

# Tracking APT28 PixyNetLoader: Evolutions from 2024 to 2026

Published: 2026-06-01 · Archived: 2026-06-10 02:01:55 UTC

## Key takeaways

- **Analysis of ~90 PixyNetLoader samples and grouping them into 4 sub-families using code similarities sharing**
- **Enabling unified detection through a single YARA rule**
- **Exposing the latest steganography mechanisms used in the 2026 March-April versions**
- **Providing PNG payload extraction script, IOCs, detection guidance and samples list**

In this article, we will examine the evolutions of the APT28 **PixyNetLoader** code family, and how, by analyzing approximately 90 samples and studying the shared code between them, we can identify **4 major different sub-families** that we will briefly detail, and produce a single YARA rule to match all of these codes.

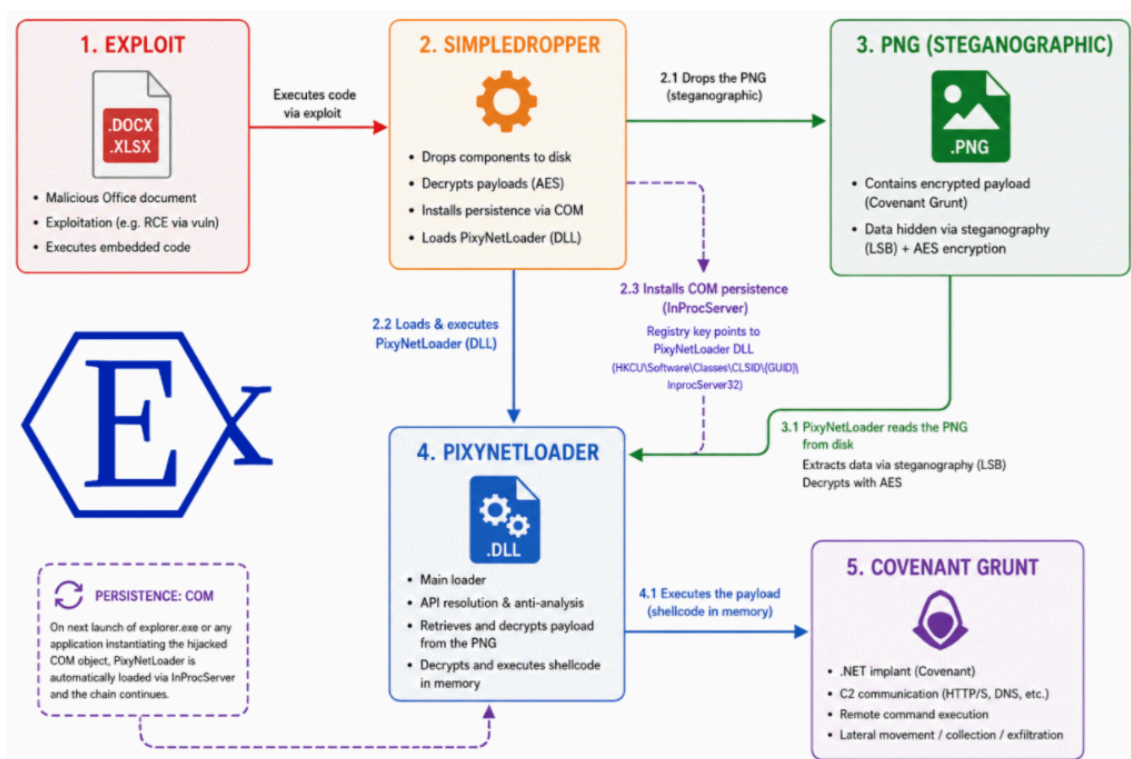
We will then focus on the latest family to date and **cover the most recent evolutions** of this malware's steganographic loading mode, which appeared in March 2026. We will detail the evolutions of this mode and **provide a Python code snippet enabling the extraction and decryption of a payload**.

We will also see **how to detect these codes** and how to potentially track their future evolutions on various platforms.

Finally, we will **provide all relevant IOCs** in the appendix.

## PixyNetLoader

This threat fits a relatively standard compromise scheme through vulnerability exploitation via a malicious .DOC file ( CVE-2026-21509 in February 2026) executing a version of the SimpleDropper code, which in turn drops a **PixyNetLoader** DLL installed via COM persistence and a PNG file. **PixyNetLoader** loads the .PNG file, extracts a **Covenant Grunt** payload from the pixels' LSBs, using the filen service for communication.



We chose to cover the **PixyNetLoader** code because it is the most likely to be detected on a network since it is installed with persistence and is not encrypted. Furthermore, [an article from CERT-UA in February 2026](#) shows that it is still current. This code was notably covered in the articles [OPERATION PHANTOM NET VOXEL](#) by Sekoia and [OPERATION NEUSPLOIT](#) by ZSCALER.

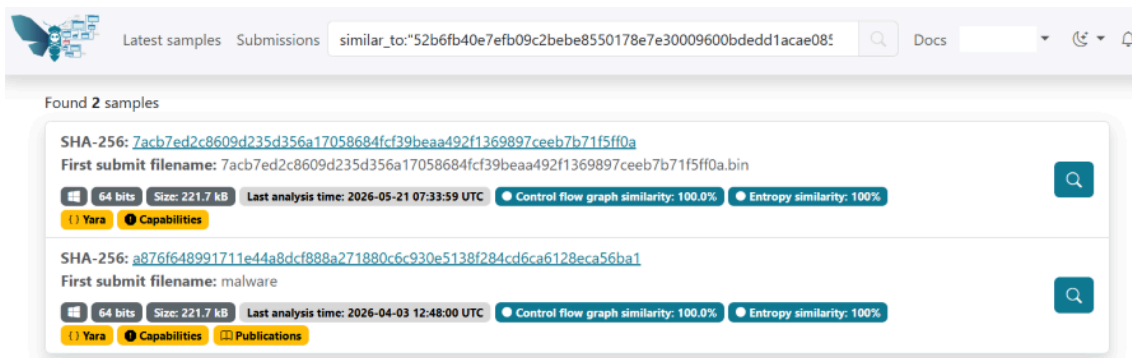
## Clustering

Let's take a first sample, [52b6fb40e7efb09c2bebe8550178e7e30009600bdeedd1acae085d753761b7598](#), referenced in the [February Cert UA article](#).

This is a **PixyNetLoader**; the DLL will be installed via COM persistence and will perform steganography to extract a shellcode from the PNG data in its companion file ( `%programdata%\Microsoft OneDrive\setup\Cache\SplashScreen.png` ), and which exports itself as `EhStorShell.dll`. The payload is contained in the least significant bits of the file's pixels.

