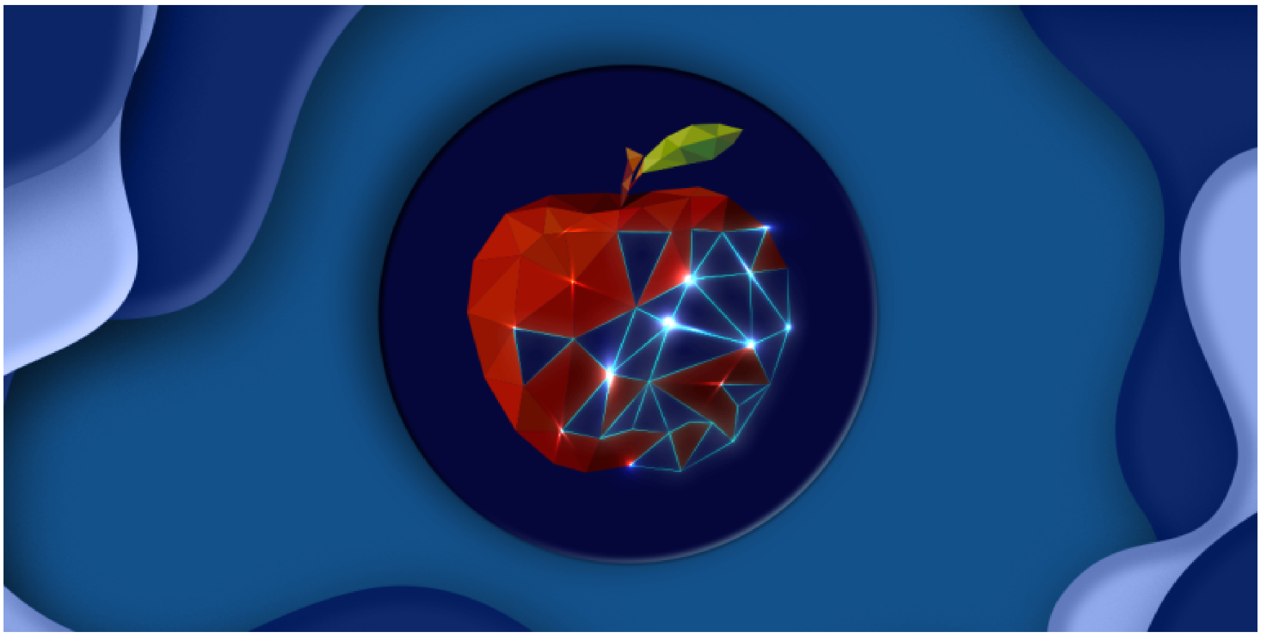
The strange link between a destructive malware and a ransomware-gang linked custom loader: IsaacWiper vs Vatet

: 5/3/2022



Cluster25 researchers, during a comparative analysis performed at the beginning of March 2022, found evidence that suggests a possible relationships between a piece of malware belonging to the **Sprite Spider** arsenal (or some elements that are or were part of it) and another malware that has recently taken part in destructive attacks against organizations and institutions in **Ukraine**.
C25 analyzes the evolution of the events that led to this analysis and the related conclusions.

INTRODUCTION

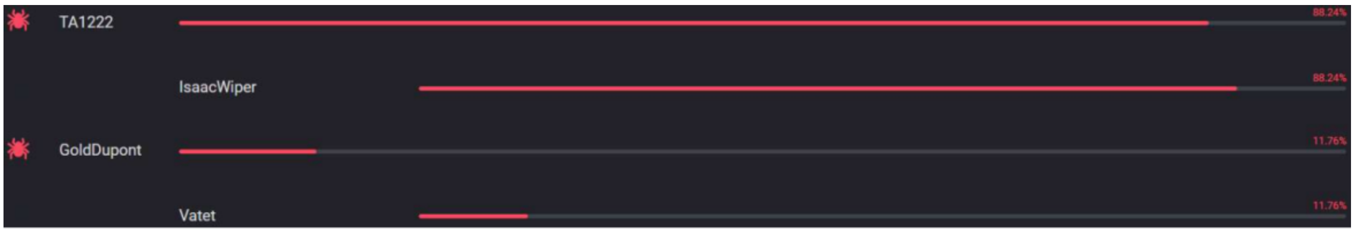
On **March 1st, 2022,** ESET researchers reported variants of a destructive malware deployed against Ukraine. (ESET Research: Ukraine hit by destructive attacks before and during the Russian invasion with HermeticWiper and IsaacWiper). On the same day C25 submitted copies of the reported samples to our internal attribution engine to acquire key points for the attribution.

