



Corso di Digital Forensics

CdLM in Informatica

Università degli Studi di Salerno

Docente: Ugo Fiore

4 – Acquisizione della memoria

Memory Acquisition

Contestualizzazione | 1/4

- **Post Mortem Analysis | Alcune Caratteristiche**

- Analisi di un *dead system* (sistema spento)
- Ripetibilità dell'analisi
- Contenuto della memoria RAM, non presente

- **Live Analysis | Alcune Caratteristiche**

- Analisi di un *live system* (sistema acceso)
- **Analisi della memoria RAM (*memory analysis*)**
 - Analisi dei processi in esecuzione
 - Analisi di alcune attività di rete
 - Ecc.
- Possibile irripetibilità di alcuni passi
- Minimizzare l'impatto sul sistema

Memory Acquisition

Contestualizzazione | 1/4

- **Post Mortem Analysis | Alcune Caratteristiche**

- Analisi di un *dead system* (sistema spento)
- Ripetibilità dell'analisi
- Contenuto della memoria RAM, non presente

Nelle precedenti lezioni, ci siamo occupati di **acquisizione di dati**, da un disco fisso, e dei processi di **file recovery** e di **data carving**

Memory Acquisition

Contestualizzazione | 1/4

Ora, ci focalizzeremo su alcuni degli aspetti principali relativi alla **live forensics**

- **Live Analysis | *Alcune Caratteristiche***
 - Analisi di un *live system* (sistema acceso)
 - **Analisi della memoria RAM (*memory analysis*)**
 - Analisi dei processi in esecuzione
 - Analisi di alcune attività di rete
 - Ecc.
 - Possibile irripetibilità di alcuni passi
 - **Minimizzare** l'impatto sul sistema

Memory Acquisition

Contestualizzazione | 2/4

- Alcuni passi della *live analysis*, potrebbero quindi risultare irripetibili, per via principalmente delle seguenti problematiche:
 - **Problematiche di carattere tecnico**
 - Non è possibile effettuare l'eventuale analisi di alcuni dati, senza alterare (quantomeno in parte) lo stato del sistema (contenuto della memoria RAM, ecc.)
 - **OSSERVAZIONE IMPORTANTE:** Anche la fase di acquisizione di alcuni dati, può provocare l'alterazione dello stato della macchina (ad esempio, l'acquisizione del contenuto della memoria RAM)
 - **Problematiche di carattere temporale**
 - È necessario considerare inoltre che lo stato della macchina, all'atto dell'attività, è frutto del «momento»
 - È estremamente complesso (se non impossibile) riprodurre lo stato

Memory Acquisition

Contestualizzazione | 3/4

- Abbiamo precedentemente osservato che non tutti i dati hanno la stessa volatilità, per cui, come visto, è assolutamente indispensabile definire un **ordine di volatilità (Order Of Volatility – OOV)**
- In virtù della natura volatile, della memoria RAM, i dati in essa contenuti, hanno generalmente **elevata priorità nell'OOV**
 - Risulta quindi importante il processo di acquisizione del contenuto della memoria (***memory acquisition***)

Memory Acquisition

Contestualizzazione | 4/4

- Abbiamo precedentemente osservato che non tutti i dati hanno la stessa volatilità, per cui, come visto, è assolutamente indispensabile definire un **ordine di volatilità (Order Of Volatility – OOV)**
- In virtù della natura volatile, della memoria RAM, i dati in essa contenuti, hanno generalmente elevata priorità nell'OOV
 - Risulta quindi importante il processo di acquisizione del contenuto della memoria (*memory acquisition*)

OSSERVAZIONE

Ci focalizzeremo su due aspetti principali:

- Acquisizione della memoria RAM (*memory acquisition*)
- Analisi della memoria RAM (*memory analysis*)

Memory Acquisition

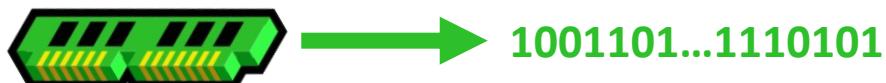
Memory Dump e Processo di Memory Acquisition

- Ricordiamo che un *memory dump* (detto talvolta *memory image*) è una «istantanea» del contenuto della memoria RAM
 - Il processo di acquisizione di un memory dump è detto *memory acquisition* (o *memory imaging*)
- Un memory dump può essere acquisito da tool esterni
- Tuttavia, talvolta, è il S.O. stesso che produce, automaticamente, un memory dump
 - Ad esempio, al verificarsi di un problema grave ed irreversibile, all'interno del sistema, il quale deve necessariamente essere riavviato poiché non può più assicurare un comportamento corretto

Memory Acquisition

Tipologie di Memory Dump

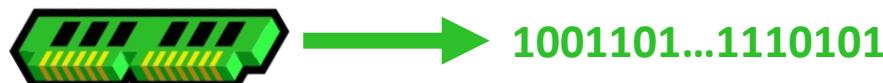
- Esistono **diverse tipologie** di memory dump, fra cui:
 - Memory dump in formato RAW
 - Memory dump prodotti automaticamente dal S.O. [sistemi Windows-based]:
 - Memory dump prodotto in seguito ad un grave crash di sistema
 - Memory dump prodotto dal processo di ibernazione (dove il processo di ibernazione è avviato dall'utente)



Memory Acquisition

Tipologie di Memory Dump

- Esistono diverse tipologie di memory dump, fra cui:
 - **Memory dump in formato RAW**
 - Memory dump prodotti automaticamente dal S.O. [sistemi Windows-based]:
 - Memory dump prodotto in seguito ad un grave crash di sistema
 - Memory dump prodotto dal processo di ibernazione (dove il processo di ibernazione è avviato dall'utente)



Memory Acquisition

Memory Dump | Formato RAW

- **Memory dump in formato RAW**

- **Pro**

- Copia «esatta» del contenuto della memoria RAM
 - Analogia con le immagini forensi, generate dal tool come, ad esempio, D3CDD, ecc.

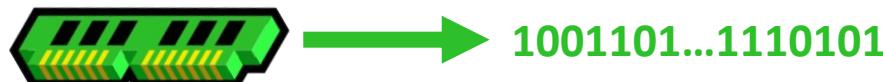
- **Contro**

- Non contiene lo stato del processore (contenuto dei registri)

Memory Acquisition

Tipologie di Memory Dump

- Esistono diverse tipologie di memory dump, fra cui:
 - Memory dump in formato RAW
 - **Memory dump prodotti automaticamente dal S.O. [sistemi Windows-based]:**
 - **Memory dump prodotto in seguito ad un grave crash di sistema**
 - Memory dump prodotto dal processo di ibernazione (dove il processo di ibernazione è avviato dall'utente)



Memory Acquisition

Memory Dump | Prodotto a causa di Gravi Crash | 1/3

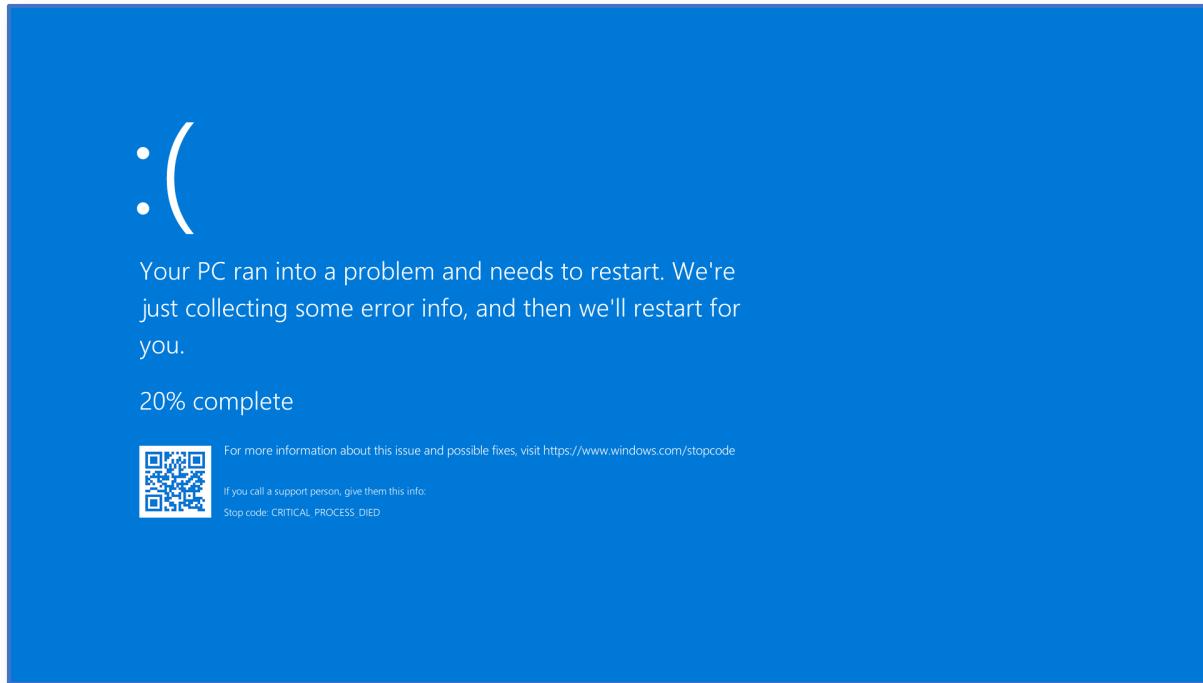
- **Memory dump prodotto in seguito ad un grave crash di sistema [sistemi Windows-based]**
 - Nei sistemi Windows-based, al verificarsi di un grave crash del S.O. (a seguito del quale è necessario effettuare il riavvio del sistema), viene generata una **BSoD** (**Blue Screen of Death**)
 - Nei recenti sistemi Windows-based, dopo aver mostrato all'utente la BSoD, il S.O. produce automaticamente un memory dump
 - È possibile utilizzare tale memory dump, al fine di individuare la possibile causa, che ha condotto al crash del sistema (ad esempio, problema con un applicativo, problema con un driver, ecc.)

Memory Acquisition

Memory Dump | Prodotto a causa di Gravi Crash | 2/3

- **Memory dump prodotto in seguito ad un grave crash di sistema [sistemi Windows-based]**

Esempio di BSOD



Fonte: https://en.wikipedia.org/wiki/Blue_Screen_of_Death#/media/File:Bsodwindows10.png

Memory Acquisition

Memory Dump | Prodotto a causa di Gravi Crash | 3/3

- **Memory dump prodotto in seguito ad un grave crash di sistema [sistemi Windows-based]**
 - **Pro**
 - Include tutte le pagine gestite dal Windows Memory Manager (sistema di gestione della memoria di Windows)
 - Include aspetti relativi all'utente ed al kernel
 - **Contro**
 - Non è disponibile su sistemi a 32 bit
 - Non acquisisce alcuni dati iniziali (ad esempio, password –cifrate– dell'autenticazione, ecc.)

Memory Acquisition

Tipologie di Memory Dump

- Esistono diverse tipologie di memory dump, fra cui:
 - Memory dump in formato RAW
 - **Memory dump prodotti automaticamente dal S.O. [sistemi Windows-based]:**
 - Memory dump prodotto in seguito ad un grave crash di sistema
 - **Memory dump prodotto dal processo di ibernazione (dove il processo di ibernazione è avviato dall'utente)**



Memory Acquisition

Memory Dump | Prodotto dal Processo di Ibernazione | 1/2

- **Memory dump prodotto dal processo di ibernazione [sistemi Windows-based]**

- Grazie al processo di ibernazione (o sospensione), è possibile spegnere il sistema, mantenendo esattamente il suo «stato» (contenuto della memoria RAM/altre memorie volatili, contenuto del disco fisso, ecc.)
 - Alla riattivazione del sistema, verrà ripristinato lo stato del sistema, al momento, immediatamente precedente, dell'avvio dell'ibernazione
 - Durante il processo di ibernazione, viene acquisito anche un memory dump (utilizzato per ripristinare il contenuto della memoria RAM, alla riattivazione del sistema)

Memory Acquisition

Memory Dump | Prodotto dal Processo di Ibernazione | 2/2

- **Memory dump prodotto dal processo di ibernazione [sistemi Windows-based]**
 - **Pro**
 - Memorizza, all'interno del file hiberfil.sys, diverse informazioni:
 - Contenuto dei registri
 - Memory dump
 - **Contro**
 - Disponibile solo per alcune versioni di Windows (da Windows 98 a Windows Vista)
 - Viene eseguita esclusivamente se l'utente richiede l'ibernazione del sistema

Memory Acquisition

Tool Esterni per la creazione di Memory Dump | 1/3

Memory Acquisition

Windows

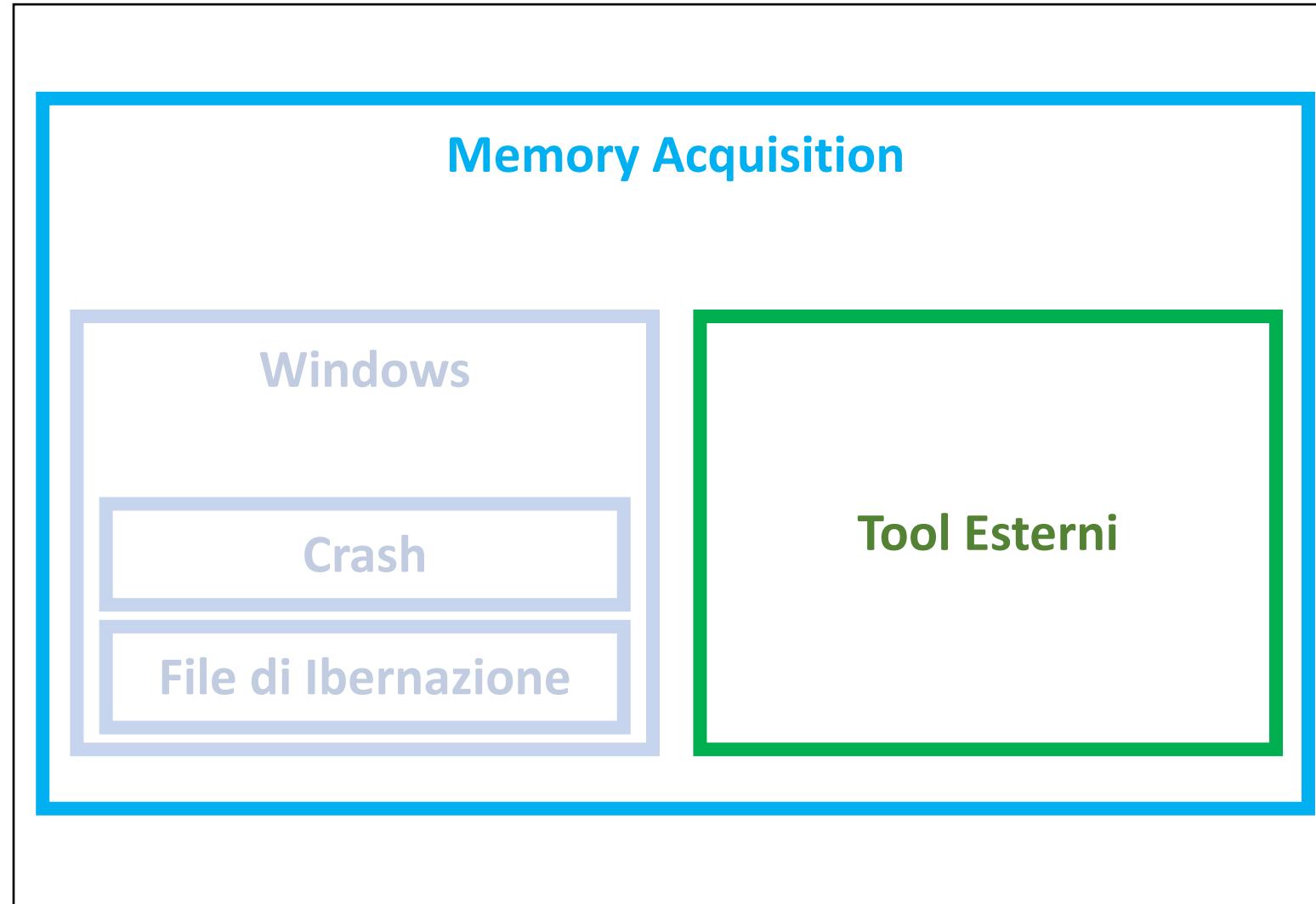
Crash

File di Ibernazione

Tool Esterni

Memory Acquisition

Tool Esterni per la creazione di Memory Dump | 1/3



Memory Acquisition

Tool Esterni per la creazione di Memory Dump | 2/3

- Vi sono molti tool esterni Open-Source e commerciali che permettono la memory acquisition
 - Un tool esterno deve essere eseguito direttamente sul live system
 - **OSSERVAZIONE IMPORTANTE:** L'esecuzione stessa di tale tool impatta sul contenuto della memoria RAM del live system
 - Alcuni tool esterni (per sistemi Windows-based)
 - FTK® Imager
 - Dumpli
 - ...



