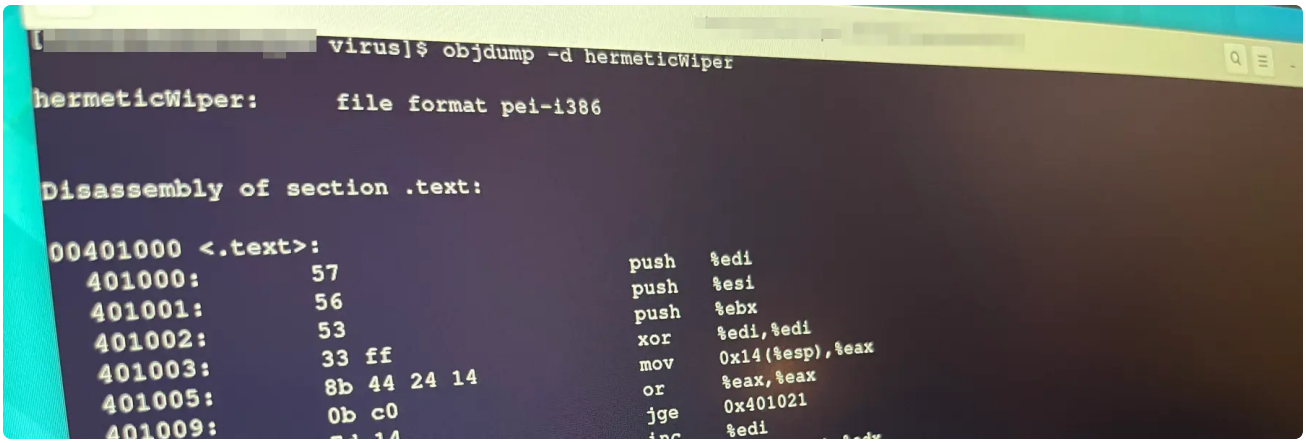
Reverse Engineering | Hermetic Wiper

englert.one/hermetic-wiper-reverse-code-engineering

von Thomas

Erstellt am 25.02.2022



Hermetic Wiper eine erste Analyse

Der Ukraine-Russland Konflikt wird auch im Internet mittels Attacken auf Computersysteme geführt. Am 23.02 hat ESET, ein Unternehmen für Sicherheitssoftware, eine neue Wiper Malware entdeckt. ESET hat die Meldung erstmals via Twitter veröffentlicht. In diesem Artikel möchte ich eine erste Analyse mittels Reverse Engineering der Malware beschreiben. Im Vorhinein ist aber bereits eines klar: die Malware löscht die komplette Festplatte. Im Rahmen meiner Studienarbeit, die bis Ende Juni veröffentlicht werden soll, werde ich den Virus genauer analysieren und weitere Erkenntnisse hier ergänzen.

Was ist ein Wiper Trojaner?

Laut virustotal.com wird die Schadsoftware als ein Trojaner mit der Untergruppe Wiper klassifiziert. Trojanische Pferde (oft auch Trojaner) tarnen sich als unbedenkliche Software oder Dokumente. Nutzer laden diese meist herunter und werden von scheinbar seriösen Quellen getäuscht. Die Malware führt später die Schadfunktionen verborgen im Hintergrund aus. Ziel ist oft andere Malware nachzuinstallieren, um sie für Folgeangriffe verwundbar zu machen.

Namensherkunft vom Hermetic Wiper

Laut sentinelone.com ist die Malware mit einem Zertifikat von Hermetica Digital Ltd signiert. Die Forscher von sentinelone.com gehen davon aus, dass es sich um eine Scheinfirma handelt, da das Zertifikat von April 2021 ist und für die Forscher nie außer im Falle von Hermetic Wiper in Erscheinung getreten ist.

