Malware Analysis - PXRECVOWEIWOEI

By Mandar Naik

Published: 2024-09-20 · Archived: 2026-04-05 18:06:11 UTC

In this post, We will do malware analysis and reverse engineering on a sample called **PXRECVOWEIWOEI** (AKA PureLogs Stealer).

The source of the sample is

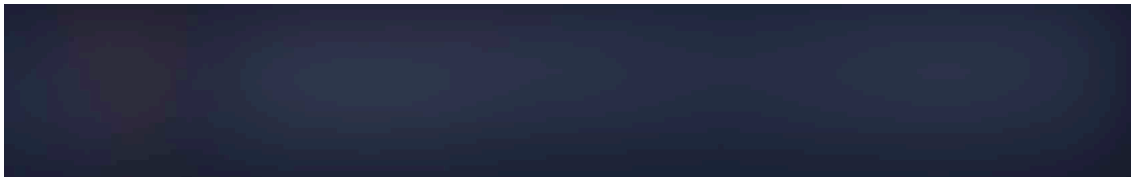
```
hxxps[://]bazaar[.]abuse[.]ch/sample/574403dce45be3a5edec18e66f16fef5e013ce99c7713479ab67c11e6f472330/#intel
```

Static Analysis

Let’s get the hash of the file first.

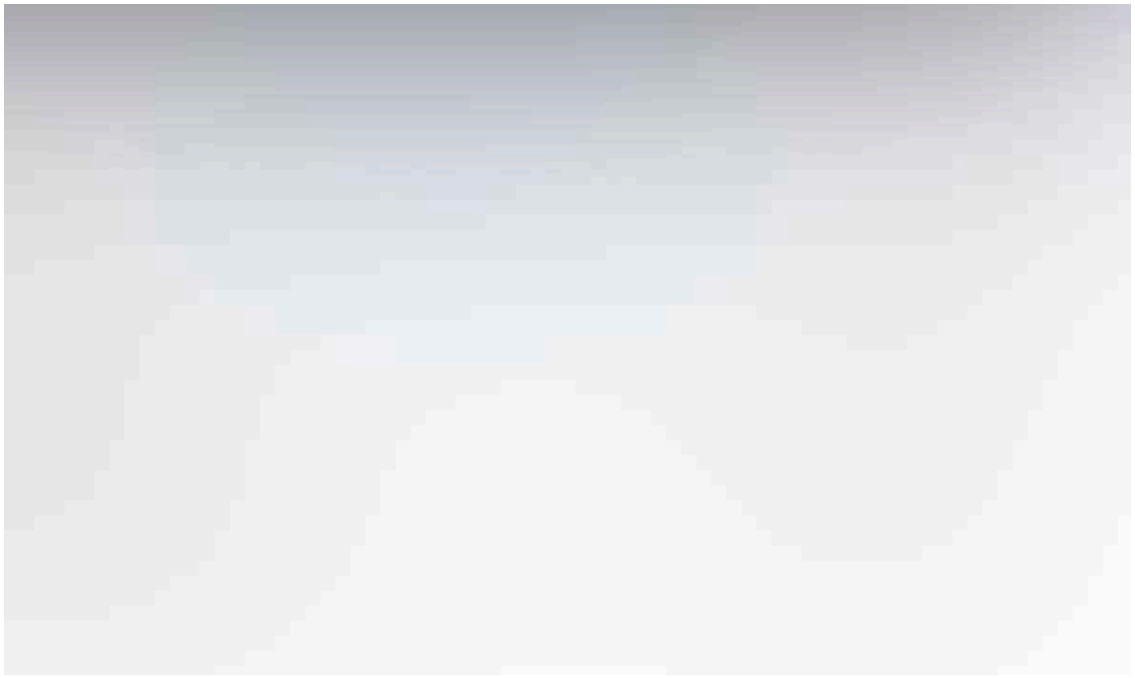
```
SHA256: 574403DCE45BE3A5EDEC18E66F16FEF5E013CE99C7713479AB67C11E6F472330
```

On VirusTotal, the file is detected as malicious by 25 engines.



The sample opened in notepad++ looks full of long strings assigned to variables with extremely long names.

The variables seem to be added purposely to distract us from the actual investigation point. The occurrence of variables is only one for each, meaning they are declared with strings but never used.

