



UNIVERSITA' DEGLI STUDI DI  
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base  
Corso di Laurea Magistrale in Ingegneria Informatica

Software Security

## ***Ransomware Conti***

Anno Accademico 2021/2022

Professore

**Roberto Natella**

Candidati

**Emma Melluso** matr. M63001176

**Carmine Pio D'Antuono** matr. M63001224

**Pasquale Gaviglia** matr. M63001188

# Indice

<b>1</b>	<b>Introduzione</b>	<b>2</b>
1.1	Modus Operandi . . . . .	3
<b>2</b>	<b>MITRE ATT&amp;CK</b>	<b>6</b>
2.1	MITRE ATT&CK sul malware Conti . . . . .	8
<b>3</b>	<b>Malware Analysis</b>	<b>11</b>
3.1	Basic Static Analysis . . . . .	12
3.2	Basic Dynamic Analysis . . . . .	20
3.3	Advanced Static Analysis - Reverse Engineering . . . . .	26
3.3.1	Dynamic Loading . . . . .	27
3.3.2	Multi-Threading . . . . .	29
3.3.3	Files Encryption . . . . .	32
3.3.4	Console Management . . . . .	35
3.4	Hybrid Analysis - Sandbox . . . . .	36
3.4.1	Malicious Indicators . . . . .	37
3.4.2	Suspicious Indicators . . . . .	38
<b>4</b>	<b>Obfuscation</b>	<b>41</b>
4.1	Windows API Hashing . . . . .	41
4.2	Obfuscation in Conti v2 . . . . .	42

# Capitolo 1

## Introduzione

**Conti** è un sofisticato modello di **RaaS** (*Ransomware-as-a-Service*) rilevato per la prima volta a dicembre 2019. Nel modello **RaaS** gli attaccanti concentrano tutti i loro sforzi nello sviluppo del software, mentre lasciano a soggetti terzi il compito di individuare le vittime ed effettuare il deploy del malware. Esso permette anche ad individui che non hanno particolari conoscenze tecniche di lanciare attacchi ransomware, semplicemente registrandosi a un servizio.

Conti è stato sviluppato dal gruppo hacker russo **Wizard Spider** e fin dal 2019 è stato utilizzato per mettere in atto attacchi di alto profilo, a tal punto che il governo statunitense decise di offrire ricompense in denaro per chiunque scoprisse e condividesse informazioni utili su tale organizzazione e sul suo modus operandi. Tra i più famosi ricordiamo:

- **Tulsa City system shutdown**
- **Irish Health Service**
- **ARMattack campaign**

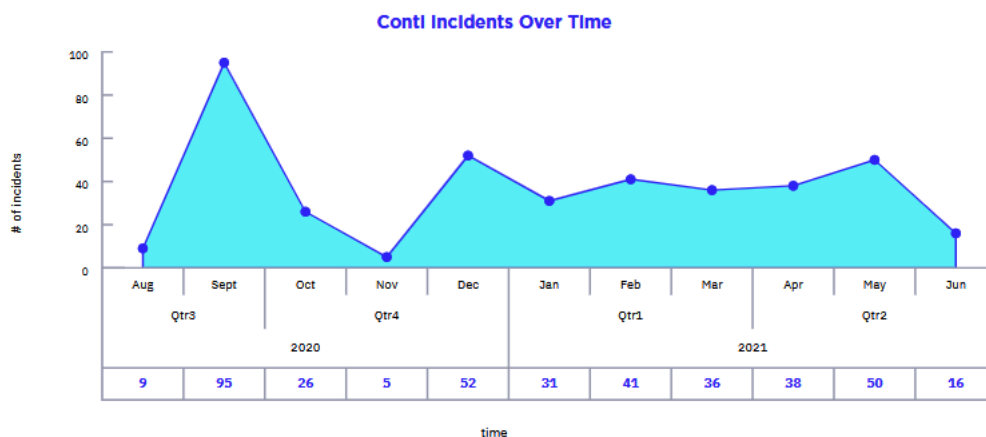


Figura 1.1: Diffusione temporale del malware Conti

In seguito al conflitto Russia-Ucraina, sul blog **Conti News** fu pubblicato un post nel quale

il gruppo hacker dichiarò il proprio supporto alla Russia.

Come conseguenza, un ricercatore ucraino rese pubblici circa 60.000 messaggi delle chat interne al gruppo oltre che il sorgente del malware. Le conversazioni vennero pubblicate a partire dal 27 febbraio 2022 mediante l'account Twitter (*@ContiLeaks*). Ciò portò alla temporanea chiusura del portale utilizzato dagli hacker per riscuotere i pagamenti derivanti dal ransomware.

Wizard Spider sfrutta un approccio iterativo col quale Conti viene continuamente evoluto e adattato in modo tale da renderlo sempre più efficace ed evasivo. Ad ogni iterazione, la firma del ransomware cambia perché tutti gli indicatori statici vengono crittografati utilizzando una logica diversa. Questo mira a raggiungere i seguenti obiettivi:

- **Miglioramento dell'offuscamento:** sin dalle prime versioni, Conti, implementava un semplice meccanismo XOR per nascondere i nomi delle APIs caricate a runtime (offuscamento delle stringhe). Da giugno 2020 esso è stato sostituito da una funzione di codifica custom.
- **Velocità:** Conti utilizza fino a 32 thread CPU simultanei per le operazioni di crittografia dei file e, a partire dall'iterazione di settembre 2020, ha commutato l'algoritmo di crittografia da AES a **CHACHA** per velocizzare ulteriormente il processo di crittografia. In questo modo viene ridotta di molto la possibilità che eventualmente la vittima riesca ad interrompere il ransomware.
- **Ottimizzazione della crittografia dei file:** dal 2 settembre 2020 è stata aggiunta una nuova logica per la crittografia dei file. La logica implementa due diverse modalità: **totale e parziale**, a seconda dell'estensione e della dimensione del file.

## 1.1 Modus Operandi

### Initial Entry

Il principale vettore di attacco utilizzato dal gruppo, sono le **email di phishing**, le quali contengono allegati dannosi come documenti Microsoft Word con all'interno macro che permettono di installare Trojan come BazarLoader e Trickbot.

In alcuni casi, cercano di sfruttare firewall vulnerabili o prendere di mira qualsiasi server **RDP** (*Remote Desktop Protocol*) con connessione Internet, al fine di ottenere l'accesso alla rete.

Il ransomware può diffondersi anche attraverso il protocollo **SMB - Server Message Block**.

### Privilege Escalation

Dopo aver stabilito l'accesso alla rete vittima, l'obiettivo degli aggressori è quello di accedere ad un account di *amministratore* di dominio o ad altri account privilegiati, in modo da eseguire il codice con i privilegi necessari e da ottenere altre informazioni utili.

### Reconnaissance

La rete locale di cui fa parte la macchina infetta viene scansionata alla ricerca di server, endpoint, backup, dati sensibili. L'obiettivo principale del malware è il Domain Controller della rete.

### Getting the Credentials

A tal proposito vengono utilizzati tool come *Mimikatz* che realizzano un dump delle credenziali dalla memoria della macchina vittima.

### Backdoors

Anche in questo caso gli attaccanti tentano di installare una *Bazaar backdoor* sulla macchina vittima in modo da connetterla al server command-and-control di Conti.

### Data Harvesting

Prima di eseguire il codice ransomware, gli aggressori mireranno a rubare quanti più dati business-critical possibili. Gli attaccanti in genere salvano questi dati sul proprio server, li trasmettono tramite e-mail o li caricano su uno o più contenitori di archiviazione cloud anonimi.

### Ransomware Deployment

L'attacco sarà lanciato (i dati sul dispositivo vittima verranno cifrati) solo dopo aver esfiltrato gran parte dei dati utili, eliminato/cifrato ogni loro eventuale backup e disattivato tutte le possibili misure di sicurezza.

Una volta eseguito sull'endpoint della vittima, Conti effettua tali operazioni:

- Cripta immediatamente i file e modifica l'estensione dei file crittografati.
- Tenta di connettersi ad altri computer sulla stessa sottorete sfruttando la vulnerabilità **EternalBlue** (CVE-2017-0144) di **SMB** (*Microsoft Windows Server Message Block*) (porta 445).
- Lascia una richiesta di riscatto in ogni cartella: *readme.txt/conti\_readme.txt*

Esso adotta una "**doppia estorsione**", ovvero non solo cripta i dati delle vittime e richiede il pagamento, ma recupera anche una copia dei dati, che gli aggressori esporranno o venderanno se la vittima si rifiuta di pagare.

## Capitolo 2

# MITRE ATT&CK

Il **MITRE** ha creato il framework *ATT&CK* con il fine di documentare tattiche, tecniche e procedure comuni che fanno parte di minacce persistenti avanzate contro le organizzazioni. Tale modello si è diffuso in vari settori come mezzo per creare un modello comune di tassonomia e relazioni per sistemi defender per comprendere ed ideare meccanismi di protezione dalle attività di attacco e da comportamenti avversi.

Esso utilizza un linguaggio comune per rendere comprensibili i diversi comportamenti malevoli dei vari malware per la community di cybersecurity. Grazie a ciò si ottiene un maggior rigore e dettaglio per comprendere le possibili minacce sfruttando i diversi strumenti offerti all'organizzazione.

La **piramide del dolore** spiega come integrare le valutazioni del MITRE con altri indicatori tipici (**IOCs**) che permettono di misurare un potenziale threat in base agli incidenti e alla ricerca delle minacce.

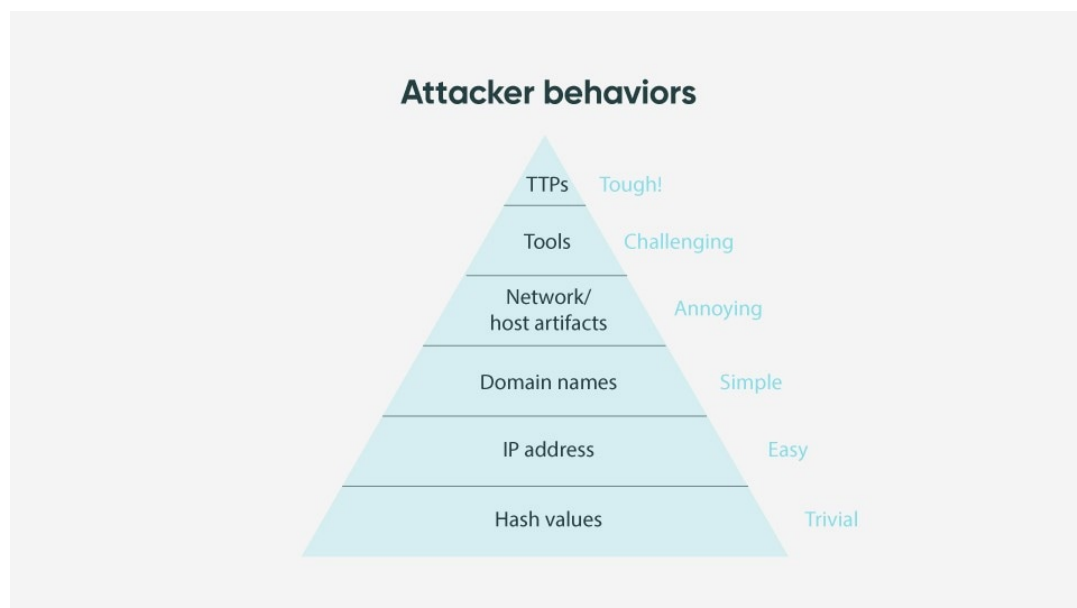


Figura 2.1: Piramide del dolore

Il framework proposto dal MITRE propone una descrizione dettagliata delle procedure precedentemente nominate nella Piramide del dolore (TTPs) adottate dall'attaccante per portare a termine un attacco. In particolare fa una distinzione tra **tecniche** e **tattiche**. Esse riguardano l'intero processo in base al quale gli utenti malintenzionati portano a compimento la loro missione, dall'inizio della fase di ricerca fino all'estrapolazione dei dati, comprendendo tutte le operazioni intermedie.

Le tattiche costituiscono un'idea generale del perché un utente malintenzionato stia eseguendo un'azione, mentre le tecniche sono le azioni a supporto della tattica.

Per il framework ATT&CK ci sono 12+2 tattiche da raggiungere tramite le tecniche. Esse sono classificate come segue:

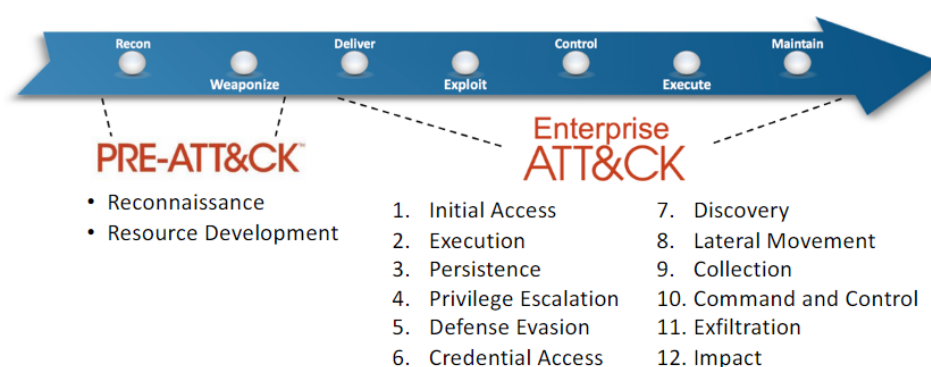


Figura 2.2: Enterprise ATT&CK

All'interno di ciascuna tattica è contenuta una serie di tecniche utilizzate da malware o gruppi di minacce nel corso della compromissione di un obiettivo e del raggiungimento dei propri scopi. Ogni tattica è, dunque, composta da una serie di tecniche e sotto-tecniche che indicano specifici comportamenti da parte di un attore. Inoltre, tali tecniche sono descritte da delle procedure.

La matrice delle TTPs permette di osservare: mitigazioni, sorgenti dati e metodi di rilevamento.



## 2.1 MITRE ATT&CK sul malware Conti

Si valutino ora le diverse tecniche e tattiche utilizzare sul malware Conti.

[illegible]

Figura 2.3: MITRE ATT&amp;CK Navigator

## Execution

L'*execution* consiste in tecniche che riguardano codice immesso dall'attaccante ed eseguito su sistema locale o remoto. Tali riguardano scopi come il raggiungimento della rete e il furto di dati:

- **Command and Scripting Interpreter: Windows Command Shell:** utilizza opzioni della linea di comando per permettere a un attaccante il controllo sulla scansione e la codifica dei file. I tool *Virus Total* e *Hybrid Analysis* riportano l'utilizzo del comando "*cmd*" molteplici volte;
- **Native API:** richiama le API durante l'esecuzione.

## Privilege escalation

Il *privilege escalation* consiste in tecniche che gli attaccanti utilizzano per ottenere autorizzazioni di livello superiore su un sistema o una rete:

- **Process Injection: Dynamic-link Library Injection:** carica DLL nei processi al fine di eludere le difese o possibilmente elevare i privilegi nel sistema. Tramite il tool *Hybrid Analysis* è possibile notare che tra i "*malicious indicators*", ne troviamo

uno in cui viene evidenziata *l'allocazione di memoria virtuale in un processo remoto*. Probabilmente è realizzata per mezzo della procedura *VirtualAlloc*, individuata negli import dell'eseguibile e descritta in seguito.

Installation/Persistence	
Allocates virtual memory in a remote process	
details	"conti_v2.exe" allocated memory in "%WINDIR%\System32\pnprnsp.dll"
source	API Call
relevance	7/10
ATT&CK ID	T1055.012 (Show technique in the MITRE ATT&CK™ matrix)

Figura 2.4: DLL injection

## Defense Evasion

La *defense evasion* consiste in tecniche che gli attaccanti utilizzano per evitare rilevamento. Tali tecniche includono la disinstallazione/disabilitazione dei software di sicurezza o l'offuscamento dei dati:

- **Obfuscated Files or Information:** utilizza meccanismi di offuscamento per il codice, le DLL e le chiamate alle API di Windows;
- **Deobfuscate/Decode Files or Information:** decripta le informazioni offuscate.