Statische Analyse von Hermetic Wiper

Die Datei ist 115 KB groß und als *PE32 executable (GUI) Intel 80386, for MS Windows* identifiziert. Das bedeutet, dass die Malware ausschließlich für das Betriebssystem Microsoft Windows entworfen und entwickelt wurde.
Mit dem Befehl *strings* unter Linux können alle in der Datei enthaltenen Strings ausgegeben werden. Dabei interessant ist die sogenannte Data Sektion, welche unter Wikipedia ausführlich beschrieben ist. Sie ist Bestandteil jeder Portable Executable. Die von mir händisch gefilterte Ausgabe ist in folgendem dargestellt:

```
NTFS
Wow64DisableWow64FsRedirection
Wow64RevertWow64FsRedirection
IsWow64Process
GCTL
.text
.text$mn
.idata$5
.rdata
.rdata$zzzdbg
.xdata$x
.idata$2
.idata$3
.idata$4
.idata$6
.data
.bss
.rsrc$01
.rsrc$02
StrStrA
wnsprintfW
StrCmpNW
StrStrIW
PathAppendW
PathAddBackslashW
StrCatBuffW
PathFileExistsW
PathFindExtensionW
PathAddExtensionW
StrToIntW
StrChrW
StrRChrW
StrStrW
SHLWAPI.dll
LZOpenFileW
LZClose
LZCopy
LZ32.dll
wcsncpy
towupper
msvcrt.dll
HeapAlloc
GetProcessHeap
DeviceIoControl
GetLastError
HeapReAlloc
HeapFree
lstrcmpA
GetSystemTimeAsFileTime
CreateFileW
CloseHandle
SetFilePointerEx
ReadFile
GetDiskFreeSpaceW
lstrlenW
```

```
WriteFile
FlushFileBuffers
CreateThread
WaitForMultipleObjects
GetModuleHandleW
GetProcAddress
GetCurrentProcess
VerSetConditionMask
VerifyVersionInfoW
FindResourceW
LoadResource
LockResource
SizeofResource
GetSystemDirectoryW
DeleteFileW
Sleep
WaitForSingleObject
SetThreadPriority
FindFirstFileW
FindNextFileW
FindClose
GetLogicalDriveStringsW
SetLastError
GetCommandLineW
GetModuleFileNameW
GetFileAttributesW
CreateEventW
SetEvent
ExitProcess
GetCurrentProcessId
GetFileInformationByHandle
KERNEL32.dll
wsprintfW
CharLowerW
USER32.dll
CryptAcquireContextW
CryptGenRandom
CryptReleaseContext
RegDeleteKeyW
OpenProcessToken
LookupPrivilegeValueW
AdjustTokenPrivileges
OpenSCManagerW
OpenServiceW
CreateServiceW
QueryServiceStatus
ChangeServiceConfigW
StartServiceW
DeleteService
CloseServiceHandle
ControlService
InitiateSystemShutdownExW
RegQueryInfoKeyW
RegEnumKeyExW
RegOpenKeyW
```

```
RegSetValueExW
RegCloseKey
ADVAPI32.dll
CommandLineToArgvW
SHELL32.dll
_except_handler3
memcpy
memset
DigiCert Inc1
www.digicert.com1+0)
"DigiCert EV Code Signing CA (SHA2)0
210413000000Z
2204142359592Z0
Private Organization1
        HE 4194691
Nicosia1
Hermetica Digital Ltd1
Hermetica Digital Ltd0
1http://crl3.digicert.com/EVCodeSigningSHA2-g1.crl07
1http://crl4.digicert.com/EVCodeSigningSHA2-g1.crl0J
http://www.digicert.com/CPS0
http://ocsp.digicert.com0H
http://cacerts.digicert.com/DigiCertEVCodeSigningCA-SHA2.crt0
DigiCert Inc1
www.digicert.com1+0)
"DigiCert High Assurance EV Root CA0
DigiCert Inc1
www.digicert.com1+0)
"DigiCert EV Code Signing CA (SHA2)0
http://ocsp.digicert.com0I
=http://cacerts.digicert.com/DigiCertHighAssuranceEVRootCA.crt0
:http://crl3.digicert.com/DigiCertHighAssuranceEVRootCA.crl0@
:http://crl4.digicert.com/DigiCertHighAssuranceEVRootCA.crl0
.http://www.digicert.com/ssl-cps-repository.htm0
DigiCert Inc1
www.digicert.com1+0)
"DigiCert EV Code Signing CA (SHA2)
```

Die Analyse zeigt, dass der Wiper wohl für das NTFS Filesystem entwickelt ist. Das lässt der Ouput in der ersten Zeile mutmaßen. Zusätzlich sind viele Funktionen in der Executable enthalten. Interessante Funktionen sind Wow64DisableWow64FsRedirection, Wow64RevertWow64FsRedirection und IsWow64Process. Diese drei Funktionen werden in Kombination genutzt, um Weiterleitungen aus dem Dateisystem zu deaktivieren.

