REPORT TECNICO

Esplorazione di Processi, Trhead, Handle e Registro di Windows

Redatto da: *Nicolò Calì Cybersecurity Student*

Data: *16/02/2026*

Oggetto: *Analisi dei processi e del registro di sistema Windows tramite Suite SysInternals.*

1. Introduzione

Il presente report documenta le attività di analisi tecnica svolte su un sistema operativo Windows, focalizzandosi sulla gestione dei **processi**, dei **thread**, degli **handle** e sulla configurazione del **Registro di Sistema**. L'esercitazione è stata condotta all'interno di una **VM**, utilizzando la suite di strumenti avanzati **Windows Sysinternals**, specificamente l'utility **Process Explorer**.

L'obiettivo principale è acquisire competenze pratiche nel monitoraggio delle risorse di sistema, nell'identificazione delle gerarchie dei processi e nella comprensione di come le modifiche al Registro di Windows influenzino il comportamento delle applicazioni.

2. Analisi della Gerarchia dei Processi

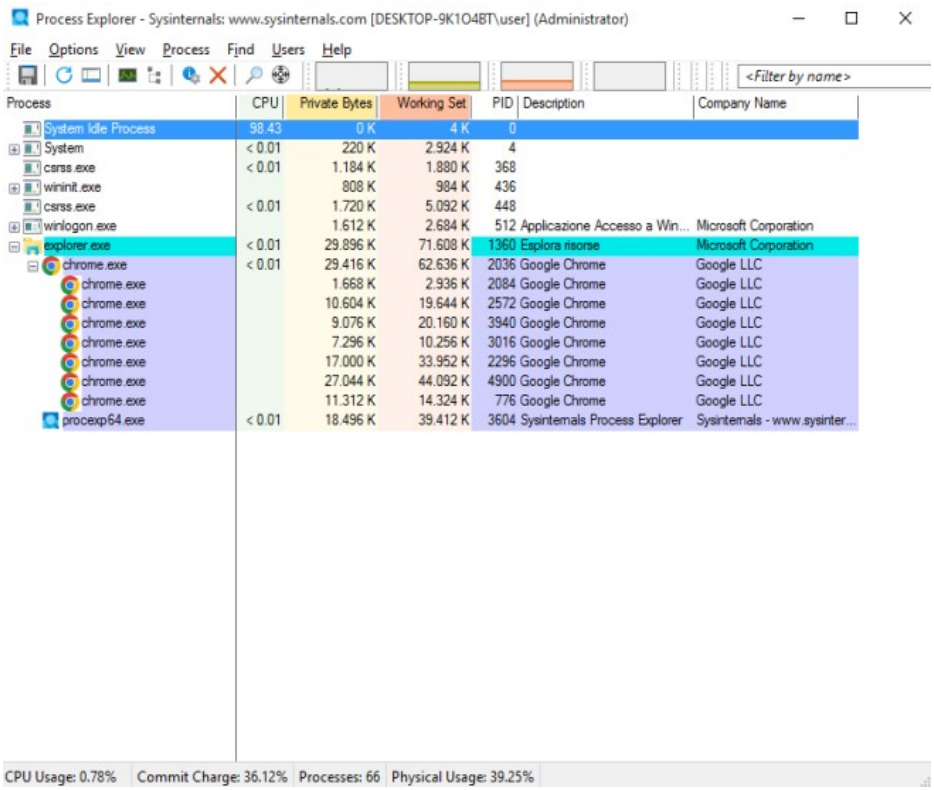

Fig 1. Process Explorer

Dall'analisi effettuata con **Process Explorer**, si evidenzia una struttura gerarchica in cui il processo `explorer.exe` ha avviato il browser. Si nota che il browser genera molteplici processi figli identici.

Questa architettura multiprocesso è fondamentale per la **Security by Isolation (Sandboxing)**: ogni scheda o estensione viene eseguita in uno spazio di memoria separato. Ciò garantisce che il crash o la compromissione di una singola scheda non impatti l'intero browser o il sistema operativo sottostante.

