Reversing and Technical Analysis of qakbot trojan:

Introduction:

Qakbot is a banking trojan initially developed with credential stealing capabilities, it was one of the top trending malware back in 2021. It steals sensitive data from the victim. It has been used by ransomware gang such as Revil, Lockbit and Prolock. It also look for financial data and sensitive information such as credentials, browser cookies etc.

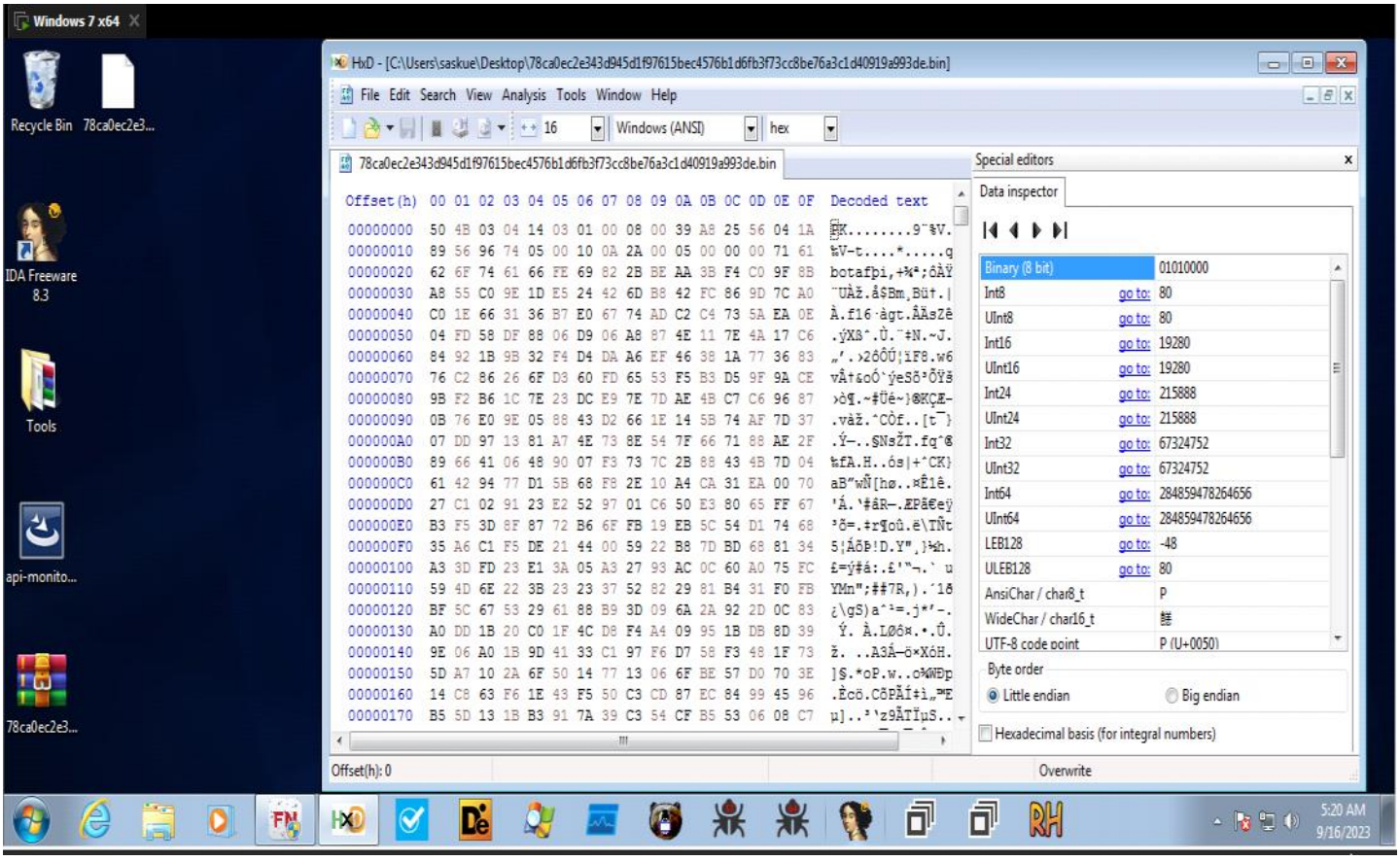
Methodology:

Initial static analysis
Initial dynamic analysis
Unpacking of the malware
Decryption of strings
Reversing and analysis
Detection Content
Yara signature
Sigma rule
Mapping to MITRE Framework

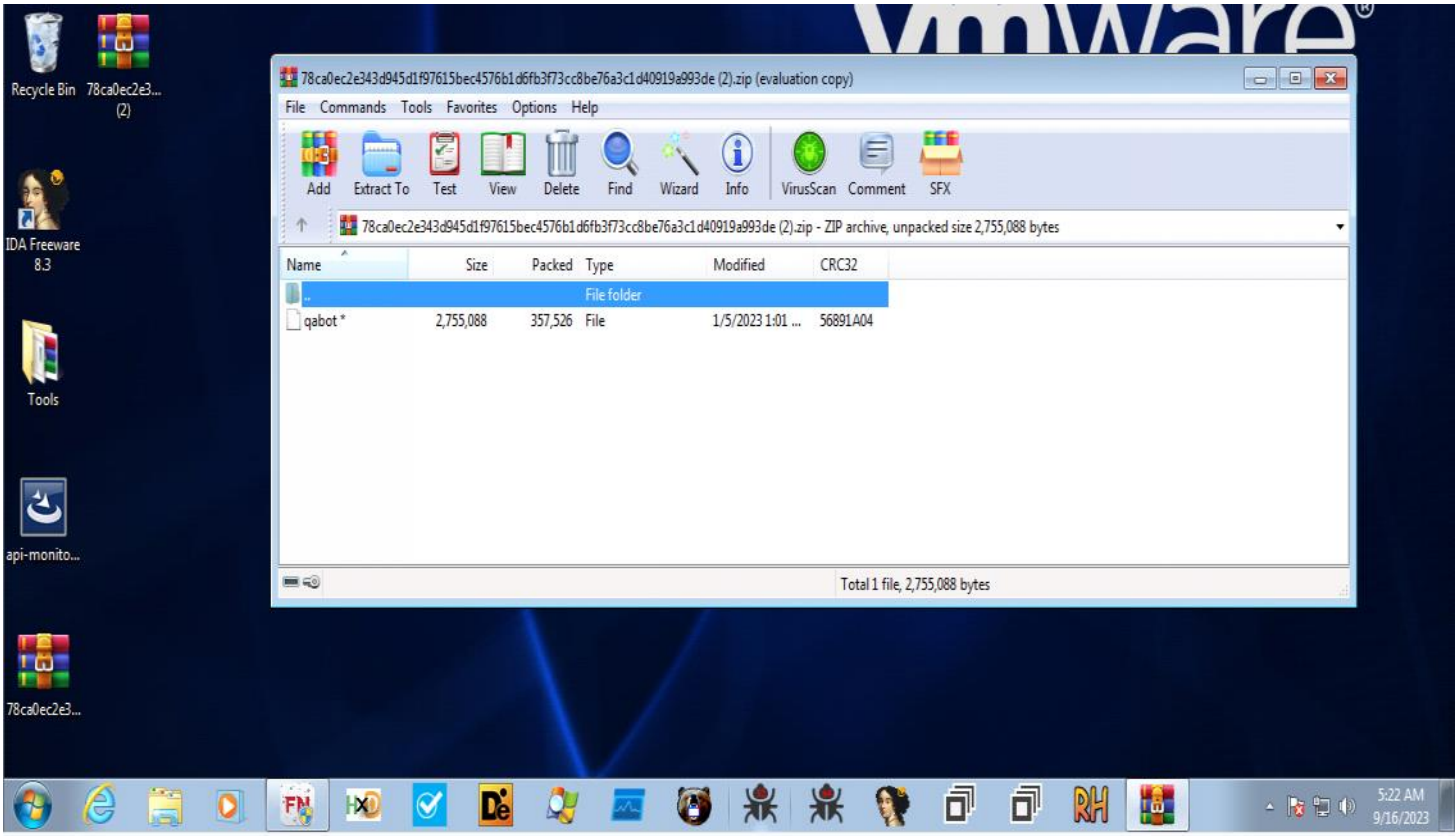
Initial static analysis:

For the initial static analysis we are going to use Hexeditor, Pestudio and DIE and will try to get as much information as possible.
First thing I always do is check out the what kind of file I am dealing with in hex editor, is it a PE or DLL etc. Apart from that if I find suspicious about resource section I will then use resource hacker to study resource section bit in detail so let's start with our
Initial static analysis.

