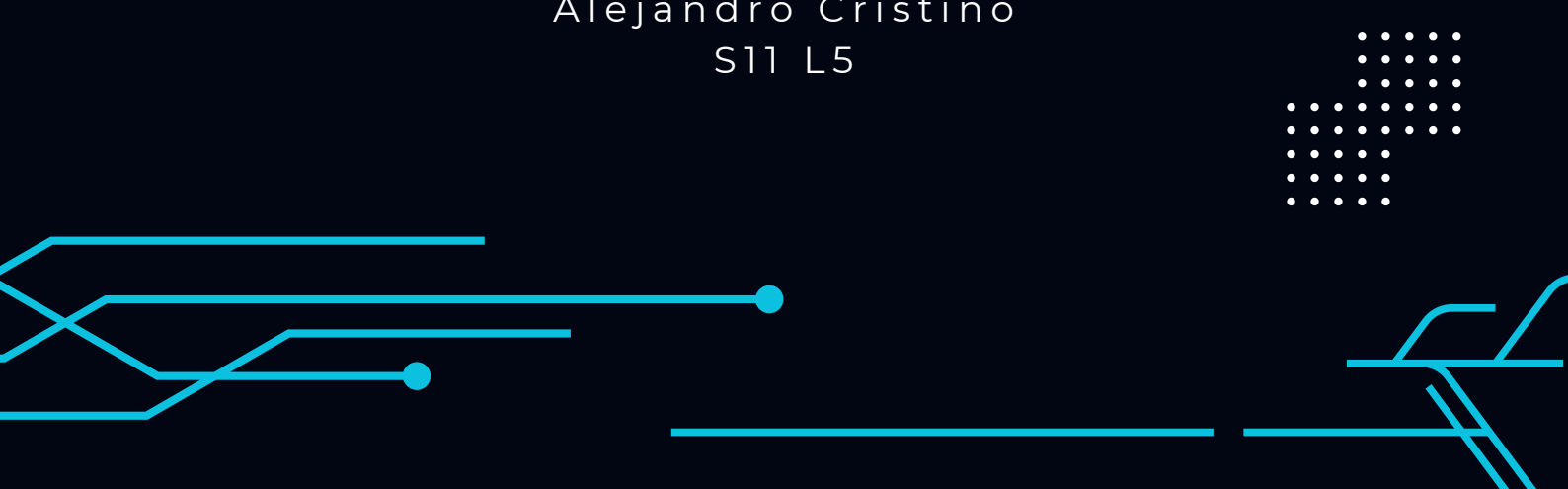




# ***REPORT***

*BONUS*

Alejandro Cristino  
S11 L5



# Traccia

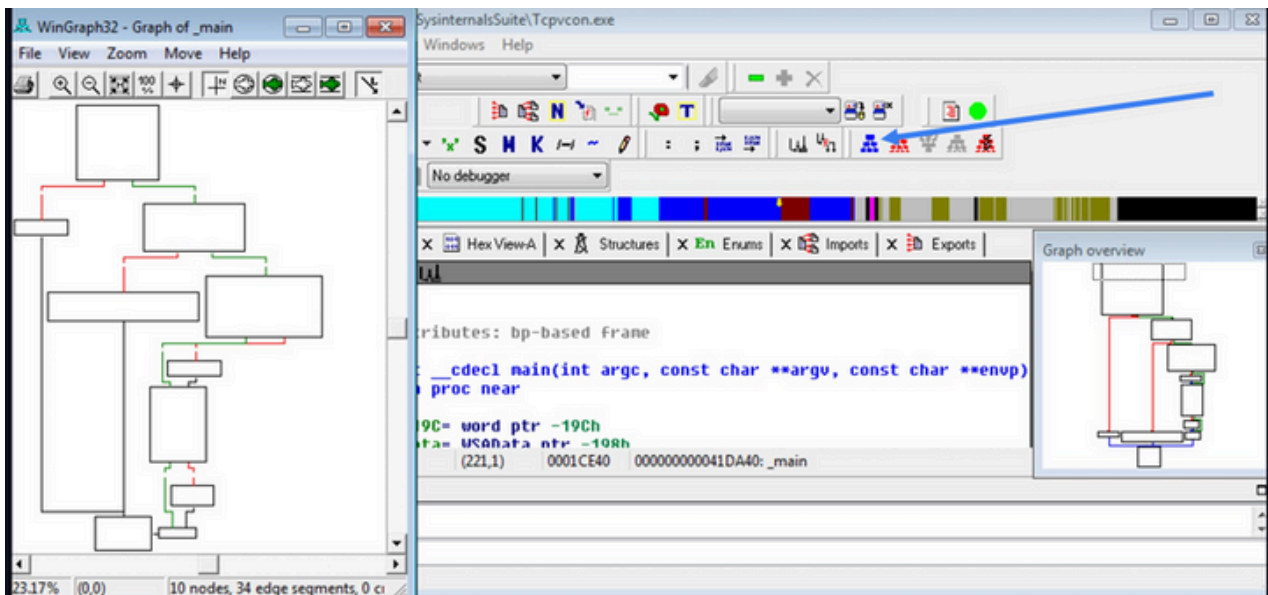
## BONUS:

Analizzare il file C: \Users\user\Desktop \Software Malware analysis\SysinternalsSuite \Tcpvcon.exe con IDA Pro

Analizzare SOLO la "funzione corrente" una volta aperto IDA La funzione corrente la visualizzo con il tasto F12 oppure con il tasto blu indicato nella slide successiva.

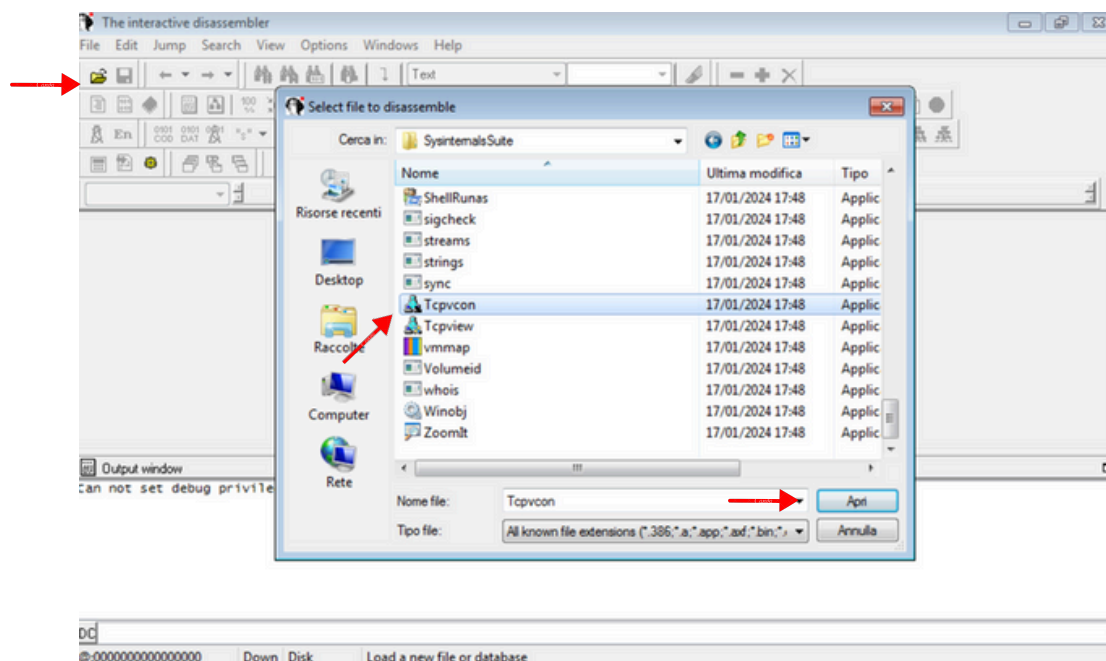
Se necessario, reperire altre informazioni con OllyDBG oppure effettuando ulteriori analisi con IDA (o altri software).

