Gh0stRat Anti-Debugging : Nested SEH (try - catch) to Decrypt and Load its Payload

E tccontre.blogspot.com/2021/02/gh0strat-anti-debugging-nested-seh-try.html



SEH tricks is not a new Anti-Debugging trick. So many malware already used this to make the manual debugging of its code time consuming and confusing. Today I will share how Gh0strat malware make use of nested SEH exception (try{} catch) as anti-debugging trick to hide its decryption routine.

This article is not to tackle the full C++ Exception Internals, but to share how IDAPRO really helps me in analyzing this type of anti-debugging tricks statically. :)

So lets start!!!

SEH:

Structure Exception handler (SEH) is one of the classic Anti-debugging tricks used by malware. where it tries to abuse the mechanism provided by operating system in managing

exceptional situation like reference to a non-existence address pointer or execution of code in a read-only page.

This is done usually by locating the pointer to SEH linked list in the stack known to be the SEH frame. The current SEH frame address is located in 0x00 offset relative to the FS (x32 bit) or GS selection (x64 bit).

0:000> dt	teb
combase! '	TEB
+0x000	NtTib : _NT_TIB
+0x01c	EnvironmentPointer : Ptr32 Void
+0x020	ClientId : _CLIENT_ID
+0x028	ActiveRpcHandle : Ptr32 Void
+0x02c	ThreadLocalStoragePointer : Ptr32 Void
+0x030	ProcessEnvironmentBlock : Ptr32 PEB
+0x034	LastErrorValue : Uint4B
+0x038	CountOfOwnedCriticalSections : Uint4B
+0x03c	CsrClientThread : Ptr32 Void
+0x040	Win32ThreadInfo : Ptr32 Void
+0x044	User32Reserved : [26] Uint4B
+0x0ac	UserReserved : [5] Uint4B
+0x0c0	WOW32Reserved : Ptr32 Void
+0x0c4	CurrentLocale : Uint4B
+0x0c8	FpSoftwareStatusRegister : Uint4B
+0x0cc	ReservedForDebuggerInstrumentation : [16] Ptr32 Void
+0x10c	SystemReserved1 : [26] Ptr32 Void
+0x174	PlaceholderCompatibilityMode : Char
+0x175	PlaceholderHydrationAlwaysExplicit : UChar
+0x176	PlaceholderReserved : [10] Char
+0x180	ProxiedProcessId : Uint4B
+0x184	_ActivationStack : _ACTIVATION_CONTEXT_STACK
+0x19c	WorkingOnBehalfTicket : [8] UChar
+0x1a4	ExceptionCode : Int4B
+0x1a8	ActivationContextStackPointer : Ptr32 _ACTIVATION_CONTEXT_STACK
+0x1ac	InstrumentationCallbackSp : Uint4B
+0x1b0	InstrumentationCallbackPreviousPc : Uint4B
+0x1b4	InstrumentationCallbackPreviousSp : Uint4B
+0x1b8	InstrumentationCallbackDisabled : UChar
+0x1b9	SpareBytes : [23] UChar

figure 1: FS[0] of x32 bit OS

0:000>	> dt	_NT_TIB						
combas	e! I	NI IIB						
+0x	000	ExceptionList	:	Ptr32	_EXCEP	TION	REGISTRATION	RECORD
+0x	004	StackBase	1	Ptr32	Void			
+0x	008	StackLimit		Ptr32	Void			
+0x	(00c	SubSystemTib		Ptr32	Void			
+0x	010	FiberData		Ptr32	Void			
+0x	010	Version		Uint48	3			
+0x	014	ArbitraryUserPoir	nte	er : Pi	tr32 Vo	id		
+0x	018	Self	:	Ptr32	NT TI	B		
0:000>	∙ dt	_EXCEPTION_REGIST	ΓR,	ATION_R	RECORD			
combas	;e!_i	EXCEPTION_REGISTR/	AT.	ION_REG	CORD			
+0x	(000	Next		Ptr32	_EXCEP	TION	REGISTRATION	RECORD
+0×	004	Handler		Ptr32	E	XCEPT	ION_DISPOSIT	ON

figure 2: The EXCEPTION_REGISTRATION_RECORD in FS[0]

When the exception is triggered, control is transfer to the current SEH handler where it will return one of the _EXCEPTION_DISPOSITION members.