Die letzten Zeilen der Ausgabe bestätigen die Annahme von sentinelone.com, dass die Malware für Hermetica Digital Ltd signiert ist. In der Ausgabe wird sogar bekannt, dass DigiCert der Herausgeber des Zertifikates ist.

Mittels des Befehls *objdump -d* ist es möglich, die Binäre Datei zu disassemblieren. Mit den Parametern *-M Intel* ist es möglich die Disassemblierung in der Intel Syntax darzustellen. Leider stand mir bisher nicht genügend Zeit zur Verfügung, den Assemblercode zu

analysieren. Ich hoffe ich werde dazu im Rahmen meiner Studienarbeit Zeit finden und den Abschnitt entsprechend ergänzen.

Analyse nach Packern und Crypters

Häufig werden oft Packer oder Crypters verwendet, um ein Reverse Code Engineering von Malware zu erschweren bzw. unmöglich zu machen. Mittels des Kommandos *objdump -D* können alle Assembler Sektionen einer Datei erzeugt werden. Im Fall von Hermetic Wiper sind folgende Sektionen vorhanden:

```
$ objdump -D hermeticWiper | grep section
Disassembly of section .text:
Disassembly of section .rdata:
Disassembly of section .data:
Disassembly of section .rsrc:
Disassembly of section .reloc:
```

Die aufgelisteten Sektionen sind übliche Sektionen für Windows Executables. Daher ist nicht davon auszugehen, dass ein Packer oder Crypter bei Hermetic Wiper verwendet wurde.

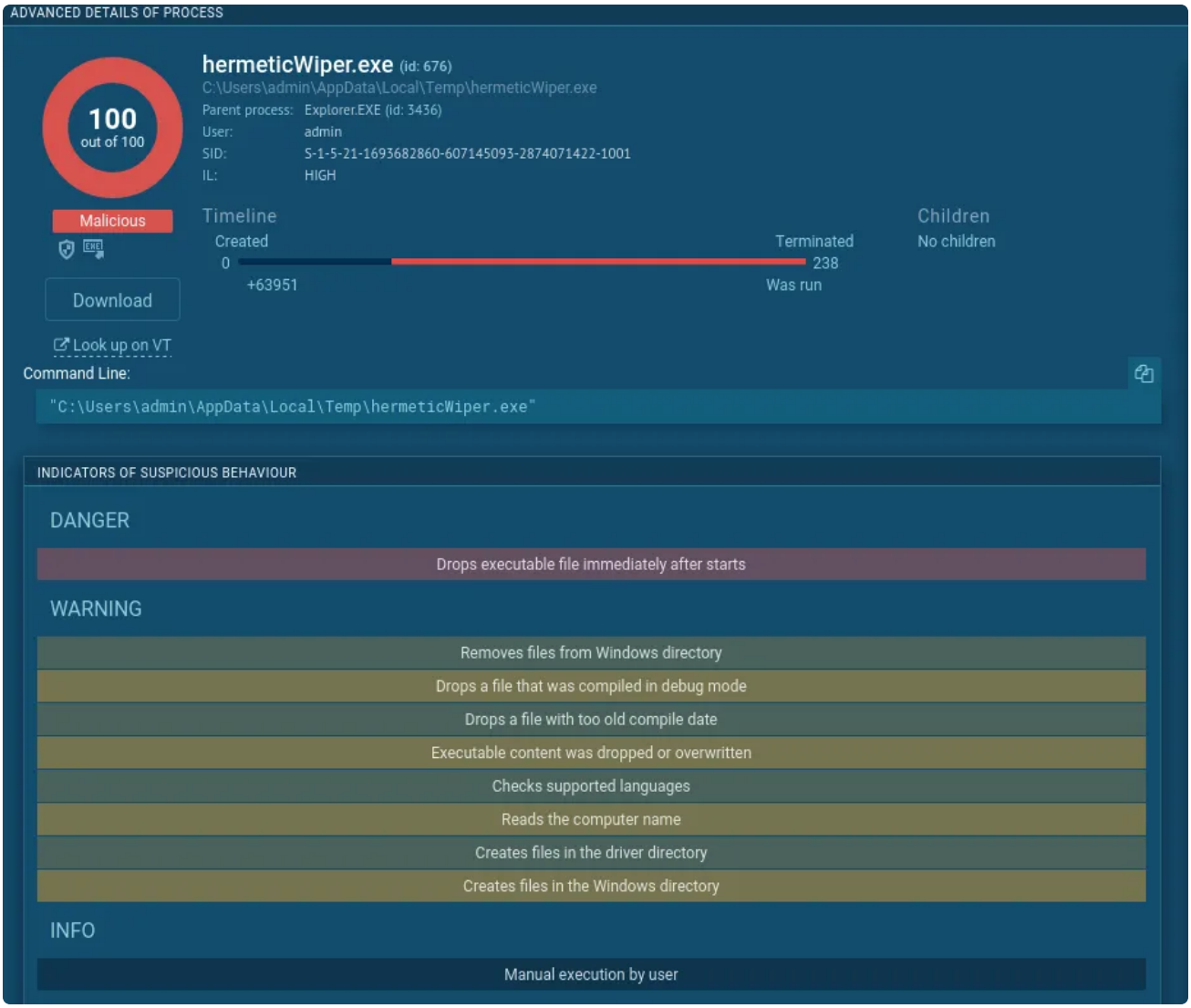
Hashsummen von Hermetic Wiper:

- **md5** - 3f4a16b29f2f0532b7ce3e7656799125
- **sha1** - 61b25d11392172e587d8da3045812a66c3385451

Mittels dieser Checksummen kann eine statische Analyse auf bekannten Seiten wie virustotal durchgeführt werden. Stand heute (25.02.2022 09:55 Uhr) wird die Malware laut VirusTotal von 37 Scannern erkannt. Darunter befindet sich auch der Microsoft Defender.

Dynamische Analyse von Hermetic Wiper

Die dynamische Analyse erfolgt bei mir im ersten Schnitt mit dem Tool any.run. Bei der Analyse stellt sich heraus, dass nach der Ausführung des Hermetic Wiper eine .sys Datei unter *C:\WINDOWS\system32\Drivers* abgelegt wird. Der Name der Datei ist immer vier Zeichen lang und verfügt über die Dateiendung .sys. Im weiteren Fall bezeichne ich die Datei als Treiberdatei. Die erzeugte Datei hat immer die gleiche Hashsumme. Das Tool file identifiziert die Datei *PE32+ executable (native) x86-64, for MS Windows*. Das weitere Verhalten von Hermetic Wiper in der Sandbox von any.run, ist im Bild dargestellt.



Resultat dynamische Analyse Hermetic Wiper

Nennenswerte besondere Verhalten gibt es in dem Fall nicht weiter zu beschreiben. Hermetic Wiper ist darauf ausgelegt, eine .sys Datei auf der Festplatte abzulegen. In der weiteren Analyse werde ich mich nun dieser Datei widmen.

Statische Analyse der erzeugten Treiberdatei

Da die Datei von Hermetic Wipe komplett neu erzeugt wird, muss eine erneute statische Analyse durchgeführt werden. Die Datei ist 17,48 KB groß und wird, wie im vorherigen Abschnitt schon beschrieben, als PE32+ executable (native) x86-64, for MS Windows von file erkannt. Die von mir gefilterte strings Ausgabe hat folgende interessante Inhalte:

```
Invalid parameter passed to C runtime function.
RSDS
h:\epm2.0\01_projectarea\00_source\epm2\mod.windiskaccessdriver\windiskaccessdriver\c

ExAllocatePoolWithTag
IoGetLowerDeviceObject
IoBuildDeviceIoControlRequest
IoDeleteSymbolicLink
ExFreePoolWithTag
RtlInitUnicodeString
IoDeleteDevice
KeSetEvent
KeInitializeEvent
IoFreeMdl
MmMapLockedPagesSpecifyCache
IoGetDeviceObjectPointer
IoBuildAsynchronousFsdRequest
IofCompleteRequest
KeWaitForSingleObject
IoFreeIrp
IoGetAttachedDeviceReference
RtlCompareUnicodeString
MmUnlockPages
ObfReferenceObject
IoCreateSymbolicLink
ObfDereferenceObject
RtlUnicodeStringToInteger
IoCreateDevice
DbgPrint
ObDereferenceObjectDeferDelete
IofCallDriver
KeBugCheckEx
ntoskrnl.exe
RtlAnsiCharToUnicodeChar
VeriSign, Inc.1
VeriSign Trust Network1;09
2Terms of use at https://www.verisign.com/rpa (c)101.0,
%VeriSign Class 3 Code Signing 2010 CA0
120423000000Z
140911235959Z0
Sichuan1
Chengdu100.
'CHENGDU YIWO Tech Development Co., Ltd.1>0<
5Digital ID Class 3 - Microsoft Software Validation v2100.
'CHENGDU YIWO Tech Development Co., Ltd.0
r*@XD
!rKl
90705
/http://csc3-2010-crl.verisign.com/CSC3-2010.crl0D
=0;09
0*0(
https://www.verisign.com/rpa0
e0c0$
http://ocsp.verisign.com0;
/http://csc3-2010-aia.verisign.com/CSC3-2010.cer0
```

```
Washington1
Redmond1
Microsoft Corporation1)0'
 Microsoft Code Verification Root0
110222192517Z
210222193517Z0
VeriSign, Inc.1
VeriSign Trust Network1:08
1(c) 2006 VeriSign, Inc. - For authorized use only1E0C
VeriSign Class 3 Public Primary Certification Authority – G50
Dhttp://crl.microsoft.com/pki/crl/products/MicrosoftCodeVerifRoot.crl0
https://www.verisign.com/cps0*
https://www.verisign.com/rpa0
```