Mi interessa soltanto il significato/funzionamento/senso di questa parte di codice visualizzato alla pagina successiva.

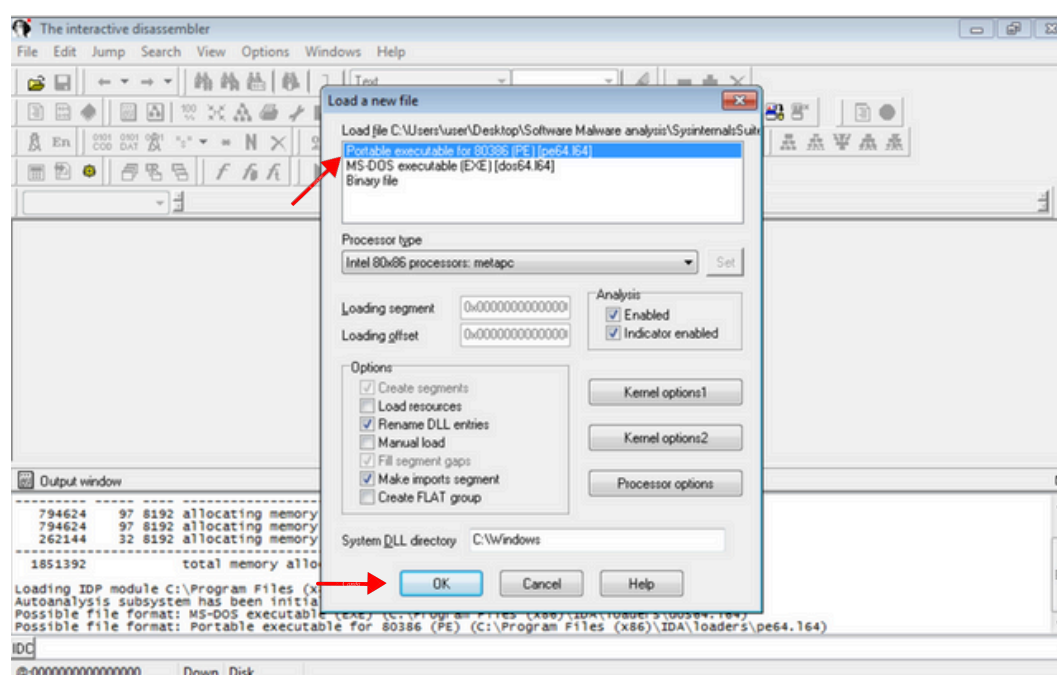


## 1.Preparazione

Per prima cosa , prepariamo il software e carichiamo il malware oggetto dell'analisi richiesta. Clicchiamo la cartella in alto a sinistra, selezioniamo il file da analizzare e clicchiamo su Apri:



Clicchiamo su “Ok” per confermare che desideriamo aprire il formato PE:



## 2. Analisi della "funzione corrente"

```

; Attributes: bp-based frame
; int __cdecl main(int argc, const char **argv, const char **envp)
_main proc near
    var_19C= word ptr -19Ch
    USADData= USADData ptr -198h
    var_4= dword ptr -4
    argc= dword ptr 8
    argv= dword ptr 0Ch
    envp= dword ptr 10h

    push    ebp
    mov     ebp, esp
    sub     esp, 19Ch
    mov     eax, dword_427284
    xor     eax, ebp
    mov     [ebp+var_4], eax
    mov     eax, [ebp+argv]
    push    eax
    lea     ecx, [ebp+argc]
    push    ecx
    push    offset aTCView ; int
    call    sub_42C0E0 ; "TCPView"
    add     esp, 0Ch
    test    eax, eax
    jnz     short loc_41D074