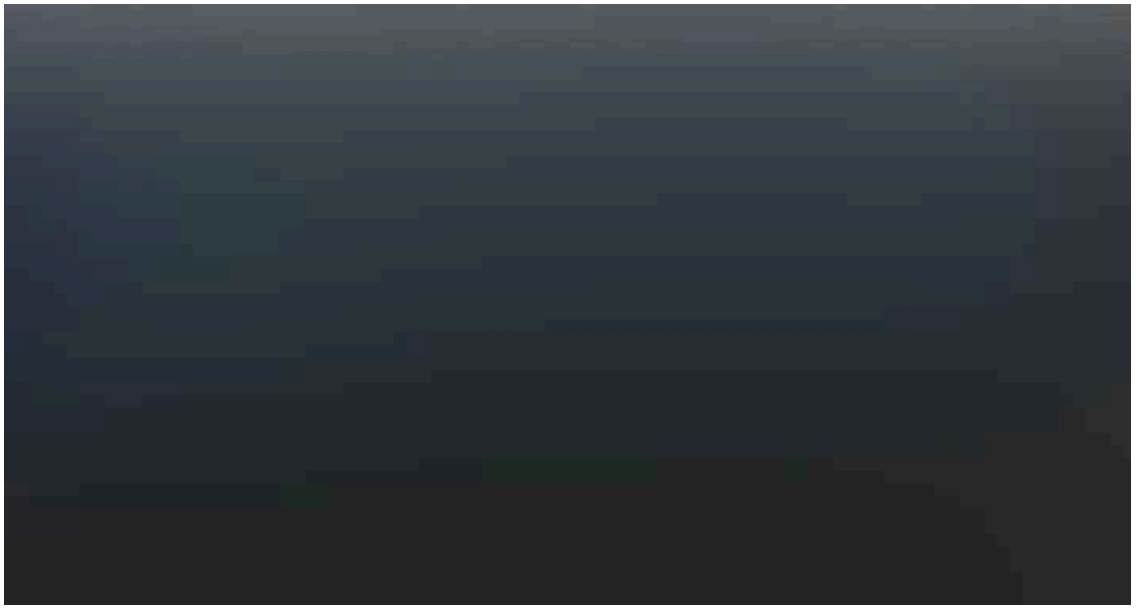


After ignoring them, we see some interesting variables that have been used repeatedly.

Let’s evaluate the variables.

The variables are interconnected. i.e. the variable *tonta* is used in variable *aligulado*, inturn that variable is used in *publicista*. We are going to focus on the content of *publicista*. We can see a base64 string, after decoding it.

The decoded base64 is a powershell script that is gibberish. Let’s evaluate the script to make it understandable.



After beautifying the script, we can connect the dots

The powershell script downloads the *DetahNoteJ.txt*, loads the content into a variable, then base64 decodes that content, eventually storing it in a variable called *$assembly*. The result of the above script can be accessed via variable *$OWjuxD* (I.e. base64 decode of *$Codigo*).

Finally, the output is executed via a powershell.

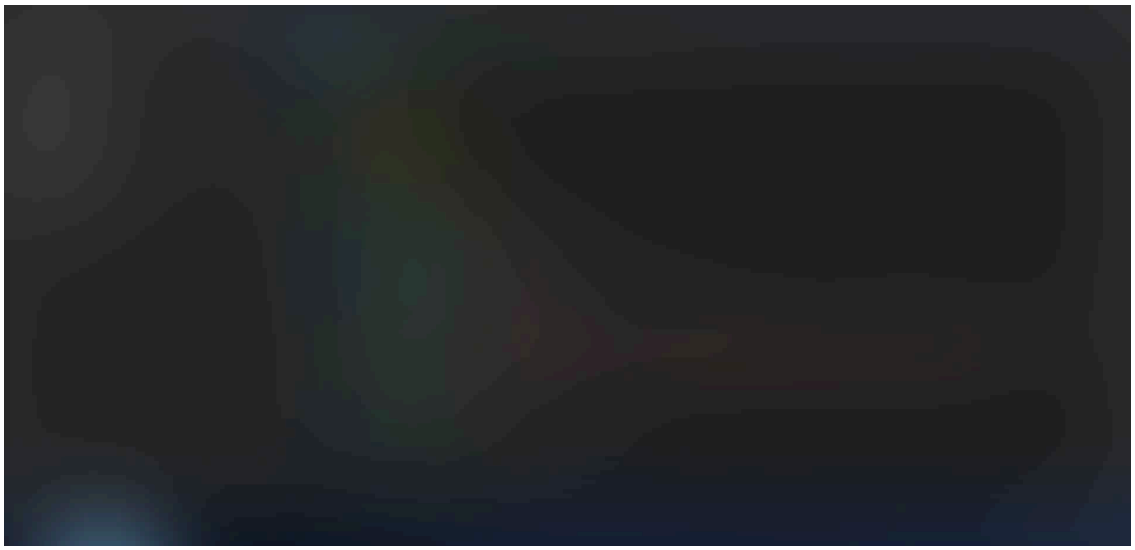
Let's check the content of a file called *DetahNoteJ.txt*. (PSSS: Shouldn't the filename be DeathNote and not DetahNoteJ?)