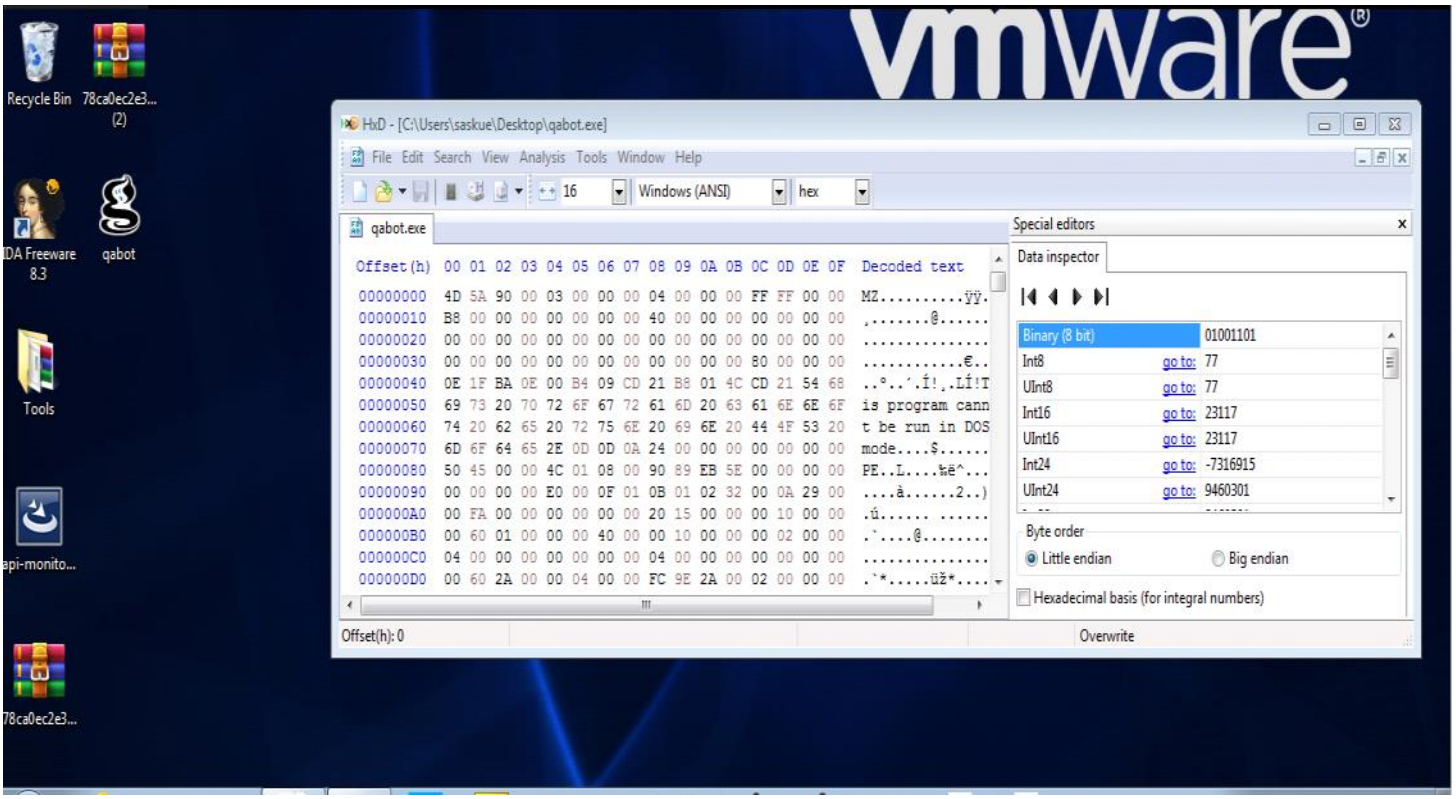
After extracting and opening the bin file in hex editor I see PK Magic bytes which means it’s a zip format therefore will have to make it in zip format and then we could have the binary file.



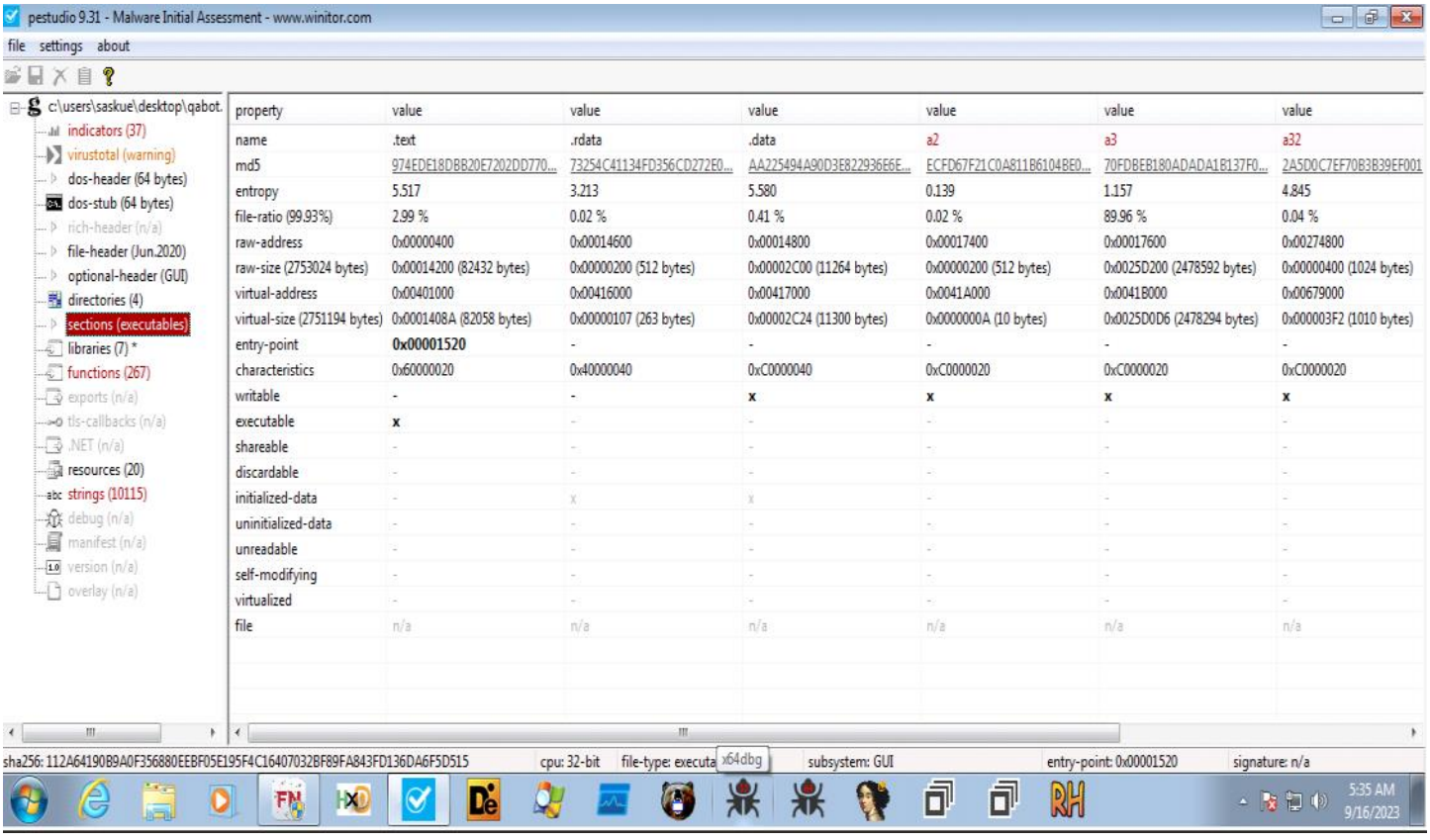
After changing the file extension we could see qakbot file so this might be the binary file on which we are going to do analysis on.