Memory Acquisition

Tool Esterni per la creazione di Memory Dump | 2/3

- Vi sono molti tool esterni Open-Source e commerciali che permettono la memory acquisition
 - Un tool esterno deve essere eseguito direttamente sul live system
 - **OSSERVAZIONE IMPORTANTE:** L'esecuzione stessa di tale tool impatta sul contenuto della memoria RAM del live system
 - Alcuni tool esterni (per sistemi Windows-based)
 - FTK® Imager
 - DumplIt



Nelle prossime slide, vedremo come è possibile acquisire un *memory dump*, mediante il tool **FTK® Imager** ed il tool **DumplIt**

II tool FTK® Imager

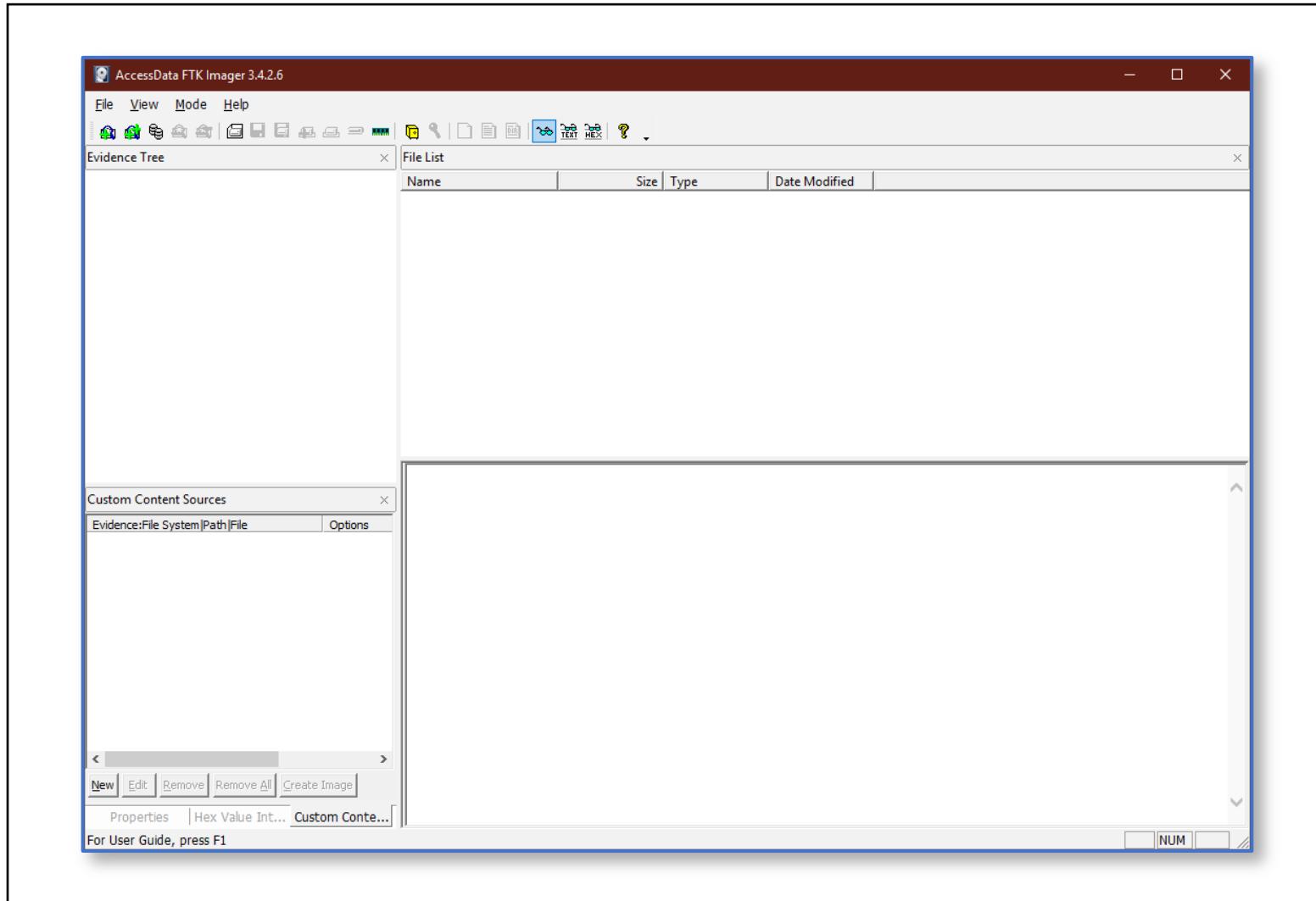
Il tool FTK Imager

Caratteristiche

- **FTK® (Forensic Toolkit®) Imager** è un software sviluppato da AccessData
- Permette di creare immagini forensi (da dischi fissi, CD/DVD, ecc.) e memory dump di differenti supporti:
- Ulteriori dettagli e maggiori informazioni sono reperibili al seguente link:
 - <https://accessdata.com/>
- Benché il tool risulti particolarmente completo ed efficace, verranno brevemente trattati solo degli aspetti relativi alla **memory acquisition**