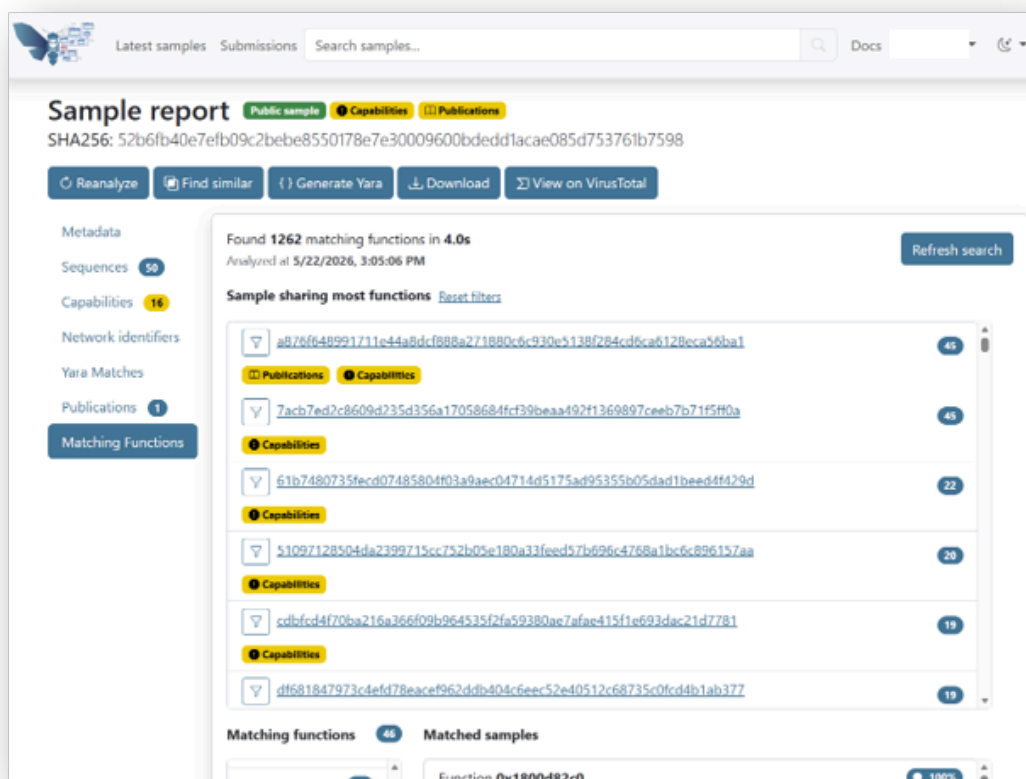
A similarity search immediately reveals 2 other samples:

- [a876f648991711e44a8dcf888a271880c6c930e5138f284cd6ca6128eca56ba1](#) referenced in the *Operation Neusploit* campaign, compiled on 2026-01-23
- [7acb7ed2c8609d235d356a17058684fcf39beaa492f1369897ceeb7b71f5ff0a](#) compiled on 2026-01-23



These are similar codes, slightly modified but based on the same source.

A [search by function similarities](#) (i.e., a function-by-function comparison with all binaries on Exalyze) brings up 87 other samples that match between 4 and 45 similar functions.



We have listed a total of 23 different versions of PixyNetLoader, ranging from 2024-04-12 to 2026-04-15. We group each version by `RICH` header hash and compilation timestamp. These 23 versions cover a total of about a hundred samples (we deliberately exclude approximately 430 samples of the 58a6e3e4 family, see below). 1 Among these 23 versions, we can find 4 different families, by cross-referencing the similar functions in each:

- **Family A**, described in Operation Phantom Net Voxel, which covers from December 2024 to July 2025, with 3 versions on January 21 and 23, 2026, described in Operation Neusplot.
- **Family 58a6e3e4**, which includes a single version from 2025-09-04. This version is particularly noteworthy because approximately 430 samples of it can be found on VirusTotal, with slightly and randomly code

modification, continuously pushed by the same user. Certain functions responsible for PNG header parsing disappear, such as functions dealing with RGBA codes, grayscale levels, etc.

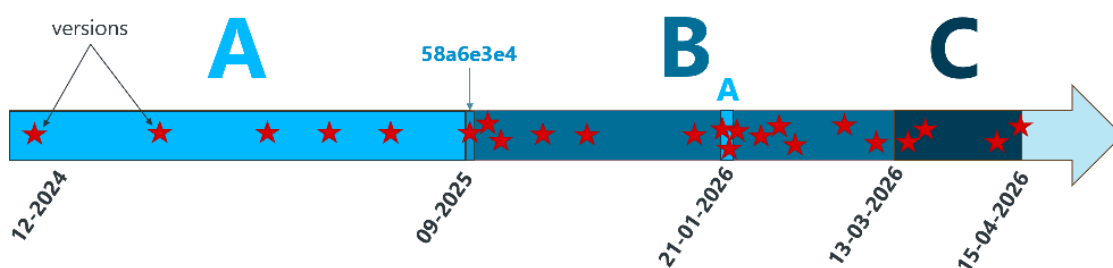
- **Family B**, which covers from September 2025 to March 2026. Some PNG lib functions persist.
- **Family C**, which begins on 2026-03-13. We note here a modification of the payload loading using a new steganography mode. This operating mode will be described in a dedicated section.

Note that some samples embed the PNG file directly as a resource.

Some codes of the C family are barely detected, such as

[a5729b6e36c0ab4798db5004700a1fe843f4d1b0811023c47b7b2972befb6360](#) which only 2 engines detect.

Details in terms of configurations (companion file, DLL export name, etc.) are provided in the appendix.

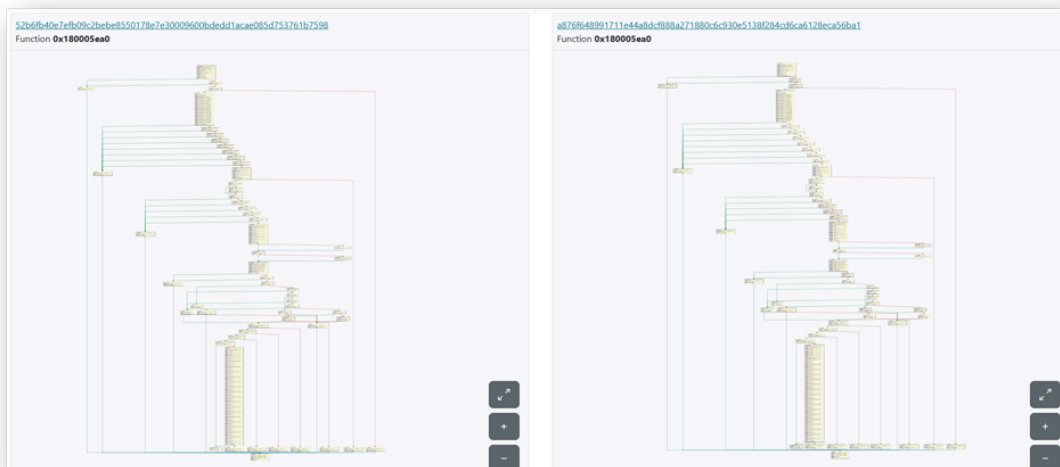


Some **PixyNetLoader** code strains are “hybrid” and possess several striking elements from earlier and later versions. We therefore simply repeated the process of binary diffing on Exalyze, create new rules to obtain more samples, and so on. It was also quite simple to reuse certain unique version info to identify other slightly different codes (or even belonging to other APT28 code families).

Writing rules targeting these similarities is interesting because it allows for better categorization of samples by internal functioning. It is by using function comparisons and defining YARA rules on common functions that we were able to establish larger families. For example:

- Code responsible for LSB processing of PNG files (Families A, B, and C)
- PNG library (useless code present in Family A)
- PE Parser (Families B and C)
- Specific string encryption (all)
- Import resolution (Family B)
- Code responsible for Family C steganography
- Pattern/magic search (Family C)

For example, [analyzing the PNG library of strain A](#) makes it quite easy to write a YARA rule.