In this Gh0strat variant it used nested SEH (try{} catch{}) that serve as anti-debugging tricks to make the debugging more confusing or let say more time consuming if analyst didn't notice the SEH.

Gh0srat: Nested SEH to decrypt its payload:

The sample we will use here contains a big data section where the encrypted gh0strat payload located. we will notice that using DIE tool or PE-bear that visualized the size of each section with quite high entropy same as text section.



figure 3: high entropy of data section

we all know there are so many faster way to bypassed this anti-debugging technique like monitoring the TIB offset 0x0 dynamically for next SEH or dumping process. In our case I will just want to share how IDA PRO will help you a lot in this case in traversing "**FuncInfo**" structure since IDAPRO resolved most of this SEH structure.

when you try to load the sample in debugger and breakpoint on some API let say, you may encounter some exception error shown in figure below. This is also a good hint that it may use SEH technique.

🕮 CPU	👰 Graph	📄 Log	📋 Notes	Bre	akpoints	Memory Map	🗐 Cal
Ⅲ ← →	74838022 74838028 74838028 74838029 74838029 74838020 74838030 74838033 74838033 74838033 74838033 74838033 74838033 74838033 74838033 74838034	8B4C 33CC E8 A 8BE5 5D C2 1 8364 ^ EB D 6A 0 58	24 54 3490000 000 24 10 00 E F		mov ecx,dwo xor ecx,esp call kernel mov esp,ebp pop ebp ret 10 and dword p jmp kernel push F pop eax	rd ptr ss:[e base.74B3F90 tr ss:[esp+1 ase.74B3B018	••••••••••••••••••••••••••••••••••••••
ecx=3 dword ptr .text:748	<pre> [esp+54] 33B022 ker</pre>	=[0019FE60 nelbase.d	[]=65BCC3E	9 #11A422	2		>
💷 Dump 🛙	1 💷 Dum	p 2 📖 🛙	ump 3 🛛 関	Dump 4	💷 Dump 5	🛞 Watch 1	√ [x]
Address	Нех					ASCII	~
77331000 77331010	36 00 38 28 00 2A	00 A4 C4 00 44 C4	<u>33 77</u> 1C (33 77 34 (00 1E 00	70 C4 33 77 0C C4 33 77	6.8.¤A3w (.*.DÄ3w4.	p 6
77331020	1E 00 20	00 EC C3	33 77 <u>1A (</u>	00 1C 00	DO C3 33 77	ìÅ3w	Ð
77331030	18 00 1A	00 <u>B4 C3</u>	33 77 20 0	00 22 00	90 C3 33 77	A3w .	"••
77331040	20 00 22	00 08 C3	<u>33 77</u> 20 0 33 77 18 0	0 14 00	EC C2 33 77	0.2. (ASW,. "Δ3w	··.4
77331060	10 00 12	00 D8 C2	33 77 00 0	0 02 00	90 5D 33 77	ØÅ3w.	
77331070	0E 00 10	00 20 82	33 77 OC 0	00 OE 00	10 82 33 77		
77331080	06 00 08	00 F0 81	33 77 06 0	00 80 00	00 82 33 77	ð.3w	
77331090	06 00 08	00 <u>F8 81</u>	<u>33 77</u> 06 (00 80 00	08 82 33 77	Ø.3W	•••
773310A0	08 00 0A	00 54 77	<u>33 77</u> 22 0	00 24 00	EO 80 33 77	Tw3w".	\$.à∨
<					00 00 00 01	1.6 5.3.0	
-				-	exception en	counter	
Command:				_		oodinter.	_
Command:							

figure 4: exception during debugging

by further checking its code using IDAPRO I notice that it uses nested SEH. yes a nested try{} catch{} exception handler to decrypt its payload. at the entry point of the malware code you will notice right away the first exception handler function registered to FS:0. Exception will be trigger by calling "call __CxxThrowException" API.



figure 5: first SEH in malware entrypoint

ehFuncInfo or the exception handler function registered in FS:0 contains some structure that may help us to figure out statically which exception handler function may be call upon the exception is trigger.