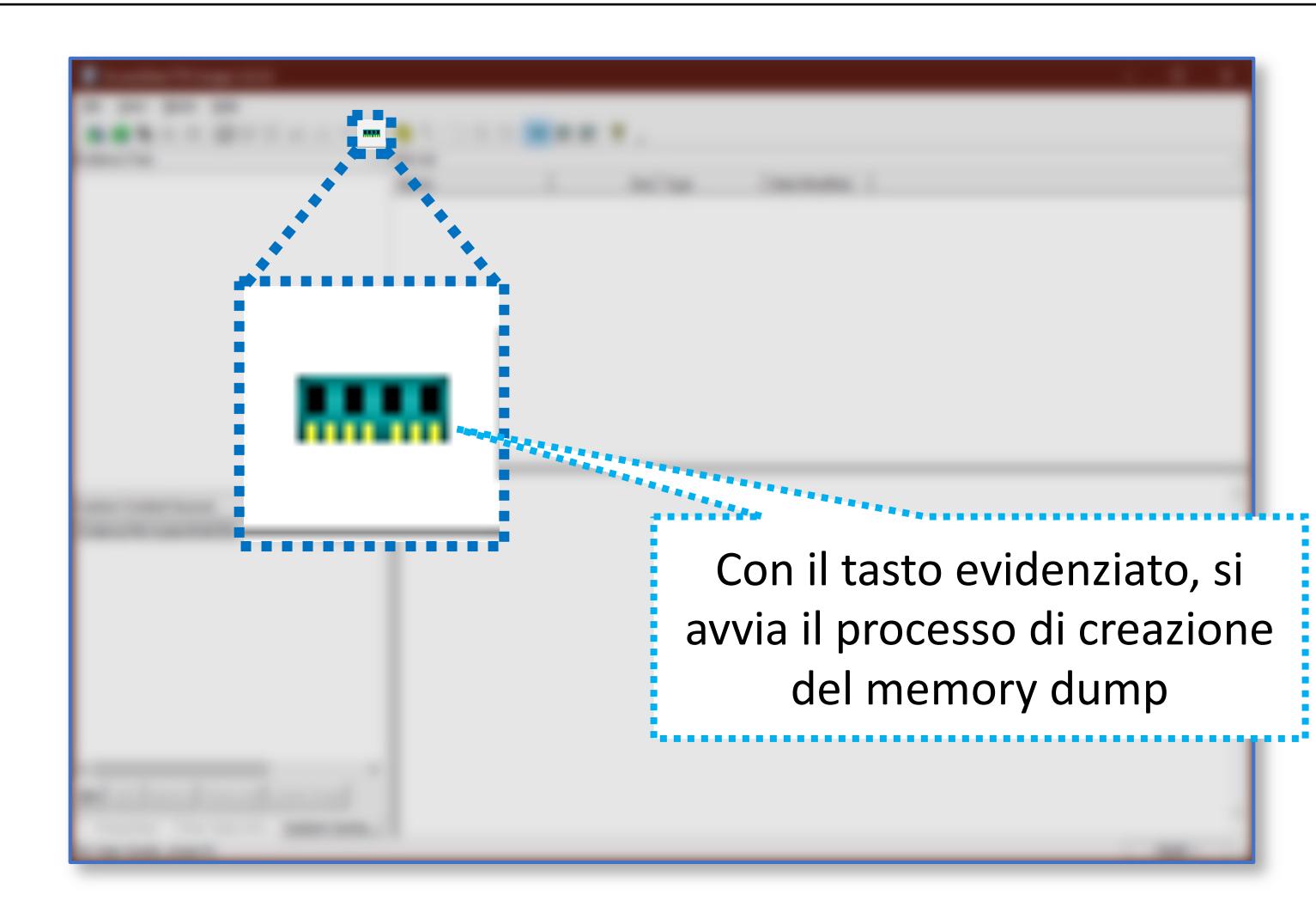
Il tool FTK Imager

Interfaccia Utente



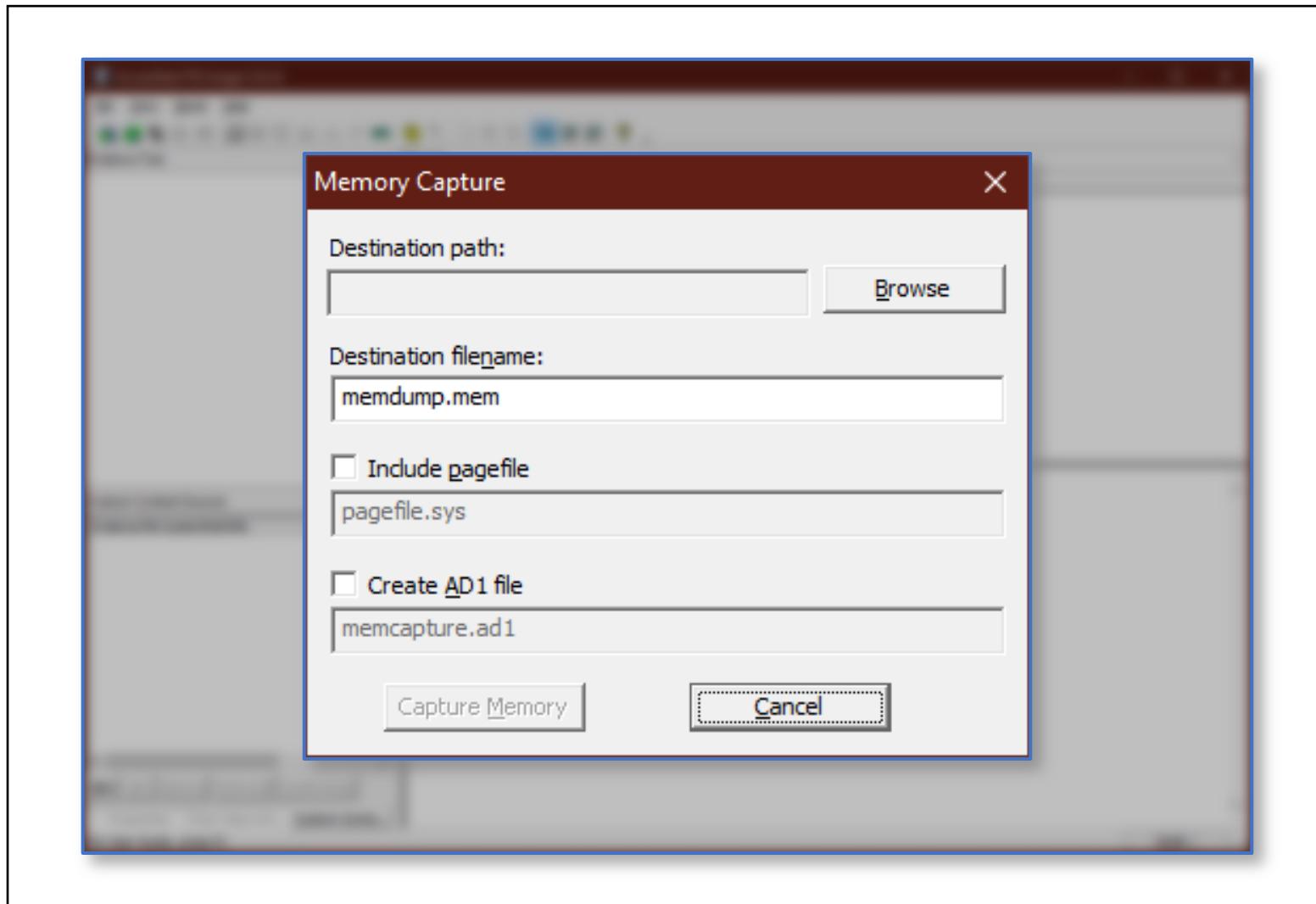
Il tool FTK Imager

Interfaccia Utente



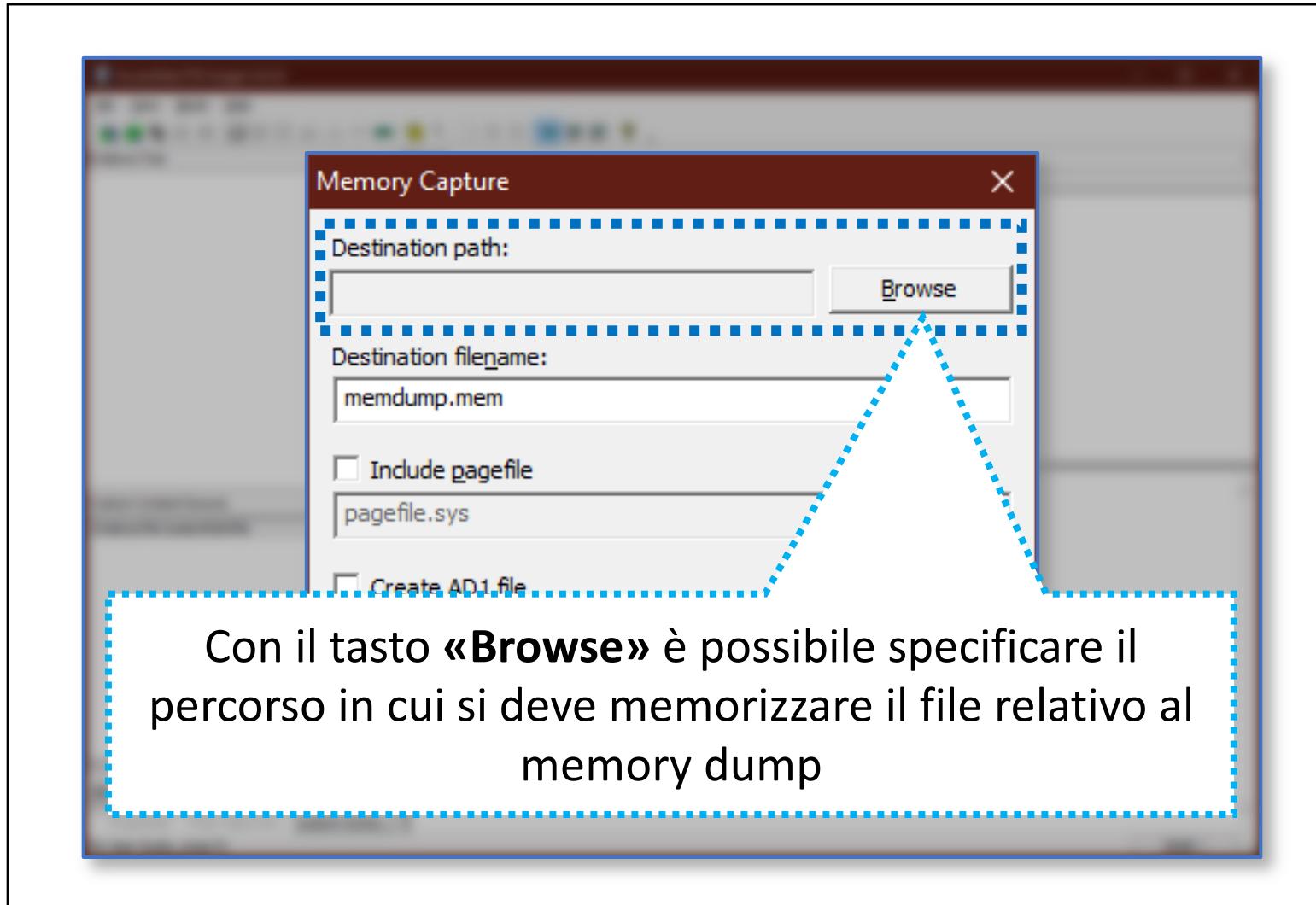
Il tool FTK Imager

Interfaccia Utente



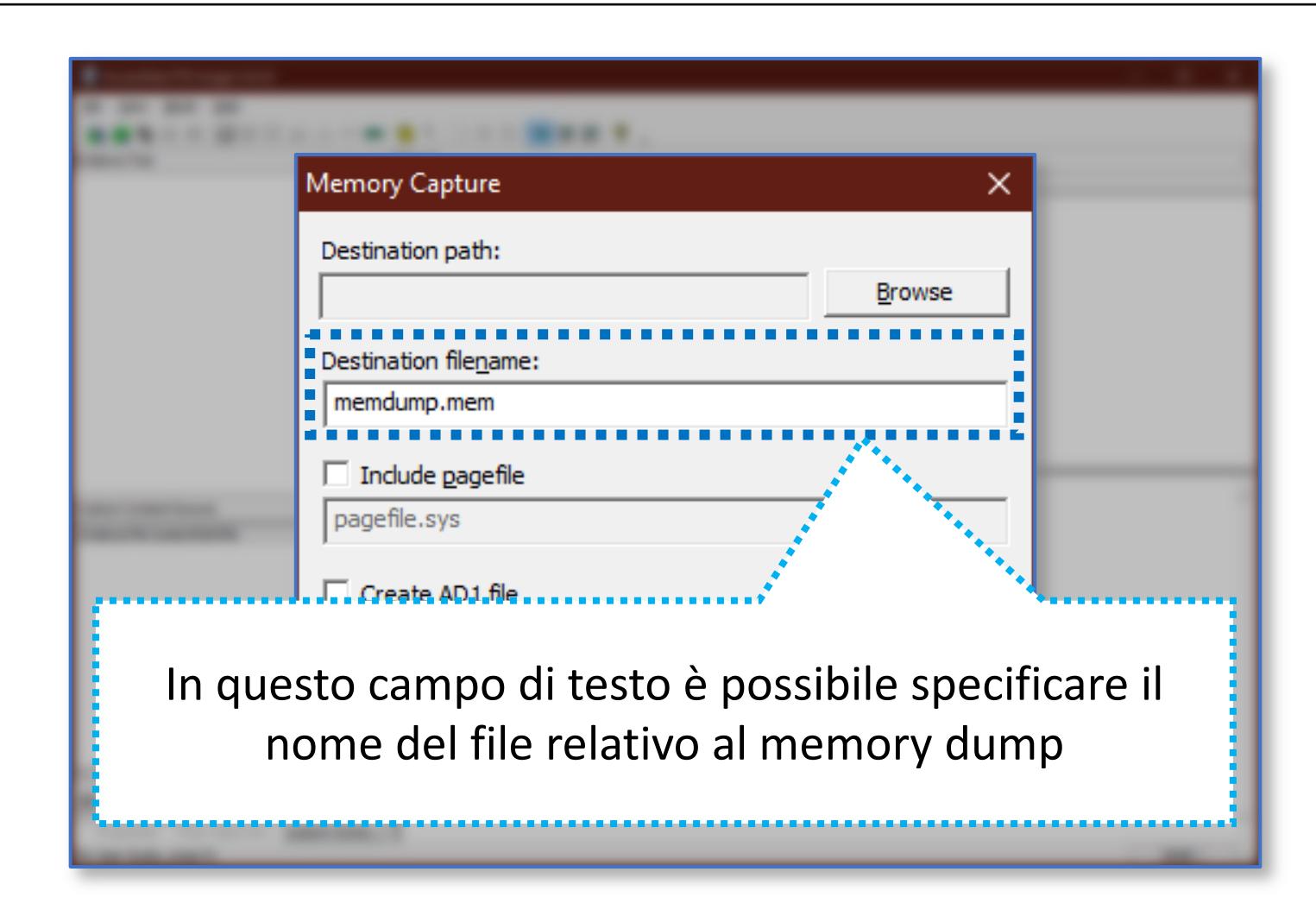
Il tool FTK Imager

Interfaccia Utente



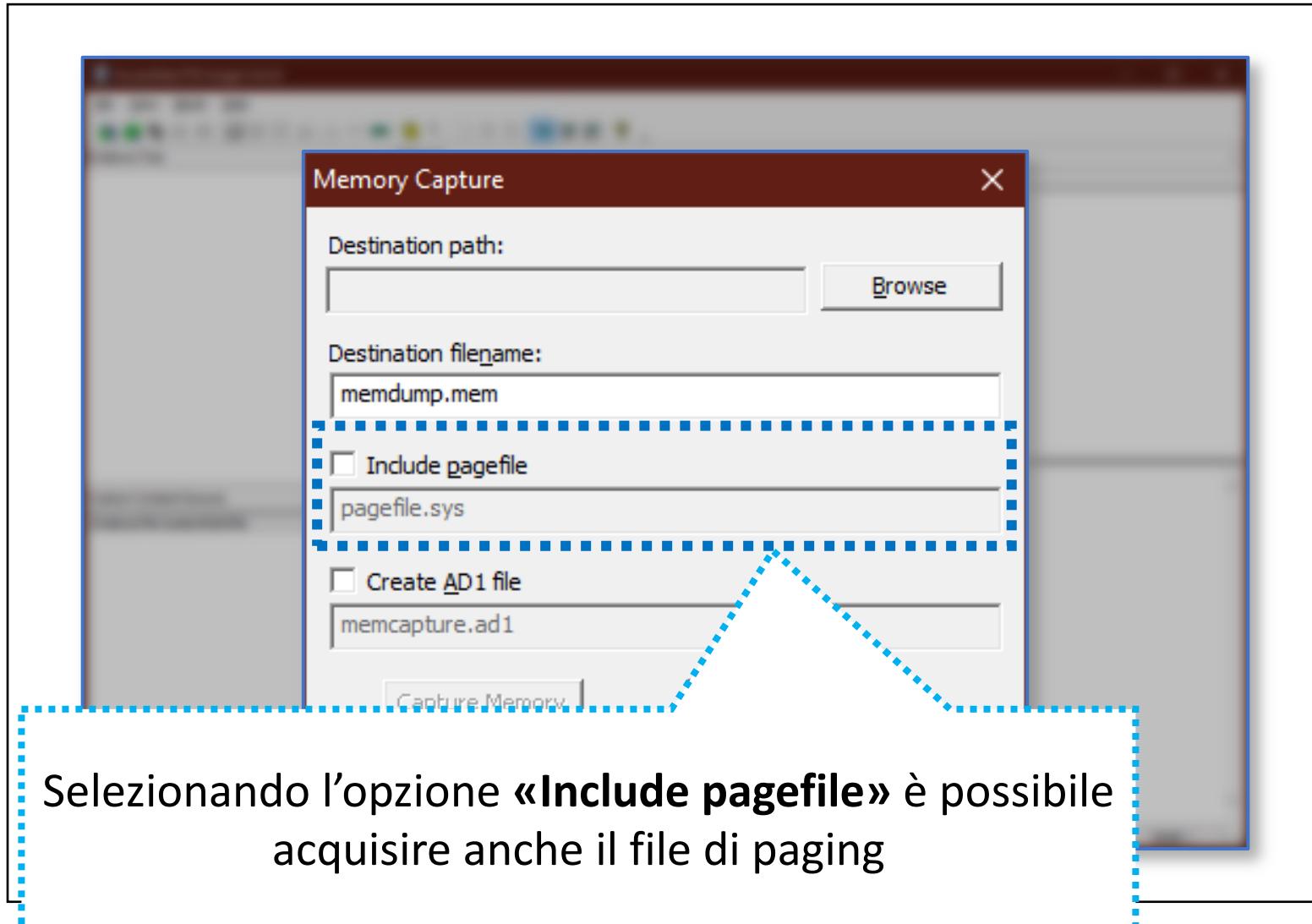
Il tool FTK Imager

Interfaccia Utente



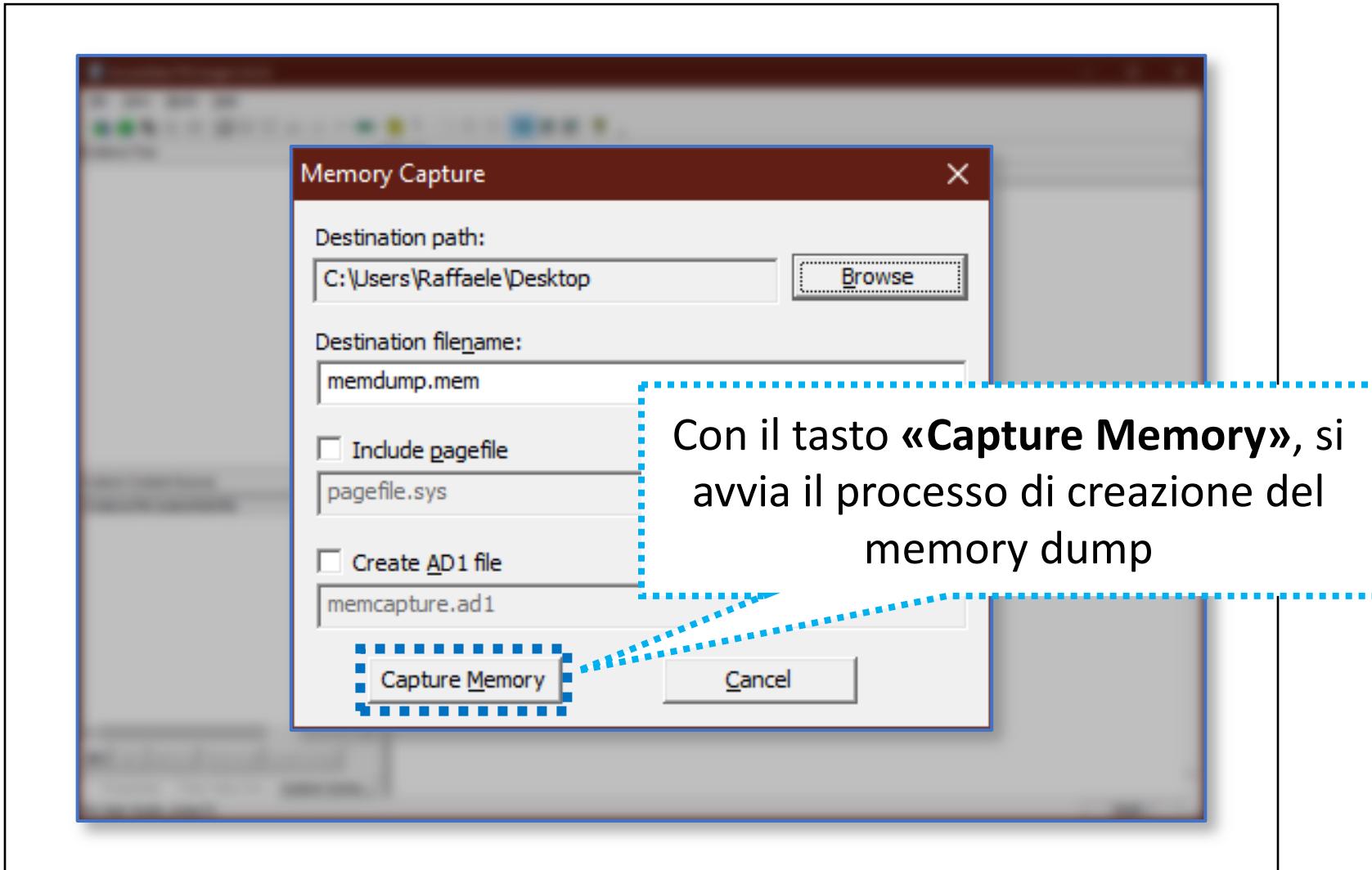
Il tool FTK Imager

Interfaccia Utente



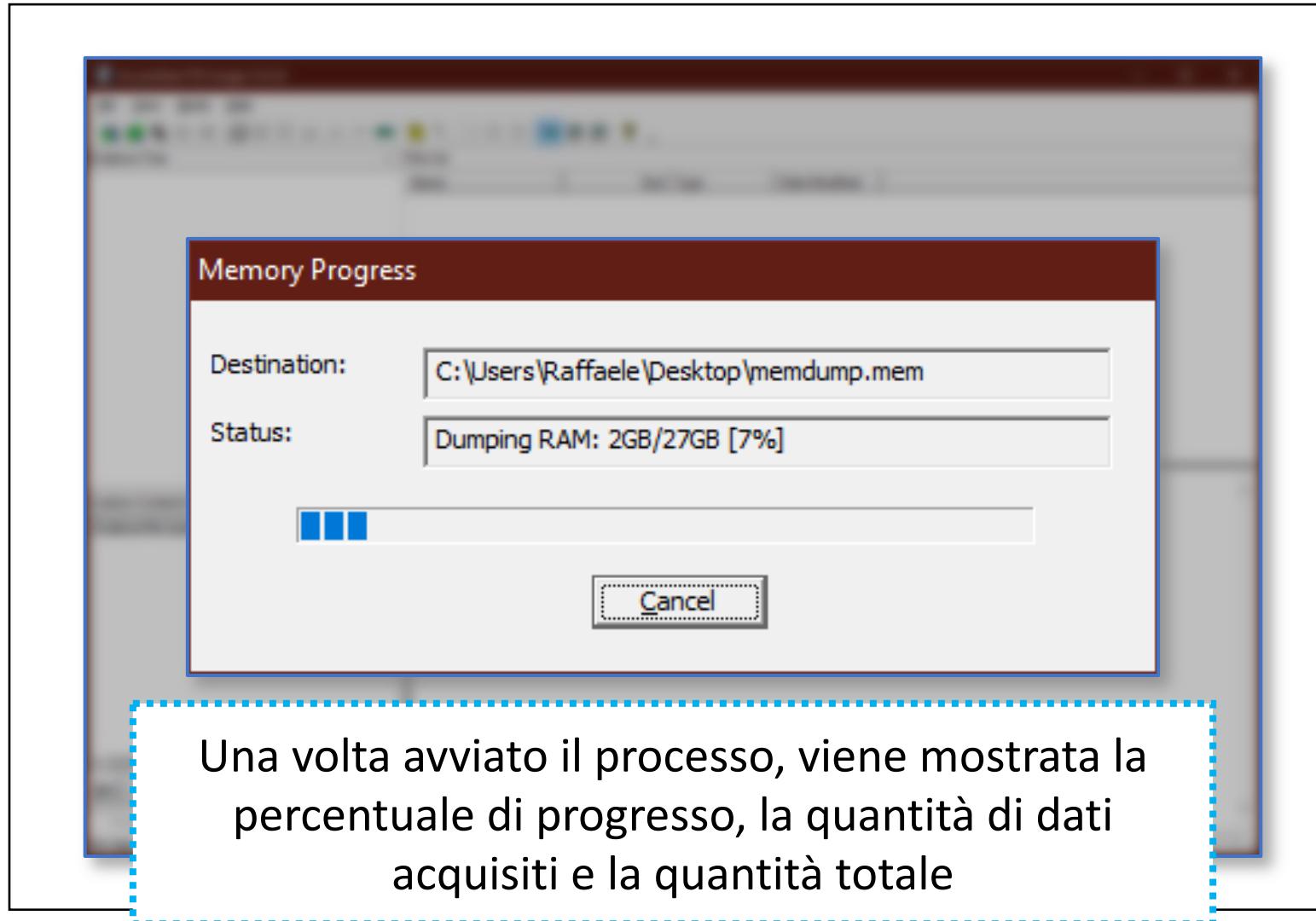
Il tool FTK Imager

Interfaccia Utente



Il tool FTK Imager

Interfaccia Utente



Il tool Dumpl

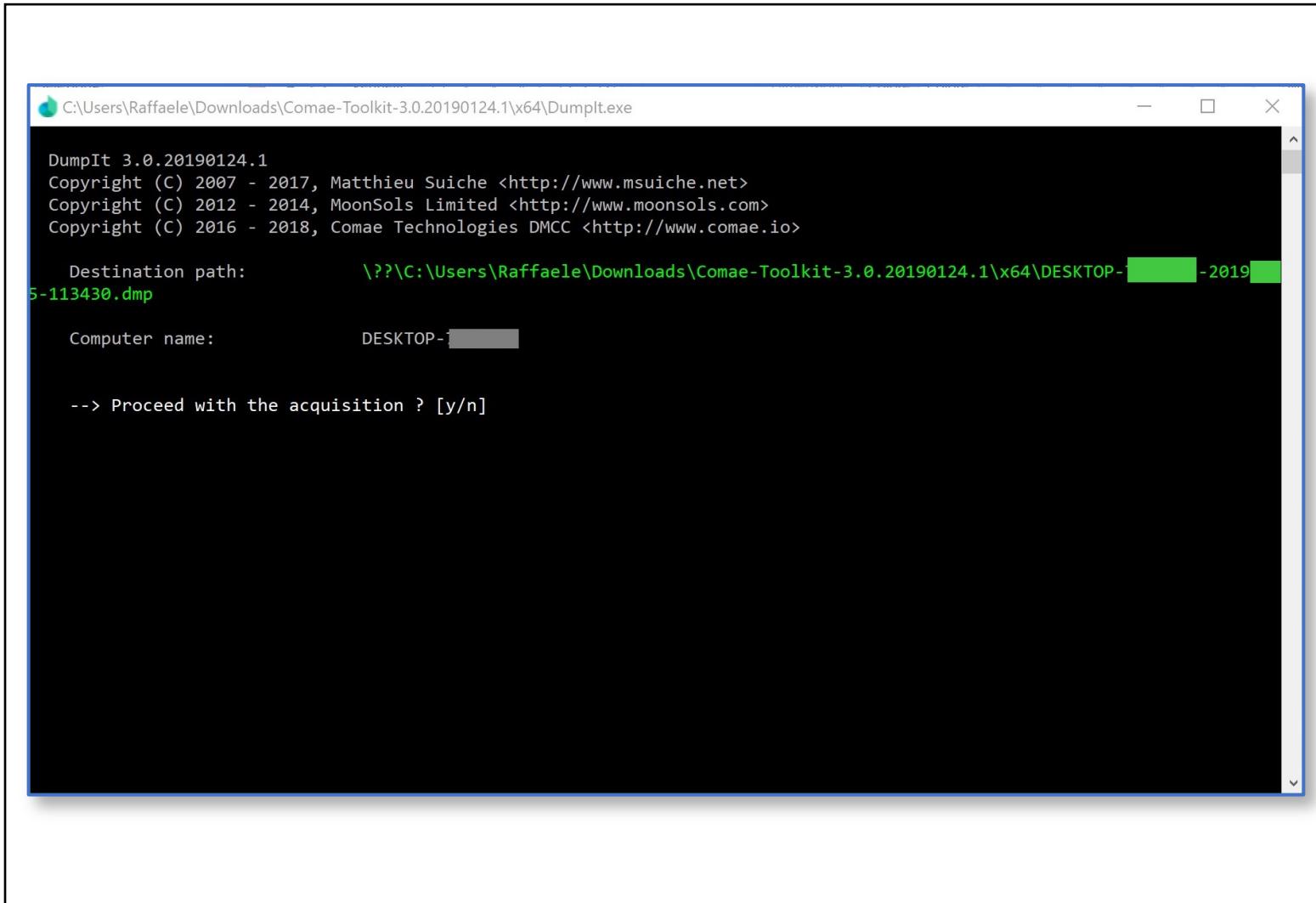
Il tool Dumpl

Caratteristiche

- **Dumpl** è utilizzabile su sistemi a 32 bit (x86) e su sistemi a 64 bit (x64)
- È un tool con **interfaccia a linea di comando**, capace di acquisire memory dump, in formato RAW, e di effettuare la comparazione tramite una funzione di hash (che deve essere specificata da linea di comando)
- Maggiori dettagli ai seguenti link:
 - <https://www.comae.com/>
 - <https://zeltser.com/memory-acquisition-with-dumpit-for-dfir-2/>

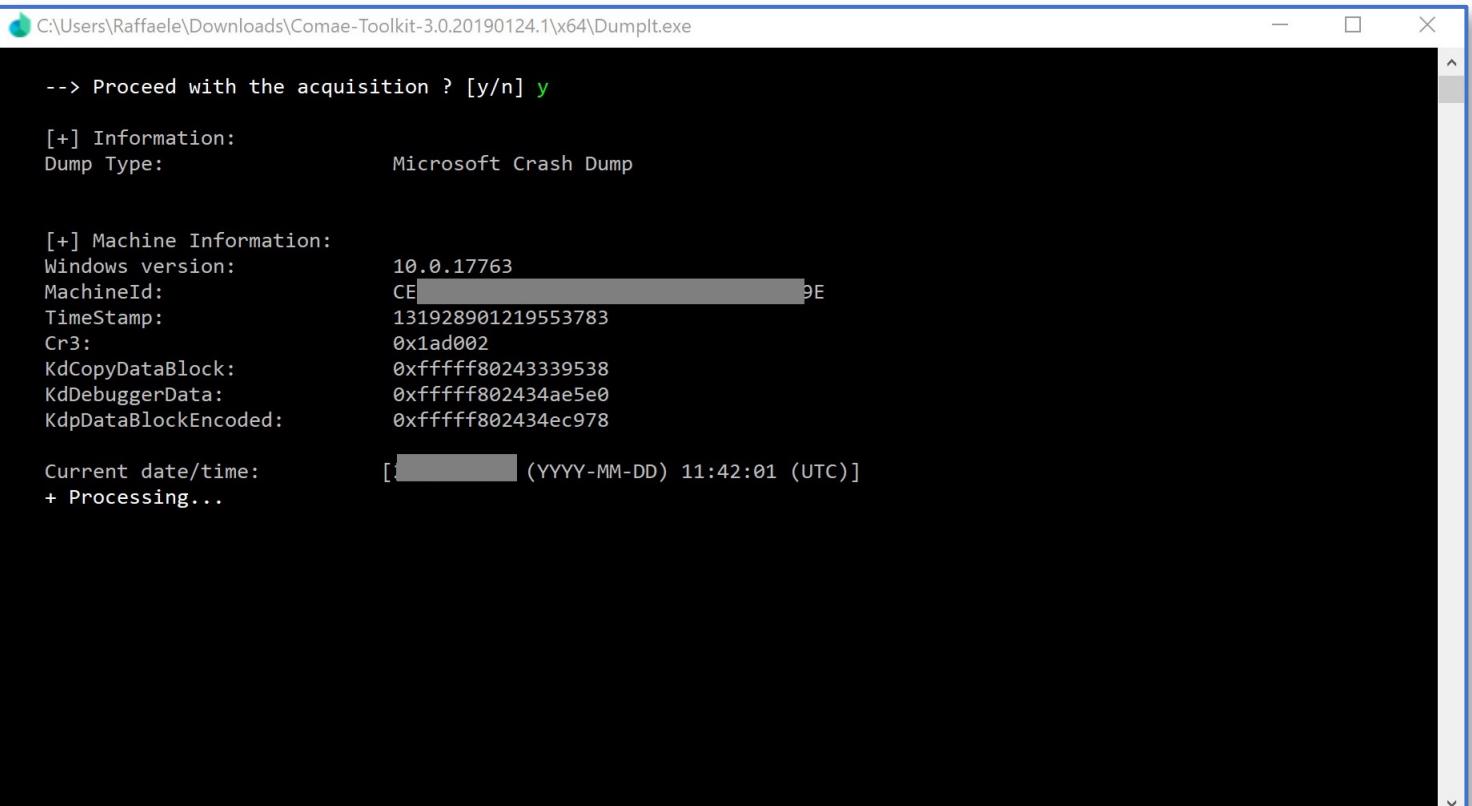
Il tool Dumpl

Interfaccia Utente | 1/2



Il tool Dumpl

Interfaccia Utente | 2/2



The screenshot shows a terminal window titled 'C:\Users\Raffaele\Downloads\Comae-Toolkit-3.0.20190124.1\x64\Dumpit.exe'. The window contains the following text:

```
--> Proceed with the acquisition ? [y/n] y

[+] Information:
Dump Type: Microsoft Crash Dump

[+] Machine Information:
Windows version: 10.0.17763
MachineId: CE[REDACTED]9E
TimeStamp: 131928901219553783
Cr3: 0x1ad002
KdCopyDataBlock: 0xfffffff80243339538
KdDebuggerData: 0xfffffff802434ae5e0
KdpDataBlockEncoded: 0xfffffff802434ec978

Current date/time: [REDACTED] (YYYY-MM-DD) 11:42:01 (UTC)
+ Processing...
```

Salvare la RAM di una macchina virtuale

Vmware e VirtualBox

- Vmware
 - Il file **.vmem** è a corredo di ogni snapshot
- VirtualBox
 - Andrà usata l'opzione **dumpvmcore** di **VBoxManage**
 - Dal file risultante si dovrà poi estrarre l'immagine della memoria in formato raw

Memory Analysis con il Volatility Framework

Volatility Framework

Memory Forensics

- All'interno della RAM (e del file di paging), è possibile individuare diversi dati, potenzialmente rilevanti
- *Esempi:*
 - Password (tipicamente cifrate)
 - Possibili informazioni dell'utente
 - Processi in esecuzione e processi nascosti
 - Ecc.
- Considerati i suddetti aspetti, è estremamente significativo effettuare una **investigazione forense sulla memoria (memory forensics)**