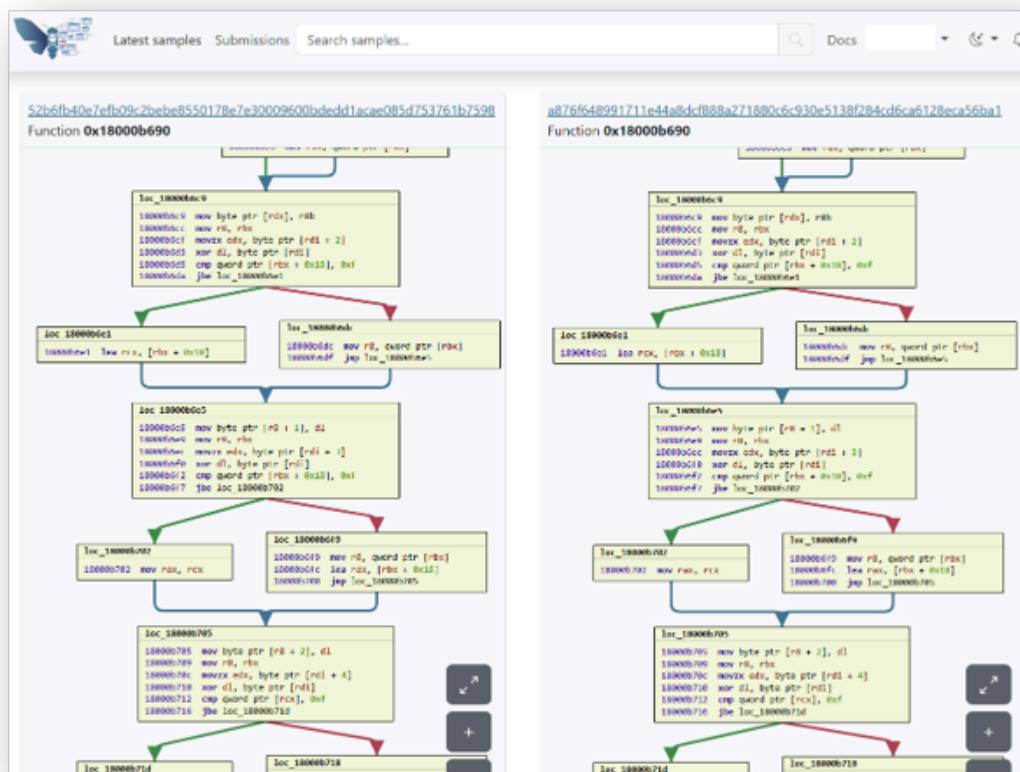
And the YARA rule:

```
private rule DllCom {
    strings:
        $com_01 = "DllRegisterServer"
        $com_02 = "DllCanUnloadNow"
        $com_03 = "DllGetClassObject"
        $str_01 = "GetProcAddress"
        $str_02 = "GetCurrentProcess"
        $str_03 = "CreateMutexW"
        $str_04 = "LoadLibraryW"
    condition:
        uint16be(0) == 0x4d5a and 2 of ($com_*) and 2 of ($str_*) and filesize < 800KB
}

rule PixyNetLoader_B_PNG {
    meta:
        author = "Exatrack"
        date = "2026-05-25"
        description = "PixyNetLoader PNG functions"
        score = 90
        tlp = "GREEN"
        sample_hash = "52b6fb40e7efb09c2bebe8550178e7e30009600bdedd1acae085d753761b7598"
    strings:
        $png_search_idat = { 80 7B 04 49 75 ?? 80 7B 05 44 75 ?? 80 7B 06 41 75 ?? 80 7B 07 54 75 ?? }
        $png_search_tIME = { 80 7B 04 74 75 ?? 80 7B 05 49 75 ?? 80 7B 06 4D 75 ?? 80 7B 07 45 75 ?? }
        $crc32_1 = { 0F B6 11 33 ?? C1 E2 08 44 8B ?? 0B D0 0F B6 41 02 C1 E2 08 0B D0 }
        $crc32_2 = { 8B C1 48 33 D0 C1 E9 08 44 0F B6 C2 49 FF C1 }
    condition:
        DllCom and all of ($png_*) and all of ($crc32_*)
}
```

## Writing a single and simple yara rule

With Exalyze matching functions feature, it is possible to isolate 4 functions that are found in the vast majority of samples:



These are [functions responsible for string encryption](#), found in almost all samples, regardless of the binary strain, sometimes with minor variations. We can therefore create a very basic but functional yara rule:

```
rule PixyNetLoaderCrypto {
  meta:
    author = "Exatrack"
    date = "2026-05-25"
    description = "PixyNetLoader crypto strings functions"
    score = 90
    tlp = "GREEN"
    sample_hash = "52b6fb40e7efb09c2bebe8550178e7e30009600bdedd1acae085d753761b7598"

  strings:
    $pattern1_01 = { 0FB6??243841B7010000008BC?418BC?D2E0B908000000 }
    $pattern1_02 = { 410FB6C34503C02407B9070000002AC8498BC348C1E80349FFC3 }

    $pattern2_01 = { 4488024C8BC30FB657023217 }
    $pattern2_02 = { 418850014C8BC30FB65703321748837B180F }

    $pattern3_01 = { 0FB6C8D2EA4132D0F6C201 }
    $pattern3_02 = { D2EA410FB6C0D0E832D0F6C201 }
    $pattern3_03 = { 0FB6C8D2EA410FB6C0C0E80332D0F6C201 }

  condition:
    $pattern1_01 && $pattern1_02 && $pattern2_01 && $pattern2_02 && $pattern3_01 && $pattern3_02 && $pattern3_03
```

```
(all of ($pattern1_*) or all of ($pattern2_*) or all of ($pattern3_*)) and DllCom
}
```

YARA results on the Exalyze platform can be found here: [https://exalyze.io/sample/search?q=yara%3A%22PixyNetLoader\\_CryptoAllVersions%22&submit=](https://exalyze.io/sample/search?q=yara%3A%22PixyNetLoader_CryptoAllVersions%22&submit=)

## PixyNetLoader Family C

Let's focus on the last family of these samples, as it the newer, less detected one, and as it introduces a new steganography mode.

We have seen this strain packaged in malicious .xls files:

<https://www.virustotal.com/gui/file/87a962c6599176e1806c0ccd1b157d3f80e3ccc288c288d039872d9683da24d9/behavior>