I really recommend to read this great presentation of hexblog regarding the Exception and RTTI:

https://www.hexblog.com/wp-content/uploads/2012/06/Recon-2012-Skochinsky-Compiler-Internals.pdf

The ehFuncInfo is a object structure that may lead you to the "AddressOfHandler" which is a address or a function address that will handle the exception encounter of the current thread.

IDAPRO really did a good job to give you some hint how to traverse that structure and lead you the said structure member of **HandlerType**. FuncInfo structure contains several member so I will just focus on the member that helps me to decrypt the payload.

Below is a simple structure starting from "FuncInfo" that may help you to look for the AddressOfHandler field member of HandlerType structure.

FuncInfoV1	<pre>struc ; (sizeof=0x1C,</pre>	mappedto_20)
		; XREF: .rdata:stru_408788/r
		; .rdata:stru_408848/r
magicNumber	dd ?	; base 16
maxState	dd ?	; base 10
pUnwindMap	dd ?	; offset
nTryBlocks		of try block
pTryBlockMap	dd ?	; offset
nIPMapEntries	dd ?	hase 10 DVA (to that
pIPtoStateMap	dd ?	; OTTSEC BYA OF CIV DIOCK
FuncInfoV1	ends	Gindy
TryBlockMapEntr	v struc : (sizeof=0x14.	mappedto 22)
	,,	: XREF: .rdata:stru 4087D8/r
		: .rdata:004087EC/r
tryLow	dd ?	; base 10
tryHigh	dd ?	; base 10
catchHigh	dd ?	; base 10
nCatches	dd ?	; base 10
pHandlerArray .	dd ?	; offset
[ryBlockMapEntr	y ends	structure that holds the
	-	AddressOfhandler field
HandlerType	<pre>struc ; (sizeof=0x10,</pre>	mappedto_23)
		; XREF: .rdata:stru_408818/r
		; .rdata:stru_408828/r
adjectives	dd ?	; base 16
рТуре	dd ?	; offset
dispCatchObj	dd ?	; base 10 address of handler registered
addressOfHandle	er dd ?	to handle the exception
HandlerType	ends	

figure 6: Traversing AddressOfhandler

The figure 6 shows that FuncInfo structure contains TryBlockMap field. this field is another structure object that contains HandlerArray field structure that holds the AddressOfHandler field. so to make use of this structure in our sample lets try to traverse the first SEH in malware entry point.

u_409478 u_409488	0, 0> align 8 UnwindMapEntry <-1, UnwindMapEntry <-1, TryBlockMapEntry <0	0> ; DATA XREF: .rda 0> 0, 1, 1, offset stru ; DATA XREF: .rda	ata:stru_4094581 1_4094A0>	o a	offset to HandlerTyp ddressOfHandler, the during	e Structure that c address that will g exception	contains the I be executed	
u_4094A0 MPORT_DESCRI	align 10h HandlerType <0, 0, 0 PTOR_KERNEL32 dd rva dd 0 dd 0	<pre>a, offset loc_40243D></pre>	10: 4094881	0		• DATA YPEE	rdata-stru 4	
000094A0 004 < A Structures 0000001C Func	dd rva aKernel32Dll 094A0: .rdata:stru_4 	; DLL Name 94A0 (Synchronized w	; catch()	// owned push call add mov retn	by 40242A offset aShellex sub_402200 esp, 4 eax, offset loc_	; "Shellex" 402450	addressOfHandle be executed b exception	r that will ny this n
0000001C 000000000 ; [0 00000000 ; [0 00000000 ; 00000000 ; 00000000 Hanc	00000008 BYTES. COLL 00000014 BYTES. COLL 0 1 for Type struc;	PSED STRUCT UnwindMap PSED STRUCT TryBlockM (sizeof=0x10, mappedt	; loc_402450:	mov pop	ecx, [ebp+var_C] edi	; CODE XREF: ; DATA XREF:	WinMain(x,x,x WinMain(x,x,x	,x)+4F↑j ,x)+4A↑o
0000000 0000000 adje 0000000 dje 0000000 disp 0000000 addr 0000000 addr 0000000 i [0	ectives dd ? pe dd ? pCatchObj dd ? ressOfHandler dd ? dlerType ends 00000010 BYTES. COLLA	; XRE ; .rd ; bas ; off ; bas ; off ; bas	F: .rdata:stru_4 ata:stru_408828, e 16 set e 10 set o. PRESS CTRL-NU	408818/r /r JMPAD+ T(D EXPAND]			