Our attribution engine is designed to find similarities between malware samples by comparing the input with a vast pool of already attributed and categorized malware.

We submitted the file corresponding to the following hash:

TYPE	VALUE
SHA256	7bcd4ec18fc4a56db30e0aaebd44e2988f98f7b5d8c14f6689f650b4f11e16c0

leading to the following results:



Cluster25 internal threat attribution engine

TA1222 is the internal **Cluster25** nomenclature about an unidentified group and **GoldDupont** is a financially motivated threat actor, that has compromised several important targets, including **Regione Lazio**. It is also called **RansomExx**, **Defray777** or **Defray**.

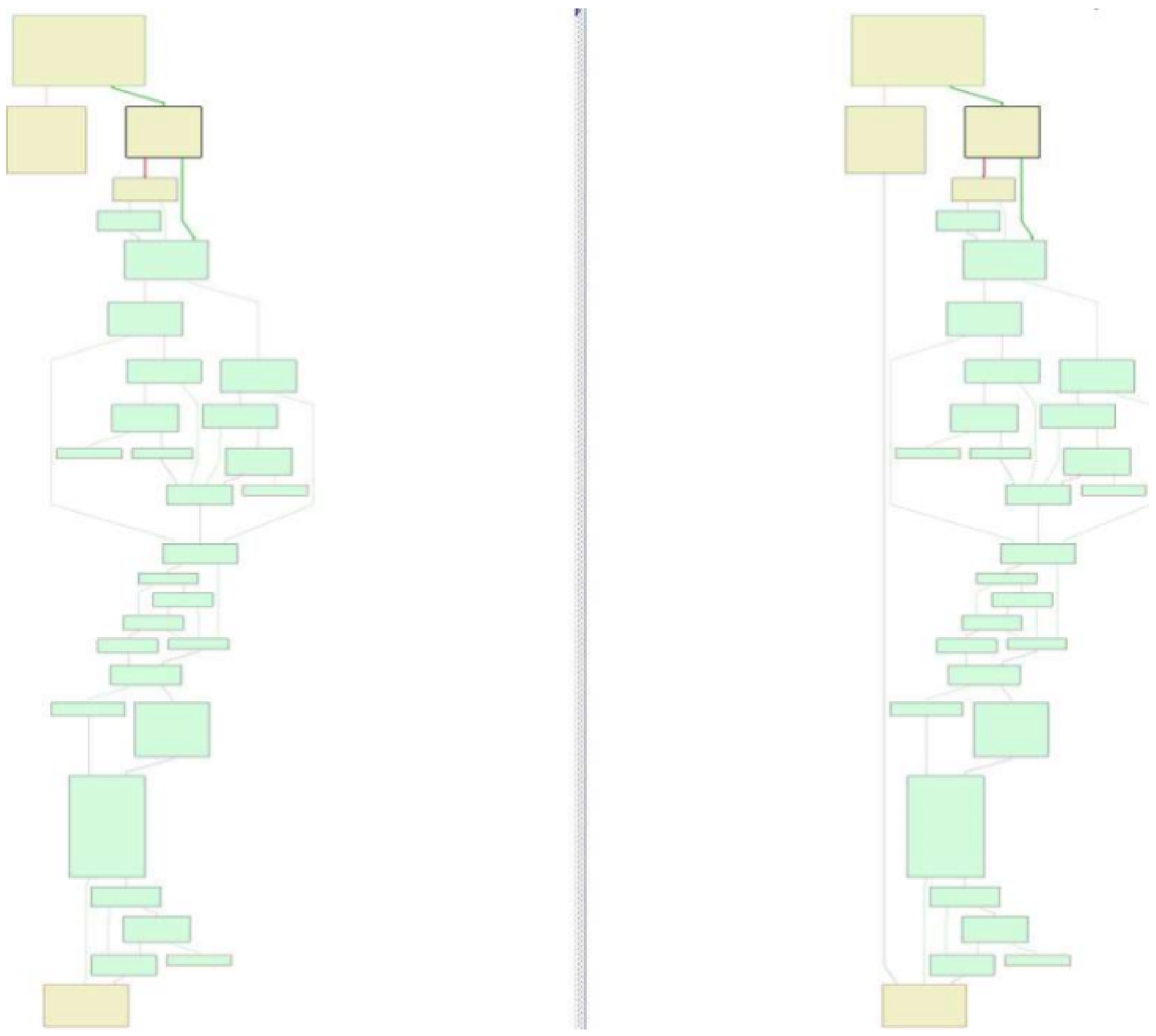
After this we manually analyzed the two samples.