Volatility Framework

Alcune Caratteristiche

- Nelle prossime slide, ci focalizzeremo sull'**analisi della memoria (*memory analysis*)**
 - **OSSERVAZIONE IMPORTANTE:** L'analisi della memoria viene condotta su un **memory dump** (non direttamente sulla memoria)
- Il **Volatility Framework** permette l'**analisi della memoria**
 - È **Open-Source**
 - È **multi-piattaforma** (disponibile per Windows, Linux, macOS/OS X)

Volatility Framework

Avvio del Tool | 1/3

- *Avvio*

```
(kali㉿kali)-[~/Desktop/volatility3]
$ python vol.py
Volatility 3 Framework 2.7.0
usage: volatility [-h] [-c CONFIG] [--parallelism [{processes,threads,off}]] [-e EXTEND] [-p PLUGIN_DIRS] [-s SYMBOL_DIRS] [-v] [-l LOG] [-o OUTPUT_DIR] [-q] [-r RENDERER] [-f FILE] [--write-config] [--save-config SAVE_CONFIG] [--clear-cache] [--cache-path CACHE_PATH] [--offline] [--filters FILTERS] [--single-location SINGLE_LOCATION] [--stackers [STACKERS ...]] [--single-swap-locations [SINGLE_SWAP_LOCATIONS ...]] plugin ...
volatility: error: Please select a plugin to run
```

- All'avvio di Volatility, verranno riportate informazioni sulla versione (nell'esempio, la versione è la 2.7) e alcune delle principali opzioni
- Si noti come il tool richieda la specifica di un plugin

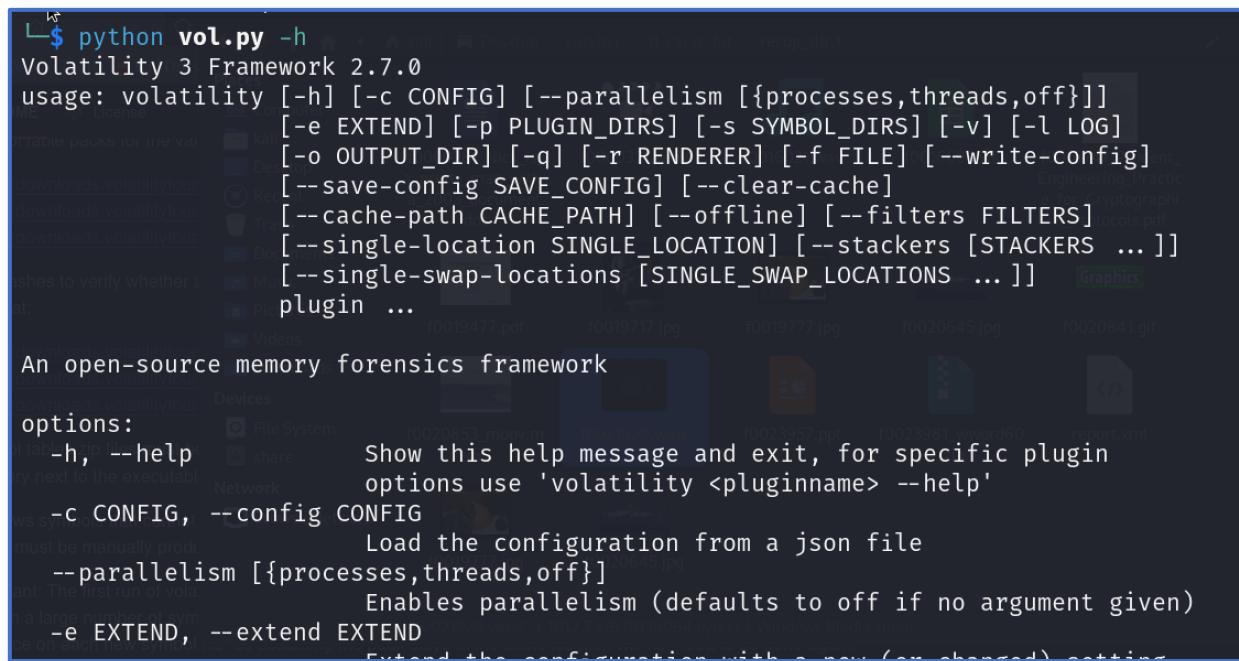
Volatility Framework

Avvio del Tool | 3/3

- È possibile ottenere ulteriori informazioni sulle opzioni, ecc., tramite l'help, visionabile utilizzando l'opzione `-h`, nel modo seguente:

```
python vol.py -h
```

- *Output (Parziale)*



The screenshot shows a terminal window on a Linux desktop environment. The user has run the command `python vol.py -h`. The output is the help documentation for the Volatility 3 Framework version 2.7.0. The help text describes various command-line options such as `-h` for help, `-c` for configuration file, `--parallelism` for parallel processing, and `-e` for extending the configuration. The terminal window is overlaid on a desktop background where several files like PDFs and images are visible.

```
$ python vol.py -h
Volatility 3 Framework 2.7.0
usage: volatility [-h] [-c CONFIG] [--parallelism [{processes,threads,off}]] [-e EXTEND] [-p PLUGIN_DIRS] [-s SYMBOL_DIRS] [-v] [-l LOG] [-o OUTPUT_DIR] [-q] [-r RENDERER] [-f FILE] [--write-config] [--save-config SAVE_CONFIG] [--clear-cache] [--cache-path CACHE_PATH] [--offline] [--filters FILTERS] [--single-location SINGLE_LOCATION] [--stackers [STACKERS ...]] [--single-swap-locations [SINGLE_SWAP_LOCATIONS ...]] plugin ...
An open-source memory forensics framework
options:
-h, --help      Show this help message and exit, for specific plugin
options use 'volatility <pluginname> --help'
-c CONFIG, --config CONFIG
               Load the configuration from a json file
--parallelism [{processes,threads,off}]
               Enables parallelism (defaults to off if no argument given)
-e EXTEND, --extend EXTEND
               Extend the configuration with a new (or changed) setting
```

Volatility Framework

Volatility ed i Plugin

- Volatility permette di estrarre specifiche informazioni dal memory dump su cui si sta conducendo l'analisi, tramite l'utilizzo dei plugin
- Ad esempio, è possibile estrarre informazioni specifiche sui processi, informazioni specifiche sulle attività di rete, ecc.
- Sono supportati molteplici plugin, dal Volatility Framework
 - La lista completa dei plugin (con una descrizione sintetica per ciascun plugin) è specificata dall'help di Volatility (opzione `-h`, vista nella slide precedente)

Volatility Framework

Formati di Memory Dump analizzabili da Volatility

- Volatility è in grado di analizzare **diversi formati** di memory dump, fra i quali:
 - Memory dump, automaticamente prodotti da sistemi Windows-based (relativi a crash/ibernazioni)
 - Memory dump, acquisiti da Virtual Machine
 - Oracle VirtualBox
 - VMWare (aventi estensione .vmem)
 - Formato RAW (estensione .dd, .img, .dmp, .mem, ...)
- Anche i memory dump, acquisiti dai tool, discussi precedentemente, possono essere analizzati da Volatility
- Maggiori informazioni sui formati supportati e su Volatility, sono disponibili al seguente link:
 - <https://www.volatilityfoundation.org/>

Volatility Framework

Utilizzo su Kali Linux | 1/4

- Sintassi Semplificata (*Descrizione*)

```
python vol.py -f <file> plugin [options]
```

Volatility Framework

Utilizzo su Kali Linux | 2/4

- Sintassi Semplificata (*Descrizione*)

```
python vol.py -f <file> plugin [options]
```

- **-f**: Permette di specificare il percorso del **file di input**
 - **OSSERVAZIONE**: il file di input deve essere un memory dump, in uno dei formati supportati

Volatility Framework

Utilizzo su Kali Linux | 3/4

- Sintassi Semplificata (*Descrizione*)

```
python vol.py -f <file> plugin [options]
```

- **-f**: Permette di specificare il percorso del **file di input**
 - **OSSERVAZIONE**: il file di input deve essere un memory dump, in uno dei formati supportati
- **plugin**: Permette di specificare il nome del plugin da utilizzare

Volatility Framework

Utilizzo su Kali Linux | 4/4

- Sintassi Semplificata (*Descrizione*)

```
python vol.py -f <file> plugin [options]
```

- **-f**: Permette di specificare il percorso del **file di input**
 - **OSSERVAZIONE**: il file di input deve essere un memory dump, in uno dei formati supportati
- **plugin**: Permette di specificare il nome del plugin da utilizzare
- **options**: Permette di specificare eventuali opzioni (se necessarie)

Volatility Framework

Profili del Volatility Framework

- L'organizzazione logica della memoria RAM, è strettamente dipendente dalla tipologia del S.O.
 - *Esempio:* Un sistema Windows-based avrà una diversa organizzazione logica, della memoria RAM, rispetto a un sistema Linux-based

Volatility Framework

Plugin Supportati

- *Lista dei plugin supportati*

Plugins:

For plugin specific options, run 'volatility <plugin> --help'

Volatility Framework

Plugin Supportati

- *Lista dei plugin supportati | 1/6*

```
banners.Banners
configwriter.ConfigWriter
frameworkinfo.FrameworkInfo
isfinfo.IsfInfo
layerwriter.LayerWriter
linux.bash.Bash
linux.capabilities.Capabilities
linux.check_afinfo.Check_afinfo
linux.check_creds.Check_creds
linux.check_idt.Check_idt
linux.check_modules.Check_modules
linux.check_syscall.Check_syscall
linux.elfs.Elfss
linux.envars.Envvars
linux.iomem.IOMem
linux.keyboard_notifiers.Keyboard_notifiers
linux.kmsg.Kmsg
linux.library_list.LibraryList
linux.lsmod.Lsmod
linux.lsof.Lsof
```

Volatility Framework

Plugin Supportati

- *Lista dei plugin supportati | 2/6*

```
linux.malfind.Malfind
linux.mountinfo.MountInfo
linux.procMaps
linux.psaux.PsAux
linux.pslist.PsList
linux.psscan.PsScan
linux.pstree.PsTree
linux.sockstat.Sockstat
linux.tty_check.tty_check
linux.vmayarascan.VmaYaraScan
mac.bash.Bash
mac.check_syscall.Check_syscall
mac.check_sysctl.Check_sysctl
mac.check_trap_table.Check_trap_table
mac.dmesg.Dmesg
mac.ifconfig.Ifconfig
mac.kauth_listeners.Kauth_listeners
mac.kauth_scopes.Kauth_scopes
mac.kevents.Kevents
mac.list_files.List_Files
mac.lsmod.Lsmod
```

Volatility Framework

Plugin Supportati

- *Lista dei plugin supportati | 3/6*

```
mac.lssof.Lsof
mac.malfind.Malfind
mac.mount.Mount
mac.netstat.Netstat
mac.proc_maps.Maps
mac.psaux.Psaux
mac.pslist.PsList
mac.pstree.PsTree
mac.socket_filters.Socket_filters
mac.timers.Timers
mac.trustedbsd.Trustedbsd
mac.vfsevents.VFSevents
timeliner.Timeliner
windows.bigpools.BigPools
windows.cachedump.Cachedump
windows.callbacks.Callbacks
windows.cmdline.CmdLine
windows.crashinfo.Crashinfo
windows.devicetree.DeviceTree
windows.dlllist.DllList
```

Volatility Framework

Plugin Supportati

- *Lista dei plugin supportati | 4/6*

```
windows.driverirp.DriverIrp  
windows.drivermodule.DriverModule  
windows.driverscan.DriverScan  
windows.dumpfiles.DumpFiles  
windows.envars.Envars  
windows.filescan.FileScan  
windows.getservicesids.GetServiceIDs  
windows.getsids.GetIDs  
windows.handles.Handles  
windows.hashdump.Hashdump  
windows.iat.IAT  
windows.info.Info  
windows.joblinks.JobLinks  
windows.ldrmodules.LdrModules  
windows.lsadump.Lsadump  
windows.malfind.Malfind  
windows.mbrscan.MBRScan  
windows.memmap.Memmap  
windows.mftscan.ADS  
windows.mftscan.MFTScan
```

Volatility Framework

Plugin Supportati

- *Lista dei plugin supportati | 5/6*

```
windows.modscan.ModScan
windows.modules.Modules
windows.mutantscan.MutantScan
windows.netscan.NetScan
windows.netstat.NetStat
windows.poolscanner.PoolScanner
windows.privileges.Privs
windows.pslist.PsList
windows.psscan.PsScan
windows.pstree.PsTree
windows.registry.certificates.Certificates
windows.registry.hivelist.HiveList
windows.registry.hivescan.HiveScan
windows.registry.printkey.PrintKey
windows.registry.userassist.UserAssist
windows.sessions.Sessions
windows.skeleton_key_check.Skeleton_Key_Check
windows.ssdt.SSDT
windows.statistics.Statistics
windows.strings.Strings
```

Volatility Framework

Plugin Supportati

- *Lista dei plugin supportati | 6/6*

```
windows.svcscan.SvcScan  
windows.symlinkscan.SymlinkScan  
windows.thrdscan.ThrdScan  
windows.truecrypt.Passphrase  
windows.vadinfo.VadInfo  
windows.vadwalk.VadWalk  
windows.vadyarascan.VadYaraScan  
windows.verinfo.VerInfo  
windows.virtmap.VirtMap  
yarascan.YaraScan
```

Volatility Framework

Memory Dump per il Testing | 1/2

- Nei prossimi esempi, verrà utilizzato un memory dump di esempio, denominato `Ozapftis.vmem`
 - Memory dump acquisito su un sistema con Windows XP, in esecuzione su una macchina virtuale (VMWare)
- Tale memory dump è gratuitamente scaricabile al seguente link:
 - <https://github.com/volatilityfoundation/volatility/wiki/Memory-Samples>



GrrCon forensic challenge ISO (also see PDF questions)	Windows XP x86
Malware Cookbook DVD	Black Energy, CoreFlood, Laqma, Prolaco, Sality, Silent Banker, Tigger, Zeus, etc
Malware - Cridex	Windows XP SP2 x86
Malware - Shylock	Windows XP SP3 x86
Malware - R2D2 (pw: infected)	Windows XP SP2 x86

Volatility Framework

Memory Dump per il Testing | 1/2

OSSERVAZIONE

Al link (ottenibile anche mediante il relativo QR Code) indicato precedentemente sono presenti diversi memory dump, su cui è possibile effettuare testing con il Volatility Framework



GrrCon forensic challenge ISO (also see PDF questions)	Windows XP x86
Malware Cookbook DVD	Black Energy, CoreFlood, Laqma, Prolaco, Sality, Silent Banker, Tigger, Zeus, etc
Malware - Cridex	Windows XP SP2 x86
Malware - Shylock	Windows XP SP3 x86
Malware - R2D2 (pw: infected)	Windows XP SP2 x86

Volatility Framework

Memory Dump per il Testing | 2/2

- Nei prossimi esempi, verrà utilizzato un memory dump di esempio, denominato Ozapftis.vmem
 - Memory dump acquisito su un sistema con Windows XP, in esecuzione su una macchina virtuale (VMWare)
- Tale memory dump è gratuitamente scaricabile al seguente link:
 - <https://github.com/volatilityfoundation/volatility/wiki/Memory-Samples>
- **NOTA:** Il memory dump verrà scaricato in formato ZIP ed è necessario estrarlo prima di procedere (dimensioni del file: ~32MB, in formato ZIP, e ~256MB una volta estratto)

Volatility Framework

Profili Suggeriti | Plugin windows.info | 1/2

- Utilizziamo il plugin windows.info, per avere indicazioni sui profili suggeriti da Volatility (simulando di non disporre di alcuna informazione, in merito al memory dump, denominato Ozapftis.vmem)
- *Esempio di Utilizzo*

```
python vol.py -f Ozapftis.vmem windows.info
```

Volatility Framework

Profili Suggeriti | Plugin windows.info | 1/2

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.info
```

- *Output | 1/2*

Variable	Value
Kernel Base	0x804d7000
DTB	0x319000
Symbols	file:///home/kali/Desktop/volatility3/volatility3/symbols/windows/ntkrnlppa.pdb/BD8F451F3E754ED8A34B50560CEB08E3-1.json.xz
Is64Bit	False
IsPAE	True
Layer_name	0 WindowsIntelPAE
memory_layer	1 FileLayer
KdDebuggerDataBlock	0x80544ce0
NTBuildLab	2600.xpsp_sp2_rtm.040803-2158
CSDVersion	2
KdVersionBlock	0x80544cb8
Major/Minor	15.2600

Volatility Framework

Profili Suggeriti | Plugin windows.info | 1/2

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.info
```

- *Output | 2/2*

```
MachineType      332
KeNumberProcessors    1
SystemTime       2011-10-10 17:06:54
NtSystemRoot     C:\WINDOWS
NtProductType   NtProductWinNt
NtMajorVersion  5
NtMinorVersion  1
PE MajorOperatingSystemVersion 5
PE MinorOperatingSystemVersion 1
PE Machine       332
PE TimeStamp      Wed Aug  4 05:58:36 2004
```

Volatility Framework

Profili Suggeriti | Plugin windows.info | 1/2

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.info
```

- *Output | 2/2*

```
MachineType      332
KeNumberProcessors 1
SystemTime       2011-10-10 17:06:54
NtSystemRoot     C:\WINDOWS
NtProductType    NtProductWinNt
NtMajorVersion   5
NtMinorVersion   1
PE MajorOperatingSystemVersion 5
PE MinorOperatingSystemVersion 1
PE Machine        332
PE TimeStamp      Wed Aug  4 05:58:36 2004
```

Viene riportato il numero di processori (o numero di core) presenti nel sistema da cui è stato acquisito il memory dump

Volatility Framework

Categorie di Plugin

- Possiamo suddividere i **plugin** del Volatility Framework, in diverse categorie
- Le categorie su cui ci soffermeremo sono le seguenti:
 - Analisi dei Processi
 - Identificazione di Attività di Rete
 - Analisi di librerie DLL (Dynamic Link Library) di Windows
 - Informazioni sul Registro di Sistema del S.O. Windows
 - Altri plugin

Volatility Framework

Categorie di Plugin

- Possiamo suddividere i **plugin** del Volatility Framework, in diverse categorie
- Le categorie su cui ci soffermeremo sono le seguenti:
 - **Analisi dei Processi**
 - Identificazione di Attività di Rete
 - Analisi di librerie DLL (Dynamic Link Library) di Windows
 - Informazioni sul Registro di Sistema del S.O. Windows
 - Altri plugin

Volatility Framework

Identificazione dei processi

- Il Volatility Framework permette di elencare i processi ed ottenere diverse informazioni su di essi (ad esempio, la data e l'ora in cui il processo è stato avviato, ecc.)
- Per ottenere tali informazioni è possibile utilizzare i seguenti plugin:
 - pslist
 - pstree
 - psscan
- Nelle prossime slide verranno approfonditi questi plugin

Volatility Framework

Identificazione dei processi

- Il Volatility Framework permette di elencare i processi ed ottenere diverse informazioni su di essi (ad esempio, la data e l'ora in cui il processo è stato avviato, ecc.)
- Per ottenere tali informazioni è possibile utilizzare i seguenti plugin:
 - pslist
 - pstree
 - psscan
- Nelle prossime slide verranno approfonditi questi plugin

Volatility Framework

Plugin pslist | 1/8

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.pslist
```