## Discovery

La *discovery* consiste in tecniche che gli attaccanti possono utilizzare per acquisire conoscenze sul sistema e sulla rete interna. Tali tecniche permettono di osservare l'ambiente e orientarsi per decidere come agire:

- **Network Share Discovery:** può enumerare condivisioni di rete SMB;
- **Process Discovery:** può enumerare i processi aperti;
- **Remote System Discovery:** capacità di enumerare gli host presenti sulla rete vittima;
- **System Network Configuration Discovery:** può recuperare la cache ARP dal sistema locale e verificare che gli indirizzi IP a cui si connette siano in locale;
- **System Network Connections Discovery:** può enumerare le connessioni di rete di un sistema compromesso.

## Lateral Movement

Il *lateral movement* consiste in tecniche che gli attaccanti utilizzano per entrare e ottenere il controllo remoto di sistemi in una rete:

- **Remote Services: SMB/Windows Admin Shares:** può diffondersi tramite SMB e crittografare i file su host diversi, compromettendo potenzialmente un'intera rete;
- **Taint Shared Content:** può diffondersi infettando altre macchine remote tramite unità di rete condivise.

## Impact

L'*impact* consiste in tecniche che gli attaccanti utilizzano per compromettere la disponibilità o l'integrità manipolando i diversi processi:

- **Data Encrypted for Impact:** utilizza un algoritmo crittografico per crittografare rapidamente i file;
- **Inhibit System Recovery:** può cancellare le Windows volume shadow copies<sup>1</sup>. Tramite i tool *Virus Total* e *Hybrid Analysis*, la linea di comando viene utilizzata per poter eliminare le shadow copies;

Analysed 9 processes in total (System Resource Monitor).

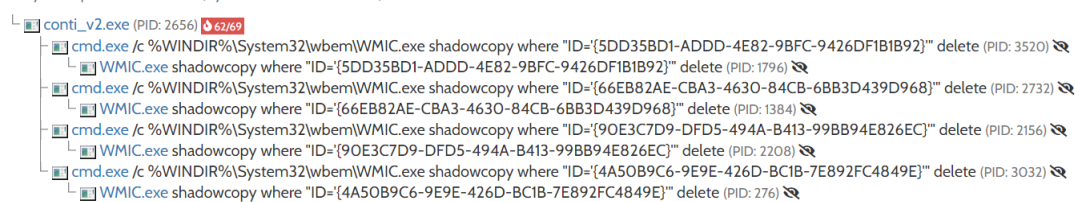


Figura 2.5: Cancellazione shadow copies

- **Service Stop:** può bloccare servizi Windows relativi a sicurezza, backup e database.

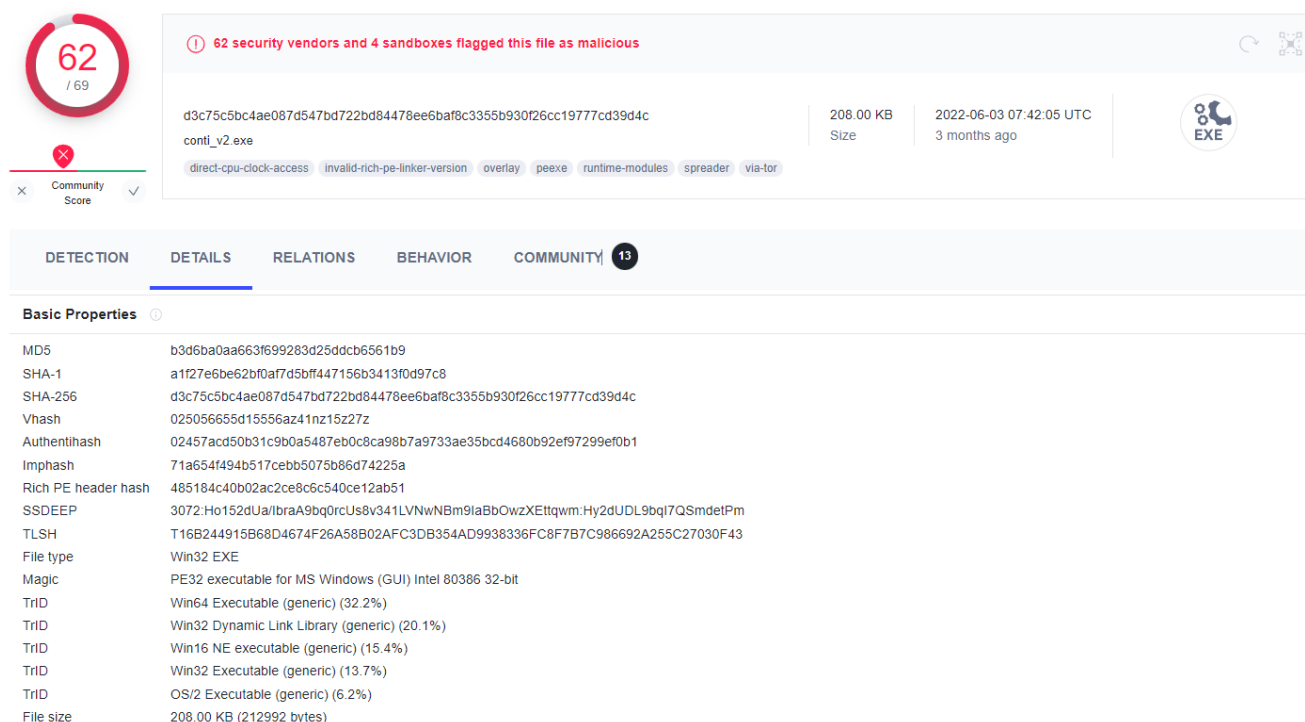
<sup>1</sup> Permette la creazione manuale o automatica di copie di backup di un file, di una cartella o di uno specifico volume ad un dato momento di tempo

## Capitolo 3

# Malware Analysis

Per la *malware analysis* è stato selezionato l'eseguibile (campione E): **conti\_v2.exe**.

Un'analisi preliminare può essere ottenuta mediante il tool **VirusTotal**<sup>1</sup>, il quale riporta una serie di informazioni utili come, ad esempio, il calcolo dell'**hash** (MD5, SHA-1 e SHA-256).



The screenshot shows the VirusTotal interface for the file **conti\_v2.exe**. At the top, a red circle indicates a score of 62/69. A warning message states: "62 security vendors and 4 sandboxes flagged this file as malicious". The file's MD5 hash is `d3c75c5bc4ae087d547bd722bd84478ee6baf8c3355b930f26cc19777cd39d4c`, and its size is 208.00 KB. The upload date is 2022-06-03 07:42:05 UTC, 3 months ago. The file is categorized as a PE32 executable for MS Windows (GUI) Intel 80386 32-bit. The 'Basic Properties' section lists various hashes and file details.

Property	Value
MD5	b3d6ba0aa663f699283d25ddcb6561b9
SHA-1	a1f27e6be2bf0af7d5bffa47156b3413f0d97c8
SHA-256	d3c75c5bc4ae087d547bd722bd84478ee6baf8c3355b930f26cc19777cd39d4c
Vhash	025056655d15556az41nz15z27z
Authentihash	02457acd50b31c9b0a5487eb0c8ca98b7a9733ae35bcd4680b92ef97299ef0b1
Imphash	71a654f494b517cebb5075b86d74225a
Rich PE header hash	485184c40b02ac2ce8c6c540ce12ab51
SSDEEP	3072:Ho152dUaIbraA9bq0rcUs8v341LVNwNBm9IaBbOwzXEtqwm:Hy2dUDL9bqI7QSmdeIPm
TLSH	T16B244915B68D4674F26A58B02AFC3DB354AD9938336FC8F7B7C986692A255C27030F43
File type	Win32 EXE
Magic	PE32 executable for MS Windows (GUI) Intel 80386 32-bit
TrID	Win64 Executable (generic) (32.2%)
TrID	Win32 Dynamic Link Library (generic) (20.1%)
TrID	Win16 NE executable (generic) (15.4%)
TrID	Win32 Executable (generic) (13.7%)
TrID	OS/2 Executable (generic) (6.2%)
File size	208.00 KB (212992 bytes)

Figura 3.1: Scansione con Virus Total

Un aspetto importante da sottolineare è che 62 fornitori di servizi di sicurezza (sui 69 totali) hanno riconosciuto nell'eseguibile un malware.

Di seguito sono riportati, invece, i nomi con cui il malware è stato distribuito e alcuni dati rilevanti sul suo studio.

<sup>1</sup><https://www.virustotal.com/gui/file/d3c75c5bc4ae087d547bd722bd84478ee6baf8c3355b930f26cc19777cd39d4c/details>

History ⓘ	
Creation Time	2020-11-06 16:33:28 UTC
First Seen In The Wild	2022-09-20 00:28:29 UTC
First Submission	2020-12-13 16:23:51 UTC
Last Submission	2022-09-20 07:36:02 UTC
Last Analysis	2022-06-03 07:42:05 UTC

Names ⓘ	
conti_v2.exe	
d3c75c5bc4ae087d547bd722bd84478ee6baf8c3355b930f26cc19777cd39d4c.exe	
Test.exe	
test.exe	
d3c75c5bc4ae087d547bd722bd84478ee6baf8c3355b930f26cc19777cd39d4c.sample	

Figura 3.2: History and Names

Nella scheda **Relations** troviamo un resoconto dei domini e indirizzi IP che il malware tenta di contattare, in più, un elenco dei file droppati sulla macchina vittima.

Infine, nella scheda **Behavior** vengono fornite le tecniche e le tattiche, secondo il framework **MITRE** utilizzate dal malware e alcune delle operazioni svolte.

### 3.1 Basic Static Analysis

La **Basic Static Analysis** consiste nell'esaminare il file eseguibile senza visualizzare le istruzioni effettive. Questa può confermare se un file è dannoso, fornire informazioni sulla sua funzionalità e talvolta consentire di produrre semplici firme di rete. La Basic Static Analysis è semplice e può essere rapida, ma è in gran parte inefficace contro malware sofisticati e può non rilevare comportamenti importanti.

L'obiettivo è quello di individuare la presenza di **stringhe**, **librerie** e **procedure**, le quali potrebbero rivelare un possibile comportamento malevolo.

Un primo indicatore sospetto è immediatamente rilevato per mezzo del tool **PEstudio**. Esso è la *file-checksum* che risulta essere pari a zero e non coincide con quella reale.

file-checksum	0x00000000
real-checksum	0x0003A4B5

Figura 3.3: Checksum in PEstudio

#### Malware unpacked

Quando utilizziamo un tool per il prelievo delle stringhe, alcune di queste potrebbero essere nascoste. In tal caso il malware prende il nome di **packed**.

Il tool **PEiD** riceve in ingresso l'eseguibile ed analizzandolo verifica se è presente o meno un **wrapper** conosciuto per la decompressione del contenuto.

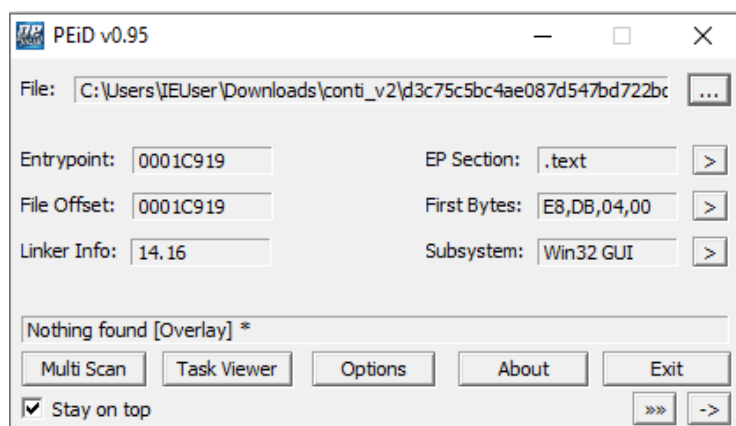


Figura 3.4: PEiD

Dato che l'eseguibile *conti\_v2.exe* è di tipo unpacked, PEiD effettivamente non riscontra la presenza di un wrapper.

È possibile, analogamente, confrontare per ognuna delle sezioni dell'eseguibile le seguenti informazioni:

- **Virtual Size**: rappresenta la dimensione occupata dalla sezione in RAM.
- **Raw Size**: rappresenta la dimensione occupata dalla sezione sul disco.

Se questi due valori sono molto diversi tra loro molto probabilmente il malware è di tipo packed. Queste informazioni vengono rilasciate dal tool **PEview**, ma similmente anche da VirusTotal:

Sections						
Name	Virtual Address	Virtual Size	Raw Size	Entropy	MD5	Chi2
.text	4096	159744	157696	6.5	6b69c80c858978885884c22b7c69d476	1011770.06
.rdata	163840	24576	24064	4.88	e3400bda3f796ab4b0db6d547f3fb0d4	1208685.5
.data	188416	12288	8704	2.42	f20e215f537fc836840b4cd20bae99ef	1285622.88
.rsrc	200704	4096	512	4.72	ad7b78e84f1d02fc883315380c423021	9292
.reloc	204800	4608	4608	6.35	3293556c0310d7bda0acc56d08d1bb89	29282.3

Figura 3.5: Virtual Size vs Raw Size

Dall'immagine possiamo ravvisare la presenza di un ulteriore parametro: l'**entropia**. Essa è una misura della casualità all'interno di un insieme di dati e spesso si fa riferimento all'**algoritmo di Shannon**. Questo restituisce un valore compreso tra 0 e 8, dove valori prossimi a 8 indicano che i dati sono molto casuali, mentre valori vicini a 0 indicano che i

dati sono molto omogenei.

I file legittimi tendono ad avere un'entropia compresa tra 4.8 e 7.2, i restanti, invece, tendono ad essere dannosi.

Possiamo, quindi, dedurre che:

- **.text, .rdata** sono sezioni *unpacked* poichè VirtualSize e RawSize coincidono e l'entropia non è indice di codice offuscato.
- **.data, .rsrc** a nostro parere presentano differenze significative tra le dimensioni sul disco e in RAM, dunque molto probabilmente sono compressi. Si è pensato che tale differenza sia dovuta all'offuscamento di alcune API all'interno dell'eseguibile. L'entropia risulta in ogni caso essere nei limiti quindi non si possono effettuare valutazioni dettagliate su di essa.

## DLL (Dynamic Link Libraries)

Per individuare le librerie, procedure e stringhe usate è stato impiegato il tool **PEstudio**:

c:\users\ieuser\downloads\conti\_v2\d3c75c5bc4

- indicators (33)
- virustotal (error)
- dos-header (64 bytes)
- dos-stub (208 bytes)
- rich-header (Visual Studio)
- file-header (Nov.2020)
- optional-header (GUI)
- directories (6)
- sections (91.83%)
- libraries (3) \***
- functions (68)
- exports (n/a)
- tls-callbacks (n/a)
- .NET (n/a)
- resources (manifest)
- strings (2470)
- debug (Nov.2020)
- manifest (asInvoker)
- version (n/a)
- overlay (unknown)

library (3)	blacklist (1)	type (1)	functions (68)	description
kernel32.dll	-	implicit	<u>65</u>	Windows NT BASE API Client DLL
user32.dll	-	implicit	<u>1</u>	Multi-User Windows USER API Client DLL
ws2_32.dll	x	implicit	<u>2</u>	Windows Socket 2.0 32-Bit DLL

Figura 3.6: PEstudio: librerie

Le librerie importate visibili sono:

- **Kernel32.dll**: questa è una DLL molto comune che contiene funzionalità di base, come l'accesso e la manipolazione di memoria, file e hardware.
- **User32.dll**: questa DLL contiene tutti i componenti dell'interfaccia utente, come pulsanti, barre di scorrimento e componenti per il controllo e la risposta alle azioni dell'utente.
- **Ws2 32.dll**: questa è una DLL di rete.