figure 7: Traversing the AddressOfHandler of SEH in malware entry point

We saw that the possible Address that will handle the exception is in 0x40243d. It works in dynamically test I did with x64dbg where I put break point on this address after the exception then press skip exception shift+f9.

00402405	68 50784000	puch 1 407850	
00402405	68 F07B4000	push 1.40/BF0	
0040240A	64:A1 00000000	mov eax,dword ptr ms:[0]	
00402410	50	push eax	
00402411	64:8925 00000000	mov dword ptr fs:[0],esp	
00402418	83EC 08	sub esp,8	
0040241B	53	push ebx	
0040241C	56	push esi	
0040241D	57	push edi	
0040241E	8D45 EC	lea eax,dword ptr ss:[ebp-14]	
00402421	8965 F0	mov dword ptr ss:[ebp-10],esp	
00402424	68 78874000	push 1.408778	
00402429	50	push eax	
0040242A	C745 FC 00000000	mov dword ptr ss:[ebp-4],0	
00402431	C745 EC 3B000000	mov dword ptr ss:[ebp-14],3B	3B:';'
00402438	E8 8D040000	call 1.4028CA	
00402430	68 40E04100	push 1.41E040	41E040:"Shellex"
00402442	E8 B9FDFFFF	call 1.402200	
00402447	83C4 04	add esp,4	
0040244A	B8 50244000	mov eax,1.402450	
0040244F	C3	ret	

figure 8: AddressOfHandler was triggered

if we follow the call function 0x402200 pushing string address "Shellex" as a parameter. you will notice again that it use another SEH to execute piece of its code. Not like the first SEH, this SEH contains 9 tryblock and HandlerOfAddress like the figure below.



figure 9: multiple try Block Map

Parsing ehFuncInfo structure Using Ida python:

In this case I decided to use IdaPython to parse all the **FrameHandler ehFuncInfo** structure to locate all **AddressOfHandler** field available for all **tryBlockMap** entries and add it as a code reference comment in its IDB. This approach help me to figure out where the decryption routine and learn multi line comment in idapython :).

the script is available here:

https://github.com/tccontre/KnowledgeBase/tree/main/malware_re_tools/gh0strat_seh_helpe r