Volatility Framework

Plugin pslist | 2/8

- *Esempio di Utilizzo*

```
python vol.py -f Ozapftis.vmem windows.pslist
```

- *Output*

```
└$ python ./volatility3/vol.py -f Ozapftis.vmem windows.pslist
Volatility 3 Framework 2.7.0
WARNING volatility3.framework.layers.vmware: No metadata file found
alongside VMEM file. A VMSS or VMSN file may be required to correctly
process a VMEM file. These should be placed in the same directory with
the same file name, e.g. Ozapftis.vmem and Ozapftis.vmss.
Progress: 100.00          PDB scanning finished
PID      PPID      ImageFileName      Offset(V)      Threads Handles SessionId
Wow64      CreateTime      ExitTime      File output

4        0        System      0x819cc830      55        162      N/A      False
N/A      N/ADisabled
536      4        smss.exe      0x81945020      3         21       N/A
False    2011-10-10 17:03:56.000000  N/A      Disabled
608      536      csrss.exe      0x816c6020      11        355      0
False    2011-10-10 17:03:58.000000  N/A      Disabled
632      536      winlogon.exe    0x813a9020      24        533      0
False    2011-10-10 17:03:58.000000  N/A      Disabled
```

Volatility Framework

Plugin pslist | 2/8

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.pslist
```

- *Output*

```
└$ python ./volatil
Volatility 3 Framework
WARNING volatility3.f
alongside VMMEM file. A
process a VMMEM file. T
the same file name, e.
Progress: 100.00
```

Mediante il plugin pslist, viene mostrata la lista dei processi (sia avviati direttamente dal sistema sia avviati dall'utente)

PID Wow64	PPID CreateTime	ImageFileName	Set(V) ExitTime	Threads Handles SessionId			
				File output			
4	0	System	0x819cc830	55	162	N/A	False
N/A	N/A	Adisabled					
536	4	smss.exe	0x81945020	3	21	N/A	
False	2011-10-10 17:03:56.000000	N/A		Disabled			
608	536	csrss.exe	0x816c6020	11	355	0	
False	2011-10-10 17:03:58.000000	N/A		Disabled			
632	536	winlogon.exe	0x813a9020	24	533	0	
False	2011-10-10 17:03:58.000000	N/A		Disabled			

Volatility Framework

Plugin pslist | 2/8

- *Esempio di Utilizzo*

```
python vol.py -f Ozapftis.vmem windows.pslist
```

- *Output*

```
└$ python ./volatility3/vol.py -f Ozapftis.vmem windows.pslist
Volatility 3 Framework 2.7.0
WARNING volatility3.framework.layers.vmware: No metadata file found
alongside VMEM file. A VMSS or VMSN file may be required to correctly
process a VMEM file. These should be placed in the same directory with
the same file name, e.g. Ozapftis.vmem and Ozapftis.vmss.
Progress: 100.00          PDB scanning finished
PID      PPID      ImageFileName      Offset(V)      Threads Handles SessionId
Wow64      N/A      C:\Windows\...      0x0000000000000000      0      0      0
4
N/A
536
False
608
False
632
False
```

Per ciascuno dei processi, vengono mostrati anche:

- PID (Process ID)
 - Riferito al processo
- PPID (Parent Process ID)
 - Riferito al processo padre

Volatility Framework

Plugin pslist | 2/8

- *Esempio di Utilizzo*

```
python vol.py -f Ozapftis.vmem windows.pslist
```

- *Output*

```
└$ python ./volatility3/vol.py -f Ozapftis.vmem windows.pslist
Volatility 3 Framework 2.7.0
WARNING volatility3.framework.layers.vmware: No metadata file found
alongside VMEM file. A VMSS or VMSN file may be required to correctly
process a VMEM file. These should be placed in the same directory with
the same file name, e.g. Ozapftis.vmem and Ozapftis.vmss.
Progress: 100.00          PDB scanning finished
PID      PPID     ImageFileName   Offset(V)      Threads Handles SessionId
Wow64    : CreateTime       ExitTime        File output
4        0           S... 0x819cc830      55       162      N/A    False
N/A      N/A Disabl...               0x81945020      3        21      N/A
536      4           sm...               0x81945020      3        21      N/A
False
608
False
632
False
```

Per ciascuno dei processi viene inoltre riportata anche la data e l'ora in cui esso è stato avviato (colonna: CreateTime)

Volatility Framework

Plugin pslist | 3/8

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.pslist
```

- *Breve Descrizione dei Principali Processi di Sistema Individuati | 1/2*

Nome Processo	Descrizione (Cenni)
smss.exe	Acronimo di S ession M anager S ub S ystem Responsabile dell'avvio della sessione utente; è inizializzato da Windows (termina generalmente quando il sistema viene spento). Inoltre, avvia il processo Winlogon (winlogon.exe) ed il processo csrss.exe
csrss.exe	Acronimo di C lient/ S erver R un-time S ub S ystem Responsabile della gestione delle applicazioni del prompt dei comandi (terminale), ecc.

Volatility Framework

Plugin pslist | 3/8

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.pslist
```

- *Breve Descrizione dei Principali Processi di Sistema Individuati | 2/2*

Nome Processo	Descrizione (Cenni)
winlogon.exe	Si occupa della fase di autenticazione (logon)/disconnessione in Windows (contrazione di Windows Logon)
scvhost.exe	Processo host generico che permette l'esecuzione (e download) dei servizi di Windows, memorizzati nelle librerie DLL (<i>maggiori dettagli sulle librerie DLL, in seguito</i>)
explorer.exe	Esegue il Windows Program Manager (o Windows Explorer), ovvero, il gestore dell'interfaccia grafica di Windows
spoolsv.exe	Esegue il servizio di spooling della stampante, ovvero un servizio che si occupa del caching dei lavori in fase di stampa, ecc. (contrazione di Spooler Service)

Volatility Framework

Plugin pslist | 4/8

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.pslist
```

- *Breve Descrizione dei Principali Processi di Sistema Individuati | 2/2*

Nome Processo	Descrizione (Cenni)
winlogon.exe	Si occupa della fase di autenticazione (logon)/disconnessione in Windows (contrazione di Windows Logon)
scvhost.exe	Processo host generico che permette l'esecuzione (e download) dei servizi di Windows, memorizzati nelle librerie DLL (<i>maggiori dettagli sulle librerie DLL, in seguito</i>)
explorer.exe	Esegue il Windows Program Manager (o Windows Explorer), ovvero, il gestore dell'interfaccia grafica di Windows
spoolsv.exe	Esegue il servizio di spooling della stampante, ovvero un servizio che si occupa del caching dei lavori in fase di stampa, ecc. (contrazione di Spooler Service)

Maggiori dettagli ed informazioni riguardo i processi di Microsoft Windows possono essere trovati al seguente link: <https://www.file.net/process/>

Volatility Framework

Plugin pslist | 5/8

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.pslist
```

- *Output*

```
└$ python ../volatility3/vol.py -f 0zapftis.vmem windows.pslist
Volatility 3 Framework 2.7.0
WARNING volatility3.framework.layers.vmware: No metadata file found
along
proce
the s
Prog
PID
Wow64
```

Nelle prossime slide, verranno esplicata
alcune osservazioni sull'output

Volatility Framework

Plugin pslist | 6/8

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.pslist
```

- OSSERVAZIONE 1 | 1/3

Nome Processo	PID	PPID
winlogon.exe	632	536
services.exe	676	632
lsass.exe	688	632
svchost.exe	848	676
svchost.exe	916	676
svchost.exe	964	676
svchost.exe	1020	676
svchost.exe	1148	676

Volatility Framework

Plugin pslist | 6/8

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.pslist
```

- **OSSERVAZIONE 1 | 1/3**

Nome Processo	PID	PPID
winlogon.exe	632	536
services.exe	676	632
lsass.exe	688	632
svchost.exe	848	676
svchost.exe	916	676
svchost.exe	964	676
svchost.exe	1020	676
svchost.exe	1148	676

I processi services.exe e lsass.exe (aventi PID rispettivamente pari a 676 e 688) sono processi figli del processo winlogon.exe (come è possibile vedere dal PPID pari a 632, uguale al PID di winlogon.exe)

Volatility Framework

Plugin pslist | 6/8

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.pslist
```

- **OSSERVAZIONE 1 | 1/3**

Nome Processo	PID	PPID
winlogon.exe	632	536
services.exe	676	632
lsass.exe	688	632
svchost.exe	848	676
svchost.exe	916	676
svchost.exe	964	676
svchost.exe	1020	676
svchost.exe	1148	676

I cinque processi svchost.exe (aventi PID rispettivamente pari a 848, 916, 964, 1020 e 1148) sono processi figli di services.exe (il quale è a sua volta un processo figlio del processo winlogon.exe, come visto precedentemente)

Volatility Framework

Plugin pslist | 7/8

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.pslist
```

- OSSERVAZIONE 2 | 1/3

Nome Processo	PID	PPID	Start (Parziale)
explorer.exe	1956	1884	17:04:39
spoolsv.exe	1260	676	17:04:00
reader_sl.exe	228	1956	17:04:41

Volatility Framework

Plugin pslist | 7/8

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.pslist
```

- OSSERVAZIONE 2 | 1/3

Nome Processo	PID	PPID	Start (Parziale)
explorer.exe	1956	1884	17:04:39
spoolsv.exe	1260	676	17:04:00
reader_sl.exe	228	1956	17:04:41

I processi `explorer.exe`, `spoolsv.exe` e `reader_sl.exe` sono stati avviati a breve distanza

Volatility Framework

Plugin pslist | 7/8

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.pslist
```

- OSSERVAZIONE 2 | 1/3

Nome Processo	PID	PPID	Start (Parziale)
explorer.exe	1956	1884	17:04:39
spoolsv.exe	1260	676	17:04:00
reader_sl.exe	228	1956	17:04:41

Il processo reader_sl.exe (con PID pari a 228) è un processo figlio del processo explorer.exe

Volatility Framework

Plugin pslist | 8/8

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.pslist
```

- OSSERVAZIONE 3 | 1/2

Nome Processo	PID	PPID
svchost.exe	964	676
wuauclt.exe	400	964

Volatility Framework

Plugin pslist | 8/8

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.pslist
```

- OSSERVAZIONE 3 | 1/2

Nome Processo	PID	PPID
svchost.exe	964	676
wuauclt.exe	400	964

Il processo wuauclt.exe (avente PID 400) è figlio del processo svchost.exe (come si può vedere dal valore del PPID)

Volatility Framework

Identificazione dei processi

- Il Volatility Framework permette di elencare i processi ed ottenere diverse informazioni su di essi (ad esempio, la data e l'ora in cui il processo è stato avviato, ecc.)
- Per ottenere tali informazioni è possibile utilizzare i seguenti plugin:
 - pslist
 - **pstree**
 - psscan
- Nelle prossime slide verranno approfonditi questi plugin

Volatility Framework

Plugin pstree | 1/2

- *Esempio di Utilizzo 1 di 2*

```
python vol.py -f 0zapftis.vmem windows.pstree
```

- *Output (parziale)*

PID Wow64	PPID CreateTime	ImageFileName	Offset(V) ExitTime	Threads Audit	Handles Cmd	SessionId Path
4 N/A	0 N/A--	System	0x819cc830 -	55	162 N/A	N/A False
* 536 False	4 2011-10-10 17:03:56.000000	smss.exe	0x81945020 N/A	3	21	N/A
			\Device\HarddiskVolume1\WINDOWS\system32\smss.exe			
			\SystemRoot\System32\smss.exe		\SystemRoot\System32\smss.exe	
** 608 False	536 2011-10-10 17:03:58.000000	csrss.exe	0x816c6020 N/A	11	355 0	
			\Device\HarddiskVolume1\WINDOWS\system32\csrss.exe			
			C:\WINDOWS\system32\csrss.exe	ObjectDirectory=\Windows		
				SharedSection=1024,3072,512	Windows=On	SubSystemType=Windows
				ServerDll=basesrv,1	ServerDll=winsrv:UserServerD1lInitialization,3	
				ServerDll=winsrv:ConServerD1lInitialization,2	ProfileControl=Off	
				MaxRequestThreads=16	\??\C:\WINDOWS\system32\csrss.exe	

Volatility Framework

Plugin **pstree** | 1/2

- *Esempio di Utilizzo 1 di 2*

```
python vol.py -f 0zapftis.vmem windows.pstree
```

- *Output (elaborato)*

PID	PPID	ImageFileName
4	0	System
* 536	4	smss.exe
** 608	536	csrss.exe
** 632	536	winlogon.exe
*** 688	632	lsass.exe
*** 676	632	services.exe
**** 832	676	vmacthlp.exe
**** 964	676	svchost.exe
***** 1920	964	wsctfy.exe
***** 400	964	wuauctl.exe
***** 1444	676	VMwareService.e
***** 1260	676	spoolsv.exe
***** 848	676	svchost.exe
***** 1148	676	svchost.exe
***** 1616	676	alg.exe
***** 916	676	svchost.exe
***** 1020	676	svchost.exe
1956	1884	explorer.exe
* 184	1956	VMwareTray.exe
* 544	1956	cmd.exe
* 228	1956	reader_s1.exe
* 192	1956	VMwareUser.exe

Volatility Framework

Plugin pstree | 1/2

- *Esempio di Utilizzo 1 di 2*

```
python vol.py -f 0zapftis.vmem windows.pstree
```

- *Output (elaborato)*

PID	PPID	ImageFileName
4	0	System
* 536	4	smss.exe
** 608	536	csrss.exe
** 632	536	winlogon.exe
*** 688	632	lsass.exe
*** 676	632	services.exe
**** 832	676	vmacthlp.exe
***** 964	676	svchost.exe
***** 1920	964	wsctfy.exe
***** 400	964	wuauctl.exe
***** 1444	676	VMwareService.e
***** 1260	676	spoolsv.exe
***** 848	676	svchost.exe
***** 1148	676	svchost.exe
***** 1616	676	alg.exe
***** 916	676	svchost.exe
***** 1020	676	svchost.exe
1956	1884	explorer.exe
* 184	1956	VMwareTray.exe
* 544	1956	cmd.exe
* 228	1956	reader_s1.exe
* 192	1956	VMwareUser.exe

Analogamente al plugin pslist, il plugin pstree mostra la lista dei processi

Volatility Framework

Plugin pstree | 1/2

- Esempio di Utilizzo 1 di 2

```
python vol.py -f 0zapftis.vmem windows.pstree
```

- Output (elaborato)

PID	PPID	ImageFileName
4	0	System
* 536	4	smss.exe
** 608	536	csrss.exe
** 632	536	winlogon.exe
*** 688	632	lsass.exe
*** 676	632	services.exe
**** 832	676	vmacthlp.exe
**** 964	676	svchost.exe
***** 1920	964	wscntfy.exe
***** 400	964	wuauctl.exe
***** 1444	676	VMwareService.exe
***** 1260	676	spoolsv.exe
***** 848	676	svchost.exe
***** 1148	676	svchost.exe
***** 1616	676	alg.exe
***** 916	676	svchost.exe
***** 1020	676	svchost.exe
1956	1884	explorer.exe
* 184	1956	VMwareTray.exe
* 544	1956	cmd.exe
* 228	1956	reader_s1.exe
* 192	1956	VMwareUser.exe

OSSERVAZIONE

In questo caso, la lista è ad «albero», infatti, vengono indentati i processi figli, in modo che risulti più immediato distinguere i processi padri ed i processi figli

Volatility Framework

Plugin pstree | 1/2

- *Esempio di Utilizzo 1 di 2*

```
python vol.py -f 0zapftis.vmem windows.pstree
```

- *Output (elaborato)*

PID	PPID	ImageFileName
4	0	System
* 536	4	smss.exe
** 608	536	csrss.exe
** 632	536	winlogon.exe
*** 688	632	lsass.exe
*** 676	632	services.exe
**** 832	676	vmacthlp.exe
**** 964	676	svchost.exe
***** 1920	964	wscntfy.exe
***** 400	964	wuauctl.exe
***** 1444	676	VMwareService.e
**** 1260	676	spoolsv.exe
**** 848	676	svchost.exe
**** 1148	676	svchost.exe
**** 1616	676	alg.exe
**** 916	676	svchost.exe
**** 1020	676	svchost.exe

Esempio 2

Sono facilmente osservabili anche le dipendenze (processo padre/processo figlio), in merito ai processi discussi precedentemente (ovvero, winlogon.exe, services.exe, svchost.exe, wuauctl.exe e lsass.exe)

Volatility Framework

Identificazione dei processi

- Il Volatility Framework permette di elencare i processi ed ottenere diverse informazioni su di essi (ad esempio, la data e l'ora in cui il processo è stato avviato, ecc.)
- Per ottenere tali informazioni è possibile utilizzare i seguenti plugin:
 - pslist
 - pstree
 - **psscan**
- Nelle prossime slide verranno approfonditi questi plugin

Volatility Framework

Plugin psscan | 1/4

- *Esempio di Utilizzo 1 di 2*

```
python vol.py -f 0zapftis.vmem windows.psscan
```

- *Caratteristiche*

- Mediante il plugin psscan, il Volatility Framework mostra la lista dei processi, includendo eventuali processi nascosti

- **NOTA IMPORTANTE:**

- I processi nascosti potrebbero essere indice della presenza di malware
- Questi ultimi, cercano di nascondersi sia all'utente sia ai software anti-malware, al fine di effettuare azioni malevoli

Volatility Framework

Plugin psscan | 2/4

- *Esempio di Utilizzo 1 di 2*

```
python vol.py -f 0zapftis.vmem windows.psscan
```

- *Output (Parziale)*

PID CreateTime	PPID ExitTime	ImageFileName File output	Offset(V) File output	Threads	Handles	SessionId	Wow64
1616 17:04:01.000000	676 N/A	alg.exe Disabled	0x156c5a0	7	99	0	False 2011-10-10
632 10 17:03:58.000000	536 N/A	winlogon.exe Disabled	0x15a9020	24	533	0	False 2011-10-
1148 10 17:04:00.000000	676 N/A	svchost.exe Disabled	0x15aeda0	12	187	0	False 2011-10-
1956 10 17:04:39.000000	1884 N/A	explorer.exe Disabled	0x15bcd0	18	322	0	False 2011-10-
688 10 17:03:58.000000	632 N/A	lsass.exe Disabled	0x15c4020	23	336	0	False 2011-10-
1920 10 17:04:39.000000	964 N/A	wscntfy.exe Disabled	0x17c4da0	1	27	0	False 2011-10-
1020 10 17:03:59.000000	676 N/A	svchost.exe Disabled	0x17daca8	5	58	0	False 2011-10-
400 10 17:04:46.000000	964 N/A	wuauctl.exe Disabled	0x17e7be0	8	173	0	False 2011-10-
848 10 17:03:59.000000	676 N/A	svchost.exe Disabled	0x187e9d0	20	194	0	False 2011-10-
608 10 17:03:58.000000	536 N/A	csrss.exe Disabled	0x18c6020	11	355	0	False 2011-10-
964 10 17:03:59.000000	676 N/A	svchost.exe Disabled	0x18c6da0	63	1058	0	False 2011-10-

Volatility Framework

Plugin psscan | 3/4

- *Esempio di Utilizzo 1 di 2*

```
python vol.py -f 0zapftis.vmem windows.psscan
```

- **OSSERVAZIONE:**

- Al fine di individuare eventuali processi nascosti (da malware, ecc.), è consigliato confrontare l'output ottenuto con il plugin pslist e l'ouput ottenuto con il plugin psscan

Volatility Framework

Categorie di Plugin

- Possiamo suddividere i **plugin** del Volatility Framework, in diverse categorie
- Le categorie su cui ci soffermeremo sono le seguenti:
 - Analisi dei Processi
 - **Identificazione di Attività di Rete**
 - Analisi di librerie DLL (Dynamic Link Library) di Windows
 - Informazioni sul Registro di Sistema del S.O. Windows
 - Altri plugin

Volatility Framework

Identificazione Servizi e Connessioni di Rete

- I servizi e le connessioni di rete, utilizzati dai processi, possono fornire molte informazioni rilevanti, quali ad esempio, indirizzi IP locali e remoti, le porte utilizzate, ecc.
- I seguenti plugin sono quelli principalmente utilizzati per carpire informazioni circa servizi e/o connessioni di rete:
 - netscan

Volatility Framework

Identificazione Servizi e Connessioni di Rete

- I servizi e le connessioni di rete, utilizzati dai processi, possono fornire molte informazioni rilevanti, quali ad esempio, indirizzi IP locali e remoti, le porte utilizzate, ecc.
- I seguenti plugin sono quelli principalmente utilizzati per carpire informazioni circa servizi e/o connessioni di rete:
 - **netscan**

Volatility Framework

Plugin netscan

- *Sintassi*