```

Questo blocco di codice sembra essere coinvolto nell'inizializzazione del programma, passando gli argomenti della riga di comando a una funzione che sembra avere qualcosa a che fare con "TCPView", un noto strumento per monitorare le connessioni TCP su Windows. La funzione chiamata (sub\_42C0E0) potrebbe essere responsabile dell'impostazione iniziale dell'applicazione, come l'apertura di una finestra o l'inizializzazione di strutture dati specifiche per il monitoraggio delle connessioni di rete.

```

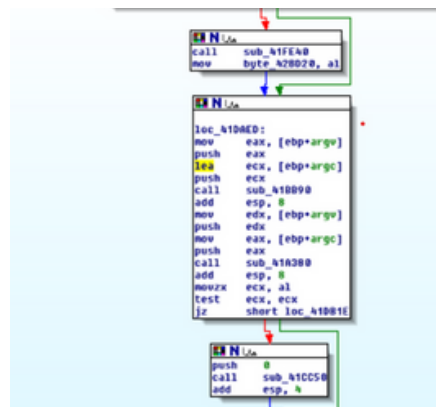
loc_41D074:
    mov     edx, 10h
    mov     [ebp+var_19C], dx
    lea     eax, [ebp+USADData]
    push    eax
    movzx   ecx, [ebp+var_19C]
    push    ecx
    call    ds:USASStartup ; uVersionRequested
    test    eax, eax
    jz      short loc_41D080

loc_41D080:
    push    offset stru_42B020 ; lpCriticalSection
    call    ds:InitializeCriticalSection
    push    offset CriticalSection ; lpCriticalSection
    call    ds:InitializeCriticalSection
    push    offset aSeDebugPrivilege ; "SeDebugPrivilege"
    call    sub_42DF50
    add     esp, 4
    call    sub_418110
    mov     byte_42B020, al
    movzx   edx, byte_42B020
    test    edx, edx
    jnz     short loc_41D080

```

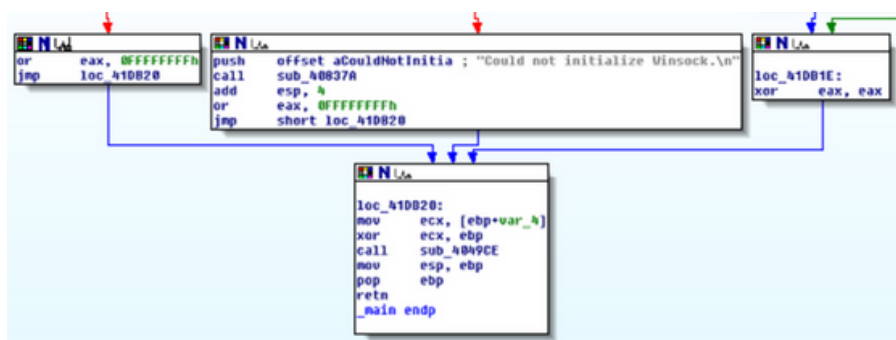
Questi blocchi di codice si occupano dell'inizializzazione di alcune componenti critiche del programma:

1. **Inizializzazione di Windows Sockets (Winsock):** Viene configurata la versione 1.1 di Winsock, necessaria per le operazioni di rete. Se WSASStartup fallisce, il programma probabilmente terminerà o restituirà un errore.
2. **Inizializzazione delle Sezioni Critiche:** Viene preparata la gestione delle sezioni critiche, necessaria per la sincronizzazione in un contesto multithreading.
3. **Impostazione del Privilegio SeDebugPrivilege:** Questo privilegio è particolarmente significativo in un contesto di analisi malware, poiché consente al programma di eseguire operazioni di debugging su altri processi, incluso quelli di sistema. Se il privilegio viene abilitato con successo, il malware potrebbe eseguire operazioni privilegiate sui processi in esecuzione.
4. **Ulteriori Funzioni:** Ci sono chiamate a funzioni specifiche (sub\_42DF50 e sub\_418110), che potrebbero contenere ulteriore logica critica, come la gestione di thread, o preparazione delle risorse di rete.