*Ws2\_32.dll* è in blacklist poiché il malware che la utilizza potrebbe connettersi alla rete o eseguire attività legate alla rete. *User32.dll* e *Kernel32.dll*, invece, sono DLL molto usate,

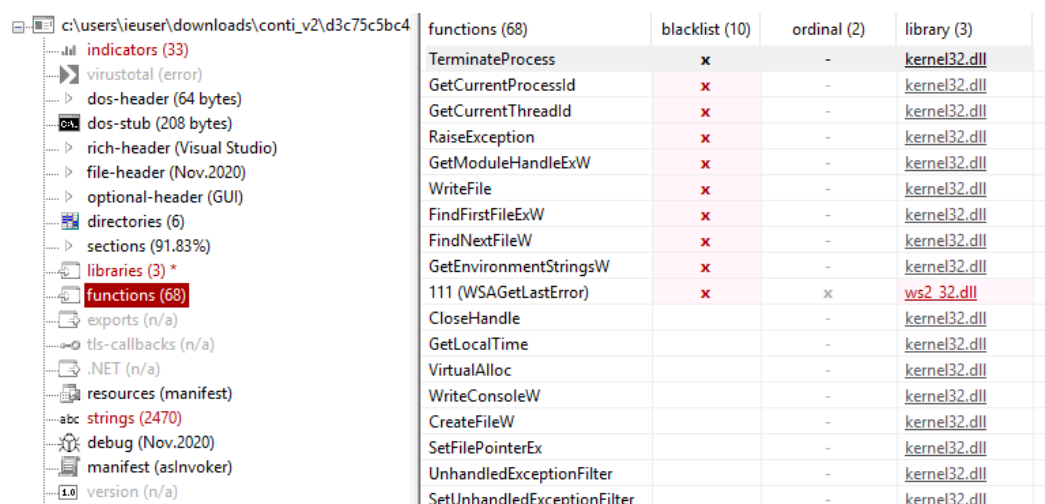
quindi, di per sé non rappresentano un problema, ma lo potrebbero diventare in base alle procedure importate.

Nonostante ciò, come abbiamo accennato precedentemente e vedremo meglio in seguito, Conti richiama molte più DLL a runtime tramite offuscamento dei nomi.

Con il tool **Dependency Walker** si ottiene una visione più completa delle DLL e delle relative funzioni importate ed esportate. Tuttavia, non è stato approfondito nel dettaglio poiché fornisce numerose informazioni complesse da ricostruire in questo punto dell'analisi.

## Procedure

Le procedure restituite da PEstudio inserite in blacklist sono:



functions (68)	blacklist (10)	ordinal (2)	library (3)
TerminateProcess	x	-	kernel32.dll
GetCurrentProcessId	x	-	kernel32.dll
GetCurrentThreadId	x	-	kernel32.dll
RaiseException	x	-	kernel32.dll
GetModuleHandleExW	x	-	kernel32.dll
WriteFile	x	-	kernel32.dll
FindFirstFileExW	x	-	kernel32.dll
FindNextFileW	x	-	kernel32.dll
GetEnvironmentStringsW	x	-	kernel32.dll
111 (WSAGetLastError)	x	x	ws2_32.dll
CloseHandle		-	kernel32.dll
GetLocalTime		-	kernel32.dll
VirtualAlloc		-	kernel32.dll
WriteConsoleW		-	kernel32.dll
CreateFileW		-	kernel32.dll
SetFilePointerEx		-	kernel32.dll
UnhandledExceptionFilter		-	kernel32.dll
SetUnhandledExceptionFilter		-	kernel32.dll

Figura 3.7: PEstudio: procedure

- **WriteFile (Kernel32.dll)**: scrive i dati nel file o nel dispositivo di input/output (I/O) specificato.
- **TerminateProcess (Kernel32.dll)**: termina il processo specificato e tutti i relativi thread.
- **RaiseException (Kernel32.dll)**: solleva un'eccezione nel thread chiamante.
- **GetModuleHandleExW (Kernel32.dll)**: recupera un handle<sup>2</sup> del modulo per il modulo specificato e incrementa il conteggio dei riferimenti del modulo a meno che non sia specificato *GET\_MODULE\_HANDLE\_EX\_FLAG\_UNCHANGED\_REFCOUNT*. Il modulo deve essere stato caricato dal processo chiamante.
- **GetEnvironmentStringsW (Kernel32.dll)**: recupera le variabili di ambiente per il processo corrente.

<sup>2</sup>Puntatore agli oggetti del Sistema Operativo, è possibile usarlo, ad esempio, per capire i programmi quali file o directory hanno aperto.



- **GetCurrentThreadId (Kernel32.dll)**: recupera l'identificatore del thread chiamante.
- **GetCurrentProcessId (Kernel32.dll)**: recupera l'identificatore del processo chiamante.
- **FindFirstFileExW (Kernel32.dll)**: cerca in una directory un file o una sottodirectory con un nome e attributi corrispondenti a quelli specificati. Se la funzione ha esito positivo, il valore restituito è un handle di ricerca utilizzato in una successiva chiamata a *FindNextFile* e il parametro *lpFindFileData* contiene informazioni sul primo file o directory trovata.
- **FindNextFileExW (Kernel32.dll)**: continua la ricerca di file da una chiamata precedente, ovvero *FindFirstFileExW*. Se la funzione ha esito positivo, il valore restituito è diverso da zero e il parametro *lpFindFileData* contiene informazioni sul file o sulla directory successiva trovata. Se la funzione ha esito negativo, il valore restituito è zero e il contenuto di *lpFindFileData* è indeterminato.
- **WSAGetLastError (Ws2\_32.dll)**: restituisce lo stato di errore per l'ultima operazione Windows Sockets non riuscita.
- **htons (Ws2\_32.dll)**: converte un indirizzo di rete dal formato Intel *little-endian* a quello di rete TCP/IP *big-endian*.

Osservando queste funzioni descritte possiamo dedurre le seguenti ipotesi:

- Scrive/modifica i file.
- Carica dinamicamente delle DLL.
- Utilizza il multi-threading.
- Naviga nel file system.
- Tenta di connettersi alla rete.

Altre procedure che non sono state segnalate dal tool ma che potrebbero rivelare il funzionamento del malware sono:

- **GetFileType (Kernel32.dll)**: recupera il tipo di file del file specificato.
- **ExitProcess (Kernel32.dll)**: termina il processo chiamante e tutti i suoi thread.
- **FreeLibrary (Kernel32.dll)**: libera il modulo della DLL caricata e, se necessario, ne diminuisce il conteggio dei riferimenti.
- **VirtualAlloc (Kernel32.dll)**: alloca, esegue il commit o modifica lo stato di un'area di pagine nello spazio degli indirizzi virtuali del processo chiamante (DLL injection).

- **GetCurrentProcess (Kernel32.dll)**: recupera uno pseudo handle per il processo corrente.
- **IsDebuggerPresent (Kernel32.dll)**: determina se il processo chiamante viene sottoposto a debug da un debugger in modalità utente.
- **CreateFileW (Kernel32.dll)**: crea o apre un file o un dispositivo I/O. I dispositivi I/O più comunemente usati sono i seguenti: file, flusso di file, directory, disco fisico, volume, buffer della console ecc..
- **WriteConsoleW (Kernel32.dll)**: scrive una stringa di caratteri in un buffer dello schermo della console iniziando dalla posizione corrente del cursore.
- **GetConsoleMode (Kernel32.dll)**: recupera la modalità di input corrente del buffer della console o la modalità di output corrente di un buffer dello schermo della console.
- **GetConsoleCP (Kernel32.dll)**: recupera la tabella codici di input utilizzata dalla console associata al processo chiamante. Una console usa la tabella codici di input per convertire l'input da tastiera nel valore del carattere corrispondente.

Le ultime quattro funzioni ci permettono di dedurre che Conti accede e scrive sulla console della macchina vittima. Infine evidenziamo la presenza dell'implementazione di un meccanismo che rileva possibili attività di debugging dell'eseguibile.

Per gestire l'esecuzione in parallelo dei thread:

- **EnterCriticalSection (Kernel32.dll)**
- **LeaveCriticalSection (Kernel32.dll)**
- **DeleteCriticalSection (Kernel32.dll)**
- **InitializeCriticalSectionAndSpinCount (Kernel32.dll)**: inizializza una sezione critica e imposta un numero di giri. Quando un thread tenta di acquisire una sezione critica che è bloccata, il thread "gira": entra in un ciclo che itera i tempi di conteggio degli spin, controllando se il blocco è stato rilasciato. Se il blocco non viene rilasciato prima del termine del ciclo, il thread passa alla modalità di sospensione per attendere il rilascio del blocco.

Per importare le DLL:

- **GetModuleFileNameW (Kernel.dll)**: recupera il percorso completo del file corrispondente al modulo specificato.
- **LoadLibraryExW (Kernel32.dll)**: carica il modulo specificato nello spazio degli indirizzi del processo chiamante. Il modulo specificato può causare il caricamento di altri moduli.

- **GetProcAddress (Kernel32.dll)**: recupera l'indirizzo di una procedura esportata o di una variabile della DLL specificata.

## Stringhe

PEiD ha individuato nell'eseguibile le seguenti stringhe:

	encoding (2)	size (bytes)	location	blacklist (9)	hint (66)	value (2470)
indicators (34)	ascii	4	0x001067C	-	utility	te k
virustotal (62/69)	ascii	16	0x0002CBB8	-	utility	expand 32-byte k
dos-header (64 bytes)	ascii	16	0x0002CBCC	-	utility	expand 16-byte k
dos-stub (208 bytes)	ascii	19	0x000300C8	-	rtti	.?AVbad_alloc@std@@
rich-header (Visual Studio)	ascii	19	0x000300E4	-	rtti	.?AVexception@std@@
file-header (Nov.2020)	ascii	21	0x00030100	-	rtti	.?AVlogic_error@std@@
optional-header (GUI)	ascii	22	0x00030120	-	rtti	.?AVlength_error@std@@
directories (6)	ascii	15	0x00030140	-	rtti	.?AVtype_info@@
sections (91.83%)	ascii	30	0x00030158	-	rtti	.?AVbad_array_new_length@std@@
libraries (3) *	ascii	11	0x0002D852	-	function	CloseHandle
functions (68)	ascii	12	0x0002D860	-	function	GetLocalTime
exports (n/a)	ascii	12	0x0002D870	-	function	VirtualAlloc
tls-callbacks (n/a)	ascii	24	0x0002D8B2	-	function	UnhandledExceptionFilter
.NET (n/a)	ascii	27	0x0002D8CE	-	function	SetUnhandledExceptionFilter
resources (manifest)	ascii	17	0x0002D8EC	-	function	GetCurrentProcess
strings (2470)	ascii	16	0x0002D900	x	function	TerminateProcess
debug (Nov.2020)	ascii	25	0x0002D914	-	function	IsProcessorFeaturePresent
manifest (aslnvoker)	ascii	17	0x0002D930	-	function	IsDebuggerPresent
version (n/a)	ascii	23	0x0002D96A	-	function	QueryPerformanceCounter
overlay (unknown)						

Figura 3.8: PEstudio: stringhe

- **Ntdll**: questa DLL è l'interfaccia per il kernel di Windows. Gli eseguibili generalmente non importano direttamente questo file, sebbene sia sempre importato indirettamente da *Kernel32.dll*.
- **A : \source\conti\_v3\Release\cryptor.pdb**: PDB è un'estensione che fa riferimento a un file database e viene generalmente installato con l'applicazione corrispondente. I file PDB utilizzati da diversi programmi di solito vengono salvati in un formato proprietario. Visual Studio utilizza i file PDB per archiviare le informazioni di debug su un programma. Nel nostro caso, il malware è stato compilato in Visual Studio, di conseguenza questo ci fa presupporre che al suo interno sono presenti informazioni di debug.
- **Mscoree.dll**: offre la possibilità di collegare informazioni, sistemi, persone e dispositivi tramite software. Il file mscoree.dll è un Microsoft Runtime Execution Engine, in altre parole contiene le funzioni fondamentali del framework Microsoft.NET.
- **Advapi32.dll**: fa parte della libreria dei servizi di API avanzati. Fornisce l'accesso a funzionalità avanzate che vengono fornite in aggiunta al kernel.
- **AppPolicyGetProcessTerminationMethod (Api-ms-win-appmodel-runtime-l1-1-2.dll)**: recupera il metodo utilizzato per terminare un processo.
- **InitalizeCriticalSectionEx (Kernel32.dll)**: inizializza una sezione critica con un numero di giri e flag facoltativi.

Come accennato precedentemente, Conti fa uso dell'algoritmo ChaCha8 per criptare i file presenti sul disco dell'host vittima. La chiave ChaCha8 è diversa per ogni file e viene criptata utilizzando una **chiave pubblica RSA** per poi essere inserita alla fine del file:

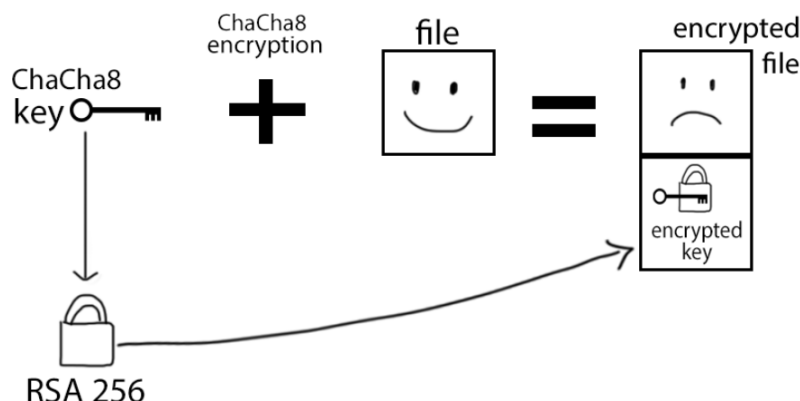


Figura 3.9: Metodo di crittografia di Conti

Gli sviluppatori per recuperare le chiavi ChaCha8 utilizzano la corrispondente **chiave privata RSA**.