INSIGHTS

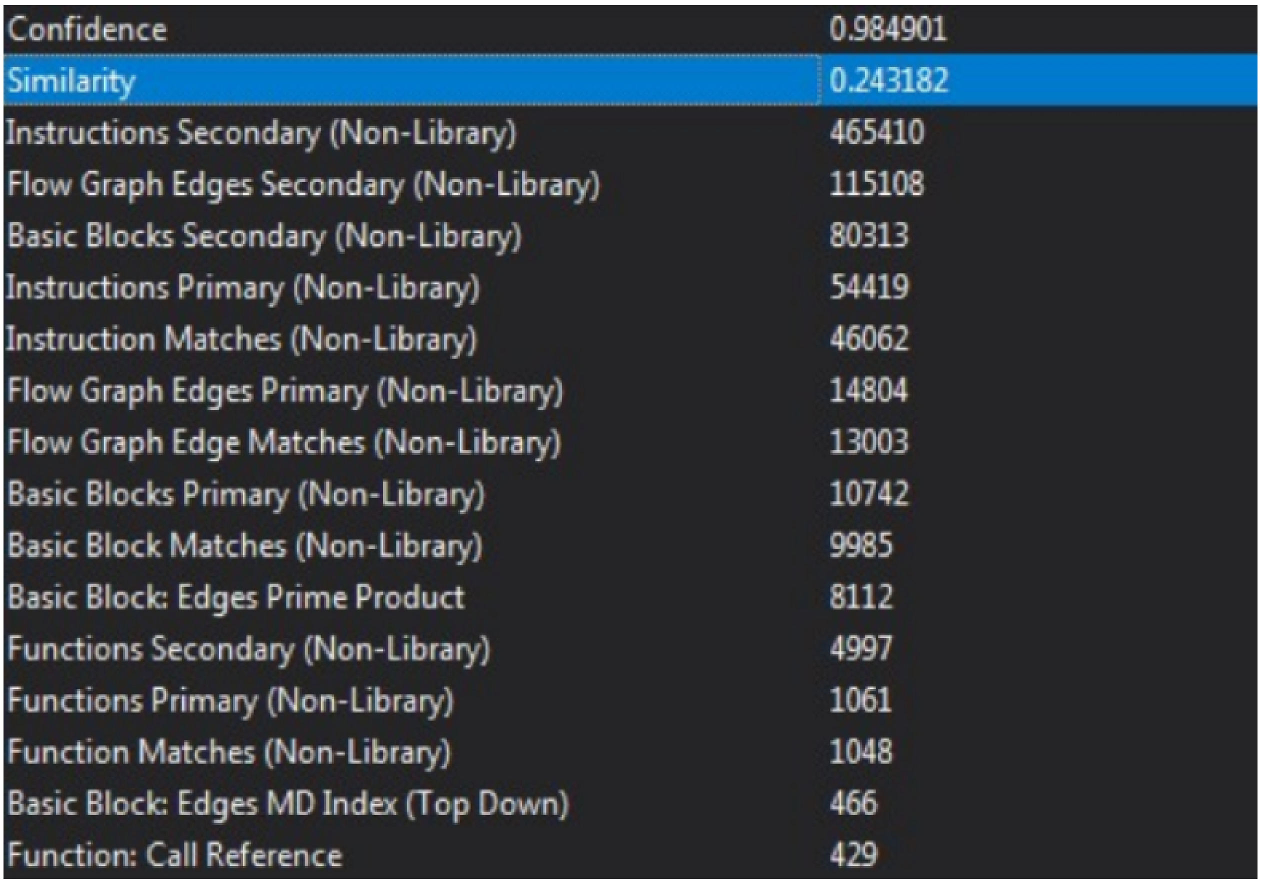
We spotted some overlaps in the source code of these two pieces of malware:

TYPE	VALUE
SHA256	7bcd4ec18fc4a56db30e0aaebd44e2988f98f7b5d8c14f6689f650b4f11e16c0
SHA256	01a2404fcf56027be610c65bbfb0f2dda9cfaf67385cb7f93f0b586e3aa6803a

In particular, code chunks at **sub_10005810** for **IsaacWiper** and **sub_100DDAE0** for **Vatet** have the following degree of similarity, as shown below graphically:

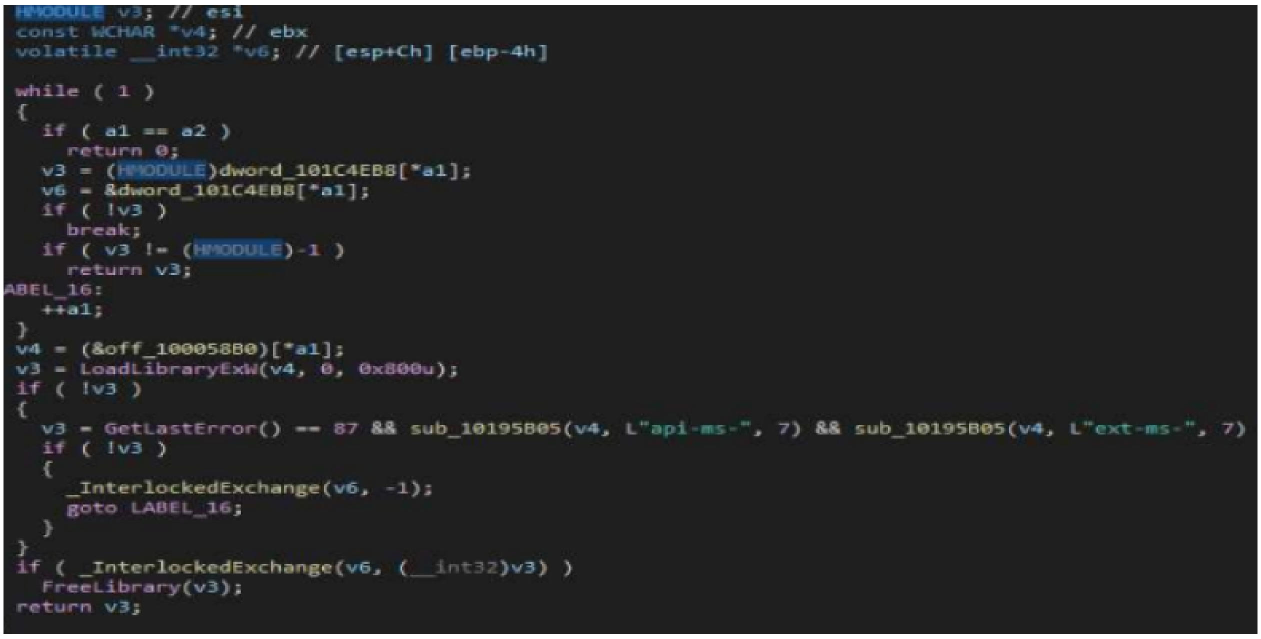


The similarity analysis returned a **24%** of rate with near **10.000** basic block that matched on non-library code