| Товар                                                                             | Категорія     |
|-----------------------------------------------------------------------------------|---------------|
| <b>Найменування</b>                                                               |               |
| <b>Квадрокоптер DJI Mavic 3 Thermal</b>                                           | Квадрокоптери |
| <b>Квадрокоптер DJI Mavic 3 Enterprise (Universal Edition)</b>                    | Квадрокоптери |
| Пропелери для квадрокоптера (2 пари)                                              | Аксесуари     |
| Захист для пропелерів квадрокоптера                                               | Аксесуари     |
| Чохол Airsight Sleeve для квадрокоптера DJI Mavic                                 | Аксесуари     |
| Зарядний хаб для акумуляторів DJI FPV Charging Hub без блоку живлення             | Аксесуари     |
| Сумка для квадрокоптера DJI Mavic та аксесуарів                                   | Аксесуари     |
| Фіксатор пропелерів для квадрокоптера DJI Mavic 3T                                | Аксесуари     |
| Акумулятор для квадрокоптера DJI Mavic 3T                                         | Аксесуари     |
| <b>Квадрокоптер Autel EVO MAX 4T</b>                                              | Квадрокоптери |
| Нижній демпфер камери для квадрокоптера Autel EVO Max 4T (SLS)                    | Аксесуари     |
| Відеоплівка підвісу для квадрокоптера Autel EVO Max 4T                            | Аксесуари     |
| Силиконовий поглинач вібрації камери для квадрокоптера Autel EVO Max 4T           | Аксесуари     |
| Ширококутна камера підвісу для квадрокоптера Autel EVO Max 4T                     | Аксесуари     |
| Модуль-розгалужувач підвісу для квадрокоптера Autel EVO Max 4T                    | Аксесуари     |
| Польотний контролер та контролер обертів двигунів для квадрокоптера Autel EVO Max | Аксесуари     |
| Плата Head Extension Board для квадрокоптера Autel EVO Max 4T                     | Аксесуари     |
| <b>Квадрокоптер DJI Matrice 4T</b>                                                | Квадрокоптери |
| Пропелери висотні 1676 для Matrice 30 Series                                      | Аксесуари     |
| Пульт дистанційного керування DJI Smart Controller Enterprise                     | Аксесуари     |
| Інтелектуальна батарея TB30 Intelligent Flight Battery                            | Аксесуари     |
| Зарядний пристрій BS30                                                            | Аксесуари     |
| "ФАНТОМ на 8 діапазонів" (300-1000 МГц; 2.4 ГГц; 5.8 ГГц)                         | РЕБ           |
| "ФАНТОМ на 5 діапазонів" (400-1000 МГц)                                           | РЕБ           |
| "ФАНТОМ на 6 діапазонів" (300-1000 МГц)                                           | РЕБ           |
| "ФАНТОМ на 11 діапазонів" (300-1200 МГц; 2.4 ГГц; 5.8 ГГц)                        | РЕБ           |

The file leads to the dropping of a **PixyNetLoader** and its companion:

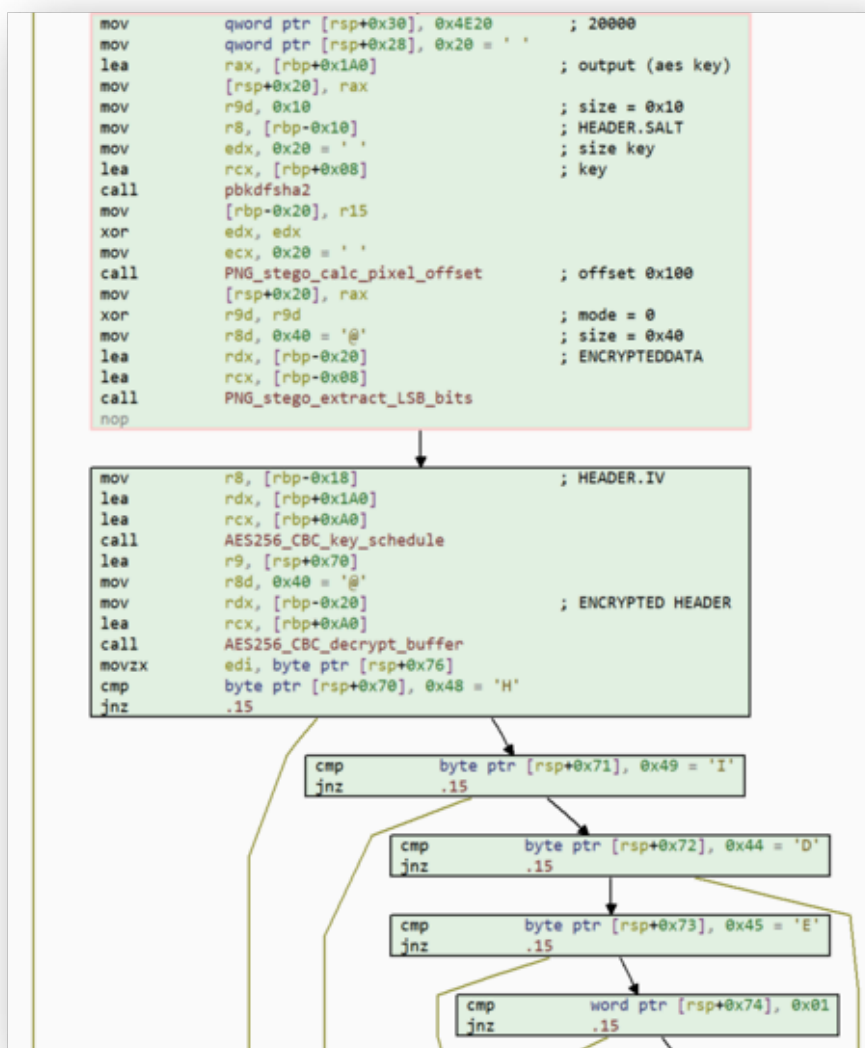
- [c6f13923f1310433b6271096957e71943543b7e41522367e4e855e653bf1c5a5](#) : PixyNetLoader 2026.04.15
- 5fce3647ca4e8bab5d981b73bf2d2dfefb9699e1f898323e8aa0d963cee46583 : PNG companion file

These resources can be extracted quite easily from the raw file.

This new version radically changes the loading of PNG files:

- A secret is embedded in the binary
- This secret undergoes a sha256 hash followed by a byte permutation
- The PNG pixel stream is read and 3 steganography modes are supported to extract data

- The first read will read an IV, a SALT, and a 0x40 byte header at several offsets
- The embedded and hashed secret is then transformed into an AES key via a PBKDF2 HMAC SHA256 of 20000 iterations using the extracted SALT
- The header is decrypted and the HIDE magic is verified. This header contains the payload offset, its size, as well as a checksum to verify decryption
- The payload is read and then decrypted with the already instantiated AES key
- This payload is then executed in RAM



We were able to create a python script for payload extraction and decryption available in the Appendix A below.

[illegible]

For the moment, all Family C versions use the LSB extraction mode 0. Modes 1 and 2 are supported but not used. Their main difference consists of the number of least significant bits extracted per pixel.

It should also be noted that the attacker uses versioning in their PNG files; interestingly, the date indicated `ca_distr_15.04.exe.shellcode` is generally roughly the same day (more or less, several have 1 day of delta) as the compilation timestamp of the **PixyNetLoader** binary that loads it.

The payload remains a **Covenant Grunt** malware (VersionInfo Publish.exe ) using **FILEN** as Cloud C2, prefixed by a shellcode allowing it to be loaded.

## Detection

Detection of these codes is relatively trivial if proper tracking is performed. The companion .png files do not vary that much, and it is possible to search for them on the network. The COM registry keys used for persistence are also relatively few. Finally, we were able to show that effective YARA rules covering the majority of these codes are also quite easy to write.