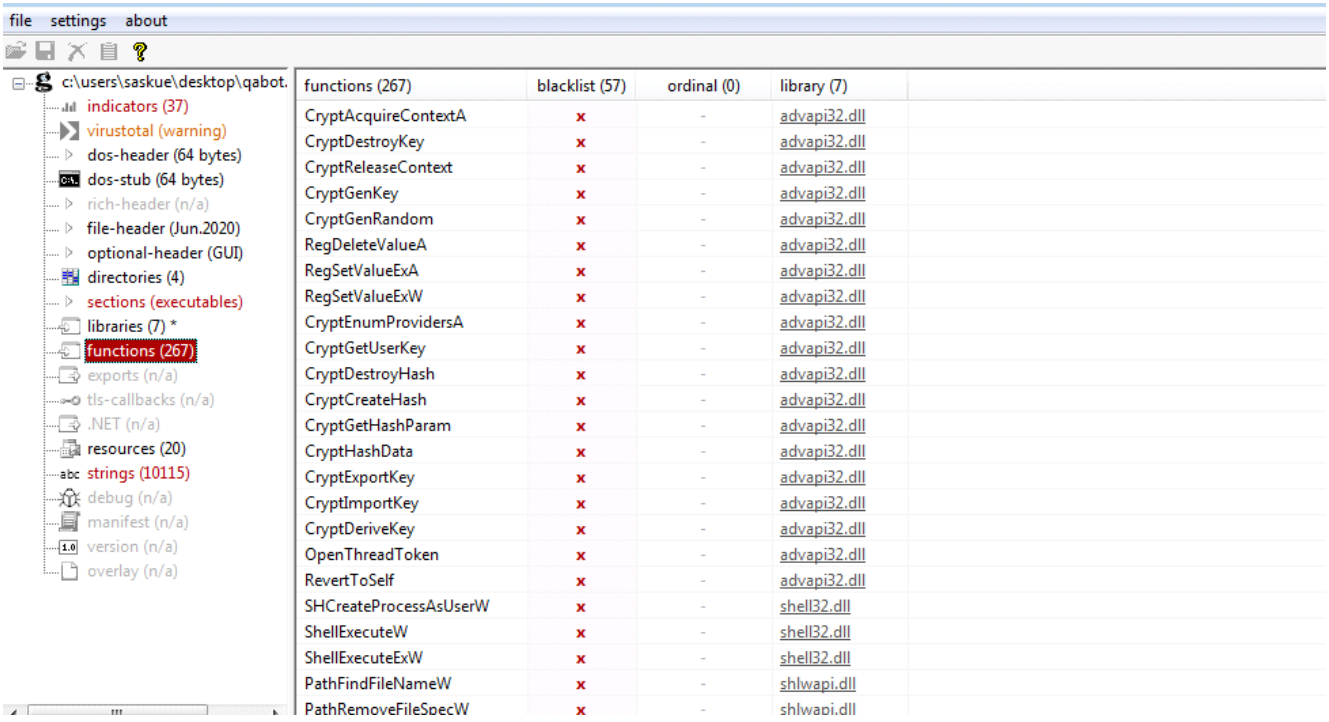
It seems this was the actual executable so let's proceed further.



Based on analysis done using pestudio I see information such as : Compiler timestamp : jun 18 2020.
As soon as I look into sections I see strange section name such as a2, a3, a32, a322. Most of these section are writeable with only text section being executable with entropy of 5.5 something to take note of.

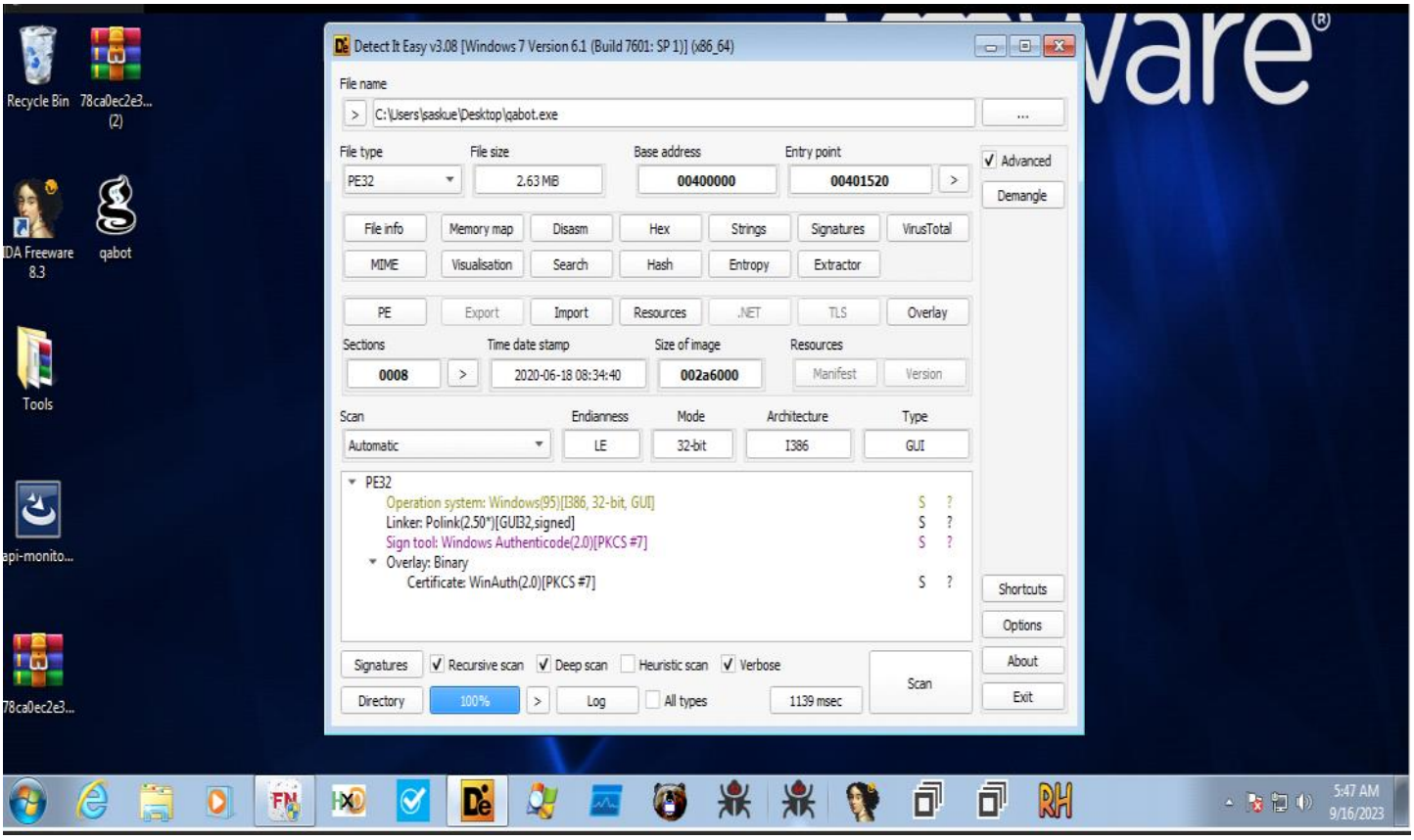


When I look over functions I could see crypto function being used heavily seems like we might encounter encryption later in the stage.