Attraverso il tool **PEview** è possibile visualizzare il contenuto della sezione **data**, al suo interno sono definiti tutti i dati globali tra cui la chiave pubblica RSA:

	pFile	Raw Data	Value
IMAGE_DOS_HEADER	0002F0A0	06 02 00 00 00 A4 00 00 52 53 41 31 00 10 00 00	.....RSA1.....
IMAGE_DEBUG_TYPE	0002F0B0	01 00 01 00 79 64 71 C8 DB FA 45 5D 8F 95 EE D8	.....y dq...E].....
MS-DOS Stub Program	0002F0C0	22 11 59 B2 A6 35 58 3A B0 E3 B6 FD 48 FF 4E B3	"..Y...5X:...H.N.
IMAGE_NT_HEADERS	0002F0D0	1D 55 C8 3F 79 86 00 2A 93 34 2A 25 3A 7E 0F A7	.U.?y...*.4*%::~..
IMAGE_SECTION_HEADER	0002F0E0	03 01 90 B8 9F BD FB 47 F9 44 A5 25 7E 95 21 5C	.....G.D.%~.!\\
IMAGE_SECTION_HEADER	0002F0F0	6E D4 42 A6 D2 B5 06 21 4A F1 E7 71 4B B1 53 F3	n.B.....J...qK.S.
IMAGE_SECTION_HEADER	0002F100	7F 31 AB 1E 57 9D DD F6 1B 1B 64 72 58 04 B1 D0	1..W.....drX...
IMAGE_SECTION_HEADER	0002F110	2D 59 E0 C4 3A 62 FA 25 6C F0 74 5F 2C 1E D5 B8	-Y...:b.%!t_....
IMAGE_SECTION_HEADER	0002F120	14 96 BE 1E A9 96 2D 64 6B 3B 0B 5E ED 9D D4 09	.....-dk;..^.....
SECTION .text	0002F130	C9 B5 4E 09 50 1C 14 D1 3B 2E 6D 99 50 D5 1B E6	.....N.P.....m.P...
SECTION .rdata	0002F140	8D BC 60 86 6B CD 61 D6 B4 43 B9 10 39 8E C8 0F	.....k.a...C...9...
SECTION .data	0002F150	29 0A 99 64 AF 10 2D 58 25 A2 1B C9 1C 91 24 07	)...d...-X%.....\$.
SECTION .rsrc	0002F160	7E B6 DF A4 13 A2 9A F3 2B CA F3 2C 4A 26 AE C1	~...s...+...J&...
SECTION .reloc	0002F170	F9 00 A7 87 A5 EB EF E2 55 49 AA AA 4B E3 C8 8A	.....UI...K...
	0002F180	CE 78 C9 DA 43 76 60 13 E2 D2 6E F3 36 3E 76 D6	.x...Cv^...n.6>v...
	0002F190	58 52 16 11 CE E1 61 0C ED 54 08 7E 57 09 E5 99	XR...a...T...~W...
	0002F1A0	CA 70 80 FA 17 96 B2 0A F2 A4 16 78 DC 63 CF 88	.p.....x...c...
	0002F1B0	0D 61 DF E6 3F F9 D5 1B CB 84 87 52 1E 06 5D 42	.a...?.....R...]B
	0002F1C0	FC 2D B9 69 BC 05 FD FC 20 5C 65 CF 5E 11 68 57	...i.....\e...hW
	0002F1D0	94 91 CE 29 82 9F 8A 6D 7E 49 AA 1D 8D C8 19 2C	...m~l.....
	0002F1E0	B2 44 0E 23 A1 FA 09 AC 6D 8F C7 B2 47 3D 66 35	.D.#.....m...G=f5
	0002F1F0	B9 5E A9 F9 48 99 11 3C 94 1D 20 22 CF B4 E8 68	^...H...<..."....h
	0002F200	BD 94 57 89 64 76 FA 77 C3 89 54 C4 F7 63 4D 5B	..W.dv..w...T...cM[
	0002F210	28 FC B6 C5 09 6E 6A 39 6D 7E 06 01 29 DC 7C A9	(...n j9m~...).l.
	0002F220	AA 75 4A C0 52 35 F4 9F 3D AA 1C C5 15 78 CD 8D	.uJ.R5...=...x...
	0002F230	76 53 F5 0F 03 03 5F EB EF 3C 82 33 8A 5B 38 7D	vS.....<...3.[8}
	0002F240	DD 70 CC 18 CE 55 14 3E F2 B6 A5 F0 6C 8B 27 DF	.p...U...>...l..'
	0002F250	8F 47 56 A0 98 29 B8 52 E4 60 4D 92 DC F9 C9 5D	.GV...).R.'M....]
	0002F260	C9 83 73 FD 5F 1E 78 38 D7 0C 5D 2B 59 29 78 E8	...s...x8...]+Y)x.
	0002F270	42 6B 23 3E 44 98 B2 5E 71 4F 3B 77 2E D3 4B 1E	Bk#>D...^qO;w...K.
	0002F280	3F D0 CE 74 EA 2B 57 72 79 65 B5 75 E9 F7 60 00	?...t.+Wrye.u...'
	0002F290	0C 32 CC 5B FE C3 DD C5 35 0F A2 AB DE B5 E5 1B	.2.[...5.....
	0002F2A0	1E 27 DC 95 1A AE 63 F9 A1 CC 66 36 04 A7 25 AB	'...c...f6...%.
	0002F2B0	E3 4C 35 C0 00 00 00 00 00 00 00 00 00 00 00	.L5.....

Figura 3.10: PEview: chiave RSA

## 3.2 Basic Dynamic Analysis

La **Basic Dynamic Analysis** è una tecnica nella quale si analizza il comportamento del malware a runtime. Per evitare che il malware si diffonda e causi danni sul sistema su cui è eseguito, adottiamo determinati passaggi per isolarlo.

La soluzione adottata è stata quella di eseguire il malware su una macchina virtuale, generando un'istanza della VM in modo tale da ripristinare lo stato della macchina prima dell'esecuzione il malware.

Al termine dell'esecuzione di *conti\_v2.exe*, ogni file sul sistema risulta crittografato ed in ogni directory del disco viene rilasciato un file di testo (*readme.txt*), nel quale viene descritto l'attacco e in più vengono forniti i contatti necessari per pagare il riscatto:

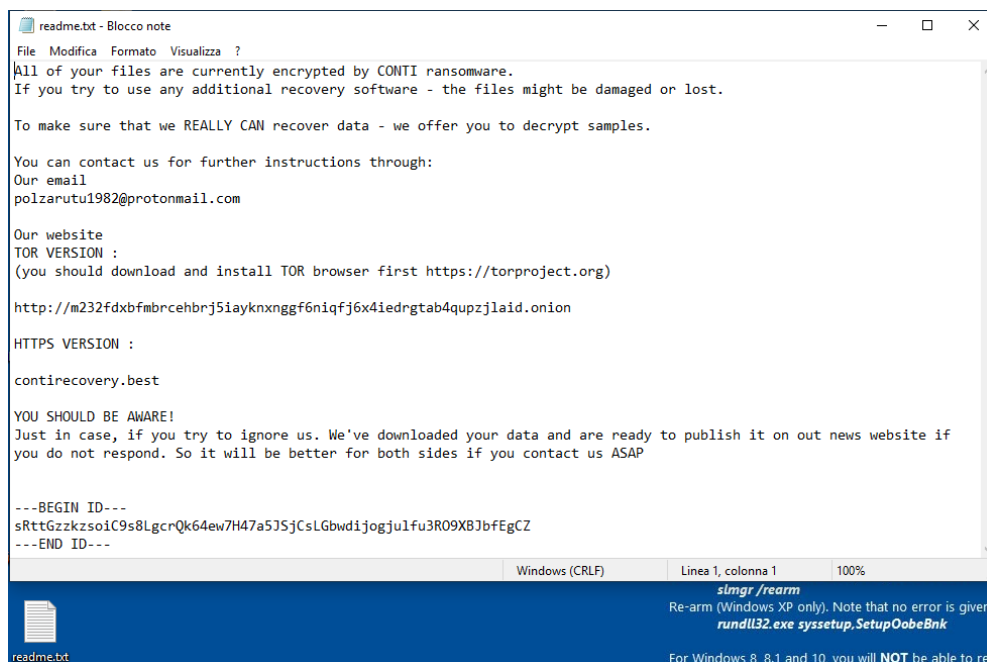


Figura 3.11: *readme.txt*

### Wireshark

**Wireshark** è un software di *packet sniffer* utilizzato per la soluzione di problemi di rete e per l'analisi.

Eseguendo il malware ed attivando lo sniffing del traffico di rete possiamo verificare il suo andamento:

22	0.082955	PcsCompu_0f:c7:c1	Broadcast	ARP	42 Who has 10.0.2.10? Tell 10.0.2.15
23	0.083171	PcsCompu_0f:c7:c1	Broadcast	ARP	42 Who has 10.0.2.11? Tell 10.0.2.15
24	0.083331	PcsCompu_0f:c7:c1	Broadcast	ARP	42 Who has 10.0.2.12? Tell 10.0.2.15
25	0.083567	PcsCompu_0f:c7:c1	Broadcast	ARP	42 Who has 10.0.2.13? Tell 10.0.2.15
26	0.083725	PcsCompu_0f:c7:c1	Broadcast	ARP	42 Who has 10.0.2.14? Tell 10.0.2.15
27	0.085113	PcsCompu_0f:c7:c1	Broadcast	ARP	42 Who has 10.0.2.16? Tell 10.0.2.15
28	0.085299	PcsCompu_0f:c7:c1	Broadcast	ARP	42 Who has 10.0.2.17? Tell 10.0.2.15
29	0.085564	PcsCompu_0f:c7:c1	Broadcast	ARP	42 Who has 10.0.2.18? Tell 10.0.2.15
30	0.085667	PcsCompu_0f:c7:c1	Broadcast	ARP	42 Who has 10.0.2.19? Tell 10.0.2.15
31	0.085796	PcsCompu_0f:c7:c1	Broadcast	ARP	42 Who has 10.0.2.20? Tell 10.0.2.15
32	0.085937	PcsCompu_0f:c7:c1	Broadcast	ARP	42 Who has 10.0.2.21? Tell 10.0.2.15
33	0.086062	PcsCompu_0f:c7:c1	Broadcast	ARP	42 Who has 10.0.2.22? Tell 10.0.2.15

```

> Frame 22: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface \Device\NPF_{08237AF4-7380-48D9-94C2-F05E6A29A5A6}, id 0
> Ethernet II, Src: PcsCompu_0f:c7:c1 (08:00:27:0f:c7:c1), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
v Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  Sender MAC address: PcsCompu_0f:c7:c1 (08:00:27:0f:c7:c1)
  Sender IP address: 10.0.2.15
  Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)
  Target IP address: 10.0.2.10

```

Figura 3.12: Richieste ARP

Come è possibile osservare, il malware comincia ad inviare richieste ARP in broadcast nella sottorete della macchina vittima con l'obiettivo di conoscere l'indirizzo MAC degli altri host che ne fanno parte. *PcsCompu\_0f:c7:c1* coincide con l'indirizzo MAC della scheda di rete dell'host su cui è stato eseguito il malware.

Esso parte dall'indirizzo IP dell'host vittima e tenta successivamente di contattare tutti i probabili host della sua stessa sottorete.

In più il malware tenta di diffondersi verso altri dispositivi sfruttando il protocollo SMB:

517	1.097099	10.0.2.15	10.0.2.2	TCP	66 1591 → 445 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
518	1.098045	10.0.2.2	10.0.2.15	TCP	60 445 → 1591 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
519	1.098108	10.0.2.15	10.0.2.2	TCP	54 1591 → 445 [ACK] Seq=1 Ack=1 Win=64240 Len=0
520	1.098329	10.0.2.15	10.0.2.2	SMB	127 Negotiate Protocol Request
521	1.098689	10.0.2.2	10.0.2.15	TCP	60 445 → 1591 [ACK] Seq=1 Ack=74 Win=65535 Len=0
522	1.099193	10.0.2.2	10.0.2.15	SMB2	592 Negotiate Protocol Response
523	1.099265	10.0.2.15	10.0.2.2	SMB2	232 Negotiate Protocol Request
524	1.099611	10.0.2.2	10.0.2.15	TCP	60 445 → 1591 [ACK] Seq=539 Ack=252 Win=65535 Len=0
525	1.100274	10.0.2.2	10.0.2.15	SMB2	654 Negotiate Protocol Response
526	1.126832	10.0.2.15	10.0.2.2	SMB2	220 Session Setup Request, NTLMSSP_NEGOTIATE

```

> Frame 520: 127 bytes on wire (1016 bits), 127 bytes captured (1016 bits) on interface \Device\NPF_{08237AF4-7380-48D9-94C2-F05E6A29A5A6}, id 0
> Ethernet II, Src: PcsCompu_0f:c7:c1 (08:00:27:0f:c7:c1), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
> Internet Protocol Version 4, Src: 10.0.2.15, Dst: 10.0.2.2
> Transmission Control Protocol, Src Port: 1591, Dst Port: 445, Seq: 1, Ack: 1, Len: 73
> NetBIOS Session Service
v SMB (Server Message Block Protocol)
  SMB Header
  Negotiate Protocol Request (0x72)

```

Figura 3.13: Protocollo SMB

SMB è un protocollo usato soprattutto dai sistemi Microsoft Windows, principalmente per condividere file, stampanti, porte seriali e comunicazioni di varia natura tra diversi nodi di una sottorete. Esso include anche un meccanismo di comunicazione autenticata tra processi.

## Regshot

**Regshot** è un tool che consente di eseguire uno snapshot del registro di sistema di Windows prima e dopo. In genere, viene utilizzato per acquisire un'istantanea del sistema prima dell'esecuzione del malware e subito dopo.

L'obiettivo è identificare eventuali modifiche al registro di sistema apportate dal malware, ciò può fornire maggiori indicazioni su ciò che è capace di fare.

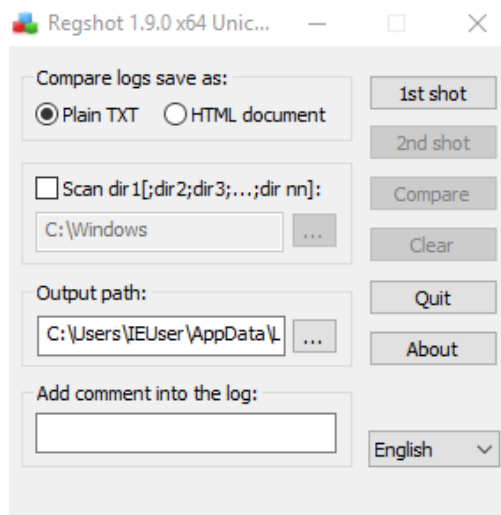


Figura 3.14: Regshot

Dopo aver eseguito i due shot è possibile confrontare i risultati ottenendo le seguenti informazioni:

- Eliminazione di 19833 chiavi.
- Aggiunta di 121 chiavi.
- Eliminazione di 23854 valori.
- Aggiunta di 1384 valori.
- Modifica di 193 valori.

Il report ottenuto contiene un numero elevato di informazioni. In linea massima possiamo affermare che Conti modifica le entry del registro di sistema di Windows per ottenere il controllo della macchina.

## Process Monitor

**Process Monitor** è un tool di monitoraggio avanzato per Windows che mostra in tempo reale file system, registro, processi/thread, rete e acquisisce eventi, ovvero le eventuali modifiche apportate su tali risorse. Tutti gli eventi registrati saranno salvati in RAM.

conti_v2.exe	2492	Process Start		SUCCESS
conti_v2.exe	2492	Thread Create		SUCCESS
conti_v2.exe	2492	Load Image	C:\Users\IEUser\Downloads\conti_v2\conti_v2.exe	SUCCESS
conti_v2.exe	2492	Load Image	C:\Windows\System32\ntdll.dll	SUCCESS
conti_v2.exe	2492	Load Image	C:\Windows\SysWOW64\ntdll.dll	SUCCESS
conti_v2.exe	2492	CreateFile	C:\Windows\Prefetch\CONTI_V2.EXE-52BFF74C.pf	NAME NOT FOUND

Figura 3.15: Caricamento di *ntdll.dll*

Non appena viene lanciato il processo *conti\_v2.exe* viene generato un thread il quale va a richiamare la libreria *ntdll.dll*.

Di seguito sono riportate tutte le operazioni effettuate dal malware sui file presenti sull'host vittima:

conti_v2.exe	2492	CreateFile	C:\metasploit-framework\embedded\mingw\bin\autoconf	SUCCESS
conti_v2.exe	2492	QueryBasicInformationFile	C:\metasploit-framework\embedded\mingw\bin\autoconf	SUCCESS
conti_v2.exe	2492	CloseFile	C:\metasploit-framework\embedded\mingw\bin\autoconf	SUCCESS
conti_v2.exe	2492	CreateFile	C:\metasploit-framework\embedded\mingw\bin\autoconf	SUCCESS
conti_v2.exe	2492	QueryStandardInformationFile	C:\metasploit-framework\embedded\mingw\bin\autoconf	SUCCESS
conti_v2.exe	2492	QueryStandardInformationFile	C:\metasploit-framework\embedded\mingw\bin\autoconf	SUCCESS
conti_v2.exe	2492	WriteFile	C:\metasploit-framework\embedded\mingw\bin\autoconf	SUCCESS
conti_v2.exe	2492	ReadFile	C:\metasploit-framework\embedded\mingw\bin\autoconf	SUCCESS
conti_v2.exe	2492	WriteFile	C:\metasploit-framework\embedded\mingw\bin\autoconf	SUCCESS
conti_v2.exe	2492	SetEndOfFileInformationFile	C:\metasploit-framework\embedded\mingw\bin\autoconf	SUCCESS
conti_v2.exe	2492	SetAllocationInformationFile	C:\metasploit-framework\embedded\mingw\bin\autoconf	SUCCESS
conti_v2.exe	2492	ReadFile	C:\metasploit-framework\embedded\mingw\bin\autoconf	SUCCESS
conti_v2.exe	2492	ReadFile	C:\metasploit-framework\embedded\mingw\bin\autoconf	SUCCESS
conti_v2.exe	2492	WriteFile	C:\metasploit-framework\embedded\mingw\bin\autoconf	SUCCESS
conti_v2.exe	2492	CloseFile	C:\metasploit-framework\embedded\mingw\bin\autoconf	SUCCESS
conti_v2.exe	2492	CreateFile	C:\metasploit-framework\embedded\mingw\bin\autoconf	SUCCESS
conti_v2.exe	2492	QueryAttributeTagFile	C:\metasploit-framework\embedded\mingw\bin\autoconf	SUCCESS
conti_v2.exe	2492	QueryBasicInformationFile	C:\metasploit-framework\embedded\mingw\bin\autoconf	SUCCESS
conti_v2.exe	2492	CreateFile	C:\metasploit-framework\embedded\mingw\bin\autoconf	SUCCESS
conti_v2.exe	2492	SetRenameInformationFile	C:\metasploit-framework\embedded\mingw\bin\autoconf	SUCCESS
conti_v2.exe	2492	CloseFile	C:\metasploit-framework\embedded\mingw\bin\autoconf	SUCCESS
conti_v2.exe	2492	CloseFile	C:\metasploit-framework\embedded\mingw\bin\autoconf.CECJF	SUCCESS

Figura 3.16: Operazioni svolte su un file

Come è possibile osservare, Process Monitor riporta le azioni svolte sul singolo file, che si riflettono con le procedure introdotte precedentemente, un esempio è *CreateFile* o *WriteFile*. In più vengono effettuate delle query sul file stesso per poterne recuperare gli attributi. Infine, quando il file viene crittografato, la sua estensione viene modificata (.*CECJF*).