## Monitoring new codes

By taking the global characteristics of these samples and looking for those that group them together:

- bad checksums and bad timestamps due to file patching
- systematically a 64-bit DLL installed in persistence
- C++ code
- crc32 capability
- exposing `microsoft corporation` as `companyname` in version info
- compilation timestamp visibly not falsified and not being a Microsoft build ID (absence of debug information in the PE metadata indicating that it is precisely a build ID)

It is possible to define Exalyze searches that cross-reference a number of them across all versions, without even using the code structure itself and focusing only on file global profiles. For example:

- [capa:“Bad Timestamps 01” capa:“Bad Checksum 02” capa:“PE\\_parseur 02” capa:“list files 01” capa:“cpp code 01” capa:“crypto CRC32” bitness:“64” size below:“800Kb” company name:“Microsoft](#)

### Corporation”

- [capa:“Bad Checksum 02” capa:“list files 01” capa:“Bad Timestamps 01” capa:“dll service 01” capa:“cpp\\_code 01” bitness:64 company\\_name:“Microsoft Corporation”](#)

The second, slightly broader search, brings up other samples. After isolating those clearly not linked to APT28, we find 2 APT28 samples that are not PixyNetLoader:

- [9faeb1c8a4b9827f025a63c086d87c409a369825428634b2b01314460a332c6c](#) APT28 **SlimAgent**
- [efa5b49bdd086125b2b7d4058d09566f1db5f183c2a6332c597322f85107667a](#) APT28 **Graphite**

We therefore see that this player is keeping certain code packaging habits, which can, in addition to bouncing from one to another, potentially make it possible to identify new families. However, it should be kept in mind that these habits are not unique to APT28 and may also be present in other players.

## Conclusion

We have seen that it was relatively easy to track these codes and identify common points allowing the creation of effective YARA rules (the analysis from start to finish took us about 6 days of work). Unsurprisingly, these are mainly reused libraries or cryptographic routines.

We also saw that the latest iteration of the code introduces a new payload extraction mode, which is more robust (separate keys) and appears set to evolve: 2 of the LSB extraction modes are currently unused.

It was also possible to track certain strains simply by searching for the unique version info used by the attacker or the rich hashes.

The attacker generates new versions relatively often, and the files are used very quickly after generation, given the delay we observed between code generation and the actual attack.

```
from PIL import Image
import sys
import hashlib, struct
import zlib
from Crypto.Cipher import AES

# WARNING : MODE 1 AND 2 have not seen in the wild yet
def extract_lsb(
    image_path: str,
    mode: int,
    byte_offset: int,
    count: int
) -> bytes:
    img = Image.open(image_path).convert("RGBA")
    pixels = bytes(img.tobytes())
    base = byte_offset
    out = bytearray()

    if mode != 0:
```

```

        raise ValueError("Only '0' LSB mode is supported for now")

    if mode == 0:
        for i in range(count):
            o = base + i * 8
            p = pixels[o:o+8]
            r8 = (p[7] & 1)
            r8 = ((r8 << 1) | (p[6] & 1)) & 0xFF
            r8 = ((r8 << 1) | (p[5] & 1)) & 0xFF
            r8 = ((r8 << 1) | (p[4] & 1)) & 0xFF
            r8 = ((r8 << 1) | (p[3] & 1)) & 0xFF
            r8 = ((r8 << 1) | (p[2] & 1)) & 0xFF
            r8 = ((r8 << 1) | (p[1] & 1)) & 0xFF
            r8 = ((r8 << 1) | (p[0] & 1)) & 0xFF
            out.append(r8)

    return bytes(out)

def derive_primary_key(initkey) -> bytes:
    raw_digest = hashlib.sha256(initkey).digest()
    # byteswap
    words = struct.unpack(">8I", raw_digest)
    key = struct.pack(">8L", *words)
    return key

def decrypt(image_path, aeskey):
    salt = extract_lsb(image_path, mode=0, byte_offset=0x0, count=0x10)
    iv = extract_lsb(image_path, mode=0, byte_offset=0x80, count=0x10)
    datadec = extract_lsb(image_path, mode=0, byte_offset=0x100, count=0x40)
    nkey = derive_primary_key(aeskey)
    derivatedkey = hashlib.pbkdf2_hmac("sha256", nkey, salt, 20000, dklen=32)
    cipher = AES.new(derivatedkey, AES.MODE_CBC, iv)
    clearheader = cipher.decrypt(datadec)

    print("---- HEADER ----")
    print(f"ORIGINAL KEY: {aeskey.hex()}")
    print(f"KEY SHA2+SWAP: {nkey.hex()}")
    print(f"SALT: {salt.hex()}")
    print(f"IV: {iv.hex()}")
    print(f"FINAL AESKEY: {derivatedkey.hex()}")
    print(f"DATA: {datadec.hex()}")

    if clearheader[:4] != b"HIDE":
        print("!!! Invalid header, bad magic")
        printable = ''.join(
            chr(b) if 32 <= b <= 126 else '.'
            for b in clearheader
        )
        print(clearheader.hex())

```

```
print(printable)
return

magic = clearheader[:4]
vers,pixel_offset,payload_size,checksum = struct.unpack("<4L", clearheader[4:0x14])
name = clearheader[0x14:].rstrip(b"\x00")
print("---- HIDE HEADER ----")
print(f"MAGIC: {magic}")
print(f"VERSION: {vers}")
print(f"PAYLOAD PIXEL OFFSET: {hex(pixel_offset)}")
print(f"PAYLOAD SIZE: {hex(payload_size)}")
print(f"CHECKSUM: {hex(checksum)}")
print(f"INTERNAL NAME: {name}")

payload_encrypted = extract_lsb(image_path, mode=0, byte_offset=pixel_offset, count=payload_size)
payload_decrypted = cipher.decrypt(payload_encrypted)
payload_decrypted = payload_decrypted[:-payload_decrypted[-1]]
crc = zlib.crc32(payload_decrypted)

print("---- PAYLOAD ----")
print(f"ENCRYPTED FIRST BYTES: {payload_encrypted.hex()[:80]}")
print(f"ENCRYPTED LAST BYTES: {payload_encrypted.hex()[-80:]}")
print(f"CLEAR FIRST BYTES: {payload_decrypted.hex()[:80]}")
print(f"CLEAR LAST BYTES: {payload_decrypted.hex()[-80:]}")
print(f"CLEAR CRC32: {hex(crc)}")

if crc != checksum:
    print("!!! Invalid data, bad checksum")
    return

open("payload.bin", "wb").write(payload_decrypted)
print("Payload is correct, has been written to <payload.bin> file")

if __name__ == "__main__":

    if len(sys.argv) < 3:
        print(f"Usage: python {sys.argv[0]} image.png aeskey")
        sys.exit(1)

    image_path = sys.argv[1]
    aeskey = sys.argv[2].encode()
    decrypt(image_path, aeskey)
```

## APPENDIX B: families

### FAMILY A

## 2024-12-04

- Compilation timestamp: 2024-12-04 15:24:42 UTC
- Exports as: prnflldr.dll
- VersionInfo.original\_filename: prnflldr.dll.10.0.19041.630
- RICH header hash: 3e285ada61d5ede750b07ff79a8dac2d
- PNG file loaded: %LocalAppData%\windows.png
- Samples:
  - [04fc885a264758b240151b893216b58d9cd9907906926f8a825cbbdf4b6301ef](#)
  - [2b0865f33870663ae9df2f90418a85a46cebea5e47b062e2ef94cc815bec45db](#)
  - [3db03895f1716c39ecc4b34a000049a520bf67101ce205c867b78907666a052e](#)
  - [6c5d347e298810b576b02aa45f04659679b20be6f26cb3c8d60a6fb04d86365d](#)
  - [7d2d930af3c220ecc459e5006bf9ecf156bd4dc634707818df90d5fb666e1912](#)
  - [888693138a8c88da46dc9143689c223531873250245c7e58683e2ecfa282a814](#)
  - [9c56917269d76bb4be4a8f51c423c44a411a628ab286c2a58f5b54345d0e000e](#)
  - [e8c9123d1539a28c89dff7365ecc760440cd48b94639aa0eeca44ca84bc7eff](#)