When I looked over the resource section I could see a huge resources being embedded into exe, it seems that there is something fishy about resource section we might have to keep monitor api such as:
1- Findresource
2- Loadresource
3- Sizeofresource

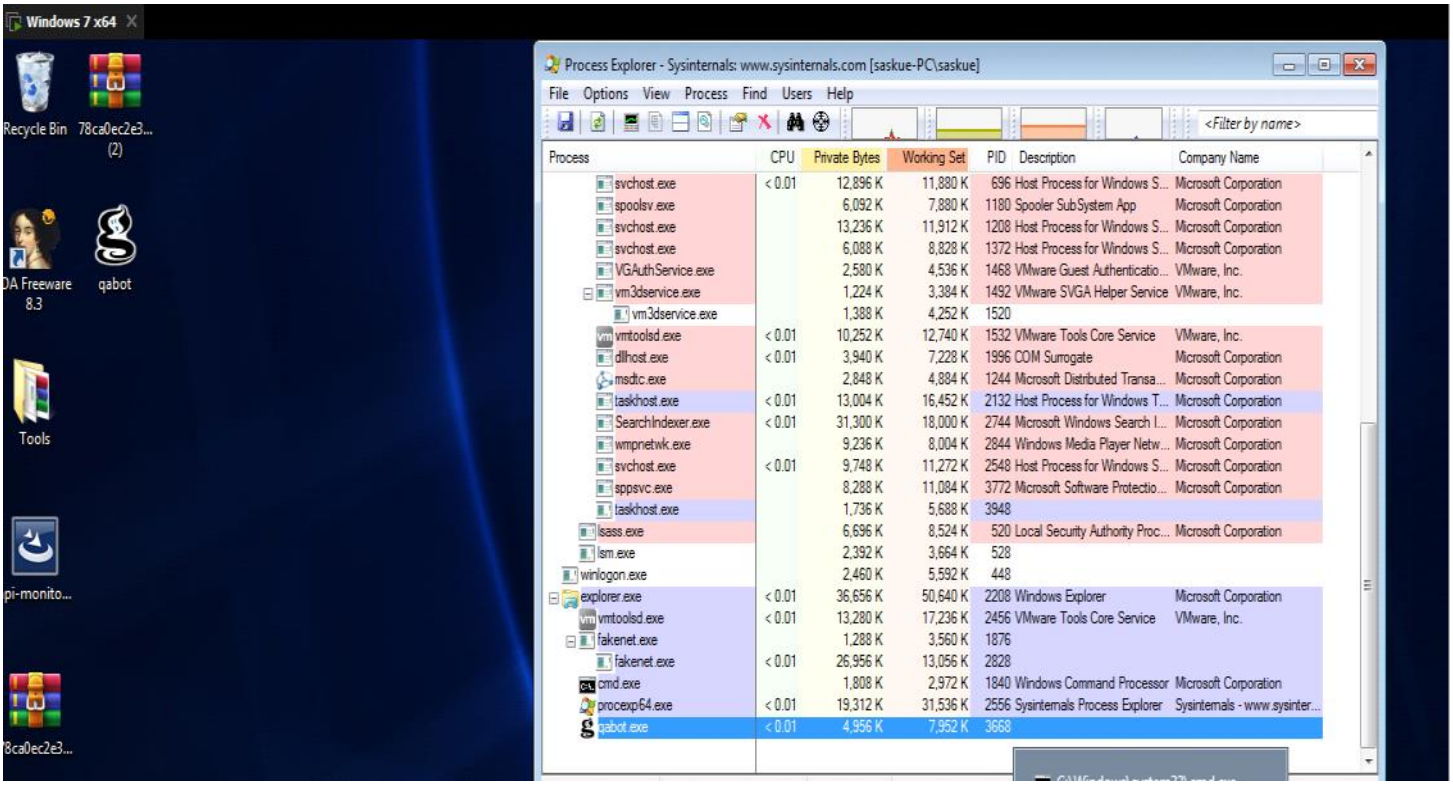
It also seems that sign tool is used to sign the executable to make it look genuine.



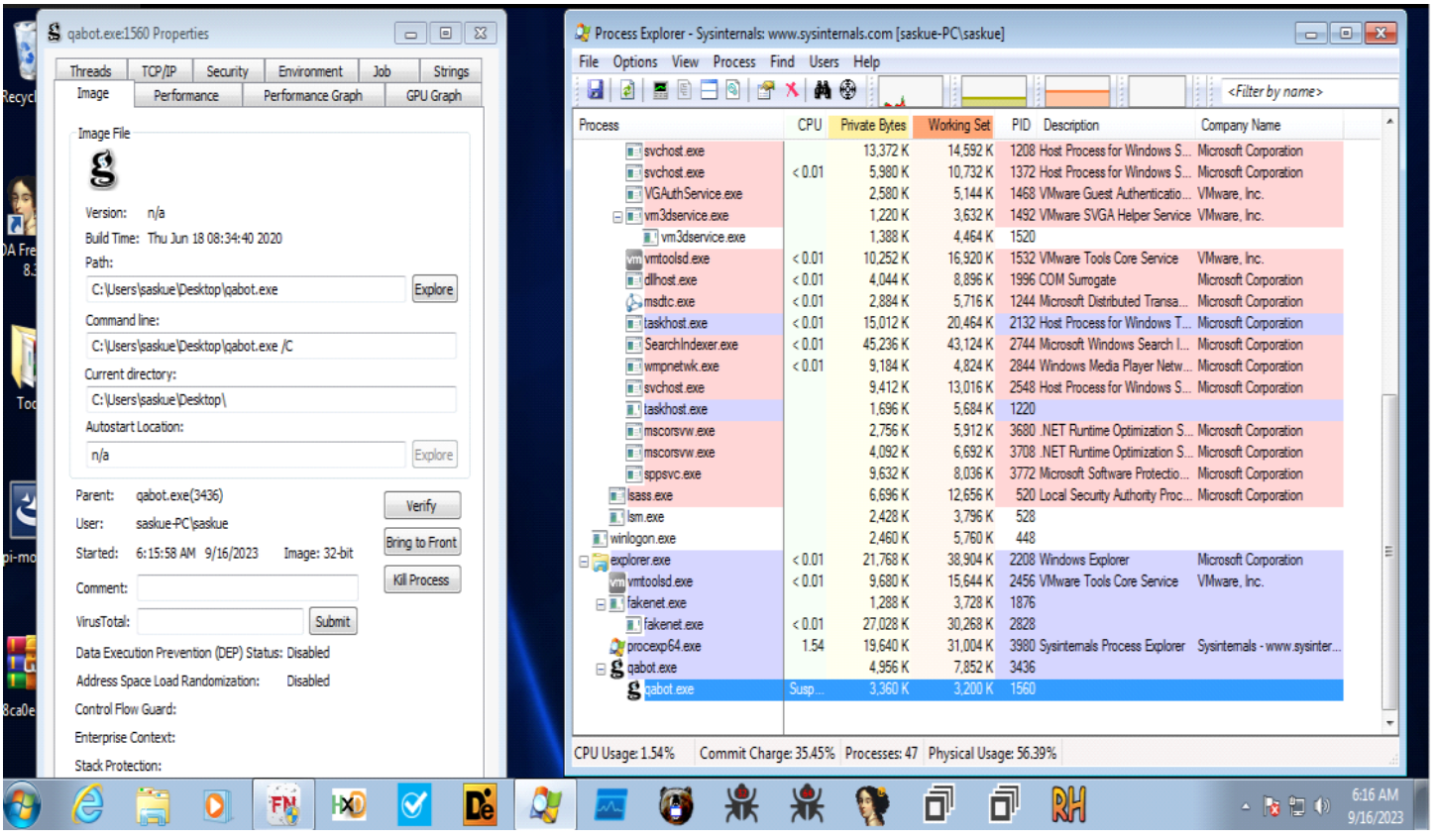
With that I will end my initial static analysis stage two things to keep in mind is :
1- Text section being executable (keep an eye over overlay too)
2- Resource section might be used
Since I know that it is packed there is no more need to do deep analysis further.