Confidence	0.984901
Similarity	0.243182
Instructions Secondary (Non-Library)	465410
Flow Graph Edges Secondary (Non-Library)	115108
Basic Blocks Secondary (Non-Library)	80313
Instructions Primary (Non-Library)	54419
Instruction Matches (Non-Library)	46062
Flow Graph Edges Primary (Non-Library)	14804
Flow Graph Edge Matches (Non-Library)	13003
Basic Blocks Primary (Non-Library)	10742
Basic Block Matches (Non-Library)	9985
Basic Block: Edges Prime Product	8112
Functions Secondary (Non-Library)	4997
Functions Primary (Non-Library)	1061
Function Matches (Non-Library)	1048
Basic Block: Edges MD Index (Top Down)	466
Function: Call Reference	429

As an example, the subroutines responsible for resolving Windows API's at runtime through **LoadLibraryExW** and **GetProcAddress** are the same on both the samples.



```
HMODULE v3; // esi
const WCHAR *v4; // ebx
volatile __int32 *v6; // [esp+Ch] [ebp-4h]

while ( 1 )
{
  if ( a1 == a2 )
    return 0;
  v3 = (HMODULE)dword_101C4EB8[*a1];
  v6 = &dword_101C4EB8[*a1];
  if ( !v3 )
    break;
  if ( v3 != (HMODULE)-1 )
    return v3;
ABEL_16:
  ++a1;
}
v4 = (&off_100058B0)[*a1];
v3 = LoadLibraryExW(v4, 0, 0x800u);
if ( !v3 )
{
  v3 = GetLastError() == 87 && sub_10195B05(v4, L"api-ms-", 7) && sub_10195B05(v4, L"ext-ms-", 7)
  if ( !v3 )
  {
    _InterlockedExchange(v6, -1);
    goto LABEL_16;
  }
}
if ( _InterlockedExchange(v6, (__int32)v3) )
  FreeLibrary(v3);
return v3;
```

The other identical subroutines are responsible for **error handling**, **heap memory allocation**, dynamic **API** resolution and **concurrency management** (through **EnterCriticalSection** and **LeaveCriticalSection** Windows API's).

SPRITE SPIDER

Sprite Spider is an active criminal crew since at least **2017** and is usually characterized by its effectiveness and speed in compromising the targeted environment. It uses offensive tools such as **Vatet** (or **VatetLoader**), **PyXie** and **CobaltStrike.** The crew is often able to evade even the most advanced endpoint and network security solutions by hiding fragments of their malicious code in **open-source** projects and adopting an approach to the operation that sees, as principle, the writing of very few files on disk heavily complicating the detection activities. In fact, the only payload written to disk is **Vatet**.

Sprite Spider usually runs as a secondary (sometimes tertiary) payload directly in memory without ever touching the disk. The actor usually prepares a unique and specific ransomware payload for each of the victims, making the implant more elusive and difficult to track.

During its execution, it has the ability to terminate some **undesirable** threads and processes such as **powershell.exe**, **rundll32.exe**, **vmnat.exe**, **wefault.exe** and **explorer.exe**. It is able to encrypt files using **AES-256** without interrupting the main functions of the system and executes specific commands to interact with the files that have already been impacted by it. Often this means that files are already encrypted when security tools, such as EDRs, are able to report potential malicious activities.