Inoltre, è stato possibile verificare che effettivamente il malware carica numerose DLL che non erano visibili nell'analisi statica:

2492	CloseFile	C:\Windows\SysWOW64\kernel32.dll	SUCCESS
2492	CloseFile	C:\Windows\SysWOW64\ws2_32.dll	SUCCESS
2492	CloseFile	C:\Windows\SysWOW64\advapi32.dll	SUCCESS
2492	CloseFile	C:\Windows\SysWOW64\Rstrtmgr.dll	SUCCESS
2492	CloseFile	C:\Windows\SysWOW64\ole32.dll	SUCCESS
2492	CloseFile	C:\Windows\SysWOW64\ole32.dll	SUCCESS
2492	CloseFile	C:\Windows\SysWOW64\metapi32.dll	SUCCESS
2492	CloseFile	C:\Windows\SysWOW64\IPHLPAPI.DLL	SUCCESS
2492	CloseFile	C:\Windows\SysWOW64\shlwapi.dll	SUCCESS
2492	CloseFile	C:\Windows\SysWOW64\shell32.dll	SUCCESS
2492	CloseFile	C:\Windows\SysWOW64\ntdll.dll	SUCCESS

Figura 3.17: Chiusura delle DLL caricate



02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1162 -> 10.0.2.16:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1163 -> 10.0.2.17:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1164 -> 10.0.2.18:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1165 -> 10.0.2.19:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1166 -> 10.0.2.20:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1167 -> 10.0.2.21:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1168 -> 10.0.2.22:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1169 -> 10.0.2.23:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1170 -> 10.0.2.24:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1171 -> 10.0.2.25:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1172 -> 10.0.2.26:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1173 -> 10.0.2.27:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1174 -> 10.0.2.28:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1175 -> 10.0.2.29:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1176 -> 10.0.2.30:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1177 -> 10.0.2.31:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1178 -> 10.0.2.32:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1179 -> 10.0.2.33:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1180 -> 10.0.2.34:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1181 -> 10.0.2.35:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1182 -> 10.0.2.36:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1183 -> 10.0.2.37:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1184 -> 10.0.2.38:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1185 -> 10.0.2.39:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1186 -> 10.0.2.40:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1187 -> 10.0.2.41:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1188 -> 10.0.2.42:microsoft-ds	SUCCESS	Length: 0, seqnum:...
02:02:...	cont_i_v2.exe	2492	TCP Reconnect	MSEDGEWIN10.Jan:1189 -> 10.0.2.43:microsoft-ds	SUCCESS	Length: 0, seqnum:...

Figura 3.18: Tentativi di connessione agli host della sottorete

Per raccogliere quanti più informazioni possibili da quest'analisi, la rete esterna è stata simulata mediante il tool **Fakenet**, il quale permette di simulare numerosi servizi tra cui:

- DNS Listener sulla porta UDP 53.
- HTTP Listener sulla porta TCP 80.
- HTTPS Listener sulla porta TCP 443.
- SMTP Listener sulla porta TCP 25.
- Raw Binary Listener sulle porte TCP e UDP 1337. È il default listener.

02:02:...	fakenet.exe	5424	UDP Receive	MSEDGEWIN10.Jan:domain -> MSEDGEWIN10.Jan:65400	SUCCESS	Length: 41, sequen...
02:02:...	svchost.exe	1108	UDP Send	MSEDGEWIN10.Jan:53527 -> 192.168.1.254:domain	SUCCESS	Length: 41, sequen...
02:02:...	fakenet.exe	5424	UDP Receive	MSEDGEWIN10.Jan:domain -> MSEDGEWIN10.Jan:53527	SUCCESS	Length: 41, sequen...
02:02:...	svchost.exe	1108	UDP Send	MSEDGEWIN10.Jan:60450 -> 192.168.1.254:domain	SUCCESS	Length: 41, sequen...
02:02:...	fakenet.exe	5424	UDP Receive	MSEDGEWIN10.Jan:domain -> MSEDGEWIN10.Jan:60450	SUCCESS	Length: 41, sequen...
02:02:...	svchost.exe	1108	UDP Send	MSEDGEWIN10.Jan:55392 -> 192.168.1.254:domain	SUCCESS	Length: 41, sequen...
02:02:...	fakenet.exe	5424	UDP Receive	MSEDGEWIN10.Jan:domain -> MSEDGEWIN10.Jan:55392	SUCCESS	Length: 41, sequen...
02:02:...	svchost.exe	1108	UDP Send	MSEDGEWIN10.Jan:49197 -> 192.168.1.254:domain	SUCCESS	Length: 41, sequen...
02:02:...	fakenet.exe	5424	UDP Receive	MSEDGEWIN10.Jan:domain -> MSEDGEWIN10.Jan:49197	SUCCESS	Length: 41, sequen...
02:02:...	svchost.exe	1108	UDP Send	MSEDGEWIN10.Jan:60630 -> 192.168.1.254:domain	SUCCESS	Length: 41, sequen...
02:02:...	fakenet.exe	5424	UDP Receive	MSEDGEWIN10.Jan:domain -> MSEDGEWIN10.Jan:60630	SUCCESS	Length: 41, sequen...
02:02:...	svchost.exe	1108	UDP Send	MSEDGEWIN10.Jan:60131 -> 192.168.1.254:domain	SUCCESS	Length: 41, sequen...
02:02:...	fakenet.exe	5424	UDP Receive	MSEDGEWIN10.Jan:domain -> MSEDGEWIN10.Jan:60131	SUCCESS	Length: 41, sequen...
02:02:...	svchost.exe	1108	UDP Send	MSEDGEWIN10.Jan:55829 -> 192.168.1.254:domain	SUCCESS	Length: 41, sequen...
02:02:...	fakenet.exe	5424	UDP Receive	MSEDGEWIN10.Jan:domain -> MSEDGEWIN10.Jan:55829	SUCCESS	Length: 41, sequen...
02:02:...	svchost.exe	1108	UDP Send	MSEDGEWIN10.Jan:52251 -> 192.168.1.254:domain	SUCCESS	Length: 41, sequen...
02:02:...	fakenet.exe	5424	UDP Receive	MSEDGEWIN10.Jan:domain -> MSEDGEWIN10.Jan:52251	SUCCESS	Length: 41, sequen...
02:02:...	svchost.exe	1108	UDP Send	MSEDGEWIN10.Jan:65189 -> 192.168.1.254:domain	SUCCESS	Length: 41, sequen...
02:02:...	fakenet.exe	5424	UDP Receive	MSEDGEWIN10.Jan:domain -> MSEDGEWIN10.Jan:65189	SUCCESS	Length: 41, sequen...
02:02:...	svchost.exe	1108	UDP Send	MSEDGEWIN10.Jan:61086 -> 192.168.1.254:domain	SUCCESS	Length: 41, sequen...
02:02:...	fakenet.exe	5424	UDP Receive	MSEDGEWIN10.Jan:domain -> MSEDGEWIN10.Jan:61086	SUCCESS	Length: 41, sequen...
02:02:...	svchost.exe	1108	UDP Send	MSEDGEWIN10.Jan:56407 -> 192.168.1.254:domain	SUCCESS	Length: 41, sequen...
02:02:...	fakenet.exe	5424	UDP Receive	MSEDGEWIN10.Jan:domain -> MSEDGEWIN10.Jan:56407	SUCCESS	Length: 41, sequen...
02:02:...	svchost.exe	1108	UDP Send	MSEDGEWIN10.Jan:62236 -> 192.168.1.254:domain	SUCCESS	Length: 41, sequen...
02:02:...	fakenet.exe	5424	UDP Receive	MSEDGEWIN10.Jan:domain -> MSEDGEWIN10.Jan:62236	SUCCESS	Length: 41, sequen...
02:02:...	svchost.exe	1108	UDP Send	MSEDGEWIN10.Jan:59527 -> 192.168.1.254:domain	SUCCESS	Length: 41, sequen...

Figura 3.19: Tentativi di connessione ad un dominio esterno

Lo screen si riferisce alle richieste fatte al DNS listener di Fakenet avente indirizzo 192.168.1.254.

## Process Explorer

**Process Explorer** è un tool per controllare i processi. Esso fornisce le funzionalità di Task Manager di Windows insieme a un ricco set di funzionalità per la raccolta di informazioni sui processi in esecuzione sul sistema.

Questo strumento permette di vedere non solo le statistiche di runtime di un processo come utilizzo di memoria e CPU, ma permette anche di vedere quali DLL sta importando il processo:

explorer.exe	< 0.01	58.572 K	133.616 K	1876	Esplora risorse	Microsoft Corporation
SecurityHealthSystray.exe		1.724 K	8.508 K	4744	Windows Security notificatio...	Microsoft Corporation
VBoxTray.exe	< 0.01	2.472 K	9.996 K	5060	VirtualBox Guest Additions Tr...	Oracle Corporation
OneDrive.exe	< 0.01	17.984 K	47.420 K	4468	Microsoft OneDrive	Microsoft Corporation
Regshot-x64-Unicode.exe		219.536 K	231.992 K	1772		
conti_v2.exe	67.64	13.632 K	19.892 K	5004		
procexp.exe		4.444 K	10.768 K	5068	Sysinternals Process Explorer	Sysinternals - www.sysinter...
procexp64.exe	< 0.01	29.744 K	64.304 K	3796	Sysinternals Process Explorer	Sysinternals - www.sysinter...
svchost.exe		3.000 K	16.504 K	828	Processo host per servizi di ...	Microsoft Corporation
ShellExperienceHost.exe	Susp...	28.772 K	46.948 K	3088	Windows Shell Experience H...	Microsoft Corporation

Name	Description	Company Name	Path
advapi32.dll	Advanced Windows 32 Base API	Microsoft Corporation	C:\Windows\SysWOW64\advapi32.dll
advapi32.dll	Advanced Windows 32 Base API	Microsoft Corporation	C:\Windows\SysWOW64\advapi32.dll
apphelp.dll	Libreria client compatibilità applicaz...	Microsoft Corporation	C:\Windows\SysWOW64\apphelp.dll
bcrypt.dll	Windows Cryptographic Primitives ...	Microsoft Corporation	C:\Windows\SysWOW64\bcrypt.dll
bcryptprimitives.dll	Windows Cryptographic Primitives ...	Microsoft Corporation	C:\Windows\SysWOW64\bcryptprimitives.dll
cfgmgr32.dll	Configuration Manager DLL	Microsoft Corporation	C:\Windows\SysWOW64\cfgmgr32.dll
clbcatq.dll	COM+ Configuration Catalog	Microsoft Corporation	C:\Windows\SysWOW64\clbcatq.dll
combase.dll	Microsoft COM for Windows	Microsoft Corporation	C:\Windows\SysWOW64\combase.dll
conti_v2.exe			C:\Users\IEUser\Downloads\conti_v2\conti_v2.exe
cryptbase.dll	Base cryptographic API DLL	Microsoft Corporation	C:\Windows\SysWOW64\cryptbase.dll
cryptsp.dll	Cryptographic Service Provider API	Microsoft Corporation	C:\Windows\SysWOW64\cryptsp.dll
dnsapi.dll	DNS Client API DLL	Microsoft Corporation	C:\Windows\SysWOW64\dnsapi.dll
gdi32.dll	GDI Client DLL	Microsoft Corporation	C:\Windows\SysWOW64\gdi32.dll
gdi32full.dll	GDI Client DLL	Microsoft Corporation	C:\Windows\SysWOW64\gdi32full.dll
imm32.dll	Multi-User Windows IMM32 API Cli...	Microsoft Corporation	C:\Windows\SysWOW64\imm32.dll
IPHLPAPI.DLL	IP Helper API	Microsoft Corporation	C:\Windows\SysWOW64\IPHLPAPI.DLL
IPHLPAPI.DLL	IP Helper API	Microsoft Corporation	C:\Windows\SysWOW64\IPHLPAPI.DLL
kemal.appcore.dll	AppModel API Host	Microsoft Corporation	C:\Windows\SysWOW64\kemal.appcore.dll
kemal32.dll	Windows NT BASE API Client DLL	Microsoft Corporation	C:\Windows\SysWOW64\kemal32.dll
kemal32.dll	Windows NT BASE API Client DLL	Microsoft Corporation	C:\Windows\SysWOW64\kemal32.dll
KernelBase.dll	Windows NT BASE API Client DLL	Microsoft Corporation	C:\Windows\SysWOW64\KernelBase.dll
locale.nls			C:\Windows\System32\locale.nls
msvc_p_win.dll	Microsoft® C Runtime Library	Microsoft Corporation	C:\Windows\SysWOW64\msvc_p_win.dll
msvcrt.dll	Windows NT CRT DLL	Microsoft Corporation	C:\Windows\SysWOW64\msvcrt.dll
mswsock.dll	Microsoft Windows Sockets 2.0 S...	Microsoft Corporation	C:\Windows\SysWOW64\mswsock.dll
ncrypt.dll	Windows NCrypt Router	Microsoft Corporation	C:\Windows\SysWOW64\ncrypt.dll
netapi32.dll	Net Win32 API DLL	Microsoft Corporation	C:\Windows\SysWOW64\netapi32.dll
netapi32.dll	Net Win32 API DLL	Microsoft Corporation	C:\Windows\SysWOW64\netapi32.dll
nsi.dll	NSI User-mode interface DLL	Microsoft Corporation	C:\Windows\SysWOW64\nsi.dll
ntasn1.dll	Microsoft ASN.1 API	Microsoft Corporation	C:\Windows\SysWOW64\ntasn1.dll
ntdll.dll	DLL del livello NT	Microsoft Corporation	C:\Windows\SysWOW64\ntdll.dll

Figura 3.20: DLL

Questo risultato ci conferma ancora una volta che *conti\_v2.exe* chiama moltissime DLL a runtime.