CACIOOTOLLOO				
ext:00402200 ;unwind {	// SEH_40	2200		
ext:00402200	push	ebp		
ext:00402201	mov	ebp, esp		
ext:00402203	push	ØFFFFFFFh		
ext:00402205	push	offset SEH_402200	hefore	
ext:0040220A	mov	eax, large fs:0	DEIDIE	
ext:00402210	push	eax		
ext:00402211	mov	large fs:0, esp		_
ext:00402218	sub	esp, 2Ch		
ext:0040221B	push	ebx		
ext:0040221C	push	esi		
ext:0040221D	push	edi		
ext:0040221E	lea	eax, [ebp+pException	nObject]	
ext:00402221	mov	[ebp+var_10], esp		
ext:00402224	push	offset	ThrowInfo	
ext:00402229	push	eax ; pi	ExceptionObject	
ext:0040222A	mov	[ebp+var_4], 0		
ext:00402231	mov	[ebp+pExceptionObjec	:t], 5Dh ; ']'	
ext:00402238	call	_CxxThrowException	<pre>38 ; _CxxThrowException(x,x)</pre>	
ext:0040223D ;				
ext:0040223D				
ext:0040223D loc_40223D:		; D/	ATA XREF: .rdata:stru_4093C8	10
ext:0040223D	push	offset aA3f0d ; "/	A3F0D"	
ext:00402242	push	14000h		•
ext:00402247	push	offset unk_40A040		6
ext:0040224C	call	sub_402150		-
ext:00402251	add	esp, 0Ch		
ext:00402254	mov	eax, offset loc_4022	25A	
ext:00402259	retn			
ovt.00400054 .				

figure 10A: before running the script



figure 10B: IDB after running the script

now with this comment we can verify all possible AddressOfHandler in each tryBlockMap entry to locate the decryption routine. Like the figure above, the first AddressOfHandler isa function waiting for the decryption key, size of the encrypted payload and the address of the encrypted payload.

);unwind { //	SEH_402	2150		
)	push	ebp		
	mov	ebp, esp		
4	push	ØFFFFFFFh		
ehFuncInfoStruc	tAddr: 0	0x409208 ptryBlockMapStructAddr: 0x409238 t	tryBlockCount: 0	x1 handlerTypeStructAddr: 0x409250 handlerOfAddres: 0x40219b
4	push	offset SEH_402150		A CONTRACT OF A CONTRACT.
λ	mov	eax, large fs:0		`
1	push	eax		
	mov	large fs:0, esp	108.	· DATA VREE · pdata stav 4003E0 a
1	sub	esp, 10h	(19D):	; DATA AREF: .Fudid:Stru_40925000
1	push	ebx	mov	eax, [ebp+var_io]
1	push	esi	mov	eux, [ebp+arg_o]
1	push	edi	mov	est, [ebptang A]
1	xor	edi, edi	inc	ecx, [ebp+arg_0]
1	mov	[ebp+var_10], esp	Inc	edi edi
1	mov	[ebp+var_14], edi	201	[ebntvar 18] eav
i -	mov	[ebp+var_18], edi	nush	edi : dwMilliseconds
1			push	al [eaviedy]
loc_402179:		; CODE XREF: sub_4	mov	d] [esitery]
1	mov	eax, [ebp+var_14]	xor	dl al
:	mov	ecx, [ebp+arg_4]	mov	[esitery] d]
-	cmp	eax, ecx	call	decryption routine
	jge	short loc_4021E8	mov	eav. esi
1	lea	ecx, [ebp+pExceptionObject]	mov	ecx, 5
i	push	offsetTI1H ; pThrowInfo	cda	conj s
1	push	ecx ; pExceptionObject	idiv	ecx
	mov	[ebp+var_4], edi	test	edx. edx
	mov	[ebp+pExceptionObject], 58h ; 'X'	inz	short loc 4021D0
i -	cail	CxxThrowException@8 ; _CxxThrowE		

figure 11: decryption routine

and once you decrypt the payload using this simple xor decryption routine. you can see right away some note worthy string of gh0strat like keylogging, creating services, regrun, download files, backdoor and etc.