## 2025-04-02 (Operation Net Voxel)

- Compilation timestamp: 2025-04-02 07:47:06 UTC
- Exports as: prnflldr.dll
- VersionInfo.original\_filename: prnflldr.dll.10.0.19041.630
- RICH header hash: 96c820dcf9be3092f9bbcaa999893db2
- PNG file loaded: %LocalAppData%\windows.png
- Samples:
  - [693c9b517ab6a792bd32459650f08f62f9ade93c1aa23f2cec140da46d217aca](#)
  - [ce5a0cb1980b777c9db820ed172ff943ced1ff0ad88b6d9605f843f7b7824caa](#)

## 2025-06-11 (Operation Net Voxel)

- Compilation timestamp: 2025-06-11 08:52:25 UTC
- Exports as: prnflldr.dll
- RICH header hash: dc1fcda71cf242410a38bd390a23df94
- PNG file loaded: %public%\pictures\WordIllustration.png
- Samples:
  - [2128151c685d55b01271cd51d7bd692c51085ac94c028c871944348b9acbe769](#)
  - [bfbdf988ddc110d67abc83d0361f97066114fdc96ad9d89338d2f68d51b25229](#)
  - [28da9b18b06d59eede211e247e2a81b66a0cf12dea2ca442bc263c2f6e1c9135](#)
  - [88e28107fbf171fdbcf4abbc0c731295549923e82ce19d5b6f6fefa3c9f497c9](#)

## 2025-07-17 (Operation Phantom Net Voxel)

- Compilation timestamp: 2025-07-17 11:33:11 UTC
- Exports as: stobject.dll
- RICH header hash: 48d561ec2a60fcde704a52f7f0cd1f8b

- PNG file loaded: %programdata%\Microsoft\DeviceSync\UIDD304d-22c6-4f29-801b-58d0685fe77b\Default.png
- Samples:
  - [393f9dd6f5cd44b7c34bec03d8cb32c4bebbe29ccf74cb2c361258d3b9ba4acc](#)
  - [4f59a15ec8954971503d0a14a31d849c1930096c31f53c8b01dee45659cdf324](#)
  - [6eddc1343d2ae7a102bf268673fc35fbc0639be4c501589a847f94efcc81ac71](#)
  - [7bb241f88dce49ac7ff73ad27b9168b7103c669d22861ecb03318a4f29f281c4](#)
  - [e6f29c0ab5e42088eae7834616eb7a7ade3d17d8d89117cc3ec67b13d16a6eb4](#)

## 2025-08-26

- Compilation timestamp: 2025-08-26 07:45:35 UTC
- Exports as: stobject.dll
- RICH header hash: a7f2aa814519714e32a30e58ee04691b
- PNG file loaded: %programdata%\Microsoft\DeviceSync\UIDD304d-22c6-4f29-801b-58d0685fe77b\Default.png
- Samples:
  - [9aea0a84e91d8dc08896ad55aafd4623de2329fed2201abb48b27d709be15fe4](#)

## 2026-01-21 (Operation Neusploit)

- Compilation timestamp: 2026-01-21 15:06:04 UTC
- Exports as: EhStorShell.dll
- RICH header hash: 653ccf5e6edb3e9fb519b747ef238606
- PNG file loaded: %programdata%\Microsoft OneDrive\setup\Cache\SplashScreen.png
- Samples:
  - [52b6fb40e7efb09c2bebe8550178e7e30009600bdedd1acae085d753761b7598](#)

## 2026-01-23a (Operation Neusploit)

- Compilation timestamp: 2026-01-23 08:09:22 UTC & 2026-01-23 12:19:38 UTC
- Exports as: EhStorShell.dll
- RICH header hash: 653ccf5e6edb3e9fb519b747ef238606
- PNG file loaded: %programdata%\Microsoft OneDrive\setup\Cache\SplashScreen.png
- Samples:
  - [a876f648991711e44a8dcf888a271880c6c930e5138f284cd6ca6128eca56ba1](#)
  - [7acb7ed2c8609d235d356a17058684fcf39beaa492f1369897ceeb7b71f5ff0a](#)

## FAMILY 58a6e3e4

### 2025-09-04

- Compilation timestamp: 2025-09-04 18:29:32 UTC
- Exports as: stobject.dll
- Remarque : Tons of resubmits on VirusTotal with slight modifications
- PNG file loaded: %programdata%\Microsoft\DeviceSync\IID11f5d-97b9-44be-812f-4fd5fd0d6d84\Sample.png

- Samples:
  - [0370fa7b7d47f40acd1a3f185a872136c9205d3066ca7d18151d41358aedc51d](#)
  - [0673ddf92a0c4ac0826c87a776f12c50f907064611ecb81316c5b2bc6c270336](#)
  - [1a8b12b856695db262d6159f66000ec5756b2270449d6cd8df36c31919cd70c4](#)
  - [1f88553336b4f23581a99a0ff1bbfacfd9c44ea60cd5330720726dba682aa1c](#)
  - [382659b350540100a050631ff12aa1b82efdb3319e92f2cfc93b553b100ae734](#)
  - [3b171233191191e393e2f1d4405e9259fd7791f4b4bf907439b7d68629791f36](#)
  - [46eb61fafa99d2ff405253571fee1b127ba01c79cbcbda52bdc7510914a309f1](#)
  - [5c97ca26f02b88c7c0652c1569cf5b197be8d2de9ccccc93dc7aea7f7d54db4c](#)
  - [62a5e2f689fd5c1bd08222010413440174dc63c66136f6d40532ccc01ea634b4](#)
  - [6b90c20b712d2e0b03dd0f39532b185d564cbadf54d360f62862a755d0397cf7](#)
  - [8401a652a02de429dd4a707d4f8f6853c0d966cee73289a20e48f2fc1da224f9](#)
  - [9231d148284b864995e35c09552a9e1cc4783e537f39d5be5e162291038c2fb3](#)
  - [92475b3aa4893906442f4131566f364d6da856e2e07b8fa81591e107d2854423](#)
  - [9a86a328e2f9ce538560b0a67f4f1e385ccb1092ee69e153e610c475c5e3e142](#)
  - [9c57572ea3d05769c5e32f532f99ae9f440de7bd14fcff6b32077d562aaeccae](#)
  - [aa6faf830fd9757b7d9bf813df5d2c165c5d12556f7f4d57a7d95306e8317e6e](#)
  - [bc4dc9d7192730671be33b6e82bb701b5b52421fee7d57c975dc67243809ce25](#)
  - [bd08e75c8e847f507ae2e0d1fed5caed68ac61ffa64cc8fefb89c0ea5e291ec7](#)
  - [c7ca1d62f5d041aa94516cc2372f8fe22cd4d026dcba5420dd17401ff870dae](#)
  - [d79e76346700d5bef67221a1f2c524fe9826f549100dab3378e97d7c8c8e92f5](#)
  - [e130181a1a6b96df49ac78786a358b7b22410f477ad112e0880480522f04e2a8](#)
  - [f3183573c84183bcb4dd72d072210ebca2104c2dbd917c2d3762ea8a3db9603f](#)