Gli handle utilizzati dall'eseguibile sono:

svchost.exe		3.276 K	6.656 K	2272	Processo host per servizi di ...	Microsoft Corporation
sihost.exe		6.460 K	24.324 K	2880	Shell Infrastructure Host	Microsoft Corporation
svchost.exe		8.768 K	32.308 K	2904	Processo host per servizi di ...	Microsoft Corporation
taskhostw.exe		5.668 K	15.136 K	2992	Processo host per attività di ...	Microsoft Corporation
ctfmon.exe		5.160 K	14.476 K	2212	Caricatore CTF	Microsoft Corporation
explorer.exe	3.02	66.088 K	122.732 K	2972	Esplora risorse	Microsoft Corporation
SecurityHealthSystray.exe		1.720 K	7.784 K	864	Windows Security notificatio...	Microsoft Corporation
VBoxTray.exe	< 0.01	2.536 K	9.416 K	5244	VirtualBox Guest Additions Tr...	Oracle Corporation
OneDrive.exe	< 0.01	17.968 K	40.408 K	5316	Microsoft OneDrive	Microsoft Corporation
d3c75c5bc4ae087d547bd722...	42.33	13.060 K	19.328 K	2632		
SearchIndexer.exe	< 0.01	20.448 K	29.040 K	3112	Microsoft Windows Search I...	Microsoft Corporation
SearchProtocolHost.exe		2.464 K	10.976 K	5428	Microsoft Windows Search P...	Microsoft Corporation
SearchFilterHost.exe		2.076 K	8.620 K	2596	Microsoft Windows Search F...	Microsoft Corporation
svchost.exe		2.672 K	15.496 K	3264	Processo host per servizi di ...	Microsoft Corporation
ShellExperienceHost.exe	Susp...	29.060 K	70.960 K	3552	Windows Shell Experience H...	Microsoft Corporation
SearchUI.exe	Susp...	104.880 K	158.312 K	3656	Search and Cortana applicati...	Microsoft Corporation

Type	Name
ALPC Port	\RPC Control\OLE49F927488E53C3C1F3B1FE9DEC0C
Desktop	\Default
Directory	\KnownDlls
Directory	\KnownDlls32
Directory	\KnownDlls32
Directory	\Sessions\1\BaseNamedObjects
Event	\KernelObjects\MaximumCommitCondition
File	C:\Windows
File	C:\Users\IEUser\Downloads\conti_v2
File	\Device\NCG
File	\Device\KsecDD
File	\Device\DeviceApi
File	C:\Windows\SysWOW64\kernel32.dll
File	C:\Windows\SysWOW64\ws2_32.dll
File	C:\Windows\SysWOW64\advapi32.dll
File	C:\Windows\SysWOW64\Rstrtmgr.dll
File	C:\Windows\SysWOW64\ole32.dll

Figura 3.21: Handle

Tra i file utilizzati dal malware troviamo quelli evidenziati precedentemente da Process Monitor.

Altri due handle importanti sono quelli relativi al mutex per la gestione dei thread in parallelo:

*Mutant\Sessions\1\BaseNamedObjects\kjkbmusop9iqkamvcrewuyy777*

*Mutant\Sessions\1\BaseNamedObjects\SM0:5596:168:WilStaging\_02*

Conti tenta di decifrare la stringa "kjkbmusop9iqkamvcrewuyy777" e di usarla come nome di un oggetto Mutex.

### 3.3 Advanced Static Analysis - Reverse Engineering

Nell'analisi di un malware è utile applicare tecniche di reverse engineering per tentare di ricostruirne in grandi linee il codice sorgente. Lo si fa mediante l'ausilio dei *disassemblatori* i quali dal codice macchina di un eseguibile ne deducono l'assembly (x86 - 32bit Intel). Il disassemblatore utilizzato in questa analisi è **IDA Pro**.



Figura 3.22: Logo *IDA Pro*

In seguito verranno descritte le caratteristiche salienti del malware riscontrate durante l'analisi. Per realizzare la ricerca si è deciso di partire dagli import e dalle stringhe ritenute "sospette" nelle analisi precedenti. Avendo già un'idea sul funzionamento di *Conti* sono stati approfonditi i seguenti aspetti:

- Caricamento a runtime di librerie ed APIs.
- Implementazione del multi-threading.
- Esplorazione del filesystem e crittazione dei files.
- Accesso in scrittura alla console per impartire comandi sulla macchina vittima.

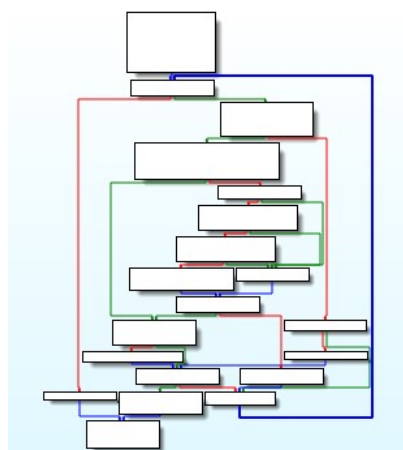
### 3.3.1 Dynamic Loading

La maggior parte delle funzioni che *Conti* utilizza vengono caricate a *runtime* sfruttando tre APIs della libreria KERNEL32: **LoadLibraryExW**, **GetModuleHandleW** e **GetProcAddress**. In questo modo non risultano visibili nell'immediato se si vanno ad analizzare gli imports in tool come *PEstudio*, ma potrebbero essere "nascoste" nelle stringhe dell'eseguibile in chiaro o cifrate.

#### **LoadLibraryExW**

Questa funzione carica il modulo specificato dalla stringa che riceve in ingresso, restituendo un handle della stessa. In IDA si evince che viene richiamata in due subroutines differenti, *sub\_41D41F* e *sub\_421A03*. In entrambe è evidente la presenza di un *ciclo while* il che fa pensare che ricevono entrambe in input un set di librerie da dover caricare dinamicamente in sequenza.



Figura 3.23: Struttura *sub\_421A03*

```

.text:0041D44B
.text:0041D44B      loc_41D44B:
.text:0041D44B  014 8B 1C 9D 58 82 42+mov     ebx, ds:lpLibFileName[ebx*4]
.text:0041D44B  014 00
.text:0041D452  014 68 00 08 00 00      push     800h           ; dwFlags
.text:0041D457  018 6A 00              push     0             ; hFile
.text:0041D459  01C 53              push     ebx           ; lpLibFileName
.text:0041D45A  020 FF 15 80 80 42 00    call     ds:LoadLibraryExW
.text:0041D460  014 8B F0              mov     esi, eax
.text:0041D462  014 85 F6              test    esi, esi
.text:0041D464  014 75 50              jnz     short loc_41D4B6

```

Figura 3.24: Focus su LoadLibraryExW invocata in *sub\_41D41F*

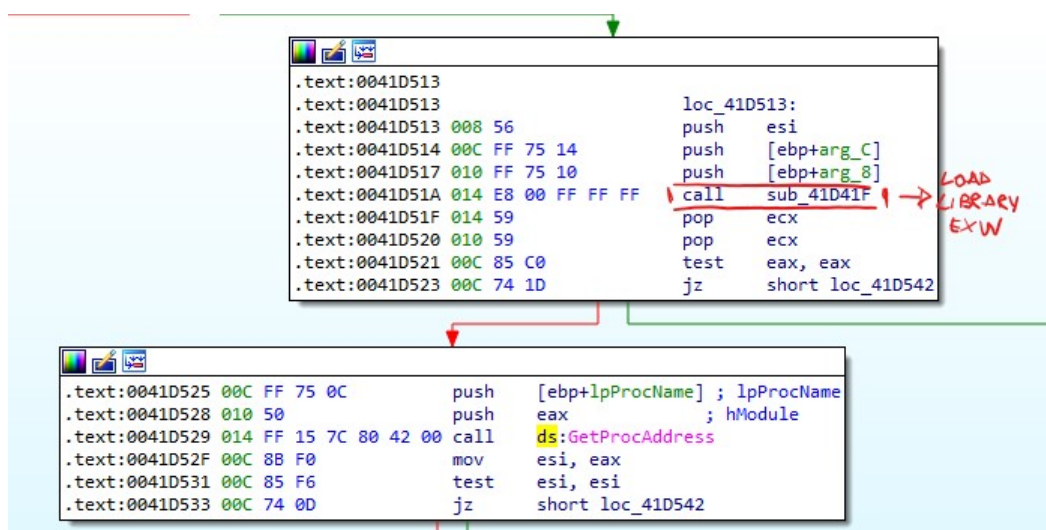
### GetModuleHandleW

Recupera l'handle del modulo che gli si passa in ingresso tramite stringa. Tra le procedure che la richiamano c'è quella di *start* dell'eseguibile.

### GetProcAddress

Riceve in input il nome di una procedura e l'handle della libreria che la contiene. Restituisce l'indirizzo di quella funzione. La ritroviamo nelle seguenti subroutines:

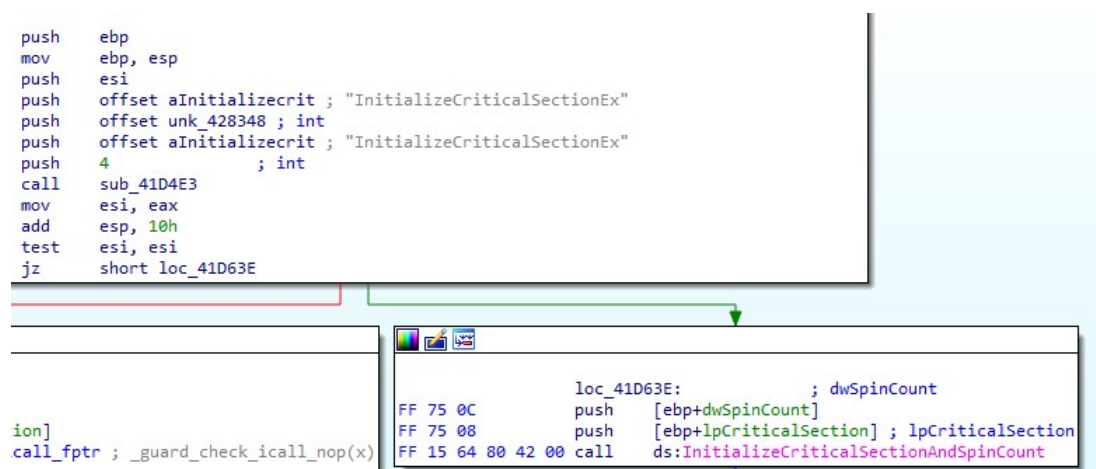
- *sub\_41D4E3*, preceduta dalla chiamata alla LoadLibraryExW. Evidentemente tale subroutine viene richiamata ogni qual volta si necessita di caricare funzioni dinamicamente.
- *sub\_41EBD0*, per importare la procedura *CorExitProcess*, preceduta dall'import di *mscoree.dll*.
- *sub\_421ACC*, molto simile alla prima subroutine dell'elenco. In essa viene caricata un'altra tipologia di funzioni tramite la libreria *api-ms-win-core-datetime-l1-1-1*.

Figura 3.25: GetProcAddress in *sub\_41D4E3*

Come già detto, la maggior parte delle funzioni vengono caricate in questo modo nell'eseguibile e nella gran parte dei casi le stringhe che specificano le procedure e i moduli sono *codificate*. Tuttavia sono state individuate in questo contesto le seguenti funzioni il cui nome è riportato in chiaro: *LCMapStringEx*, *AreFileApisANSI*, *LocaleNameToLCID*, *AppPolicy-GetProcessTerminationMethod*, *InitializeCriticalSectionEx*, *FlsAlloc*, *FlsFree*, *FlsGetValue* e *FlsSetValue*. Probabilmente gli autori del malware non hanno ritenuto necessario offuscarle.

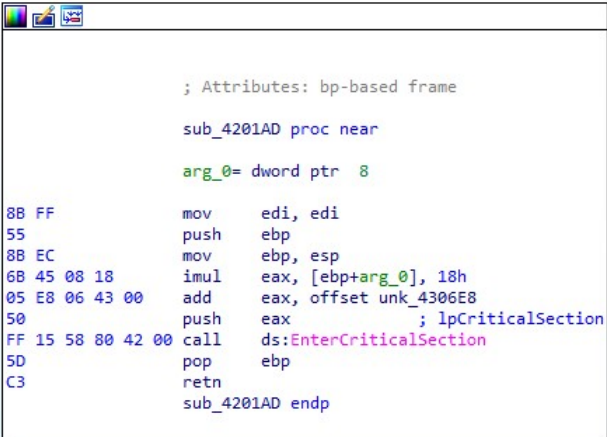
### 3.3.2 Multi-Threading

Conti implementa il multi-threading per poter incrementare la rapidità con cui vengono cifrati i dati nel filesystem del dispositivo vittima. Per la gestione della loro concorrenza è inizializzata una sezione critica tra di loro condivisa nelle subroutines *sub\_421CAD* e *sub\_41D606*.

Figura 3.26: InitializeCriticalSectionEx in *sub\_421CAD*

A tal proposito sono da evidenziare le due procedure **InitializeCriticalSectionEx** caricata a runtime e **InitializeCriticalSectionAndSpinCount** importata in maniera standard. Entrambe inizializzano appunto una sezione critica, la seconda imposta anche uno *spin count* ovvero un timestamp che coincide con il tempo che un thread attende (nel momento in cui trova la sezione critica occupata) prima di invocare la wait sul semaforo che regola l'accesso all'area di memoria.

Dopo averla inizializzata, bisogna regolarne l'accesso. Lo si fa sfruttando le procedure **EnterCriticalSection** e **LeaveCriticalSection**. Esse sono richiamate da tre subroutines ciascuna, ognuna delle quali presenta la CriticalSection in *sub\_4201AD*.



```

; Attributes: bp-based frame

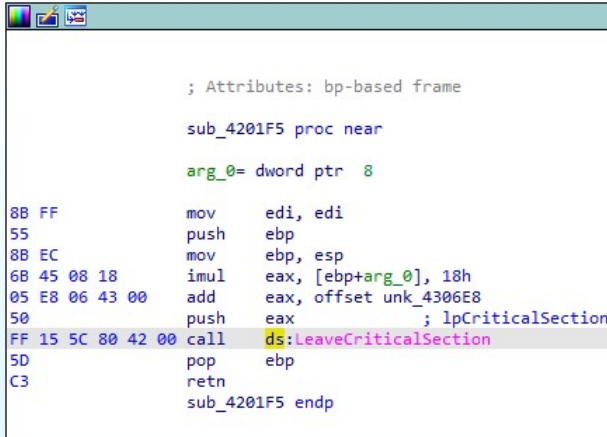
sub_4201AD proc near

arg_0= dword ptr 8

8B FF      mov     edi, edi
55         push    ebp
8B EC      mov     ebp, esp
6B 45 08 18 imul    eax, [ebp+arg_0], 18h
05 E8 06 43 00 add     eax, offset unk_4306E8
50         push    eax
FF 15 58 80 42 00 call    ds:EnterCriticalSection
5D         pop     ebp
C3         retn
sub_4201AD endp

```

Figura 3.27: EnterCriticalSectionEx in *sub\_4201AD*



```

; Attributes: bp-based frame

sub_4201F5 proc near

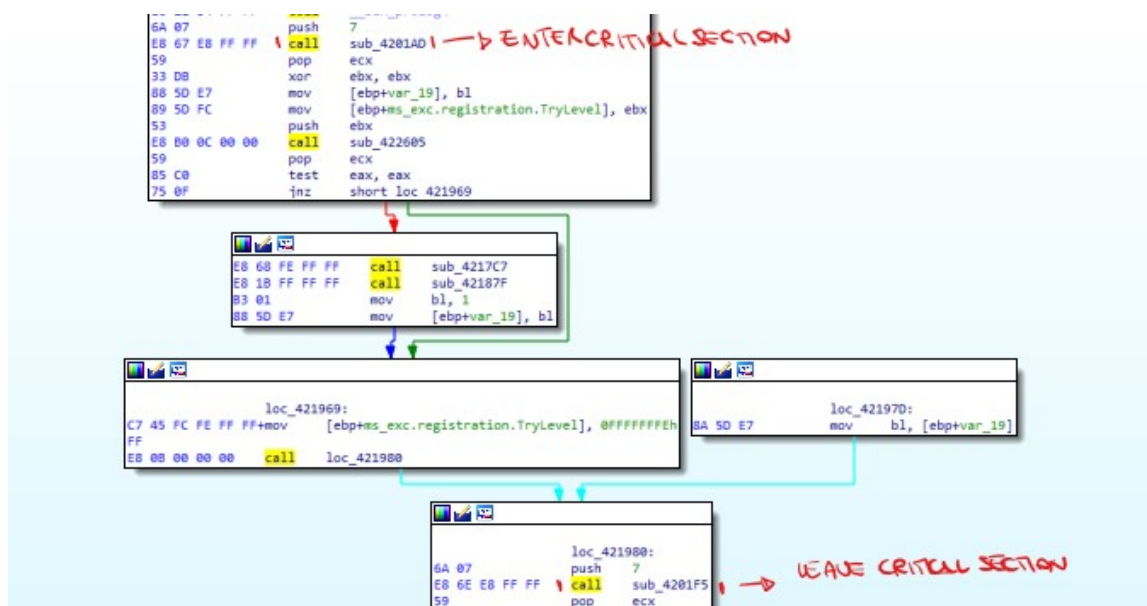
arg_0= dword ptr 8

8B FF      mov     edi, edi
55         push    ebp
8B EC      mov     ebp, esp
6B 45 08 18 imul    eax, [ebp+arg_0], 18h
05 E8 06 43 00 add     eax, offset unk_4306E8
50         push    eax
FF 15 5C 80 42 00 call    ds:LeaveCriticalSection
5D         pop     ebp
C3         retn
sub_4201F5 endp

```

Figura 3.28: LeaveCriticalSectionEx in *sub\_4201F5*

Tali funzioni vengono richiamate in sequenza in svariati contesti di subroutines più articolate che evidentemente descrivono le operazioni realizzate dai threads tra cui la crittografia.

