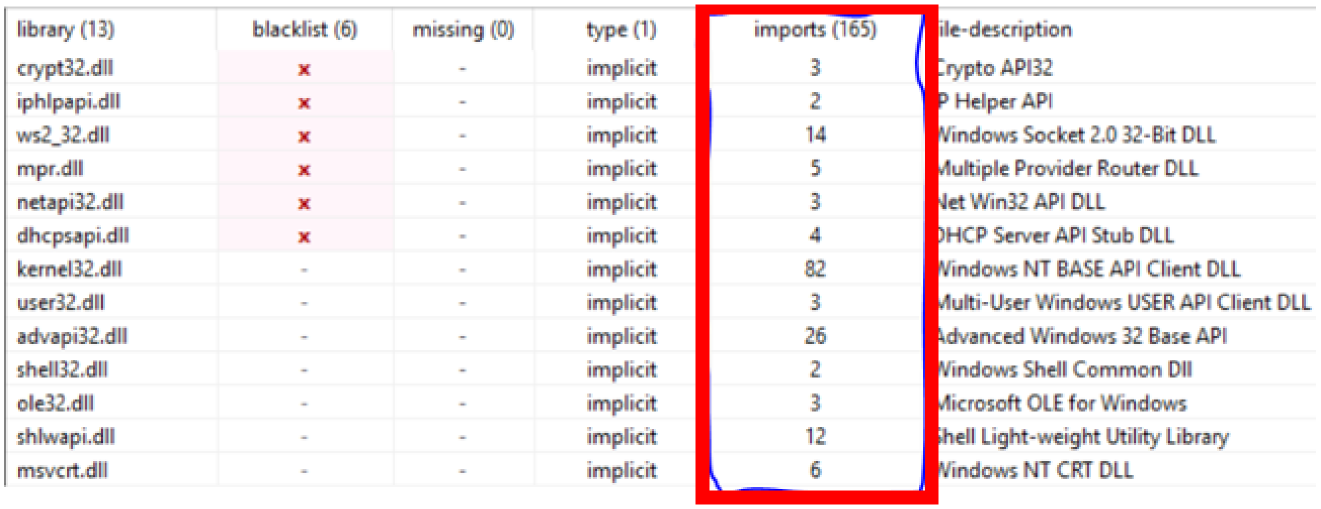
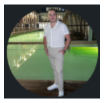
Petya/Not Petya Ransomware:

medium.com/@llandu/petya-not-petya-ransomware-9619cbbb0786

Ilan Duhin January 30, 2023



library (13)	blacklist (6)	missing (0)	type (1)	imports (165)	ile-description
crypt32.dll	x	-	implicit	3	Crypto API32
iphlpapi.dll	x	-	implicit	2	P Helper API
ws2_32.dll	x	-	implicit	14	Windows Socket 2.0 32-Bit DLL
mpr.dll	x	-	implicit	5	Multiple Provider Router DLL
netapi32.dll	x	-	implicit	3	Net Win32 API DLL
dhcpsapi.dll	x	-	implicit	4	DHCP Server API Stub DLL
kernel32.dll	-	-	implicit	82	Windows NT BASE API Client DLL
user32.dll	-	-	implicit	3	Multi-User Windows USER API Client DLL
advapi32.dll	-	-	implicit	26	Advanced Windows 32 Base API
shell32.dll	-	-	implicit	2	Windows Shell Common Dll
ole32.dll	-	-	implicit	3	Microsoft OLE for Windows
shlwapi.dll	-	-	implicit	12	Shell Light-weight Utility Library
msvcrt.dll	-	-	implicit	6	Windows NT CRT DLL



Ilan Duhin

Jan 30

.

7 min read

Writer: Ilan Duhin

Executive Summary:

Petya is a family of encrypting malware that targets OS of windows to infect MBR (master boot record) to execute payload that encrypt a hard drive's file system table.

Petya spread over the network by using the exploit MS17–010 Vulnerability known as **EternalBlue.** It also spreads by stealing **user names & passwords** and spreading across network shares.

> **Static Analysis:**

when I opened the ransomware in **IDA**, it started at the 10007D39 address with the function DLLEntryPoint, so although the file extension is .exe, I guess it is actually DLL.

PeStudio: in the indicators tab we see two of them that can suspicious to me.

The date of that specified is , probably our suspicious payload
In the **Imports** tab, I will be looking for interesting api calls that I want to investigate later in IDA/Debugger to see which values they contain.

it means that the new process that is created, in the security context of the calling process. If the calling process is impersonating another user, the new process uses the token for calling process.
Both of them is used to grab the TokenSessionId for terminal service sessions.

> It works like that: a new thread is created in suspended mode. The impersonated token replace the current thread token with SetThreadToken and the thread is resumed. This thread is then used to execute the SMB RCE as the impersonated user.

is used to connect to a server by using default credentials for the impersonated token and then cancel the connection.
The function that encrypts data. In our situation, I guess it would try to encrypt the MFT because it is ransomware.
In the **Library** tab, I checked which library the ransomware import and which function it exports. In picture below I see **many import DLL** which give me a clue that the binary **isn't packed**.