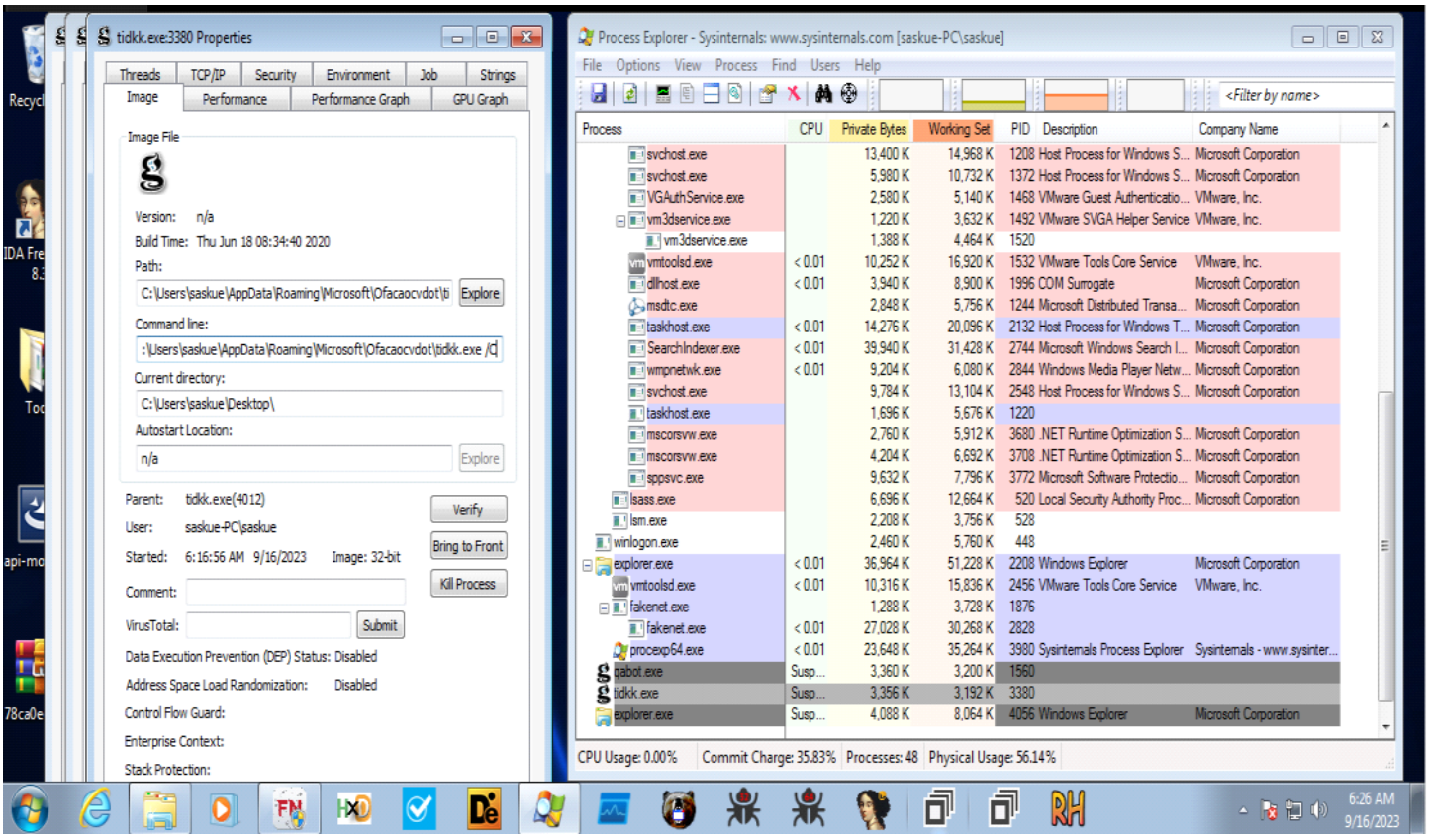
Initial Dynamic Analysis:

For the initial dynamic analysis we will use processexplorer, procmon and api_tracer to study the behavior of malware and how its packing the main payload into process memory.

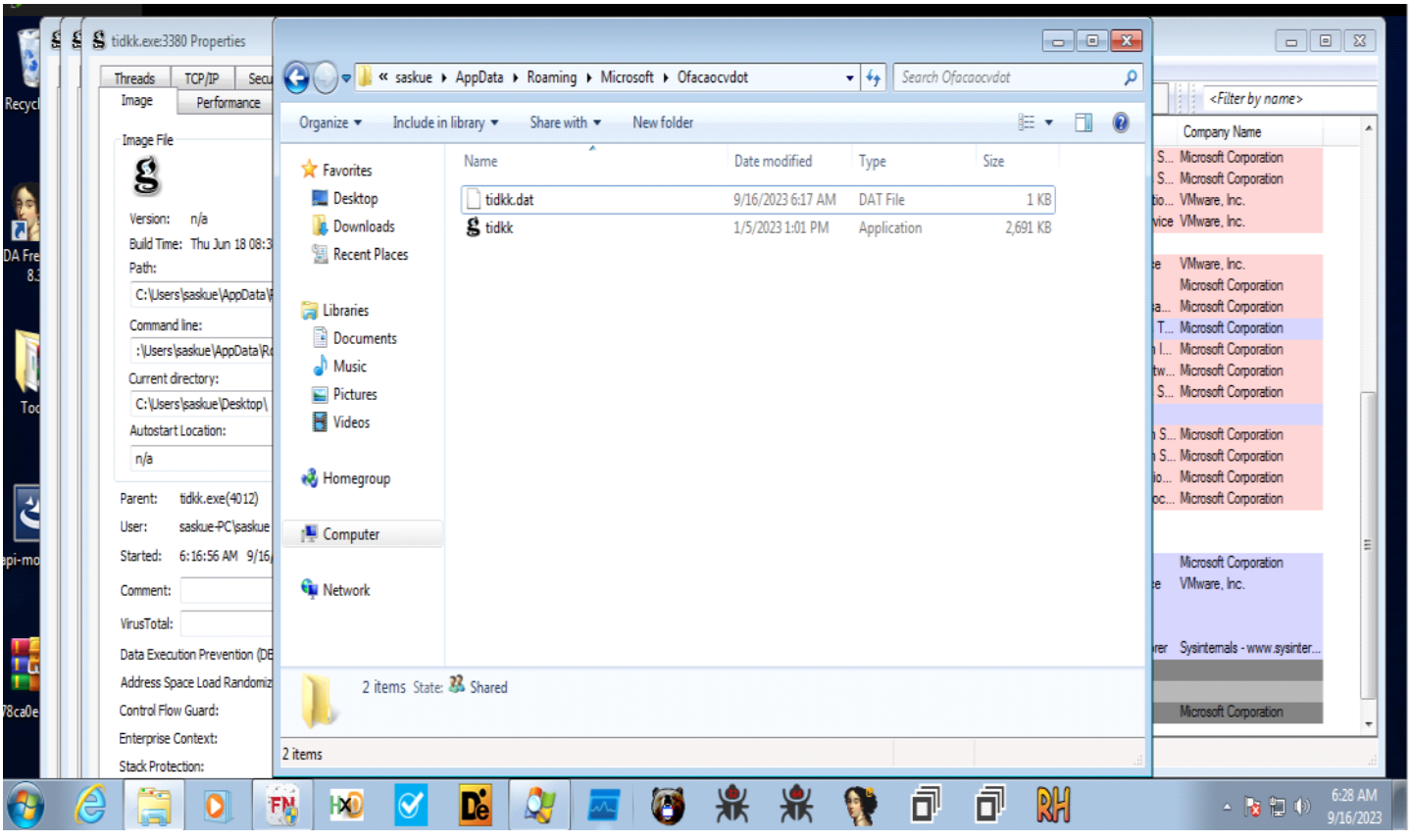


When clicked on qbot it launch another copies of itself with /C parameter, after that the child copy exit and then it dropped a executable inside the directory C:\Users\saskue\AppData\Roaming\Microsoft\Ofacaocvdot, here Ofacaocvdot is random directory name. Inside it there was tidkk file which was launched as new process again follow same pattern of launching copy of itself with /C parameter.