Figura 3.29: `sub_421933`

Ad esempio, la funzione riportata in figura 3.29 richiama ulteriori due funzioni le quali contengono al loro interno le procedure: *GetStartupInfoW*, *GetFileType* e *GetStdHandle*. Esse sono propedeutiche all'esecuzione di operazioni sui files come lettura o scrittura.



### 3.3.3 Files Encryption

#### FindFirstFileExW & FindNextFileW

La ricerca di tutti i file che si trovano in una cartella avviene mediante una procedura iterativa nella *sub\_420784*.

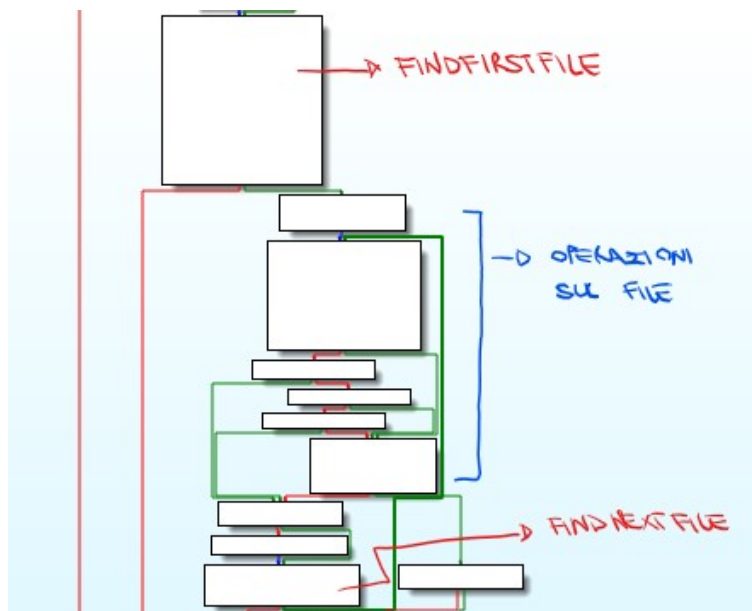
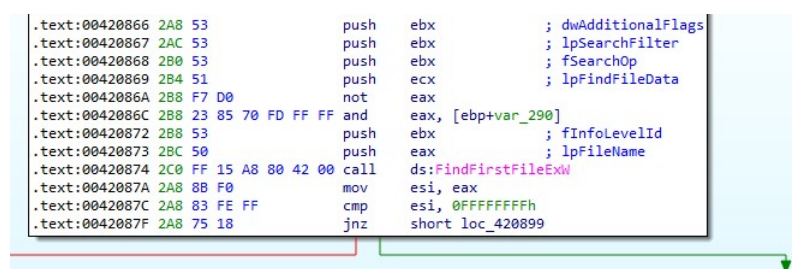


Figura 3.30: *sub\_420784*

E' una procedura abbastanza complessa che può essere suddivisa in quattro parti:

- Check sulla validità del path del file individuato (probabilmente ne viene scelto uno di default da cui partire che viene passato per mezzo di variabili globali opportunamente cifrate).
- Invocazione di *FindFirstFileExW* a partire dal nome specificato. La funzione ritorna un handle a quel file, se lo trova.
- Invocazione di *FindNextFileW*, in maniera iterativa (loop).
- Invocazione di *FindClose* nel caso in cui non si trovino più files, che chiude l'handle generato da *FindFirstFile* ed evidentemente chiude la ricerca in quel path.

Prima di realizzare le sue operazioni, il malware scarta i seguenti files: "." (directory stessa) e ".." (directory padre).

Figura 3.31: *FindFirstFileExW* sub\_420784Figura 3.32: *FindNextFileW* sub\_420784

La subroutine precedentemente descritta è invocata in maniera iterativa da un'altra funzione individuata da IDA: *sub\_4204E0*.

Figura 3.33: *sub\_4204E0*

Probabilmente il suo scopo è quello di "navigare" nel filesystem per cercare directory da cifrare.

## WriteFile

Utilizzata per sovrascrivere i file originali con quelli cifrati (riceve in input l'handle del file in questione).

Viene richiamata da quattro funzioni le quali sono tutte a sua volta invocate da un'unica subroutine, la *sub\_4248DC*. Anche in quest'ultima è presente la *WriteFile*.

Le quattro subroutines hanno tutte una struttura più o meno simile e probabilmente ne è stata implementata più di una per aumentare la confusione del codice.

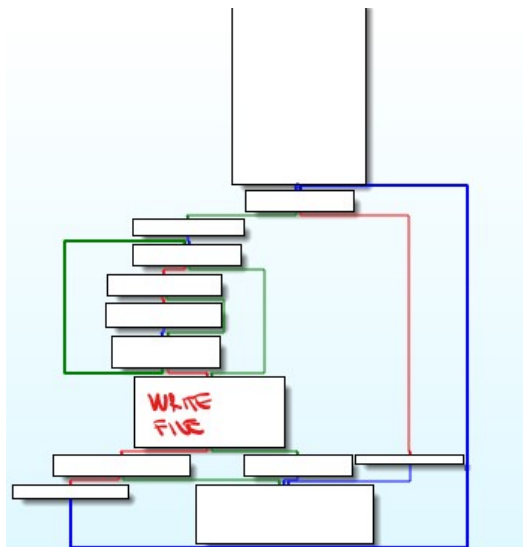


Figura 3.34: *sub\_4245D5*

Di seguito è riportata la struttura della già citata *sub\_4248DC* a partire da cui sono richiamate sia le routine che invocano la *WriteFile* che quella relativa alla *WriteConsole* (descritta dettagliatamente in seguito).

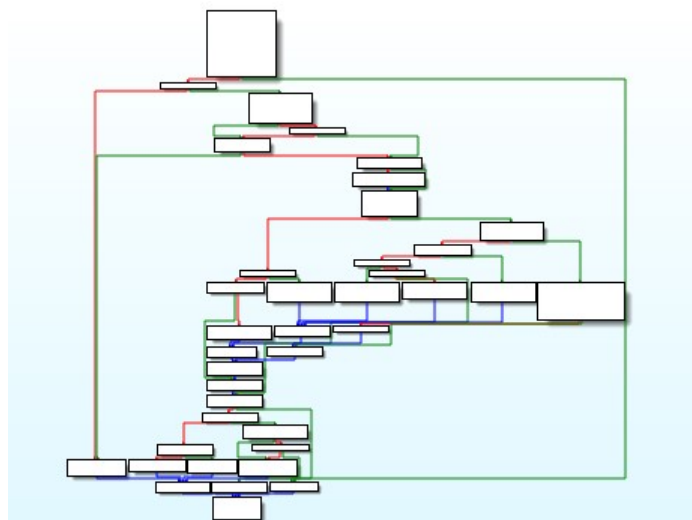


Figura 3.35: *sub\_4248DC*

### 3.3.4 Console Management

Il ransomware Conti riesce ad ottenere l'accesso in scrittura alla console dell'host vittima. Molto probabilmente in questo modo esegue dei comandi per customizzare le operazioni eseguite sul dispositivo oltre che dropare i *readme.txt* contenenti il messaggio per il riscatto. Le principali subroutines interessate in questo contesto sono:

- **sub\_424F94**

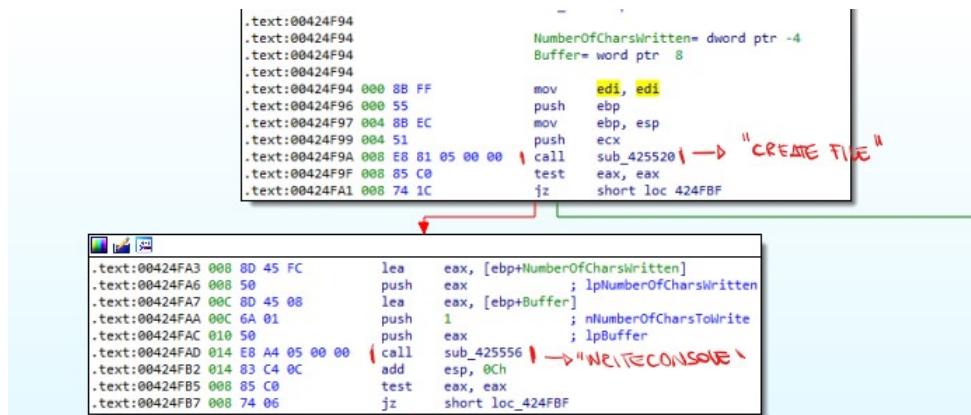


Figura 3.36: sub\_424F94

Essa innanzitutto richiama la *sub\_425520*, dopodichè passa sullo stack la stringa che ha intenzione di scrivere (*lpBuffer*) e invoca la *sub\_425556*.

A questo punto, l'unica cosa che la prima *sub\_425520* fa è richiamare un'altra routine, la *sub\_425501* che al suo interno esegue la *CreateFileW*.

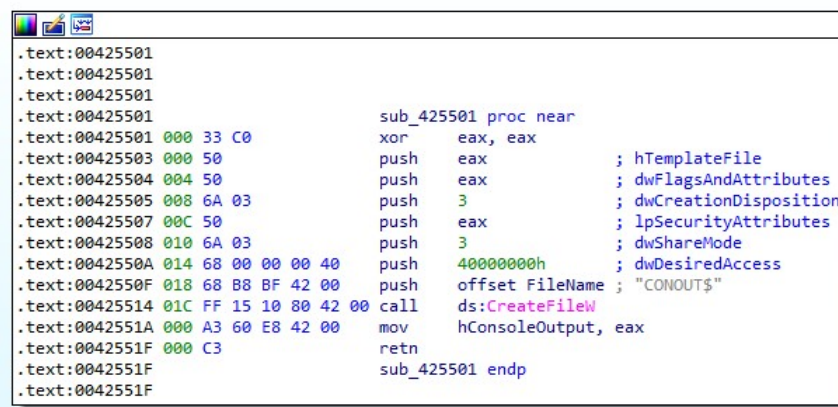


Figura 3.37: sub\_425501

Il parametro *FileName* che le viene passato è *CONOUT\$*, il che sta ad indicare che tale funzione in questo caso restituisce l'handle dell'*active screen buffer* (apre/genera una console e ne restituisce il riferimento).

A questo punto, ritornando alla `sub_424F94`, dopo aver ottenuto l'accesso alla console viene invocata la `sub_425556`.

- `sub_425556`

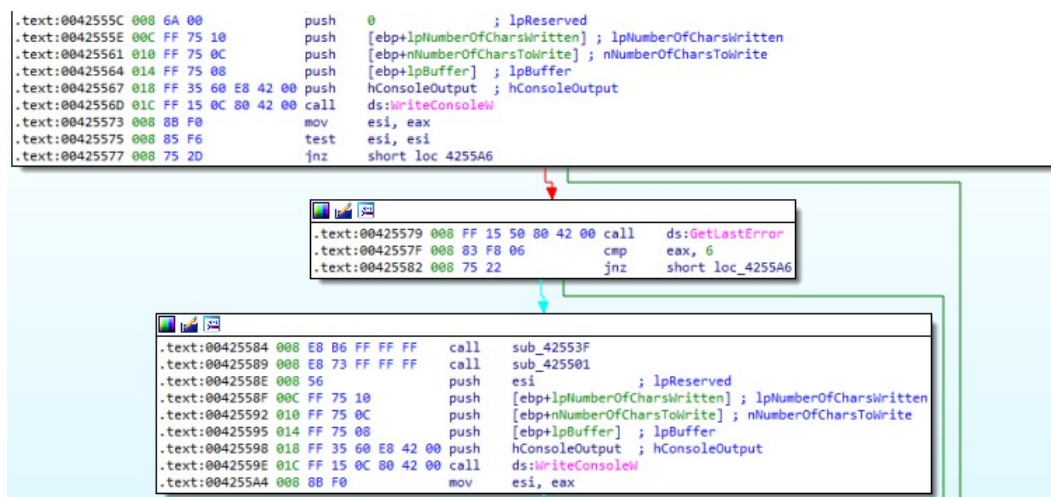


Figura 3.38: `sub_425556`

In cui avviene la scrittura del contenuto di `lpBuffer` per mezzo della `WriteConsoleW`.

### 3.4 Hybrid Analysis - Sandbox

Una **sandbox** è un meccanismo per eseguire applicazioni in uno spazio limitato. Solitamente fornisce un ristretto e controllato set di risorse al programma che deve essere testato. Le sandbox solitamente vengono utilizzate per eseguire programmi non testati o non attendibili, non verificati o provenienti da terze parti non riconosciute (come utenti o siti web), senza rischiare di infettare il dispositivo dove viene eseguita l'applicazione.

Con l'ulteriore progresso dello sviluppo della tecnologia sandbox e con l'aumento della richiesta di un metodo rapido per testare il software, abbiamo assistito all'introduzione delle sandbox online. Si tratta di siti Web in cui è possibile inviare un campione e ricevere un rapporto sulle azioni del campione osservate dalla sandbox online.

**Hybrid Analysis**<sup>3</sup> è un servizio di analisi dei file che combina i dati di runtime con l'analisi del dump della memoria per estrarre tutti i possibili percorsi di esecuzione anche per il malware più evasivo.

Tutti i dati estratti dal motore di Hybrid Analysis vengono elaborati automaticamente e integrati nei report di analisi del malware.

<sup>3</sup><https://www.hybrid-analysis.com/sample/d3c75c5bc4ae087d547bd722bd84478ee6baf8c3355b930f26cc19777cd39d4c/63297692bcacac004c2642d3>

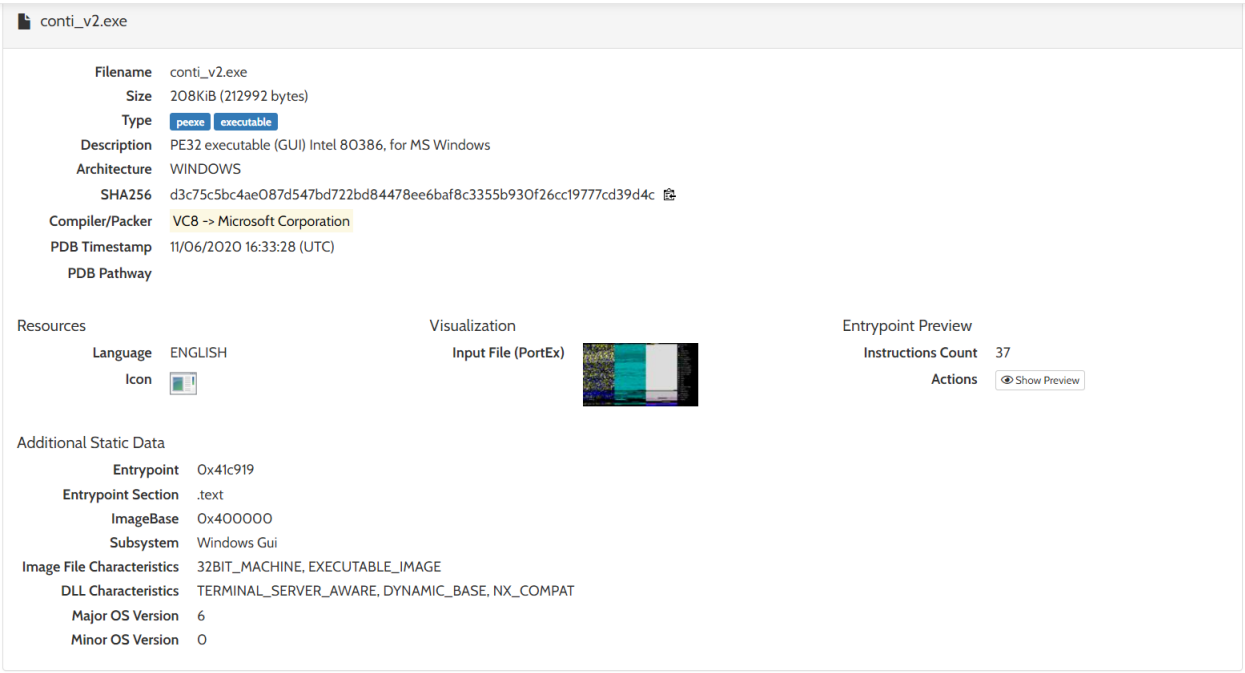


Figura 3.39: Hybrid Analysis

Una parte dell'analisi effettuata da Hybrid è simile a quella offerta da VirusTotal, ad esempio, vengono fornite stringhe, DLL caricate con le relative procedure importate.