UP] [Right][Down] [Num Lock] / - + 0 1 2 3 4 5 6 7 8 9 . [INSERT] [DELETE] [BACKSPACE][N
[F3] [F4] [F5] [F6] [F7] [F8] [F9] [F10] [F11] [F12] ~ @ # \$ % ^ & * () _ ·
D F G H J K L : " Z X C V B N M < > ? -= [CTRL] [WIN] [1 m±] [WIN] [Print Screen] [Scr
ageDown] [Left] [Up] [Right] [Down] [Num Lock] / - + 0 1 2 3 4 5 6 7 8 9 . [INSERT] [DELET
√ »χ√ »ղ√ »⊷√ »№ »ң√ »Σ√ »∣√ »+∩ »⊢⊓ »(⊓ »4⊓ »H□
∑² ⊨°² ⊨°² ⊨•°² ⊨•₩ ⊨⊨₩ ⊨LW ⊨(W ⊨KW ⊨HW ⊨XW ⊨KW ⊨KW ⊨KW ⊨KW ⊨KW ⊨KW ⊨KW ⊨KW = 360tray.exe :
xe "=p" wywhat QQPCRTP.exe QQTT-H_#T BaiduSd.exe "" baiduSafeTray.exe " Moust KvMonXP.exe " 12 RavMonD.exe
e comodo y spiber.exe DR.HEB acs.exe Outpost V3SVC.exe
UD32 Knsotray.exe jptu indensky.exe p=p avcenter.exe −1[#1 asnusp.exe Avast="top]#4, rtvscan.exe tij* remupd.exi 201 d. unter internet internet internet internet internet available to the second
An advisting Advisting Advisting 21 - Mindows 2 - Mindows 2 - Mindows 2 - Mindows 2012 - GetDondwithen Advisting 2 - All Hitsels
tion Enternnise Edition Risiness Edition Starter Edition Cluster Server Edition Datacenter Edition Datacenter Edition (core in
tion) Enterprise Edition for Itanium-based Systems Small Business Server Small Business Server Premium Edition Standard Edit:
Windows Server 2003 R2. Windows Storage Server 2003 Windows Home Server Windows XP Professional x64 Edition Windows Server 2003
ion for Itanium-based Systems Datacenter x64 Edition Enterprise x64 Edition Standard x64 Edition Compute Cluster Edition Data
Windows XP Home Edition Professional Windows 2000 Professional Datacenter Server Advanced Server Server (build
Windows.8 %u %u EXPLORER.EXE %s 🕴 🖓 🖓 🕂 🖓 🕊 🖓 🕉 NULL ProcessorNameString HARDWARE\DESCRIPTION\System\G
Ke \ .key
[IDF] Xs
[m]#Σ:]Xd-Xd-Xd Xd:Xd:Xd
[-r]; (sackSpace> (Enter> During CorrEctory ClassBarny, Cathering Beighter Esthetics Barbitetics Application Estuate (States)
Fightme OpenProxy CloseProxy Sesnedownerivinge Sesnedownerivinge Application Security System Sistem/Contention (Contention) (Content
stas caute astantidat de stateitopentidamento artar ministrotopenditas as appricaciónstratoponetenetatopentidamento Re RefRe (kau
- A second s
: : SYSTEM\CurrentControlSet\Services\%s Remark Group %4d-%.2d-%.2d %.2d:%.2d InstallTime ref.tbfull.com:15950 user
eSessionId Kernel32.dll %s Win7 Description %s\shell\open\command ~%1 %1 WinStae\Default c:\%s GetModuleFileNameA ker
L32.dll SetFileAttributesA KERNEL32.dll CreateDirectoryA KERNEL32.dll USER32.dll GetFileAttributesA KERNEL32.dll %
V Ole32.dll Coinitialize CoUninitialize CocreateInstance Oleaut32.dll SysFreeString Friendly Name «HHZ beb Technika Control (Name) (Na Name) (Name) (Name
ark installing proof iswoneeprocess kernelsz.dli ntall.dli ktigetutversionnunders jomgsoog wig war sooj war soo Daal Julii Lala Lala Lala Lala Avanta Da Mee Lala Avanta Da Mee Lala Avanta Lala Avanta Lala Avanta Lala Avanta
AVER AVER AVER AVER AVER AVER AVER AVER
A DEAL AND A DEAL
TEM\CurrentControlSet\Control\class\{4D36E968-E325-11CE-8FC1-080028E10318}\0000 Find GPU Graphics error 0 □ 0 C
GetNoduleFileNameA CreateMutexA ReleaseMutex GetLastError CloseHandle Sleep IstrcatA GetTickCount WaitForSingleO
etEvent TerminateThread GetVersionExA GetExitCodeProcess ExpandEnvironmentStringsA GetSystemInfo GetSystemDirectoryA MoveF
s User32.dll wsprintfA ExitWindowsEx MessageBoxA IsWindowVisible SendMessageA EnumWindows MSVCRT.dll strcmp strlen memcj
hostbyname htons connect send recv closesocket setsockopt WSAIoctl select getsockname gethostname ADVAPI32.dll Set
rviceA StartServiceA CloseServiceHandle QueryServiceStatus ControlService CreateServiceA ChangeServiceConfig2A DeleteServ:
reateProcessAsUserA wininet.dll CreateToolhe1p32Snapshot KERNEL32.dll user32.dll GetThreadDesktop GetUserObjectInformat:
ntinreadid user32.dll upenDesktopA CloseDesktop Wainet.dll InternetOpenA MSIE 6.0 InternetOpenUPIA InternetReddHigh
GREIPTIVILEGES LOUKOPPTIVILEGEVALUEA RETHEISZ.GII OULUUTTEHLIPTOLESS REARELSZ.GII OULUSLETYOP 8+ C:VZ.XXI XSS 61 Avt ANNADIS 2010 BARNADUSINE DA DARDARDIKSZ PARADENJALMA BARCHARDIKA BARCHARDIKA
ant ADVARIDZ.DZI REGULETYELDELA REGULERA REGULERALDE REGULERALDER REGULERALDE REGULERALDE ADVARIDZ.DII. Jannanizuri Barlincaleu I
copening car in a range period a contract of the contract of t