```
python vol.py -f <MemoryDump> windows.netscan
```

- *Descrizione*

- Per memory dump, acquisiti da sistemi Windows-based, con una versione di Windows Vista o superiore (32 o 64 bit), è possibile utilizzare il plugin netscan
 - Tale plugin permette di individuare tracce di attività di rete, relative a connessioni basate su protocolli TCP e UDP

- *Caratteristiche*

- Per ciascuna attività di rete, individuata, vengono riportate diverse informazioni, fra cui:
 - Indirizzo IP locale (e la relativa porta) ed indirizzo IP remoto (e la relativa porta)
 - Data/ora relativa al momento in cui la connessione è stata avviata
 - Ecc.

Volatility Framework

Categorie di Plugin

- Possiamo suddividere i **plugin** del Volatility Framework, in diverse categorie
- Le categorie su cui ci soffermeremo sono le seguenti:
 - Analisi dei Processi
 - Identificazione di Attività di Rete
 - **Analisi di librerie DLL (Dynamic Link Library) di Windows**
 - Informazioni sul Registro di Sistema del S.O. Windows
 - Altri plugin

Volatility Framework

Analisi librerie DLL di Windows | 1/2

- Una **libreria DLL** (Dynamic Link Library) è costituita da porzioni di codice (simili a *funzioni*) e da dati
- Più processi possono utilizzare simultaneamente la medesima libreria DLL
- Grazie alle librerie DLL è possibile progettare un programma e suddividerlo in moduli (dove ogni modulo è una libreria DLL)
 - Un processo può caricare *dinamicamente* (dynamic) la libreria DLL (se installata)

Volatility Framework

Analisi librerie DLL di Windows | 1/2

- Una libreria DLL (Dynamic Link Library) è costituita da porzioni di codice (simili a *funzioni*) e da dati
 - Più moduli possono condividere la stessa memoria.
 - Grazie alla modularità, un programma può essere composto da più moduli.
- VANTAGGI**
- Riduzione dei tempi di caricamento, poiché i moduli vengono caricati solo in caso vi sia effettiva necessità
 - Un processo può caricare *unicamente* (dynamic) la libreria DLL (se installata)

Volatility Framework

Analisi librerie DLL di Windows | 2/2

- L'identificazione dei processi, che caricano delle librerie DLL, e la versione delle librerie stesse, può essere di aiuto nella correlazione di diversi processi
- Inoltre, mediante i processi e le librerie DLL, è possibile eventualmente mettere in relazione eventuali account multipli, di un utente
- Fra i vari plugin, preposti per l'analisi delle DLL, possiamo individuare i seguenti:
 - verinfo
 - dlllist
- I suddetti due plugin verranno delineati nelle prossime slide

Volatility Framework

Analisi librerie DLL di Windows | 2/2

- L'identificazione dei processi, che caricano delle librerie DLL, e la versione delle librerie stesse, può essere di aiuto nella correlazione di diversi processi
- Inoltre, mediante i processi e le librerie DLL, è possibile eventualmente mettere in relazione eventuali account multipli, di un utente
- Fra i vari plugin, preposti per l'analisi delle DLL, possiamo individuare i seguenti:
 - **verinfo**
 - dlllist
- I suddetti due plugin, verranno delineati nelle prossime slide

Volatility Framework

Plugin verinfo | 1/2

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.verinfo
```

- *Caratteristiche*

- Mostra informazioni estremamente dettagliate, in riferimento alle librerie DLL, utilizzate da processi generati da Portable Executable (**PE**) di Windows
 - Un eseguibile PE è strutturato in modo tale che possa contenere tutto il necessario per l'esecuzione in Windows, senza utilizzo di ulteriori file o librerie esterne
 - **NOTA:** L'output prodotto da Volatility, utilizzando il plugin verinfo, risulta particolarmente esteso

Volatility Framework

Plugin verinfo | 2/2

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.verinfo
```

- *Output (Parziale)*

PID	Process	Base	Name	Major	Minor	Product	Build
N/A	N/A	0x804d7000	ntoskrnl.exe	5	1	2600	2180
N/A	N/A	0x806ce000	hal.dll	-	-	-	-
N/A	N/A	0xf9e9c000	kdcom.dll	-	-	-	-
N/A	N/A	0xf9dac000	BOOTVID.dll	-	-	-	-
N/A	N/A	0xf986d000	ACPI.sys	-	-	-	-
N/A	N/A	0xf9e9e000	WMILIB.SYS	-	-	-	-
N/A	N/A	0xf985c000	pci.sys	-	-	-	-
N/A	N/A	0xf999c000	isapnp.sys	-	-	-	-
N/A	N/A	0xf9db0000	compbatt.sys	-	-	-	-
N/A	N/A	0xf9db4000	BATTC.SYS	-	-	-	-
N/A	N/A	0xf9ea0000	intelide.sys	-	-	-	-
N/A	N/A	0xf9c1c000	PCIINDEX.SYS	-	-	-	-
N/A	N/A	0xf99ac000	MountMgr.sys	-	-	-	-
N/A	N/A	0xf983d000	ftdisk.sys	-	-	-	-
N/A	N/A	0xf9ea2000	dmload.sys	-	-	-	-
N/A	N/A	0xf9817000	dmio.sys	-	-	-	-

Volatility Framework

Plugin verinfo | 2/2

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.verinfo
```

- *Output (Parziale)*

1920	wscntfy.exe	0x1000000	wscntfy.exe	5	1	2600	2180
1920	wscntfy.exe	0x7c900000	ntdll.dll	-	-	-	-
1920	wscntfy.exe	0x7c800000	kernel32.dll	-	-	-	-
1920	wscntfy.exe	0x77c10000	msvcrt.dll	6	1	8638	2180
1920	wscntfy.exe	0x77d40000	USER32.dll	-	-	-	-
1920	wscntfy.exe	0x77f10000	GDI32.dll	5	1	2600	2180
1920	wscntfy.exe	0x7c9c0000	SHELL32.dll	-	-	-	-
1920	wscntfy.exe	0x77dd0000	ADVAPI32.dll	5	1	2600	2180
1920	wscntfy.exe	0x77e70000	RPCRT4.dll	5	1	2600	2180
1920	wscntfy.exe	0x77f60000	SHLWAPI.dll	6	0	2900	2180
1920	wscntfy.exe	0x10000000	mfc42u1.dll	4	2	1	0
1920	wscntfy.exe	0x71ab0000	WS2_32.dll	5	1	2600	2180
1920	wscntfy.exe	0x71aa0000	WS2HELP.dll	5	1	2600	2180
1920	wscntfy.exe	0x71f60000	snmpapi.dll	5	1	2600	2180
1920	wscntfy.exe	0x77c00000	VERSION.dll	5	1	2600	2180
1920	wscntfy.exe	0x773d0000	comctl32.dll	-	-	-	-
1920	wscntfy.exe	0x20000000	xpsp2res.dll	5	1	2600	2180
1920	wscntfy.exe	0x5ad70000	uxtheme.dll	6	0	2900	2180

Volatility Framework

Analisi librerie DLL di Windows | 2/2

- L'identificazione dei processi, che caricano delle librerie DLL, e la versione delle librerie stesse, può essere di aiuto nella correlazione di diversi processi
- Inoltre, mediante i processi e le librerie DLL, è possibile eventualmente mettere in relazione eventuali account multipli dell'utente
- Fra i vari plugin, preposti per l'analisi delle DLL, possiamo individuare i seguenti:
 - verinfo
 - **dlllist**
- I suddetti due plugin, verranno delineati nelle prossime slide

Volatility Framework

Plugin **dlllist** | 1/3

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem dlllist
```

- *Caratteristiche*

- Per ciascun processo, vengono mostrate tutte le librerie DLL, utilizzate da tale processo

Volatility Framework

Plugin `dlllist` | 2/3

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem dlllist
```

- *Output (Parziale)*

PID	Process	Base	Size	Name	Path	LoadTime	File output
536	smss.exe	0x48580000		0xf000	smss.exe		
	\SystemRoot\System32\smss.exe			N/A	Disabled		
536	smss.exe	0x7c900000		0xb0000	ntdll.dll		
	C:\WINDOWS\system32\ntdll.dll			N/A	Disabled		
608	csrss.exe	0x4a680000		0x5000	csrss.exe		
	\??\C:\WINDOWS\system32\csrss.exe			N/A	Disabled		
608	csrss.exe	0x7c900000		0xb0000	ntdll.dll		
	C:\WINDOWS\system32\ntdll.dll			N/A	Disabled		
608	csrss.exe	0x75b40000		0xb000	CSRSRV.dll		
	C:\WINDOWS\system32\CSRSRV.dll			N/A	Disabled		
608	csrss.exe	0x75b50000		0x10000	basesrv.dll		
	C:\WINDOWS\system32\basesrv.dll			N/A	Disabled		
608	csrss.exe	0x75b60000		0x4a000	winsrv.dll		
	C:\WINDOWS\system32\winsrv.dll			N/A	Disabled		
608	csrss.exe	0x77d40000		0x90000	USER32.dll		
	C:\WINDOWS\system32\USER32.dll			N/A	Disabled		

Volatility Framework

Plugin dlllist | 2/3

Esempio 1

Il processo smss.exe, con PID pari a 536, sta utilizzando la libreria ntdll.dll (memorizzata nella cartella C:\WINDOWS\system32)

PID	Proc	Size	Name	Path	LoadTime	File output
536	smss.exe	0x48580000		0xf000	smss.exe	
	\SystemRoot\System32\smss.exe			N/A	Disabled	
536	smss.exe	0x7c900000		0xb0000	ntdll.dll	
	C:\WINDOWS\system32\ntdll.dll			N/A	Disabled	
608	csrss.exe	0x4a680000		0x5000	csrss.exe	
	\??\C:\WINDOWS\system32\csrss.exe			N/A	Disabled	
608	csrss.exe	0x7c900000		0xb0000	ntdll.dll	
	C:\WINDOWS\system32\ntdll.dll			N/A	Disabled	
608	csrss.exe	0x75b40000		0xb000	CSRSRV.dll	
	C:\WINDOWS\system32\CSRSRV.dll			N/A	Disabled	
608	csrss.exe	0x75b50000		0x10000	basesrv.dll	
	C:\WINDOWS\system32\basesrv.dll			N/A	Disabled	
608	csrss.exe	0x75b60000		0x4a000	winsrv.dll	
	C:\WINDOWS\system32\winsrv.dll			N/A	Disabled	
608	csrss.exe	0x77d40000		0x90000	USER32.dll	
	C:\WINDOWS\system32\USER32.dll			N/A	Disabled	