3. Analisi di Thread e Handle

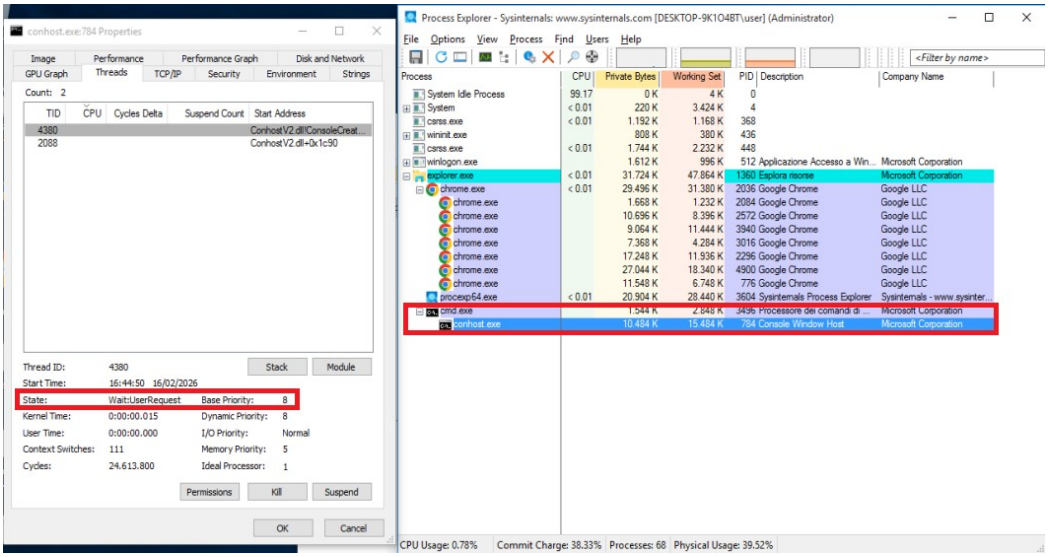

Fig 2. Threads

Analizzando le proprietà del processo **`conhost.exe`**, sono stati identificati i **Thread** attivi. Come evidenziato nello screenshot, i thread si trovano in stato di **`Wait:UserRequest`**.

Ciò indica che il processo è in attesa di input da parte dell'utente e non sta consumando attivamente cicli della CPU.

Questo dimostra come il sistema operativo gestisca le risorse assegnando tempo di calcolo solo quando necessario.