figure 12: strings upon decryption

Conclusion:

In this article we just focus on some basic internals of SEH frameHandler and how to look for all possible HandlerOfAddress that may executed upon the trigger of registered SEH. we also learned how IDAPRO did a really good job in giving you all the needed structure for try block entries where you can use IDApython to make your static analysis more easier. :)

IOC:

https://bazaar.abuse.ch/sample/70ac339c41eb7a3f868736f98afa311674da61ae12164042e4 4d6e641338ff1f/

yara:

import "pe"

rule gh0st_rat_loader { meta:

```
author = "tcontre"
    description = "detecting gh0strat loader"
    date = "2021-02-22"
sha256 = "70ac339c41eb7a3f868736f98afa311674da61ae12164042e44d6e641338ff1f"
  strings:
    mz = \{ 4d 5a \}
    $code = { 40 33 FF 89 45 E8 57 8A 04 10 8A 14 0E 32 D0 88 14 0E FF 15 ?? ?? ?? ??
8B C6 B9 ?? 00 00 00 }
    $str1 = "Shellex"
    $str2 = "VirtualProtect"
  condition:
    ($mz at 0) and $code and all of ($str*)
  }
rule gh0st rat payload {
  meta:
    author = "tcontre"
    description = "detecting gh0strat payload in memory without MZ header in memory"
    date = "2021-02-22"
sha256 = "edffd5fc8eb86e2b20dd44e0482b97f74666edc2ec52966be19a6fe43358a5db"
  strings:
  $dos = "DOS mode"
  $av str1 = "f-secure.exe"
  $av str2 = "Mcshield.exe"
  $av str3 = "Sunbelt"
  $av str4 = "baiduSafeTray.exe"
  $clsid = "{4D36E972-E325-11CE-BFC1-08002BE10318}"
  $s1 = "[WIN]"
  $s2 = "[Print Screen]"
  $s3 = "Shellex"
  $s4 = "HARDWARE\\DESCRIPTION\\System\\CentralProcessor\\0"
  $s5 = "%s\\%d.bak"
```

condition:

```
($dos at 0x6c) and 2 of ($av_str*) and 4 of ($s*) and $clsid
```

}

References:

https://www.hexblog.com/wp-content/uploads/2012/06/Recon-2012-Skochinsky-Compiler-Internals.pdf