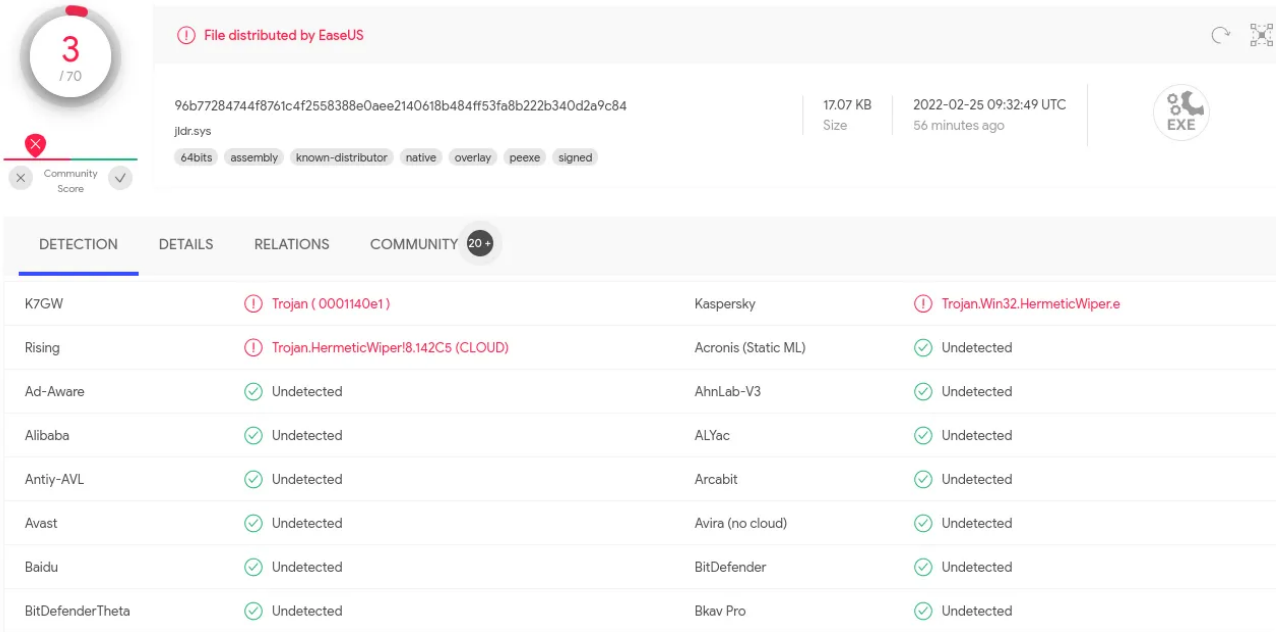
Wie genau die Malware vorgeht, ergibt die statische Analyse natürlich nicht. Dennoch lassen sich mögliche Verhaltensweisen bereits identifizieren. Die Analyse mit Strings ergibt, dass die FunktionIoDeleteDevice verwendet wird. Das lässt darauf zurückführen, dass mit der Treiberdatei ggf. Daten von der Festplatte gelöscht werden sollen.
Interessant ist, dass VeriSign in dieser Datei ein Rolle spielt. Es ist auch möglich eine Datei unter dem Link http://csc3-2010-crl.verisign.com/CSC3-2010.crl herunterzuladen, die Stand jetzt mir keine Erkenntnisse liefert.