Volatility Framework

Plugin dlllist | 3/3

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem dlllist -pid 544
```

- *Output (Parziale)*

544	cmd.exe	0x77120000		
	C:\WINDOWS\system32\OLEAUT			
544	cmd.exe	0x77be0000		
	C:\WINDOWS\system32\MSACM3			
544	cmd.exe	0x77c00000		
	C:\WINDOWS\system32\VERSIO			
544	cmd.exe	0x7c9c0000		
	C:\WINDOWS\system32\SHELL3			
544	cmd.exe	0x77f60000		
	C:\WINDOWS\system32\SHLWAP			
544	cmd.exe	0x769c0000		
	C:\WINDOWS\system32\USEREN			
544	cmd.exe	0x5ad70000		
	C:\WINDOWS\system32\UxTheme.dll		N/A	Disabled
544	cmd.exe	0x10000000	0x59000	mfc42ul.dll
	C:\WINDOWS\system32\mfc42ul.dll		N/A	Disabled
544	cmd.exe	0x71ab0000	0x17000	WS2_32.dll
	C:\WINDOWS\system32\WS2_32.dll		N/A	Disabled

Esempio 2

Il processo cmd.exe, con PID 544, sta utilizzando diverse librerie DLL (memorizzate principalmente nella cartella C:\Windows\system32), fra le quali c'è mfc42ul.dll

Volatility Framework

Categorie di Plugin

- Possiamo suddividere i **plugin** del Volatility Framework, in diverse categorie
- Le categorie su cui ci soffermeremo sono le seguenti:
 - Identificazione di processi
 - Identificazione Servizi e Connessioni di Rete
 - Analisi librerie DLL (Dynamic Link Library) di Windows
 - Informazioni sul Registro di Sistema del S.O. Windows
 - Altri plugin

Volatility Framework

Informazioni sul Registro di Windows | 1/6

- Il **registro** è una componente dei sistemi Windows-based e **memorizza informazioni** in merito a diversi aspetti, **impostazioni e preferenze**, fra cui:
 - Impostazioni/Preferenze del S.O. stesso
 - Impostazioni/Preferenze di alcuni programmi
 - Impostazioni/Preferenze di driver
 - Ecc.
- Vengono inoltre memorizzate anche alcune **preferenze dell'utente**
- Dal punto di vista forense, il **registro** è **una risorsa notevole**, poiché è una sorta di database, contenente tantissimi **valori**, dai quali estrarre informazioni

Volatility Framework

Informazioni sul Registro di Windows | 2/6

Anatomia del Registro di Windows | Cenni | 1/3

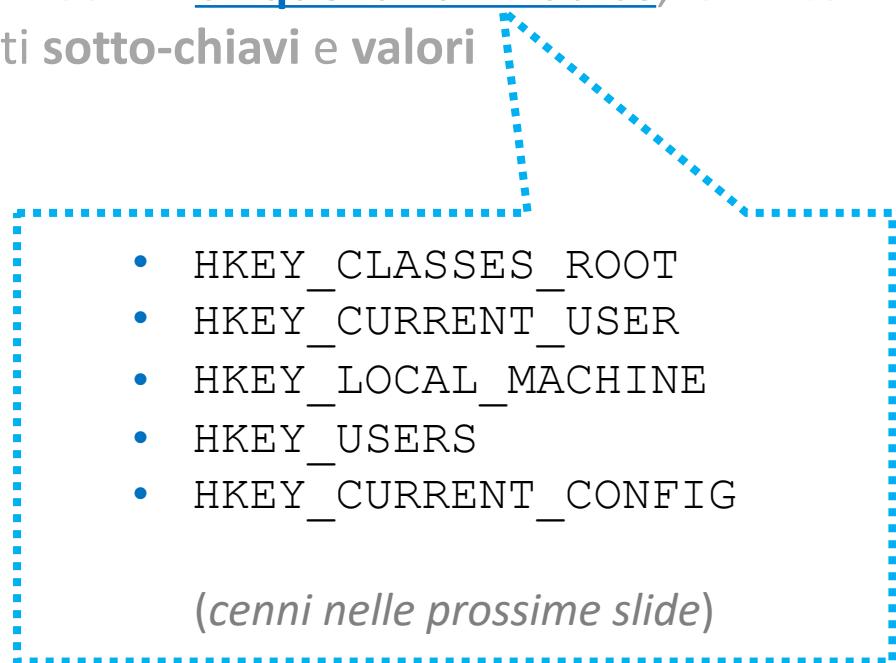
- Il registro ha una **struttura gerarchica** ad albero
- È suddiviso in **cinque chiavi radice**, all'interno delle quali sono presenti **sotto-chiavi e valori**

Volatility Framework

Informazioni sul Registro di Windows | 2/6

Anatomia del Registro di Windows | Cenni | 1/3

- Il registro ha una **struttura gerarchica ad albero**
- È suddiviso in **cinque chiavi radice**, all'interno delle quali sono presenti **sotto-chiavi e valori**

- 
- HKEY_CLASSES_ROOT
 - HKEY_CURRENT_USER
 - HKEY_LOCAL_MACHINE
 - HKEY_USERS
 - HKEY_CURRENT_CONFIG

(cenni nelle prossime slide)

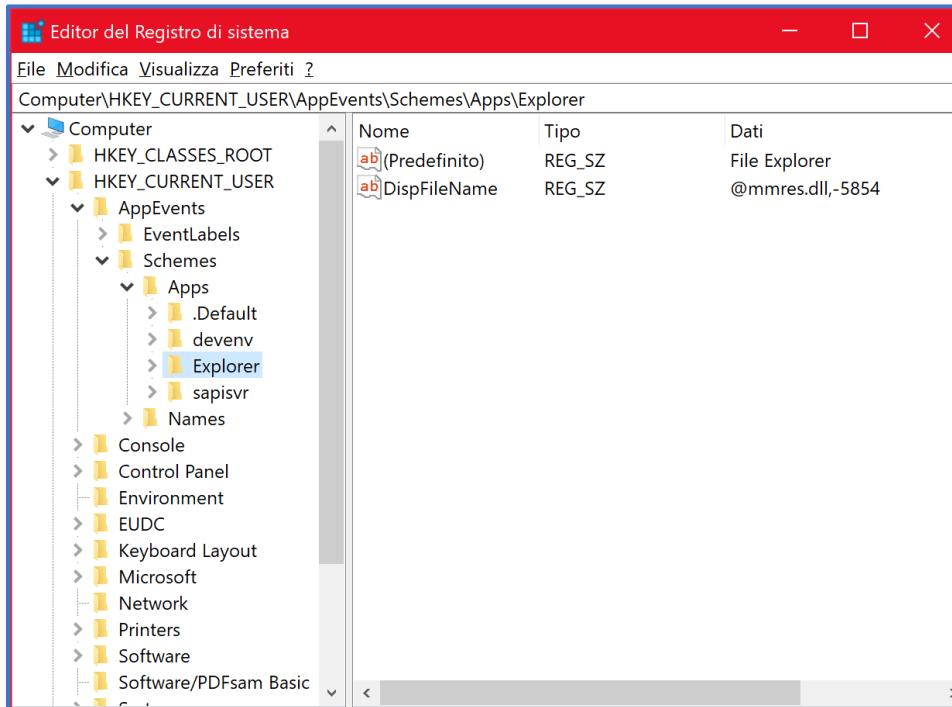
Volatility Framework

Informazioni sul Registro di Windows | 3/6

Anatomia del Registro di Windows | Cenni | 1/3

- Il registro ha una **struttura gerarchica** ad albero
- È suddiviso in **cinque chiavi radice**, all'interno delle quali sono presenti **sotto-chiavi e valori**

Rappresentazione Grafica del Registro [Editor del Registro di sistema]



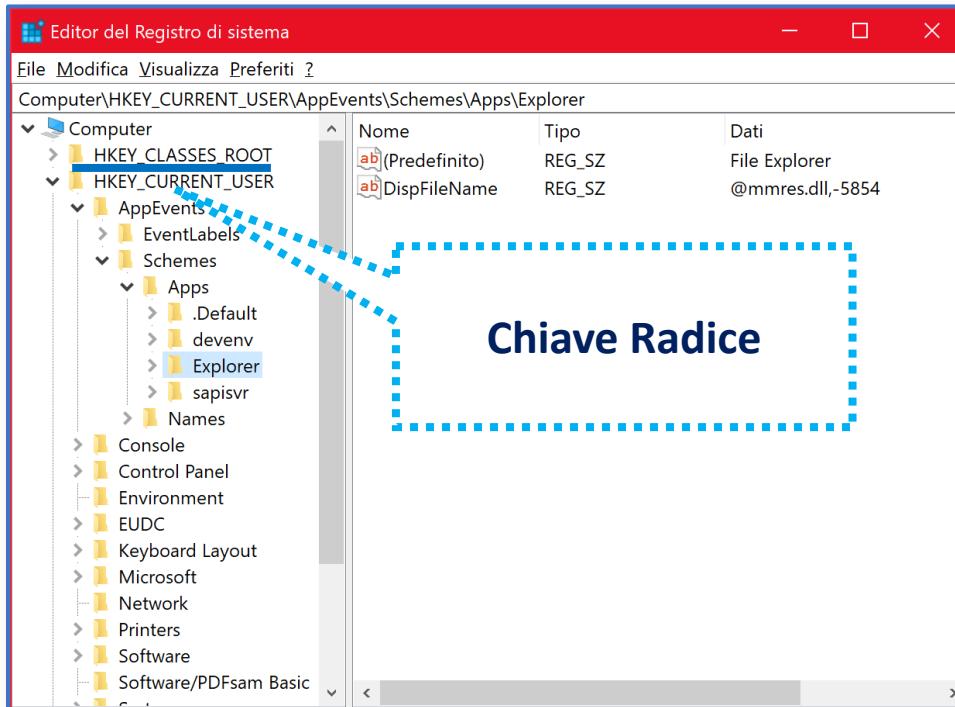
Volatility Framework

Informazioni sul Registro di Windows | 3/6

Anatomia del Registro di Windows | Cenni | 1/3

- Il registro ha una **struttura gerarchica** ad albero
- È suddiviso in **cinque chiavi radice**, all'interno delle quali sono presenti **sotto-chiavi e valori**

Rappresentazione Grafica del Registro [Editor del Registro di sistema]



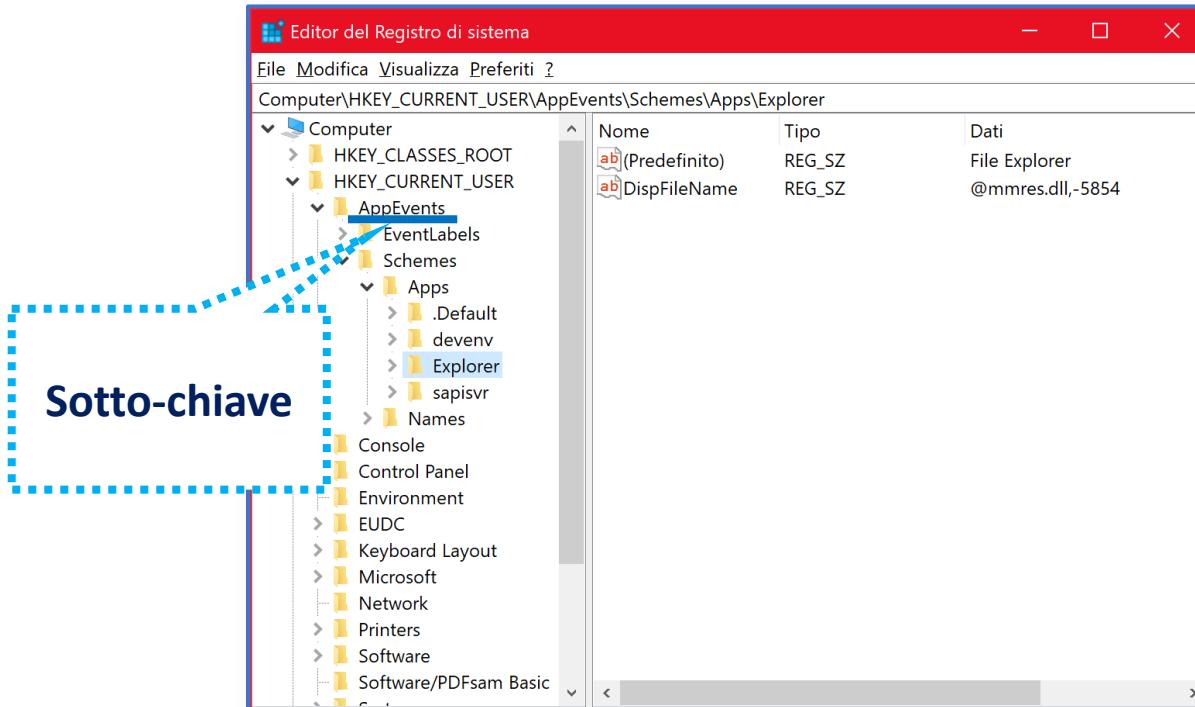
Volatility Framework

Informazioni sul Registro di Windows | 3/6

Anatomia del Registro di Windows | Cenni | 1/3

- Il registro ha una **struttura gerarchica** ad albero
- È suddiviso in **cinque chiavi radice**, all'interno delle quali sono presenti **sotto-chiavi e valori**

Rappresentazione Grafica del Registro [Editor del Registro di sistema]



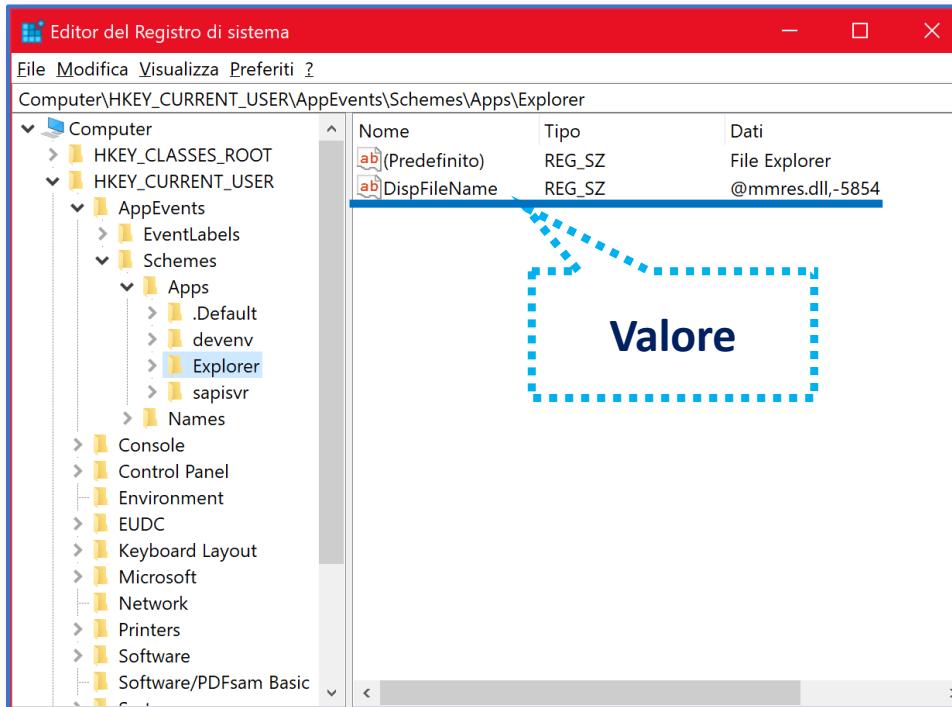
Volatility Framework

Informazioni sul Registro di Windows | 3/6

Anatomia del Registro di Windows | Cenni | 1/3

- Il registro ha una **struttura gerarchica** ad albero
- È suddiviso in **cinque chiavi radice**, all'interno delle quali sono presenti **sotto-chiavi e valori**

Rappresentazione Grafica del Registro [Editor del Registro di sistema]



Volatility Framework

Informazioni sul Registro di Windows | 4/6

Anatomia del Registro di Windows | Cenni | 2/3

CHIAVE RADICE	Descrizione (<i>Cenni</i>)
HKEY_CLASSES_ROOT	<p>Describe il comportamento di Windows, in relazione ad alcune azioni dell'utente</p> <p><i>Esempi</i></p> <ul style="list-style-type: none">• Cosa deve fare il sistema quando viene collegato un nuovo drive al PC (ad esempio, una Penna USB, ecc.)• Quale programma deve essere avviato, dal sistema, all'apertura di un file con una specifica estensione
HKEY_CURRENT_USER	Contiene le configurazioni di Windows e le configurazioni del software installato, in relazione all'utente autenticato (ad esempio, sfondo selezionato, variabili d'ambiente, configurazione del layout della tastiera, ecc.)

Volatility Framework

Informazioni sul Registro di Windows | 5/6

Anatomia del Registro di Windows | Cenni | 3/3

CHIAVE RADICE	Descrizione (<i>Cenni</i>)
HKEY_LOCAL_MACHINE	Contiene configurazioni in merito alla maggior parte del software installato, configurazioni in merito a Windows stesso ed informazioni, relative ad hardware e driver
HKEY_USERS	Sono presenti dati ed informazioni, in merito a tutti gli utenti connessi al sistema
HKEY_CURRENT_CONFIG	Contiene sotto-chiavi che puntano a configurazioni riferite all'hardware in uso

Volatility Framework

Informazioni sul Registro di Windows | 6/6

- Ogni chiave radice corrisponde ad uno o più file, presenti nel file system
 - I suddetti file vengono spesso chiamati *hive* (letteralmente, *alveari*)
- Poiché il registro è potenzialmente acceduto, in maniera frequente, dal S.O., alcune informazioni sugli hive ed alcune informazioni del registro, sono generalmente mantenute in memoria
- Il Volatility Framework prevede diversi plugin, per individuare informazioni sugli hive e sul registro
- Un plugin particolarmente utile è il seguente:
 - hivelist

Volatility Framework

Plugin hivelist | 1/3

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem hivelist
```

- *Caratteristiche*

- Mediante il plugin hivelist, è possibile ottenere informazioni sugli *hive*, individuati all'interno del memory dump
- Viene indicato il percorso degli *hive*, all'interno del file system
 - In questo modo, sarà possibile localizzare gli *hive*, ed analizzarli successivamente (*maggiori dettagli nelle prossime slide*)

Volatility Framework

Plugin hivelist | 2/3

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem hivelist
```

- *Output (Parziale)*

Offset	FileFullPath	File output
0xe1bf6b60	\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat	Disabled
0xe1bb2b60	\Device\HarddiskVolume1\Documents and Settings\Administrator\NTUSER.DAT	Disabled
0xe1a4db60	\Device\HarddiskVolume1\Documents and Settings\LocalService\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat	Disabled
0xe1991b60	\Device\HarddiskVolume1\Documents and Settings\LocalService\NTUSER.DAT	Disabled
0xe1844458	\Device\HarddiskVolume1\Documents and Settings\NetworkService\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat	Disabled
0xe183e008	\Device\HarddiskVolume1\Documents and Settings\NetworkService\NTUSER.DAT	Disabled
0xe1544b60	\Device\HarddiskVolume1\WINDOWS\system32\config\software	Disabled
0xe154db60	\Device\HarddiskVolume1\WINDOWS\system32\config\SAM	Disabled
0xe154d008	\Device\HarddiskVolume1\WINDOWS\system32\config\default	Disabled
0xe1544008	\Device\HarddiskVolume1\WINDOWS\system32\config\SECURITY	Disabled
0xe13b5a40		Disabled
0xe1018388	\Device\HarddiskVolume1\WINDOWS\system32\config\system	Disabled
0xe1008b60		Disabled

Volatility Framework

Plugin hivelist | 2/3

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem hivelist
```

- *Output (Parziale)*

Offset	FileFullPath	File output
0xe1bf6b60	\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Application Data\Microsoft\Windows\UserClass.dat	Disabled
0xe1bb2b60	\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\LocalService\Local	Disabled
0xe1a4db60	\Device\HarddiskVolume1\Documents and Settings\NetworkService\Local	Disabled
0xe1991b60	\Device\HarddiskVolume1\Documents and Settings\NetworkService\Network	Disabled
0xe1844458	\Device\HarddiskVolume1\Documents and Settings\SYSTEM\Control Panel\Font	Disabled
0xe183e008	\Device\HarddiskVolume1\Documents and Settings\SYSTEM\Control Panel\Font\Font	Disabled
0xe1544b60	\Device\HarddiskVolume1\WINDOWS\system32\config\software	Disabled
0xe154db60	\Device\HarddiskVolume1\WINDOWS\system32\config\SAM	Disabled
0xe154d008	\Device\HarddiskVolume1\WINDOWS\system32\config\default	Disabled
0xe1544008	\Device\HarddiskVolume1\WINDOWS\system32\config\SECURITY	Disabled
0xe13b5a40		Disabled
0xe1018388	\Device\HarddiskVolume1\WINDOWS\system32\config\system	Disabled
0xe1008b60		Disabled

Percorso degli hive all'interno del file system

Volatility Framework

Plugin hivelist | 3/3

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem window.hivelist -filter  
SECURITY
```

- *Output (Parziale)*

- Offset FileFullPath File output

```
0xe1544008  
\Device\HarddiskVolume1\WINDOWS\system32\config\SECURITY  
Disabled
```

Volatility Framework

Categorie di Plugin

- Possiamo suddividere i **plugin** del Volatility Framework, in diverse categorie
- Le categorie su cui ci soffermeremo sono le seguenti:
 - Analisi dei Processi
 - Identificazione di Attività di Rete
 - Analisi di librerie DLL (Dynamic Link Library) di Windows
 - Informazioni sul Registro di Sistema del S.O. Windows
 - **Altri plugin**

Volatility Framework

Categorie di Plugin

- Possiamo suddividere i **plugin** del Volatility Framework, in diverse categorie
- Le categorie su cui ci soffermeremo sono le seguenti:
 - Analisi dei Processi
 - Identificazione di Attività di Rete
 - Analisi di librerie DLL (Dynamic Link Library) di Windows
 - Informazioni sul Registro di Sistema del S.O. Windows
 - **Altri plugin**
 - filescan
 - timeliner
 - malfind

Volatility Framework

Categorie di Plugin

- Possiamo suddividere i **plugin** del Volatility Framework, in diverse categorie
- Le categorie su cui ci soffermeremo sono le seguenti:
 - Analisi dei Processi
 - Identificazione di Attività di Rete
 - Analisi di librerie DLL (Dynamic Link Library) di Windows
 - Informazioni sul Registro di Sistema del S.O. Windows
 - **Altri plugin**
 - **filescan**
 - **timeliner**
 - **malfind**

Volatility Framework

Plugin filescan | 1/2

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem filescan
```

- *Caratteristiche*

- Un processo che intende creare, scrivere o leggere un certo file, deve effettuarne preliminarmente l'apertura
 - Windows mantiene, in memoria, i «*riferimenti*» di tutti i file aperti
- Tramite il plugin filescan, vengono cercati, all'interno del memory dump, tutti i suddetti *riferimenti*
- Il plugin filescan elencherà quindi **tutti i file aperti** ed eventuali **file non visibili, da alcuni tool standard** (i file non visibili potrebbero potenzialmente essere nascosti da un malware)
 - Per ciascun file, verrà riportato il percorso completo ed i permessi effettivamente garantiti al file

Volatility Framework

Plugin `filescan` | 1/2

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.filescan
```

- *Output (Parziale) | 1/2*

Offset	Name	Size
0x156bcb0	\Endpoint	112
0x156f100	\W32TIME	112
0x15a9a70	\{9B365890-165F-11D0-A195-0020AFD156E4}	112
0x15ac5c8	\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.2180_x-ww_a84f1ff9	112
0x15ac6b0	\WINDOWS\Media\Windows XP Startup.wav	112
0x15ac8f0		
	\WINDOWS\WinSxS\x86_Microsoft.VC80.MFC_1fc8b3b9a1e18e3b_8.0.50727.762_x-ww_3bf8fa05\mfc80u.dll	112
0x15ad318	\WINDOWS\system32\webcheck.dll	112
0x15ad740	\WINDOWS\system32\themeui.dll	112
0x15ad858	\Endpoint	112
0x15adb98	\WINDOWS\system32\ega.cpi	112
0x15ae208	\Program Files\Windows NT\Accessories	112
0x15ae3d0	\WINDOWS\system32\moricons.dll	112
0x15afbf0	\WINDOWS\Fonts\framdit.ttf	112
0x15afe08	\WINDOWS\system32	112

Volatility Framework

Plugin filescan | 1/2

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.filescan
```

- *Output (Parziale) | 2/2*

OSSERVAZIONE IMPORTANTE

- Nell'elenco dei file, riportato dal plugin filescan, è possibile individuare anche tracce di file eseguibili (.exe)

0x15b40b8	\WINDOWS\explorer.exe	112
0x15b7028	\WINDOWS\system32\wscntfy.exe	112
0x16865c0	\WINDOWS\system32\imapi.exe	112
0x16883b0	\WINDOWS\system32\imapi.exe	112
0x168b820	\Program Files\VMware\VMware Tools\vmacthlp.exe	112
0x168c730	\WINDOWS\system32\mshearts.exe	112
0x16cd368	\Program Files\VMware\VMware Tools\vmacthlp.exe	112
0x16cd5d0	\WINDOWS\system32\smss.exe	112
0x1795e98	\Program Files\VMware\VMware Tools\VMwareTray.exe	112
0x17977c8	\Program Files\VMware\VMware Tools\VMwareUser.exe	112

Volatility Framework

Plugin filescan | 1/2

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.filescan
```

- *Output (Parziale) | 2/2*

OSSERVAZIONE IMPORTANTE

- Nell'elenco dei file, riportato dal plugin filescan, è possibile individuare anche tracce di file eseguibili (.exe)

0x1998d78	\WINDOWS\system32\cmd.exe	112
0x1a07ed8	\WINDOWS\system32\cmd.exe	112

È importante considerare anche gli eseguibili (ed eventualmente anche le librerie DLL), poiché potrebbero essere stati aperti da malware, al fine di modificarli, iniettandovi del codice malevole

Volatility Framework

Plugin filescan | 1/2

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.filescan
```

- *Output (Parziale) | 2/2*

OSSERVAZIONE IMPORTANTE

- Nell'elenco dei file, riportato dal plugin filescan, è possibile individuare anche tracce di file eseguibili (.exe)

0x1998d78	\WINDOWS\system32\cmd.exe	112
0x1a07ed8	\WINDOWS\system32\cmd.exe	112

In questo caso sono state individuate due tracce relative all'eseguibile cmd.exe

Volatility Framework

Plugin filescan | 1/2

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem windows.filescan
```

- *Output (Parziale) | 2/2*

OSSERVAZIONE IMPORTANTE

- Nell'elenco dei file, riportato dal plugin filescan, è possibile individuare anche tracce di file eseguibili (.exe)

0x15b8128	\WINDOWS\system32\mfc42ul.dll	112
0x1a32840	\WINDOWS\system32\mfc42ul.dll	112

e anche due tracce relative alla libreria mfc42ul.dll

Volatility Framework

Categorie di Plugin

- Possiamo suddividere i **plugin** del Volatility Framework, in diverse categorie
- Le categorie su cui ci soffermeremo sono le seguenti:
 - Analisi dei Processi
 - Identificazione di Attività di Rete
 - Analisi di librerie DLL (Dynamic Link Library) di Windows
 - Informazioni sul Registro di Sistema del S.O. Windows
 - Altri plugin
 - filescan
 - **timeliner**
 - malfind

Volatility Framework

Plugin **timeliner** | 1/2

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem timeliner
```

- *Caratteristiche*

- Il plugin `timeliner` risulta particolarmente utile agli investigatori, poiché permette di ottenere una *timeline* degli eventi individuati nel memory dump
- Gli eventi vengono raggruppati in base alla data e all'orario
- Alcuni esempi di eventi:
 - Avvio di un processo
 - Utilizzo di una librerie DDL
 - Utilizzo del registro di Windows
 - Ecc.