Location of second payload from where it was loaded and launched as new process .



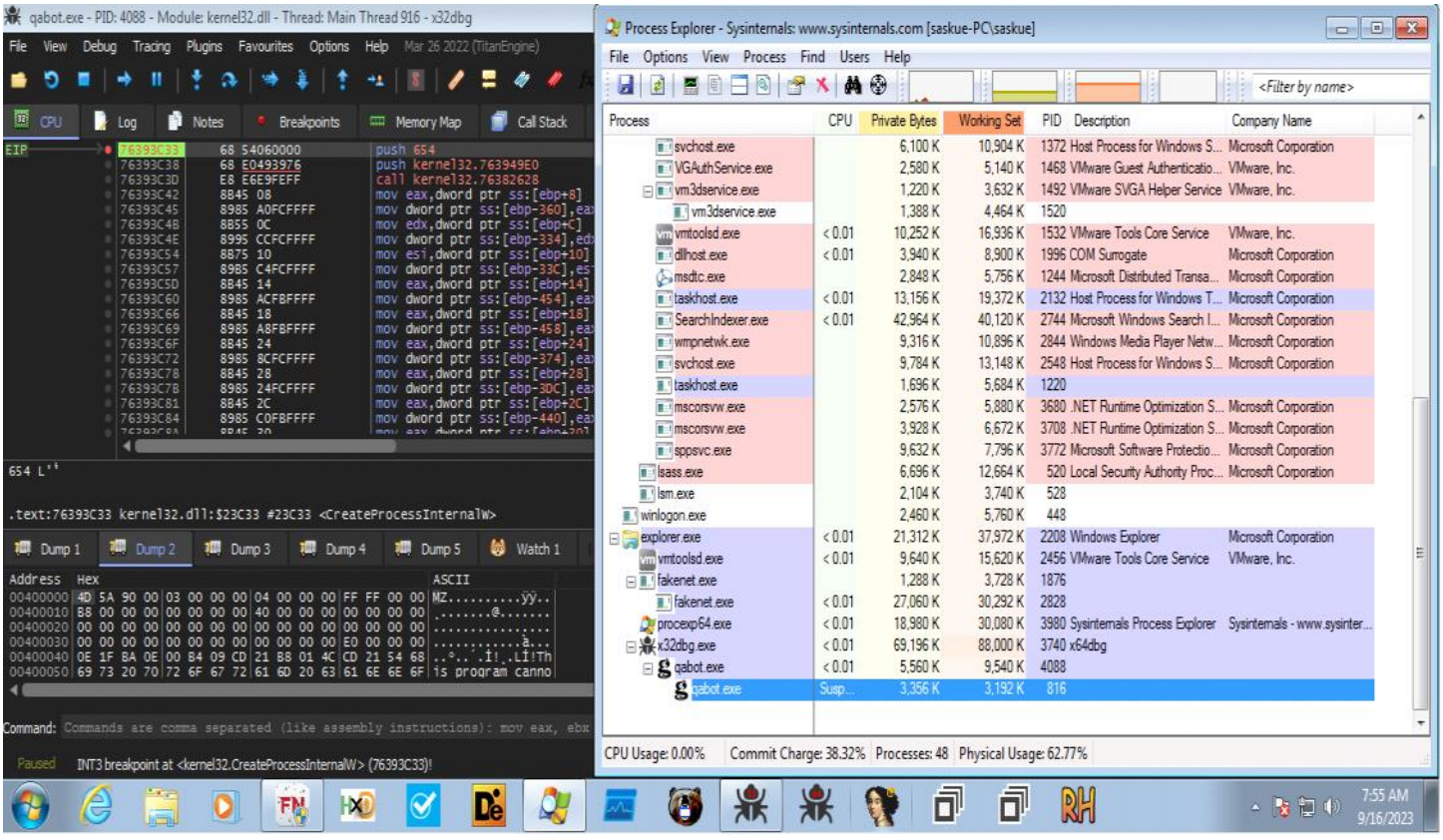
Later on I could see explorer.exe being launched could be possible that injection might happened inside the

explorer.exe.
It does seems that explorer.exe will be the process inside which the final payload might be injected.

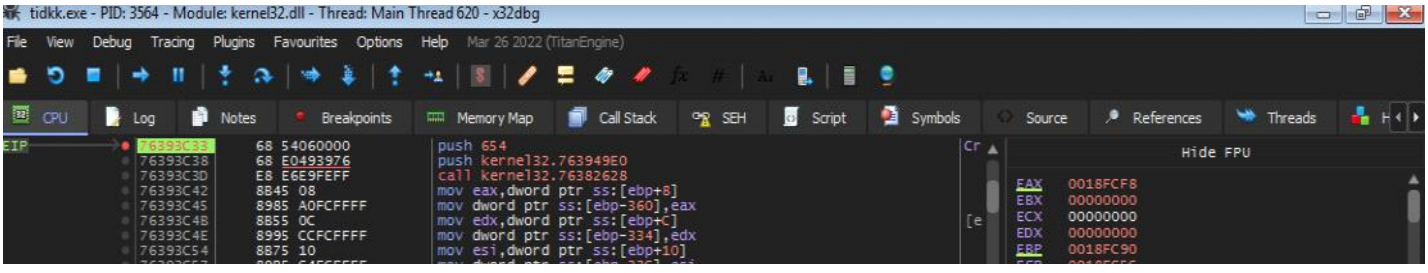
Unpacking of the malware:

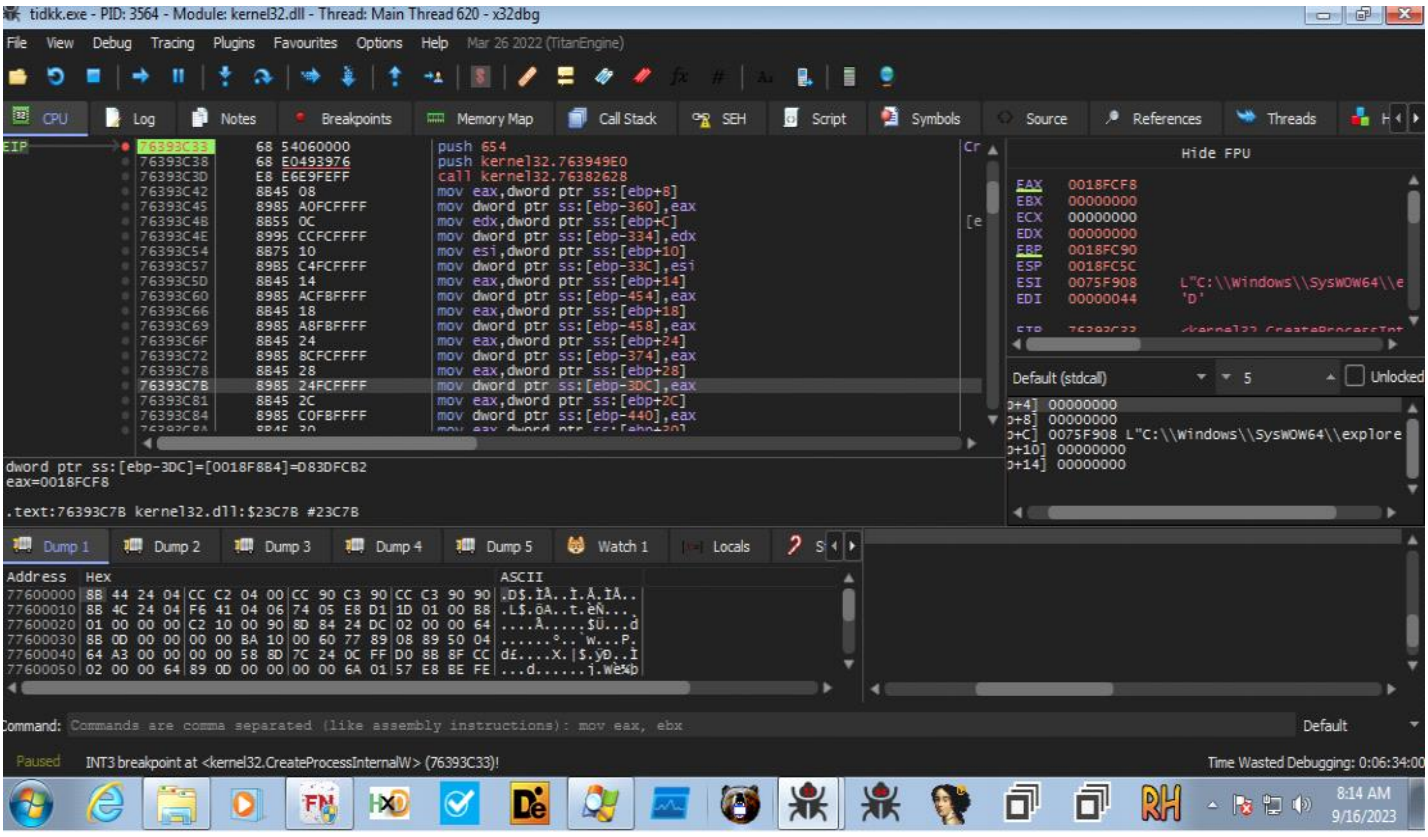
For unpacking of the malware we are gonna use x32dbg and will set the breakpoints on interesting api along with using process explorer to suspend the subsequent child process so that we could finally reach to final process i.e explorer.exe.

Usually an interesting api to look out for :
VirtualAlloc,
VirtualAllocEx,
VirtualProtect,
CreateprocesA/W,
resumethread,
Sethreadcontext,
Createremotethread,
CreateprocessinternalW

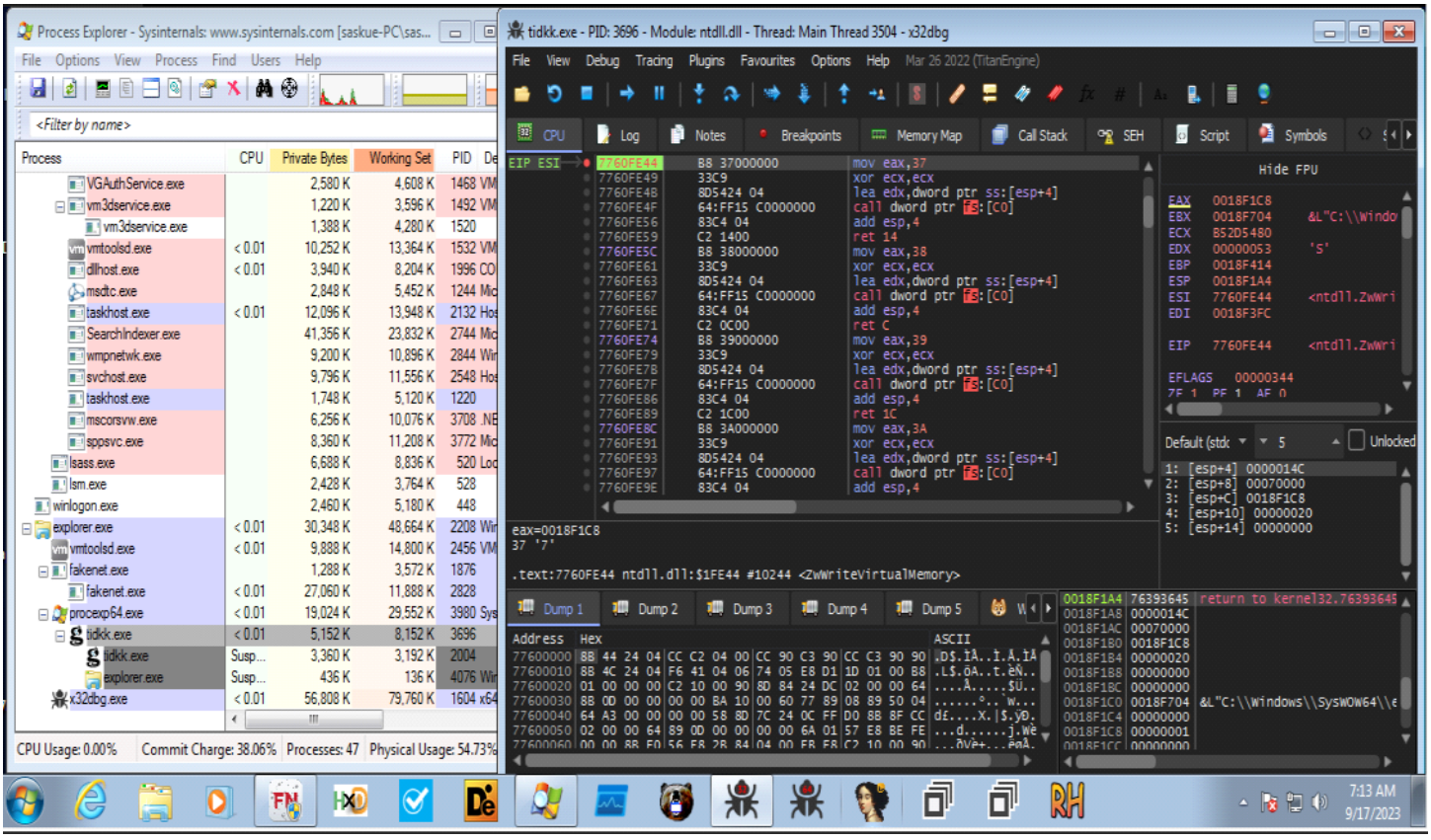


We are going to enter into second stage of unpacking after this TIDKK.EXE From previously mention weird directory will be launch as parent process, after which it will launch copy of itself as child process.
Let's proceed further to unpack the final payload.





Start the tidkk.exe with x32dbg and put breakpoints on createprocessinternalW and we can see that explorer.exe being launched. Injection is happening inside the explorer.exe.