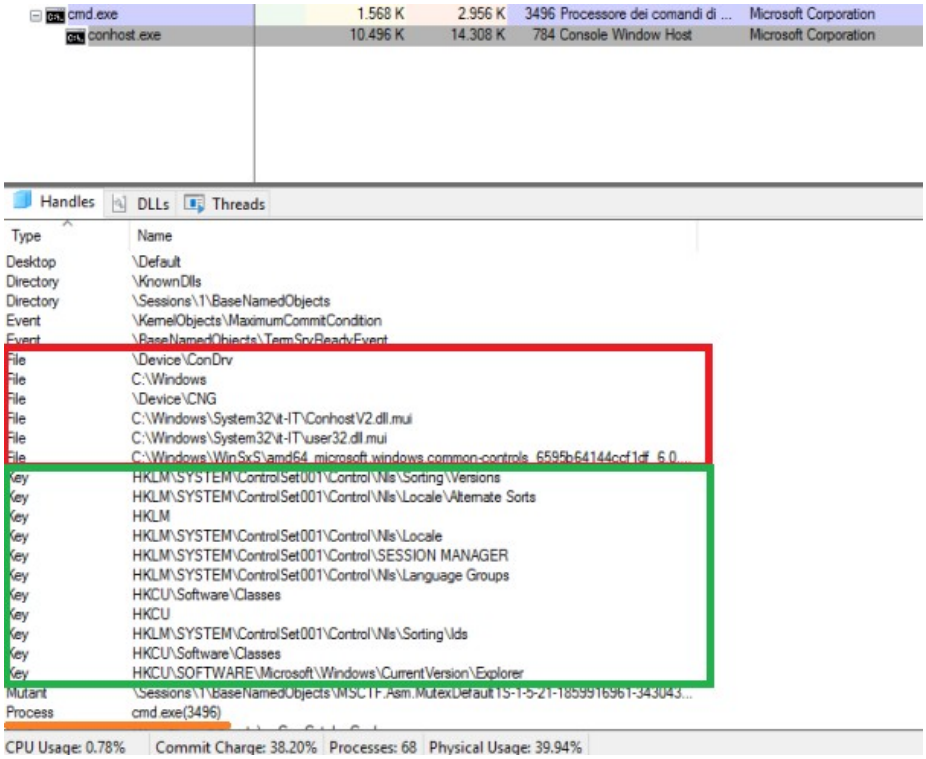

Fig 3. Handle

Successivamente, attivando la vista '**Handles**', è stato possibile esaminare le risorse a cui il processo ha accesso. L'analisi rivela handle di tipo:

- **Process:** È presente un riferimento esplicito al processo `cmd.exe`, confermando il legame operativo tra la console host e l'interprete dei comandi.
- **Key:** Sono visibili handle verso chiavi di registro (**`HKLM`**, **`HKCU`**), indicando che il processo interroga il database di configurazione di Windows.
- **File:** Il processo mantiene aperti riferimenti a librerie di sistema (**.dll**) necessarie alla sua esecuzione.

4. Manipolazione del Registro di Sistema

L'ultima fase dell'esercitazione ha riguardato l'interazione con il Registro di Sistema. Tramite il tool **`regedit`**, è stata localizzata la chiave di configurazione di **Process Explorer** nel seguente percorso:

`HKEY_CURRENT_USER\Software\Sysinternals\Process Explorer`

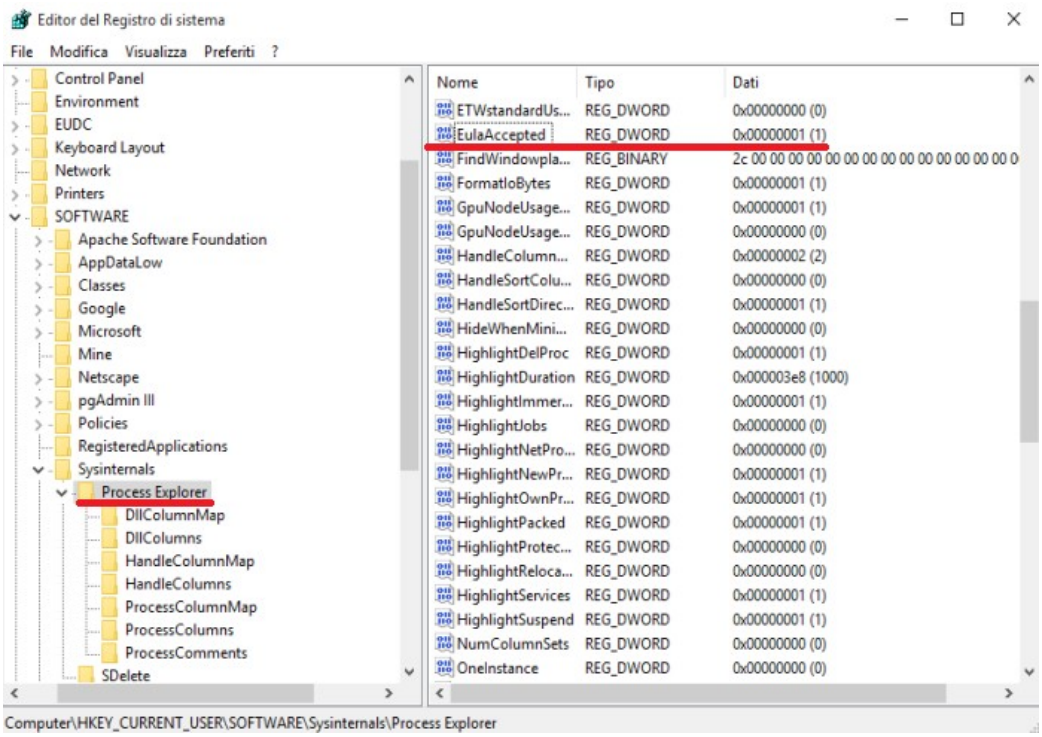

Fig 4. Registro di Sistema

È stata identificata la voce **EulaAccepted**, il cui valore booleano '**1**' indicava la precedente accettazione della licenza.

Modificando manualmente tale valore a '**0**' e riavviando l'applicazione, si è verificato che il software ha perso la 'memoria' della configurazione precedente, riproponendo la finestra di accettazione dei termini di licenza.

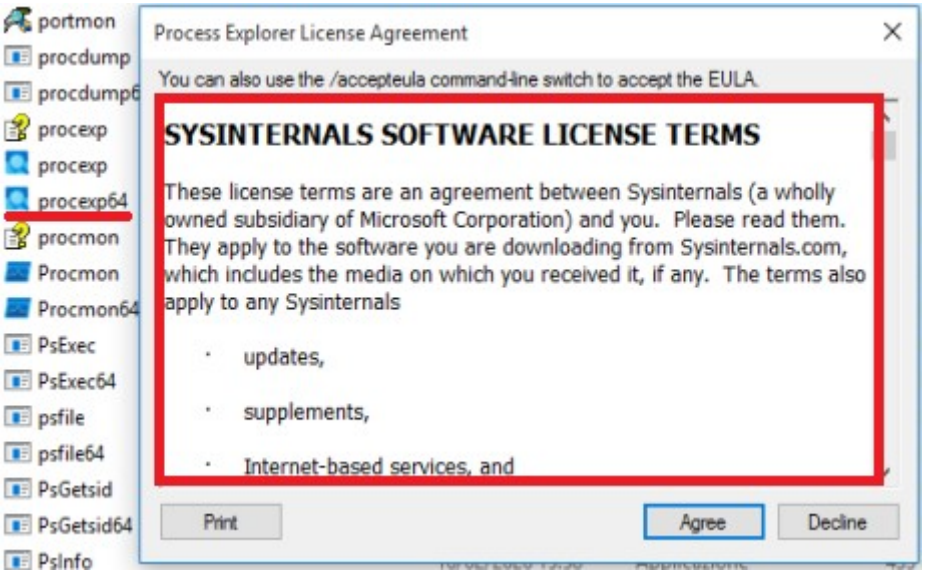

Fig 5. License Agreement procexp64

Questo dimostra come le applicazioni si affidino al Registro per memorizzare stati e configurazioni persistenti.

5. Conclusioni

L'attività svolta ha permesso di analizzare in profondità le meccaniche di gestione delle risorse del sistema operativo Windows.

Attraverso l'utilizzo del ***Process Explorer***, è stato possibile visualizzare concretamente la gerarchia dei processi e comprendere come il sistema isoli le applicazioni per garantire stabilità e sicurezza.

L'ispezione di **Thread** e **Handle** ha chiarito il funzionamento dei programmi: i primi come unità di esecuzione e i secondi come puntatori alle risorse di sistema (file, chiavi di registro).

Infine, la manipolazione diretta del **Registro di Sistema** ha dimostrato il suo ruolo critico come database centralizzato per la persistenza delle configurazioni, evidenziando come una semplice modifica di una chiave possa alterare il comportamento di un software senza ricorrere al codice sorgente.