After base64 decoding and checking the file type

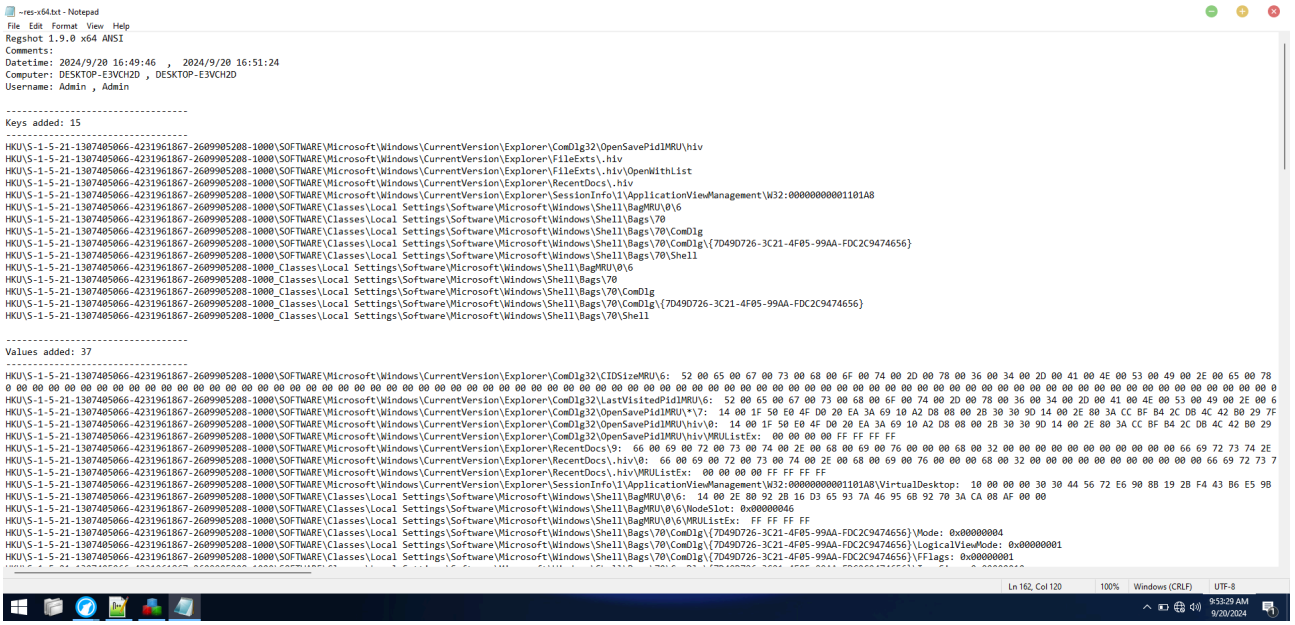
Now we can check the compiler or packer used to compile or protect the program; we can use *DIE* for this.

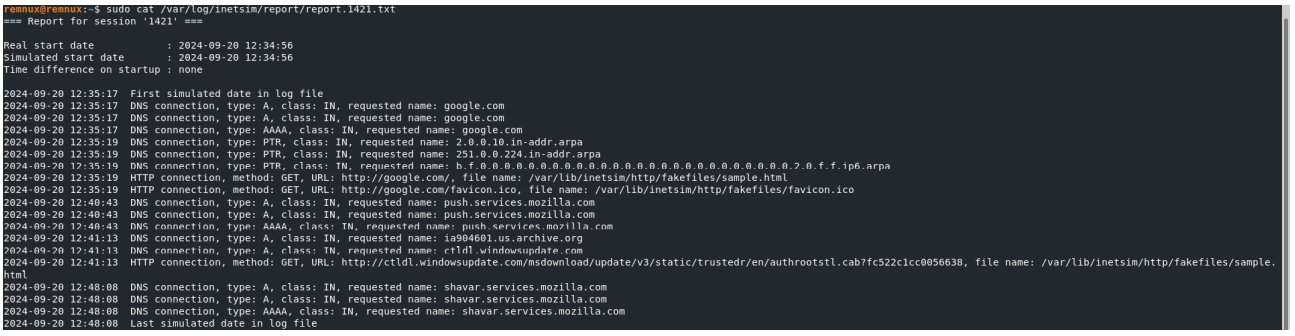
Dynamic Analysis

The sample is compiled in .NET without any packer or crypter used. we can directly decompile it using *dnSpy*



After a night with tea by my side, I was not able to understand the logic behind few variables, and was not able to decrypt them either. Let’s directly execute the sample.





My platform architecture seems incompatible, the program exited with some errors.

Atlast, we get no intel from dynamic analysis.

On VirusTotal, the file *DetahNoteJ.txt* is detected by 17 engines.

```
SHA256: 16912B71BDEFBB0B9E0B0E71D85B0095880D4DC250239E7D26E12454F7F6BADF
```

Whereas the base64 decoded content of *DetahNoteJ.txt* is a DLL file, detected by 7 engines.

```
SHA256: 97164081607B6FDB9B095CB01BB0A818FC77DB92DAD38B910B05A90160748756
```

We meet next time dissecting another sample or comming up with an evasion technique until then **čau čau**