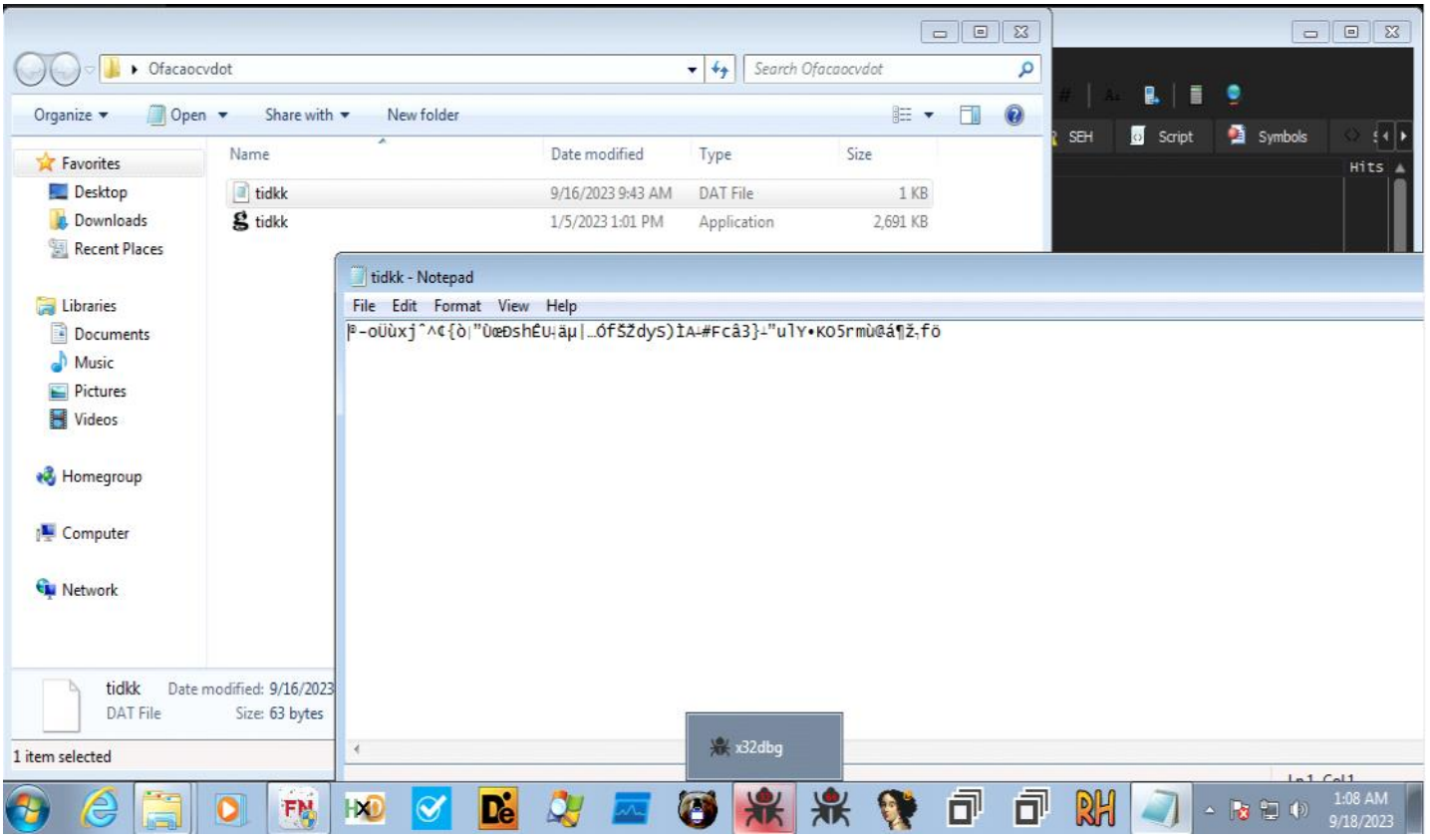


The injected code load and decrypt one of resource which happens to be an executable (dll).

Since this sample is old therefore the c2 communication will not happen therefore I have to stop it here only.

Decryption of strings:

For decryption of strings we are going to rely on debugger and decrypt it dynamically if we encounter any interesting payload.



This file is C2 Config file containing:
1- C2 IP: 187.163.101.137
2- C2 Port: 995
3- Victim IP: XX.XX.XX.XX
4- Others (campaign id, c2 hardcoded)

Reversing and technical analysis:

WMI Queries made:

Win32_OperatingSystem
Win32_ComputerSystem
Win32_Bios
Win32_DiskDrive
Win32_PhysicalMemory
Win32_Product
SELECT * FROM Win32_OperatingSystem
SELECT * FROM AntiVirusProduct
SELECT * FROM Win32_Processor
select Caption,Description,Vendor,Version,InstallDate,InstallSource,PackageName from Win32_Product
select Caption,Description,DeviceID,Manufacturer,Name,PNPDeviceID,Service,Status from Win32_PnPEntity

- Malware check for internet connection as well as wanted to know the public facing ip address of victim via query made to http[:]//www.ip-adress.com.
- Sample tries to get the config from after checking internet connection and anti analysis check

- It query for system information such as computer_name and username.
- It also check for browser activity and perform info harvesting (if found).
- It also check for uninstalled applications (HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall)

Detection Content:

Indicators of compromise:

MD5 Hash ==> BCE0DF8721504D50F4497C0A0A2C090D
MD5 Hash ==> 974EDE18DBB20E7202DD7700B2E2AD0B
MD5 Hash ==> 73254C41134FD356CD272E048086AE0A
MD5 Hash ==> AA225494A90D3E822936E6EF86814962
MD5 Hash ==> ECFD67F21C0A811B6104BE0E01BF28C3
MD5 Hash ==> 70FDBEB180ADADA1B137F01387735893
MD5 Hash ==> 2A5D0C7EF70B3B39EF00107D7D6E6591
MD5 Hash ==> 3D2763A495754638C850DFB862438399
MD5 Hash ==> 2407DE61EB885FFE21CDF2F0F70F637D
IP ==> 187.163.101.137
IP ==> 193.248.44.2
URL ==> https[:]//24.139.132.70/t3
URL ==> https[:]//68.174.15.223/t3
URL ==> https[:]//96.232.203.15/t3
Url ==> https[:]//108.188.116.179/t3
Url ==> https[:]//89.137.211.239/
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

https[:]//96.56.237.174/
https[:]//207.255.161.8/
https[:]//69.92.54.95/
https[:]//67.209.195.198/
https[:]//207.255.161.8/
https[:]//89.137.211.239/
https[:]//96.56.237.174/
https[:]//96.232.203.15/
https[:]//96.232.203.15/
https[:]//68.174.15.223/
https[:]//84.247.55.190/
https[:]//5.107.220.84/
https[:]//5.107.220.84/
https[:]//69.92.54.95/
https[:]//68.174.15.223/
https[:]//84.247.55.190/

Yara Rules:

https://github.com/Deepanjalkumar/Malware-Signature/blob/main/Qakbot/qbot.yar

qbot.yar

```
rule qakbot_rules
{
    meta:
        author="Deepanjal Kumar"
        description="Qakbot is a banking trojan initially developed with credential stealing capabilities, it was one
        org="operation falcon"
        date="9/18/2023"
        md5="BCE0DF8721504D50F4497C0A0A2C090D"

    strings:
        $registry = "HKCU\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run"
        $c2ip="187.163.101.137"
        $reshash="2407DE61EB885FFE21CDF2F0F70F637D"

    condition:
        $registry and $c2ip and $reshash
}
```

Sigma Rules:

https://github.com/Deepanjalkumar/Malware-Signature/blob/main/Qakbot/qbot.yml

qbot.yml

```
title: Qakbot banking trojan
ruletype: Sigma
author: Deepanjal kumar
date: 2023/18/09
description: Qakbot is a banking trojan initially developed with credential stealing capabilities, it was one of the t
detection:
    SELECTION_1:
        EventID: 4688
    SELECTION_2:
        Channel: Security
    SELECTION_3:
        Hashes:
        - '*MD5=BCE0DF8721504D50F4497C0A0A2C090D*'
        - '*SHA256=112A64190B9A0F356880EEBF05E195F4C16407032BF89FA843FD136DA6F5D515*'
        - '*IMPHASH=D0677C37536C81BB8834FE6FCBC66574*'
    SELECTION_4:
        md5: F472B4AE02E5956F4F25CCFD6ECFDA4B
    SELECTION_5:
        sha256: 112A64190B9A0F356880EEBF05E195F4C16407032BF89FA843FD136DA6F5D515
    SELECTION_6:
        Imphash: D0677C37536C81BB8834FE6FCBC66574
    condition: ((SELECTION_1 and SELECTION_2) and ((SELECTION_3) or
        (SELECTION_4 or SELECTION_5 or SELECTION_6)))
falsepositives:
- Unknown
id: cf0c254b-22f1-4b2b-8221-e12d178sjdjak
```

Mapping to MITRE FRAMEWORK:

MITRE TECHNIQUE NAME	MITRE ID
Windows management instrumentation	ID: T1047
Sandbox evasion	ID: T1497
Account discovery	ID: T1087
Browser information discovery	ID: T1217