## FAMILY B

### 2025-09-10

- Compilation timestamp: 2025-09-10 18:37:19 UTC
- Exports as: stobject.dll
- RICH header hash: 4ecb81d65056ff4c631ed412d17badd8
- PNG file loaded: %programdata%\Microsoft\DeviceSync\IID11f5d-97b9-44be-812f-4fd5fd0d6d84\Sample.png
- Samples:
  - [c48b0301102857dab6e74ad4fd4194bf803c793448616ec9ff31a8524877c1d9](#)

### 2025-09-18a

- Compilation timestamp: 2025-09-18 00:57:33 UTC
- Exports as: stobject.dll
- RICH header hash: 1e8b4451ea7e06a50bf1cca90e233863
- PNG file loaded: %programdata%\Microsoft\DeviceSync\8acd6e71-bf10-4800-aeec-7de00edc9781\background.png
- Samples:
  - [50b9cf048494a07be074e9fccf776dc9df5a502441378a6f83c1ba81b6ca3ee7](#)

## 2025-09-18b

- Compilation timestamp: 2025-09-18 01:12:59 UTC
- Exports as: stobject.dll
- RICH header hash: 29e2a7b7793275bcd0e5bdda4bdce2d1
- PNG file loaded: %programdata%\Microsoft\DeviceSync\8acd6e71-bf10-4800-aeec-7de00edc9781\background.png
- Samples:
  - [2303c8c2d04b74e23ad36ab4bd8572d9182f3d7c696f1389e4e017656488c216](#)
  - [77e243d607f21f69cf8e1e5400c0fca6a250a4632e29a09f0faf9c347da54cd1](#)
  - [9cfd08c6cd5198e6374b504c2e177336cc61b2e51573d175f0e49ce791e745b0](#)
  - [e5a4f511763954392dc70a56a8106920715a9d6132774eaafd13ca40136908fa](#)

## 2025-09-29

- Compilation timestamp: 2025-09-29 17:52:25 UTC
- Exports as: stobject.dll
- RICH header hash: 29e2a7b7793275bcd0e5bdda4bdce2d1
- PNG file loaded: %programdata%\Microsoft\DeviceSync\8acd6e71-bf10-4800-aeec-7de00edc9781\background.png
- Samples:
  - [808e0684bf4a13d807d2c47c60ce694f0501a1fcdf1e9e3533eda653d4f543aa](#)

## 2025-10-14

- Compilation timestamp: 2025-10-14 23:02:08 UTC
- Exports as: adwapi32.dll
- VersionInfo.original\_filename: adwapi32.dll.10.0.19041.2
- RICH header hash: 6b2e5dc05eeae55ef7ad6ba2c186d794
- PNG file loaded: %programdata%\Microsoft\DeviceSync\8acd6e71-bf10-4800-aeec-7de00edc9781\background.png
- Samples:
  - [2d0d4db1933ebb7dae8e75fb256b9519784a7d542fa2f2acd33916604119cca4](#)

## 2026-01-16

- Compilation timestamp: 2026-01-16 17:29:19 UTC
- Exports as: adwapi32.dll
- RICH header hash: 650072184b3d32ae65a3b90abe5a991b
- PNG file loaded: %programdata%\Microsoft\DeviceSync\8acd6e71-bf10-4800-aeec-7de00edc9781\background.png
- Samples:
  - [4fe2e88ccba9923b65123571378869606073d8a655587fef41db1710212687ba](#)
  - [61b7480735fecdd07485804f03a9aec04714d5175ad95355b05dad1beed4f429d](#)
  - [64dc08cf36cb729aaa9d97d16c5aebd3a104e52f2a3f9b0a69299139cc2a137e](#)

## 2026-02-10

- Compilation timestamp: 2026-02-10 17:27:12 UTC
- Exports as: EdgeSyncPr.dll
- VersionInfo.original\_filename: EdgeSyncPr.dll.11.0.19041.662
- RICH header hash: f5d0eb2e10479422403810649a6b4a54
- PNG file loaded: %programdata%\Microsoft\DeviceSync\EdgeSync\start.png
- Samples:
  - [ab68161147a5436b262afd7ecfeda4cc1d42598115b638dd98d6f105fcf2a534](#)
  - [e59af29f66b1e8e4b4e42c74acc112c6e42ec0154c26e7b367c59e0797fa6013](#)

## 2026-02-17

- Compilation timestamp: 2026-02-17 16:51:58 UTC
- Exports as: EdgeSyncPr.dll
- VersionInfo.original\_filename: EdgeSyncPr.dll.11.0.19041.662
- RICH header hash: 9b8f31d6ac5b92b3041e586ccc43b41d
- PNG file loaded: %programdata%\Microsoft\DeviceSync\EdgeSync\start.png
- Samples:
  - [7137be6bf1a84c69e4ea868727d39997ebb1a9c75401cea0aa60ccd392e6a4ad](#)
  - [c177649197c6d2dc9b37007f72079d4f55ca53f372defc95fbb69f99e5980767](#)
  - [cdbfcd4f70ba216a366f09b964535f2fa59380ae7afae415f1e693dac21d7781](#)

## 2026-02-25

- Compilation timestamp: 2026-02-25 15:47:24 UTC
- Exports as: EdgeSyncPr.dll
- VersionInfo.original\_filename: EdgeSyncPr.dll.11.0.19041.662
- RICH header hash: b05fa3e2fe9f9b43a64a97f31e936da6
- PNG file loaded: %programdata%\Microsoft\DeviceSync\EdgeSync\start.png
- Samples:
  - [51097128504da2399715cc752b05e180a33feed57b696c4768a1bc6c896157aa](#)

## 2026-03-02

- Compilation timestamp: 2026-03-02 20:17:54 UTC
- Exports as: EdgeSyncPr.dll
- VersionInfo.original\_filename: EdgeSyncPr.dll.11.0.19041.662
- RICH header hash: 79917408d8d43a665e99c4d046827c0d
- PNG file loaded: %programdata%\Microsoft\DeviceSync\EdgeSync\start.png
- Samples:
  - [4c18a1f2987ca6c56d585cc74eb96abd25f1aa64f33381a3d007ad1f1bba4e59](#)
  - [62d3b82ac3688b1c00adce7cd241de2a50c24caac4ed6b8e46b16da1266457eb](#)
  - [8a076877e1feedfd105507e9785ea61d8cdecce9ba462323d7c27c3e0c278bf](#)
  - [c4e0c9a14820ecc4227a174c42d832e40ccd169aed4a10082decc06116c489ee](#)
  - [df681847973c4efd78eacef962ddb404c6eec52e40512c68735c0fcd4b1ab377](#)

## 2026-03-10

- Compilation timestamp: 2026-03-10 17:29:59 UTC
- Exports as: EdgeSyncPr.dll
- VersionInfo.original\_filename: EdgeSyncPr.dll.11.0.19041.662
- RICH header hash: b5a2251a7d96003a34d12175736535c3
- PNG file loaded: %programdata%\Microsoft\DeviceSync\EdgeSync\start.png
- Samples:
  - [32538331fb8a0159e5c71d6fa84be5bf3c14b7deb6ffee0276fa1898263b9cbd](#)
  - [39f24e078c66ed10702e31854211542d3d554af5d4c93bbe0bded9cf0349a759](#)
  - [57184dae6676e603e3984bbbe9be2b1ebf2c9bdc7b0b2ec460cc3e8258f963d8](#)
  - [b67f3d7c8332875866c3703174f11a3a773d00b98e80d3d4ed5e5450c32faf69](#)
  - [d4a7a3f60428d400ed4e544df4b200e8f867536279c37b388ab515d74a0e069c](#)