Volatility Framework

Plugin **timeliner** | 2/2

- *Esempio di Utilizzo*

```
python vol.py -f 0zapftis.vmem timeliner
```

- *Output (Parziale)*

Plugin Date	Description	Created Date	Modified Date	Accessed Date	Changed
Progress: 88.00			Running plugin Sessions...		
MFTScan	MFT FILE_NAME entry for N/AN/A		1601-11-24 10:21:30.000000		N/A
MFTScan	MFT FILE_NAME entry for N/AN/A		1601-11-24 10:21:30.000000		N/A
MFTScan	MFT FILE_NAME entry for N/AN/A		1601-11-24 10:21:30.000000		N/A
MFTScan	MFT FILE_NAME entry for N/AN/A		1601-11-24 10:21:30.000000		N/A
MFTScan	MFT FILE_NAME entry for N/AN/A		1601-11-24 10:21:30.000000		N/A
MFTScan	MFT FILE_NAME entry for N/AN/A		1601-11-24 10:21:30.000000		N/A
MFTScan	MFT FILE_NAME entry for N/AN/A		1601-11-24 10:21:30.000000		N/A
MFTScan	MFT FILE_NAME entry for N/AN/A		1601-11-24 10:21:30.000000		N/A
MFTScan	MFT FILE_NAME entry for acgenral.dll	2004-08-04 12:00:00.000000	2004-08-04 12:00:00.000000		
		2004-08-04 12:00:00.000000	2010-11-06 18:10:23.000000	2010-11-06	
		13:02:44.000000			

Volatility Framework

Categorie di Plugin

- Possiamo suddividere i **plugin** del Volatility Framework, in diverse categorie
- Le categorie su cui ci soffermeremo sono le seguenti:
 - Analisi dei Processi
 - Identificazione di Attività di Rete
 - Analisi di librerie DLL (Dynamic Link Library) di Windows
 - Informazioni sul Registro di Sistema del S.O. Windows
 - **Altri plugin**
 - filescan
 - timeliner
 - **malfind**

Volatility Framework

Plugin malfind | 1/4

- *Esempio di Utilizzo 1 di 2*

```
python vol.py -f 0zapftis.vmem malfind
```

- *Caratteristiche*

- Il plugin malfind aiuta gli investigatori nell'individuazione di eventuali malware, riportando codice potenzialmente malevolo iniettato nella memoria
 - Si basa su alcune caratteristiche proprie dei malware, per individuare codice potenzialmente malevolo
 - È importante identificare eventuali malware, poiché potrebbero aver svolto operazioni malevoli, all'insaputa dell'utente
 - Pertanto, l'investigatore dovrebbe essere in grado di individuare quali sono tali operazioni, onde evitare di attribuirle all'utente
 - **NOTA:** L'output di Volatility, con il plugin malfind, può essere particolarmente esteso

Volatility Framework

Plugin malfind | 2/4

- Esempio di Utilizzo 1 di 2

```
python vol.py -f 0zapftis.vmem malfind
```

- Output (Parziale)

PID	Process	Start VPN	End VPN	Tag	Protection
CommitCharge ateMemory	Priv File output		Notes	Hexdump	Disasm
608	csrss.exe	0x7f6f0000	0x7f7effff		Vad
	PAGE_EXECUTE_READWRI				
TE	0	0	Disabled		N/A
c8	00 00 00 ba	01 00 00		
ff	ee ff ee 08	70 00 00p..		
08	00 00 00 00 fe	00 00 00		
00	00 10 00 00 20	00 00 00		
00	02 00 00 00 20	00 00 00		
8d	01 00 00 ff ef	fd 7f		
03	00 08 06 00 00 00	00 00 00		
00	00 00 00 00 00 00	00 00 00		
0x7f6f0000:	enter	0, 0			
0x7f6f0004:	mov	edx, 0xff000001			

Volatility Framework

Plugin malfind | 3/4

- *Esempio di Utilizzo 2 di 2 | Opzione --pid*

```
python vol.py -f 0zapftis.vmem malfind --pid 632
```

- *Caratteristiche | Opzione --pid*

- Mediante l'opzione `--pid` è possibile specificare il PID di uno dei processi
 - Nell'esempio, il PID specificato è pari a 632 (ovvero, il processo `winlogon.exe`)
 - Verranno mostrate le informazioni, descritte precedentemente, in merito al processo con il PID specificato (ovvero, 632)

Volatility Framework

Plugin malfind | 4/4

- *Esempio di Utilizzo 2 di 2 | Opzione -p*

```
python vol.py -f 0zapftis.vmem malfind --pid 632
```

- *Output (Parziale)*

PID	Process	Start VPN	End VPN	Tag	Protection	Notes
CommitCharge	PrivateMemory			File output		
Hexdump	Disasm					
632	winlogon.exe	0x64f0000		0x64f3fff	VadS	
	PAGE_EXECUTE_READWRITE	4	1	Disabled	N/A	
00 00 00 00 00 00 00 00					
00 00 00 00 00 00 00 00					
00 00 00 00 00 00 00 00					
00 00 00 00 00 00 00 00					
00 00 00 00 00 00 00 00					
00 00 00 00 00 00 00 00					
00 00 00 00 00 00 00 00					
00 00 00 00 29 00 29 00	).				
01 00 00 00 00 00 00 00					
0x64f0000:	add	byte ptr [eax], al				
0x64f0002:	add	byte ptr [eax], al				

Ulteriori Tool Utili

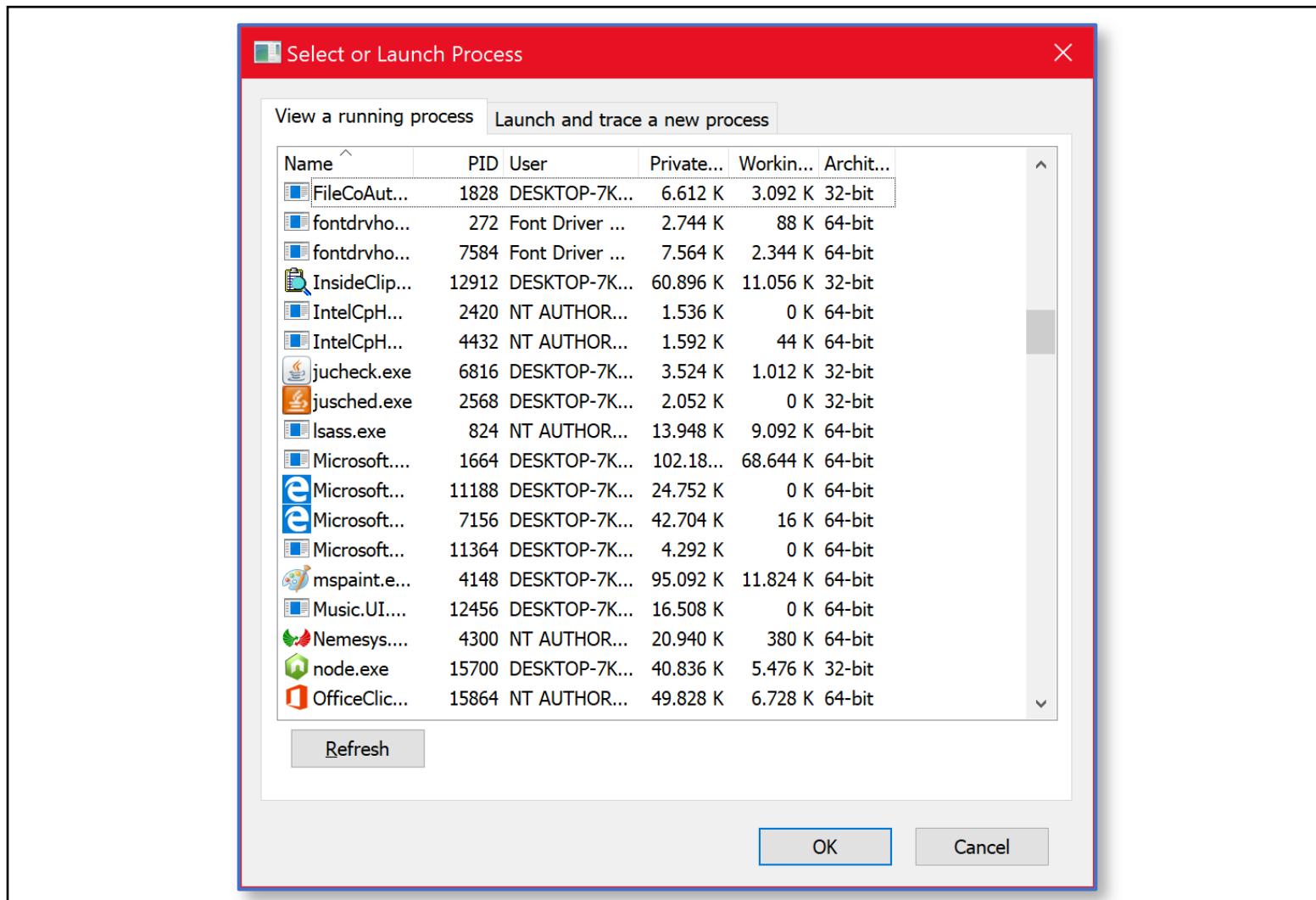
Ulteriori Tool Utili

Il tool VMMap | 1/3

- Il tool **VMMap** è in grado di fornire **dettagli accurati sulla memoria** (fisica e virtuale), utilizzata da un processo in Windows
- Mostra la **suddivisione della memoria, allocata per un processo**
- Mostra **utili rappresentazioni grafiche** della memoria ed ulteriori informazioni dettagliate
- Il tool è sviluppato da Mark Russinovich
- Da utilizzare direttamente sul live system
- Gratuitamente scaricabile
- Maggiori informazioni e dettagli al seguente link:
 - <https://docs.microsoft.com/en-us/sysinternals/downloads/vmmap>

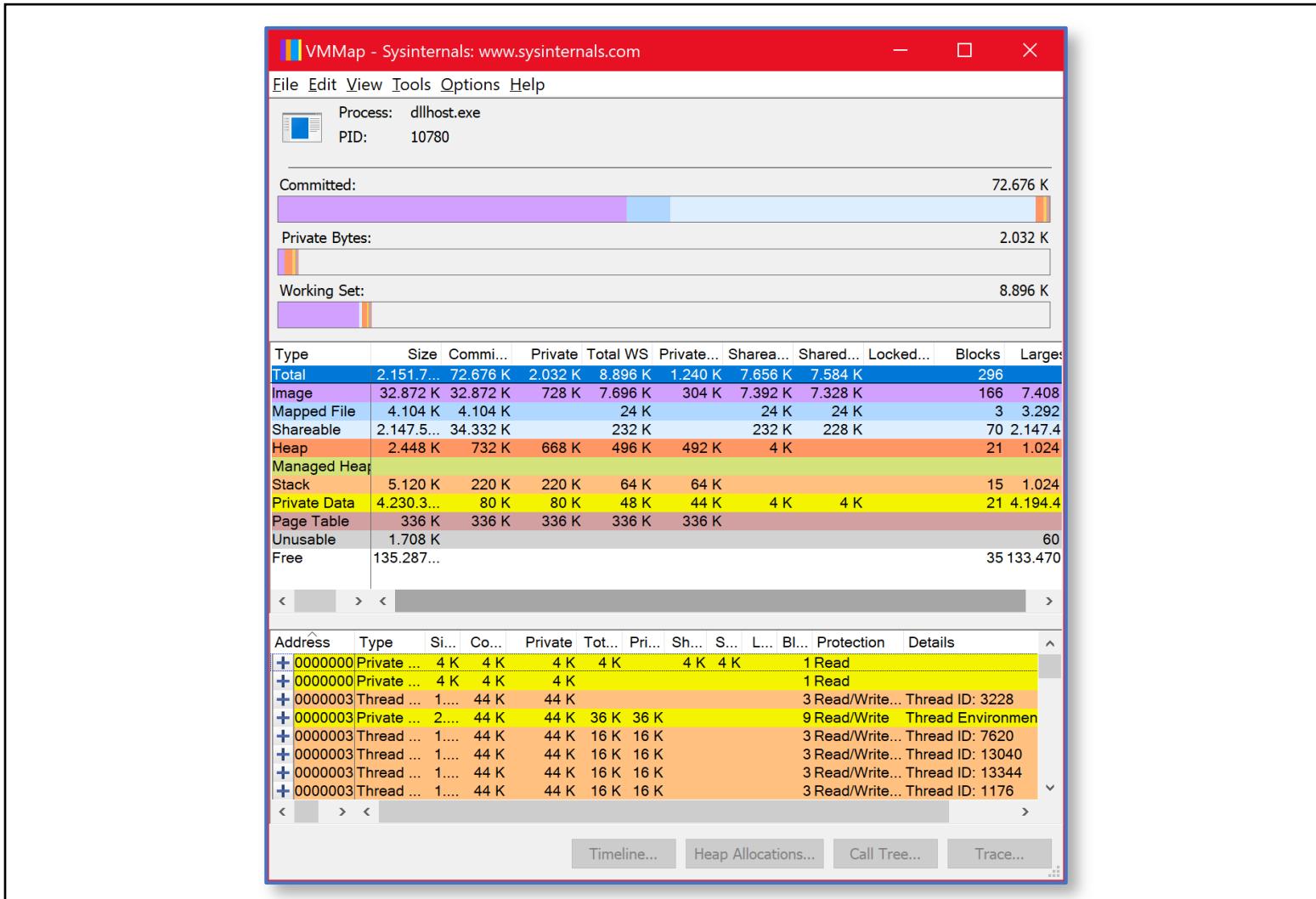
Ulteriori Tool Utili

Il tool VMMap | 2/3



Ulteriori Tool Utili

Il tool VMMap | 3/3



Ulteriori Tool Utili

Il tool InsideClipboard | 1/2

- Negli Appunti di Windows (*Clipboard*) potrebbero essere presenti delle informazioni di interesse all'investigazione forense

Ulteriori Tool Utili

Il tool InsideClipboard | 1/2

- Negli Appunti di Windows (Clipboard) potrebbero essere presenti le informazioni di interesse all'investigazione



- Componente del S.O.
- Permette lo scambio di dati (testo, immagini, ecc.), da una applicazione all'altra
 - *Alcune Operazioni, in cui tale componente è coinvolto:*
 - Copia e Incolla
 - Taglia e Incolla
 - Drag & Drop (trascinamento)
 - I dati «copiati» e/o «tagliati», vengono temporaneamente mantenuti in memoria RAM

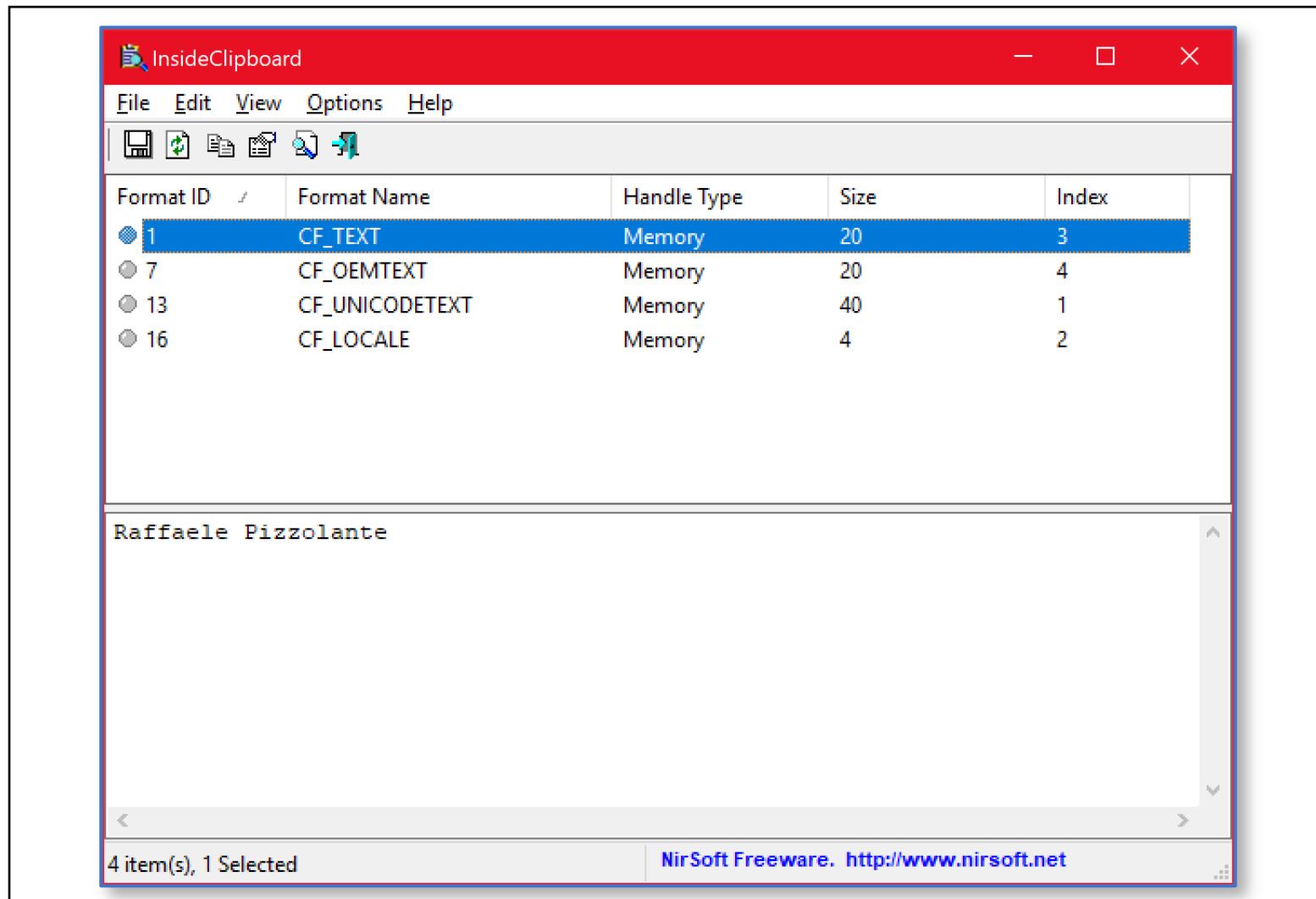
Ulteriori Tool Utili

Il tool InsideClipboard | 1/2

- Negli Appunti di Windows (*Clipboard*) potrebbero essere presenti delle informazioni di interesse all'investigazione forense
- Il tool freeware **InsideClipboard**, sviluppato dalla **NirSoft**, permette di visualizzare i dati contenuti negli appunti di Windows
 - Visualizzazione dei dati, presenti negli appunti, in diversi formati (es., ASCII, esadecimale, ...)
- Da utilizzare direttamente sul live system, non necessita di installazione e file aggiuntivi
- Maggiori informazioni e dettagli al seguente link:
 - https://www.nirsoft.net/utils/inside_clipboard.html

Ulteriori Tool Utili

Il tool InsideClipboard | 2/2



Riferimenti Bibliografici

- **Digital Forensics with Kali Linux, Shiva V.N. Parasram, Packt Publishing, 2017**
 - Capitolo 7
- **Practical Windows Forensics, Ayman Shaaban, Konstantin Sapronov, Packt Publishing, 2016**
 - Capitolo 11
- **Command Reference | Volatility Framework**
 - <https://github.com/volatilityfoundation/volatility/wiki/Command-Reference>
- **VMMMap**
 - <https://docs.microsoft.com/en-us/sysinternals/downloads/vmmap>
- **InsideClipboard**
 - https://www.nirsoft.net/utils/inside_clipboard.html