The interesting libraries I focus on them is:

· **Crypt32.dll** — will use possibly crypto functions.

· **Advapi32.dll** — probably will be responsible for restarting the OS system (I guess because the ransomware wants to reboot the machine after she encrypts all files).

· **Shlwapi.dll** — function that works for strings & filesystems paths.

· **Ws2_32.dll** — it contains windows sockets api, I guess for setting up some sockets.

Another way to find interesting things about ransomware is by reading her string. To do this I use **BinText.**

The strings are readable strings & **useful output which means it's not packed!**

the file extension that the ransom will be looking for.
the messages that will show on the victim screen

> **Dynamic Analysis:**

As we see at static analysis that needs to face with **DLL**, I try to run it with Rundll32 at the **Entry point** and see what happens.

How do we know what the Entry point argument is? Well, one of the suspicious strings that I found was: **the string is described that the ransomware is run by rundll32 as a child process with the #1 argument (which is the first value in library).**

Also when dropping the dll into IDA, at the **Export tab** we can see there is one export function that we should run.

running the ransomware with the first function of export tab
We see in **Process Hacker** that the dll running by cmd & rundll32 that we use earlier.

Double-clicking on rundll32.exe, memory & strings tab we see interesting strings that running in the memory:

When we enter into **task scheduler** itself we see the task really created and ready to run.

Mapping . It means that the ransom wants to gain access to the physical disk and encrypts the so the file system will not be readable.
The string below describe that
In simple words, our sample leaves no trace.

When the victim's computer will reboot the Petya will We will see it later after restarting our VM.
The messages that show up on victim's screen.
In addition, I have opened the **Procmon** earlier with a number of filters to capture interesting processes like:

- Operation is
- Process name is
- Process Name contains

And this is what we got!

We saw that the ransomware created a **scheduled task to shut down the infected machine at a specific time, and created tmp file "181E.tmp" in the AppData\Local\Temp path.**

It seems that the malware try to connect to
IDA:

- One of the thing I am doing when I dropped the file into IDA is to look on top of the **scale**. It indicates an interview of the situation of the code. For example, the **olive** color is an **unexplored code**, and the **pink** one is **external symbols** that can indicate to us if our sample using external DLL libraries. (Basically it's a table that shows us memory location of every symbol — API calls of the malware code)

- The second thing I always look in the **imports tab.** Very important because we see which API calls the malware use.

For example, as we see in static analysis of the API call of **WNetAddConnection2w,** we can see her **references** of her by pressing "**x"** and maybe also our malware connection by disabling the ASLR & set BP.

Extension files that are targeted by the ransomware (you can find it in the)
malware "overwrites" the **MBR**. it pushes a file his name "Physicaldrive0" with a number of arguments it including himself and **then it goes to the DeviceControl function (I guess to specify the device driver).**

When we double clicking on the file, we see the interesting arguments it pushes.

Another interesting thing is one of the arguments that DeviceControl includes and this is :**hDrive.** its argument that gives a handle on the driver and retrieves information about the physical disk, file, thread, etc.

Network Enumeration:

After the malware get **ComputerNameExW** and before she created **CreateThread** we can see condition (jz) with a very interesting argument that calls **IpStratAddress** which is the beginning of rebuilding the SMB protocol.

If we go down little bit (after the thread creation) we see **two calls to functions.**

In the first one, we see a pushing argument like **GetExtendedTcpTable** which means "retrieve a list of TCP endpoints". **In other words get tcp connection of the local machine.**

In the second, we see **GetIpNetTable** which means "give me your local network ip".

The enumeration of SMB staring with the API call **GetAdaptersInfo**. The goal of this call is to get subnetmask of all network interfaces like workstations/servers. At the end of the call, we have conditions which means jump if not zero to API call which retrieves inetaddr and closes the socket **or** jumps to API call LocalFree which checks the free space on disk and give clue if it is a **server or either workstation.**

It checks it with **NetServerGetInfo** api call.

When we dive into the call of "Check_If_Server_Or_Not" we see **three arguments that's pushed into the call of NetServerGetInfo** (bufptr, level, servername).

The **first** parameter contains 65h object which is equal to 101 in decimal. (just search in google "how much is 0x65 in hexadecimal").

The **second** one is empty which means equal to 0.

From MSDN: **bufptr** is points to a server 101 info structure.

** So **101** for me is the action of the malware return servername, type, and infrastructure. **I recommend describing this by inserting a comment (“;”).**

Source: https://docs.microsoft.com/en-us/windows/win32/api/lmserver/ns-lmserver-server_info_101

Conclusion & Activities:

· Dropped files

· Token impersonation

· Network node enumeration

· SMB copy and remote execution

· SMBv1 exploitation via EternalBlue

· Recon and write malware to admin$ on the remote target

· MBR ransomware

· Physical drive manipulation

· MFT encryption

· System shutdown