VATET LOADER

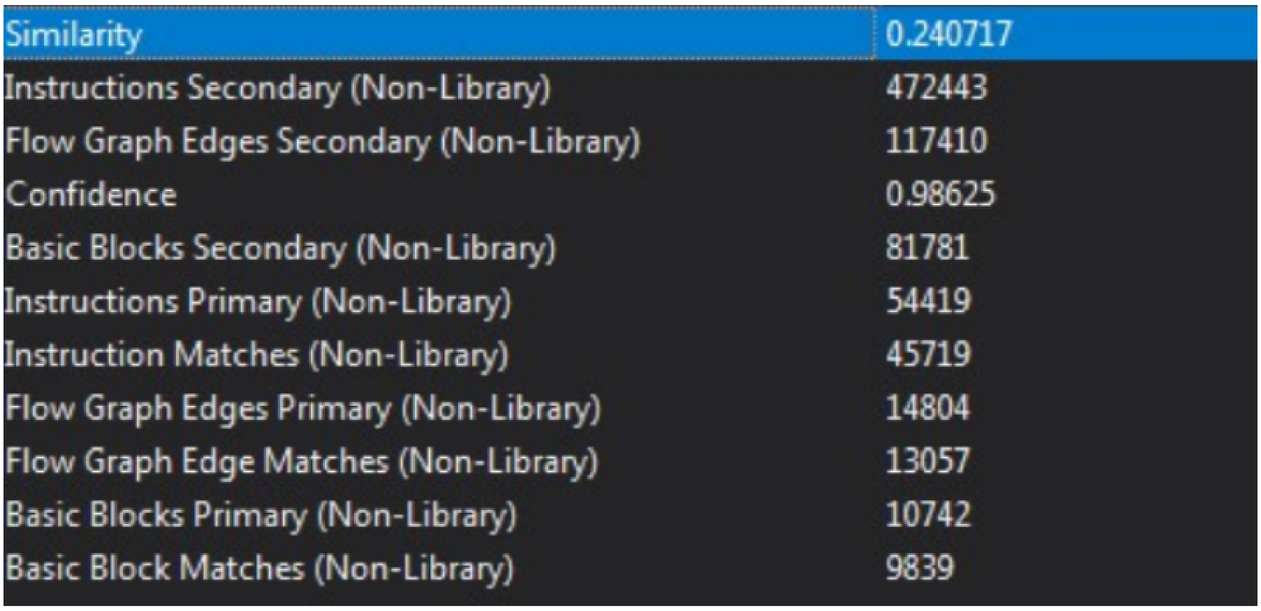
Vatet is a loader that executes custom shellcodes, protected by different encoding layers from the local disk or network shares. This loader is typically known to use opensource applications, found on **GitHub** or other repositories, that the threat actor modifies to load the final payload (e.g., **CobaltStrike**).

Based on this, we proceeded with our investigation on open-source applications that may share these portions of code. We have attributed these routines to a well-known project known **Rainmeter** (https://github.com/rainmeter/rainmeter/), already used in the past by **Vatet** to drop **CobaltStrike** payloads.

Rainmeter is a desktop customization tool that allows users to customize their desktop using skins. During a legit installation, Rainmeter creates an executable (**rainmeter.exe**) and a DLL (**rainmeter.dll**) responsible for reading configuration files and facilitating the desktop customization.

In the past, **Vatet** used a compromised and modified version of **rainmeter.dll** to lure the victim and install a **CobaltStrike** malicious payload. However, during a comparative analysis, we noticed a different behavior on the **IsaacWiper** and **Vatet** samples. In particular, to lower the detection rates, this time the malware developers have integrated in the sample's different utilities subroutines of the **rainmeter.dll**, instead of entirely using the legit DLL.

In fact, analyzing the legit **rainmenter.dll** we noticed a lot of similarities in identical subroutines regarding the dynamic API, resolving and the concurrency management. In support of our hypothesis, we performed another comparative analysis from the **IsaacWiper** sample and the **Rainmeter DLL**. Again, we can notice a 24% similarity with 10.000 basic block matches on non-library subroutines like reported following:



Similarity	0.240717
Instructions Secondary (Non-Library)	472443
Flow Graph Edges Secondary (Non-Library)	117410
Confidence	0.98625
Basic Blocks Secondary (Non-Library)	81781
Instructions Primary (Non-Library)	54419
Instruction Matches (Non-Library)	45719
Flow Graph Edges Primary (Non-Library)	14804
Flow Graph Edge Matches (Non-Library)	13057
Basic Blocks Primary (Non-Library)	10742
Basic Block Matches (Non-Library)	9839

CONCLUSIONS

According to collected evidences, the threat actor behind the destructive attacks against **Ukraine** used **open-source** code to build **IsaacWiper** variants. In particular, they extracted and **re-used** some utility functions of the library **rainmeter.dll**. Although it is not possible to directly associate the two pieces of malware with the same actor (as **rainmeter** is an open-source project and therefore it's freely accessible), it is a curious coincidence that it has already been used in the past, as reported by **Palo Alto UNIT42** researchers (https://unit42.paloaltonetworks.com/vatet-pyxie-defray777/), during **Sprite Spider** intrusions.
However this could at least suggest that the threat actor behind **IsaacWiper** knew it well enough to extract specific parts and reuse its code for very specific tasks.

Written by: Cluster25