Auf eine Analyse des Kommandos objdump verzichte ich hier mangels Zeit, wie bei Hermetic Wiper. Ich hoffe ich kann eine Analyse im Rahmen meiner Studienarbeit nachliefern. Interessant ist, dass die Disassemblierung zwei Sektionen erzeugt: .text und INIT.

Checksummen der Teriberdatei

- **md5** - 6106653b08f4f72eeaa7f099e7c408a4
- **sha1** - 0e84aff18d42fc691cb1104018f44403c325ad21

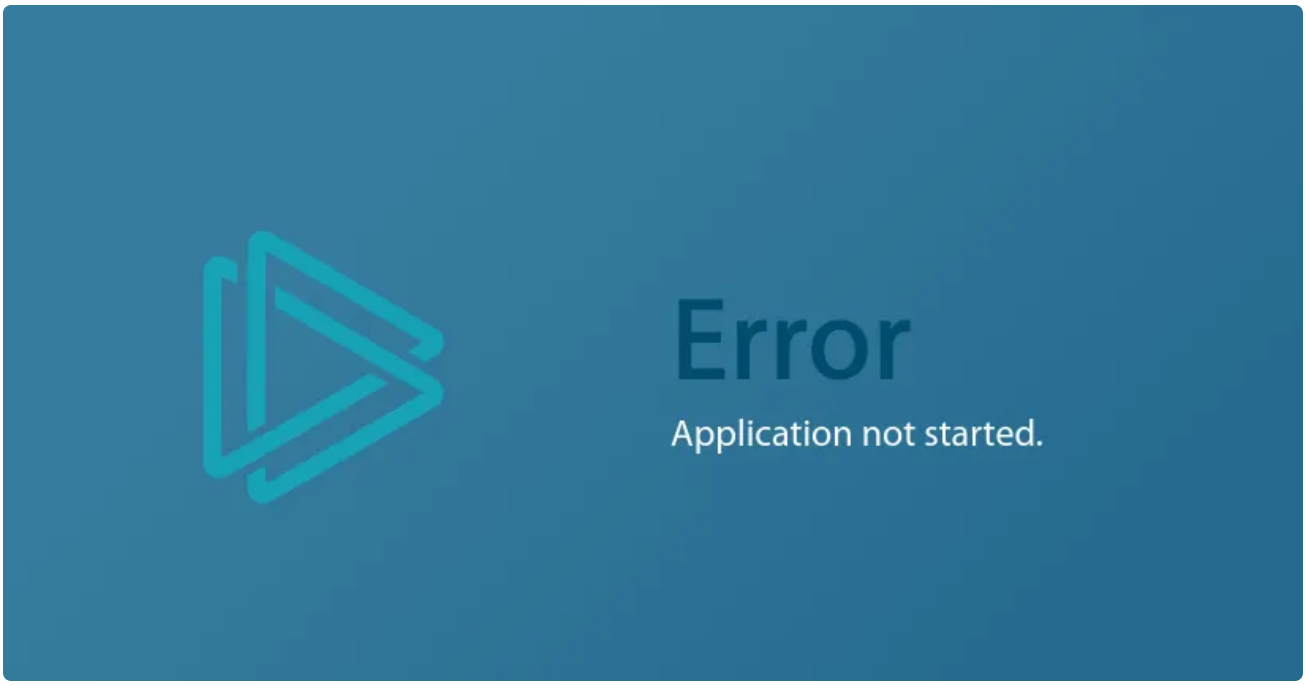
Die Checksummen sind bisher (25.02.2022 11:27 Uhr) weitgehend unbekannt. Bei VirusTotal schlagen erst drei von 71 Scannern an. Siehe Bild



Virustotal Bekanntheit der erzeugten .sys Datei am 2022-02-25 09:32:49 UTC

Dynamische Analyse der erzeugten Treiberdatei

Das Verhalten nach der Ausführung der Treiberdatei ist relativ flott erklärt: Das System ist kaputt. Dabei werden vermutlich alle Daten der Festplatte inkl. MBR gelöscht. Eine Analyse mittels any.run ist leider nicht möglich, da das System crasht. Ich werde, falls es im Rahmen meiner Studienarbeit mit einer anderen Sandbox anaylsiert wird, die Erkenntnisse teilen.



Any.run crash nach der Ausführung der erzeugten .sys