## FAMILY C

### 2026-03-13

- Compilation timestamp: 2026-03-13 19:39:01 UTC
- Exports as: EdgeSyncPr.dll
- VersionInfo.original\_filename: EdgeSyncPr.dll.11.0.19041.662
- RICH header hash: a6b6147a892c1d64a78a6d2a9086af59
- PNG file loaded: %programdata%\Microsoft\DeviceSync\EdgeSync\start.png
- Samples:
  - [6c8fdb3b96fa419afcd9ee61bee1eaa819ae9f0202cd3e2f22cca230aee34b](#)
  - [c1c8b70653f06a193285fba2ba1819ade2c05c9c80c734454cf61ccde685a744](#)
  - [dfeed404edb6809f4212b3856f1834844e05df7ea901d915f86a634f756e6624](#)

### 2026-03-19

- Compilation timestamp: 2026-03-19 18:08:02 UTC
- Exports as: EdgeSyncPr.dll
- VersionInfo.original\_filename: EdgeSyncPr.dll.11.0.19041.662
- RICH header hash: 162f98e27b567c2d1dcf1a26ad3c4ddc
- PNG file loaded: %programdata%\Microsoft\DeviceSync\EdgeSync\start.png
- Samples:
  - [1e28ac7c1f1b33046b3faef828a03d7666f57f90c0d7c864a27736f965bfb047](#)
  - [2adaccdd87760e05b6d7ee7dbbe705a6decad982f1813e248950c6169b10251f](#)
  - [2b78afe14fd883435dff43721382a355853e3c4ac5279d4ab9fac37b57f0502](#)
  - [4771feca8632b1840c7a92d121d1afb8d7d00f042355d1bb20265c1655975be9](#)
  - [7f5ac522a92a0e13cea771a482d69aace7df83c9340f1a069acbe5dd3afc7dfe](#)
  - [a5729b6e36c0ab4798db5004700a1fe843f4d1b0811023c47b7b2972befb6360](#)
  - [bc2dbf203f434b41700c551bb282f017fada9f8eefa807712f103f97ab6173e2](#)

### 2026-04-10

- Compilation timestamp: 2026-04-10 00:53:53 UTC
- Exports as: EdgeSyncPr.dll
- VersionInfo.original\_filename: EdgeSyncPr.dll.11.0.19041.662
- RICH header hash: 42b510520b574ba1086375730519fad9
- PNG file loaded: %programdata%\Microsoft\DeviceSync\EdgeSync\EdgeLogo.png
- Samples:
  - [31093f23161e5945947d9f0d9c710e4fd6d142254a4d040ac5f205fe08c0a089](#)
  - [6bedc8e0ae61475115b21f3d7a2237797969cefa5b2286c3ada455e4a8a3dc64](#)
  - [f611a21a95bfa8253c32416bb3505d21d5c21722ea293a3451d3eb831125ba25](#)

## 2026-04-15

- Compilation timestamp: 2026-04-15 20:21:47 UTC
- Exports as: EdgeSyncPr.dll
- VersionInfo.original\_filename: EdgeSyncPr.dll.11.0.19041.662
- RICH header hash: 6d7ddf39527c04c16523b74bd3d8b700
- PNG file loaded: %programdata%\Microsoft\DeviceSync\EdgeSync\EdgeLogo.png
- Samples:
  - [44ef866c8a5af6ab9e9e8bc51bcb9aed2fc125771c8d89aa05db5375a6a3b5c5](#)
  - [5771f7198b04eeaeec1ec3b6f4fbbd567e51b1dd14220298537903c03b912f06](#)
  - [724873377aee2ca23771a9c85fb60a2a041e17d7b460fb14564dec04a2da3b48](#)
  - [c6f13923f1310433b6271096957e71943543b7e41522367e4e855e653bf1c5a5](#)

## Appendix C: IOCS

### PNG files paths

PNG files paths

```
%LocalAppData%\windows.png
%programdata%\Microsoft OneDrive\setup\Cache\SplashScreen.png
%programdata%\Microsoft\DeviceSync\EdgeSync\EdgeLogo.png
%programdata%\Microsoft\DeviceSync\EdgeSync\start.png
%public%\pictures\WordIllustration.png
%programdata%\Microsoft\DeviceSync\IID11f5d-97b9-44be-812f-4fd5fd0d6d84\Sample.png
%programdata%\Microsoft\DeviceSync\8acd6e71-bf10-4800-aeec-7de00edc9781\background.png
%programdata%\Microsoft\DeviceSync\UIDD304d-22c6-4f29-801b-58d0685fe77b\Default.png
```

CLSID registry keys used (non exhaustive)

```
\Classes\CLSID\{68DDBB56-9D1D-4FD9-89C5-C0DA2A625392}\InProcServer32\
\Classes\CLSID\{D9144DCD-E998-4ECA-AB6A-DCD83CCBA16D}\InProcServer32\
\Classes\CLSID\{2227A280-3AEA-1069-A2DE-08002B30309D}\InProcServer32\
```

### DLL files paths

## Installation paths (non exhaustive)

```
%APPDATA%\microsoft\protect\altio32.dll
%ALLUSERPROFILE%\usopublic\data\user\ehstoreshell.dll
C:\ProgramData\prnflldr.dll
C:\ProgramData\Microsoft OneDrive Storage\MimeTypes\Default\mimeobj.dll
C:\ProgramData\USOShared\Logs\User\svrobj.dll
C:\ProgramData\USOShared\Logs\User\adwapi32.dll
C:\ProgramData\USOShared\Logs\User\FlightConfig.dll
C:\ProgramData\Microsoft\DeviceSync\EdgeSyncPr.dll
```

## Exploits

Several exploits (DOC, XLS), non exhaustive list

```
580cd001739f95c343c0b3f16ab2c274b54d126e1c35eaf3b7377add37435f22 drops ab681611
0773a145bff130fa527ecbf80400a6d630fa5ec0a53f6a252a7cf62fb63cf8d5 drops 57184dae
3e63952cdae714bcbdbf9a264e122c264f1f3e66d68cc6702ed1a8389153bb5c drops 57184dae
c752290553f23dc6dc3d0e78581b7a20c78571092895129513d0c195f6de360f drops 61b74807
f066bcb45151fdfe2f2921dd7cb4a09ed583f514ca62960263623365465553f drops 61b74807
8c1dc9732884c6078b23953b78314a8d0d8b8d9fe42e5f97a7cd09b8ace943a9 drops 52b6fb40
5c2a2c49e200a2d048f477440da75ff4a99c676943f6f7cac1ce70190520f998 drops 7acb7ed2
0003699a517af0a969058d3ad971704c11bb8bca2cb79994fe55cfbe6425fb68 drops 7bb241f8
7cfab5f53bbbd05c3d123393f2a6b41ff3cd46821b902d0b3eddd65b6476a99d drops 7bb241f8
8f049b3a100747167eb87fb3a134e446d9057f179b4f334a5a4006369605095a drops ce5a0cb1
57253f322504e0a8256d46f31c19e228b8c55a14ee18e759936c71941c8ee4ad drops 88e28107
17c697082bb95d05d5e761ca4a9cfd5ff10ff1a547a9639991924e8448f4d54 drops e5a4f511
```

---

Source: [https://blog.exatrack.com/Tracking\\_APT28\\_PixyNetLoader/](https://blog.exatrack.com/Tracking_APT28_PixyNetLoader/)