Hybrid è stato utilizzato per integrare e completare l'analisi del ransomware. I suoi risultati ci hanno permesso di avere una panoramica più completa riguardo il suo funzionamento.

3.4.1 Malicious Indicators



Figura 3.40: Hooks file system APIs

Il malware si aggancia e sfrutta funzioni di sistema per accedere ad informazioni riservate, come già osservato nelle analisi precedenti.

## Ransomware/Banking

Deletes volume snapshots (often used by ransomware)

The analysis extracted a known ransomware file

**details** Found dropped filename "\_ReadMe\_.txt" which has been seen in the context of ransomware (Indicator: README\_.txt)  
 Found dropped filename "README.html" which has been seen in the context of ransomware (Indicator: README.html)

**source** Extracted File

**relevance** 5/10

Figura 3.41: README.txt are known ransomware files

## Spyware/Information Retrieval

Hooks internet related APIs

**details** "send@WS2\_32.DLL" in "conti\_v2.exe"

**source** Hook Detection

**relevance** 10/10

**ATT&CK ID** T1056.004 (Show technique in the MITRE ATT&CK™ matrix)

Figura 3.42: Hooks Internet related APIs

Molte di queste procedure non sono state evidenziate dalle precedenti analisi molto probabilmente perchè offuscate. Hybrid usa tool che ne consentono la decifratura.

## System Destruction

Deletes volume snapshots (often used by ransomware)

**details** Deletes volume snapshots files "WMIc.exe" with commandline "shadowcopy where "ID="{5DD35BD1-ADDD-4E82-9BFC-9426DF1B1B92}" delete" (Show Process)  
 Deletes volume snapshots files "WMIc.exe" with commandline "shadowcopy where "ID="{66EB82AE-CBA3-4630-84CB-6BB3D439D968}" delete" (Show Process)  
 Deletes volume snapshots files "WMIc.exe" with commandline "shadowcopy where "ID="{90E3C7D9-DFD5-494A-B413-99BB94E826EC}" delete" (Show Process)  
 Deletes volume snapshots files "WMIc.exe" with commandline "shadowcopy where "ID="{4A50B9C6-9E9E-426D-BC1B-7E892FC4849E}" delete" (Show Process)

**source** Monitored Target

**relevance** 10/10

**ATT&CK ID** T1490 (Show technique in the MITRE ATT&CK™ matrix)

Figura 3.43: Deletes volume snapshots (often used by ransomware)

Come già accennato uno degli obiettivi del malware è quello di eliminare qualunque copia di backup (shadowcopy) individuata nel dispositivo vittima.

### 3.4.2 Suspicious Indicators

Reads the active computer name

**details** "conti\_v2.exe" (Path: "HKLM\SYSTEM\CONTROLSET001\CONTROL\COMPUTERNAME\ACTIVECOMPUTERNAME"; Key: "COMPUTERNAME")  
 "WMIc.exe" (Path: "HKLM\SYSTEM\CONTROLSET001\CONTROL\COMPUTERNAME\ACTIVECOMPUTERNAME"; Key: "COMPUTERNAME")

**source** Registry Access

**relevance** 5/10

**ATT&CK ID** T1012 (Show technique in the MITRE ATT&CK™ matrix)

Figura 3.44: Reads the active computer name

Valore utilizzato in genere come identificativo univoco della macchina.



Reads the cryptographic machine GUID

```

details "conti_v2.exe" (Path: "HKLM\SOFTWARE\MICROSOFT\CRYPTOGRAPHY"; Key: "MACHINEGUID")
        "WMIC.exe" (Path: "HKLM\SOFTWARE\MICROSOFT\CRYPTOGRAPHY"; Key: "MACHINEGUID")
source Registry Access
relevance 10/10
ATT&CK ID T1082 (Show technique in the MITRE ATT&CK™ matrix)

```

Figura 3.45: Reads the cryptographic machine GUID

Il ransomware accede ad informazioni private che caratterizzano univocamente la macchina vittima (molto probabilmente per realizzare un'escalation dei privilegi).

Reads configuration files

```

details "conti_v2.exe" read file "%PROGRAMFILES%\desktop.ini"
        "conti_v2.exe" read file "C:\Users\desktop.ini"
        "conti_v2.exe" read file "C:\Program Files\Cleaner\winapp2.ini"
        "conti_v2.exe" read file "C:\Program Files\Mozilla Firefox\crashreporter.ini"
        "conti_v2.exe" read file "C:\Program Files\Mozilla Firefox\platform.ini"
        "conti_v2.exe" read file "C:\Program Files\Mozilla Firefox\update-settings.ini"
        "conti_v2.exe" read file "C:\Program Files\Mozilla Firefox\updater.ini"
        "conti_v2.exe" read file "C:\Program Files\Mozilla Maintenance Service\updater.ini"
        "conti_v2.exe" read file "C:\Users\%USERNAME%\desktop.ini"
        "conti_v2.exe" read file "C:\Program Files\Cleaner\cc_config.ini"
        "conti_v2.exe" read file "C:\Program Files\Mozilla Firefox\application.ini"
        "conti_v2.exe" read file "C:\Program Files\HNC\HCell80\PrivateInfo.ini"
        "conti_v2.exe" read file "C:\Program Files\HNC\HShow80\PrivateInfo.ini"
source API Call
relevance 4/10

```

Figura 3.46: Reads configuration files

#### Ransomware/Banking

The analysis extracted file with a known ransomware suffix

```

details Found dropped filename "dictionary.alcatel-lucent.aaa" which has been seen in the context of ransomware (Indicator: .aaa)
source Extracted File
relevance 10/10

```

Figura 3.47: The analysis extracted file with a known ransomware suffix

Sono stati individuati files con suffisso noto *.aaa* e tipico di alcuni ransomware.

#### System Security

Calls an API typically used to enable or disable privileges in the specified access token

```

details "WMIC.exe" called "AdjustTokenPrivileges" (UID: 00000000-00001796)
        "WMIC.exe" called "AdjustTokenPrivileges" (UID: 00000000-00001384)
        "WMIC.exe" called "AdjustTokenPrivileges" (UID: 00000000-00002208)
        "WMIC.exe" called "AdjustTokenPrivileges" (UID: 00000000-00000276)
source API Call
relevance 10/10
ATT&CK ID T1134 (Show technique in the MITRE ATT&CK™ matrix)

```

Figura 3.48: Calls APIs used to enable or disable privileges of an access token





Figura 3.49: Reads information about supported languages

## Capitolo 4

# Obfuscation

Nello sviluppo di software, l'**obfuscation** prevede di utilizzare codice sorgente che sia di difficile comprensione sia per gli esseri umani che per i computer.

I programmatori cercano di offuscare il codice con lo scopo di prevenire manipolazioni, reverse engineering, o creazioni di codici per cercare di decifrare il codice sorgente. Tale programmazione può essere fatta manualmente o tramite dei tool denominati *offusicatori*, i quali prevedono di introdurre variazioni al codice per complicarne la lettura.

Esiste una grande varietà di offusicatori che variano da strumenti accademici a tool open source fino a tool commerciali, che utilizzano i più disparati linguaggi di programmazione. Allo stesso modo in cui esistono offusicatori, vi sono anche i **deoffusicatori** che hanno lo scopo di effettuare trasformazioni inverse rispetto a quelle tipicamente adottate dagli offusicatori.

### 4.1 Windows API Hashing

La tecnica utilizzata dall'operazione di *Windows API Hashing* consiste nel rendere l'analisi del malware più complessa nascondendo le APIs di Windows importate nella *Import Address Table of the Portable Execution*.

Il problema per gli sviluppatori di un malware è che se si ha un **PE** (*Portable Executable*) con la sua **IAT** (*Import Address Table*) intatta, risulta molto semplice farsi un'idea riguardo le capacità di una PE: per esempio se è in grado di richiamare funzionalità di rete tramite il caricamento di un binario come *Ws\_32.dll*, il quale assume sicurezza nelle capacità della rete, o nell'importazione di qualche funzione che potrebbe modificare e mettere a rischio il sistema.

Per tale motivo gli autori del malware cercano di rendere l'analisi iniziale molto più complessa e non rendere percepibili le loro intenzioni solo guardando la IAT iniziale, e a tale scopo si possono utilizzare delle API hashing per nascondere le chiamate API sospette dalla IAT. In questo modo, quando un analista analizza il codice malevolo tramite un tool che permette

la visione delle stringhe utilizzate oppure un parser di PE, le chiamate alle APIs di Windows saranno nascoste dalla IAT del PE.

## 4.2 Obfuscation in Conti v2

Dalla diffusione delle prime versioni del ransomware Conti, esso ha subito drastiche mutazioni nell'ambito dell'offuscamento.

Inizialmente (versione 2019), esso implementava un semplice meccanismo di XOR che permetteva di nascondere le *APIs* invocate a runtime. In seguito, tale meccanismo è stato migliorato con l'introduzione di funzioni di encoding custom per l'offuscamento delle stringhe. Una delle prime costatazioni che si può fare vedendo semplicemente le dimensioni elevate del malware rispetto alla media, è che esso risulta ampiamente offuscato. Tale caratteristica è dovuta al fatto che esso implementa delle funzioni di decrittazione singole per ogni stringa; ciò comporta una serie di loop per ogni stringa, il che aumenta ampiamente le dimensioni del codice.

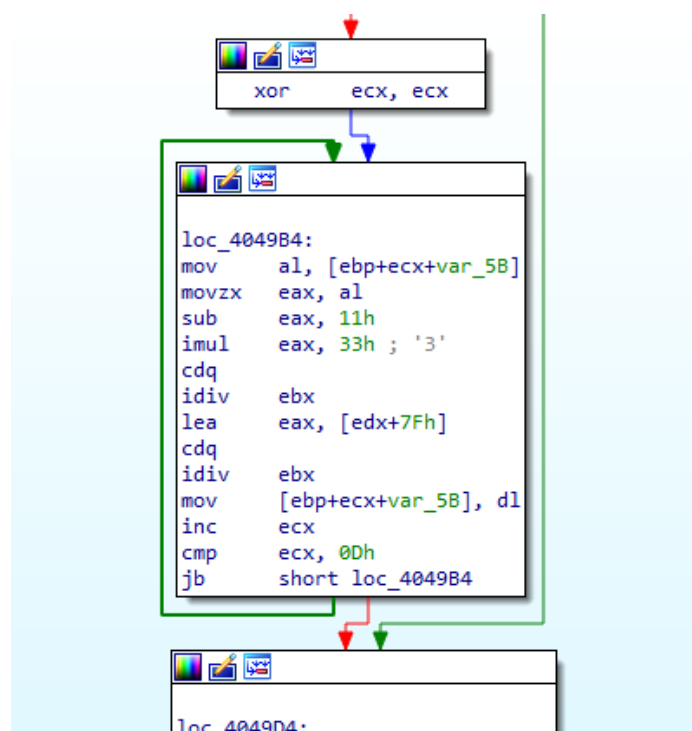


Figura 4.1: Loop di decrittazione di una stringa

Tale situazione è ampiamente osservabile tramite la scomposizione dell'eseguibile fatta con il tool di reverse engineering *IDA Pro*, ove è possibile notare una serie di loop per decifrare le stringhe nella *subroutine 401010*.

Dunque, queste stringhe sono caricate sullo stack ove vengono decodificate con il fine di ottenere dei numeri interi. Dopodiché, tramite una funzione di **switch** tali valori numerici sono associati alle funzioni che saranno, poi, richiamate.

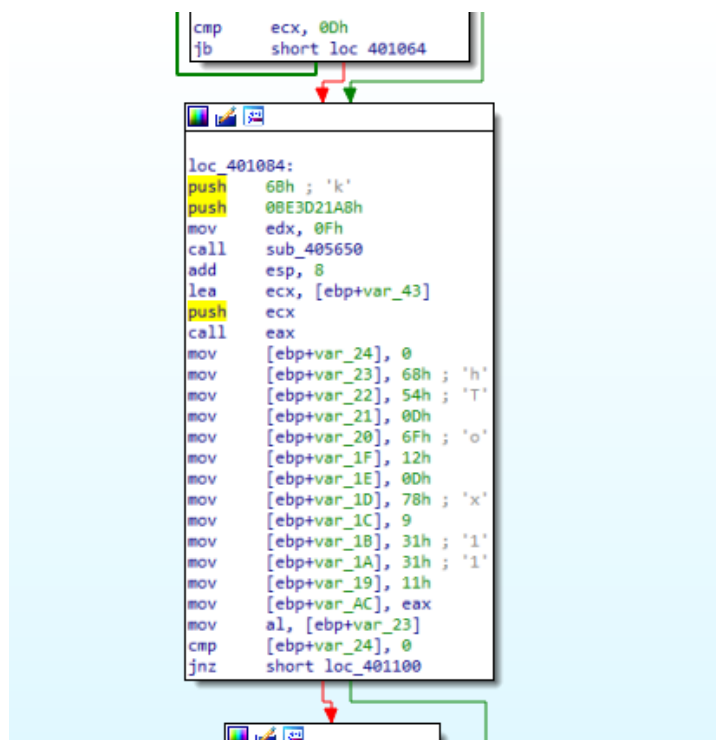


Figura 4.2: Passaggio dei parametri sullo stack (encrypted string)

Le funzioni associate a tali interi rappresentano la metodologia per ritrovare le DLL tramite: hash API e offset inserito nel buffer API.

Dopo aver ritrovato i nomi delle DLL, il malware Conti farà delle operazioni per comparare se gli hash del nome che sta cercando combaciano con quelle dei parametri ritrovati; se i due combaciano verrà ritrovato l'indirizzo della funzione.

Risulta fondamentale l'algoritmo di hashing utilizzato dal malware: **Murmur2A**.

```
#define mmix(h,k) { k *= m; k ^= k >> r; k *= m; h *= m; h ^= k; }

uint32_t MurmurHash2A ( const void * key, int len, uint32_t seed )
{
    const uint32_t m = 0x5bd1e995;
    const int r = 24;
    uint32_t l = len;

    const unsigned char * data = (const unsigned char *)key;

    uint32_t h = seed;

    while(len >= 4)
    {
        uint32_t k = *(uint32_t*)data;

        mmix(h,k);

        data += 4;
        len -= 4;
    }

    uint32_t t = 0;

    switch(len)
    {
        case 3: t ^= data[2] << 16;
        case 2: t ^= data[1] << 8;
        case 1: t ^= data[0];
    };

    mmix(h,t);
    mmix(h,l);

    h ^= h >> 13;
    h *= m;
    h ^= h >> 15;

    return h;
}
```

Figura 4.3: Algoritmo di hashing

Tale algoritmo utilizza una costante seed impostata a **0x5B2D**, utilizzata come stringa ASCII. Dopo aver trovato l'indirizzo dell'API, Conti lo aggiunge nel suo array di APIs all'offset sopraindicato, il che aiuta a ridurre il tempo di lookup nella tabella per la risoluzione di eventuali ricerche già effettuate.