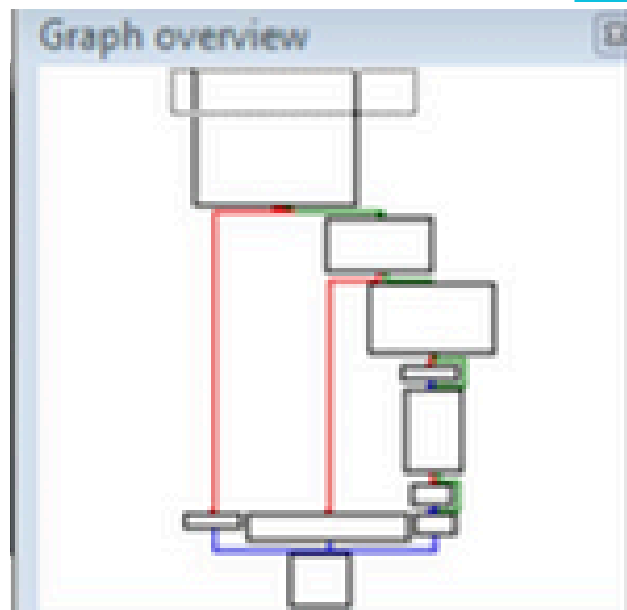
Questo blocco di codice continua la manipolazione e l'analisi degli argomenti passati al programma (tramite argv e argc). La sequenza sembra riguardare la validazione o il processamento di questi argomenti attraverso due funzioni (sub\_41BB90 e sub\_41A380), con un controllo finale per determinare se il programma deve continuare o terminare.

1. sub\_41BB90 e sub\_41A380: Queste funzioni sono cruciali, in quanto si occupano del trattamento degli argomenti passati al programma. Potrebbero gestire opzioni della riga di comando, controllare la validità degli argomenti o configurare l'ambiente di esecuzione basato su questi parametri.
2. sub\_41CC50: Senza ulteriori informazioni, questa funzione potrebbe essere utilizzata per eseguire una finalizzazione o una procedura di uscita, soprattutto se si è verificato un errore o se gli argomenti passati non sono validi.



Questa sezione del codice gestisce la terminazione del programma, assicurando che tutte le risorse siano correttamente rilasciate e che venga restituito un codice di errore appropriato se qualcosa è andato storto.

1. **Errore di Inizializzazione Winsock:** Se la funzione `WSAStartup` non riesce a inizializzare la libreria Winsock, il programma stampa un messaggio di errore e termina con un codice di errore -1. L'impostazione di `eax` a `0xFFFFFFFF` (ovvero -1) è una convenzione comune per indicare che si è verificato un errore. Molti programmi e API usano -1 come codice di errore per indicare che un'operazione non è andata a buon fine.
2. **Pulizia Finale:** Indipendentemente dal percorso di esecuzione (che si tratti di successo o errore), il programma esegue una funzione di pulizia finale prima di terminare. Questa funzione può gestire la deallocazione delle risorse o la chiusura delle connessioni aperte.
3. **Terminazione senza Errori:** Se il programma arriva al blocco `loc_41D81E`, significa che non si sono verificati errori gravi, e il programma termina restituendo 0.



## 2.1 Significato/funzionamento/senso di questa parte di codice

Dopo l'analisi precedente dei vari blocchi del codice assembly fornito, possiamo trarre le seguenti conclusioni:

### 1. Inizializzazione delle Risorse di Rete (Winsock)

- **Scopo:** La parte iniziale del codice è focalizzata sull'inizializzazione della libreria Winsock (WSAStartup). Winsock è un'API fondamentale per le applicazioni Windows che devono interagire con la rete, gestendo connessioni TCP/IP.
- **Funzionamento:** Il programma tenta di avviare Winsock alla versione 1.1, il che è tipico per applicazioni che hanno bisogno di operare a un livello basso sulla rete, gestendo direttamente socket TCP.
- **Gestione degli Errori:** Se l'inizializzazione fallisce, viene mostrato un messaggio di errore "Could not initialize Winsock" e il programma termina restituendo un codice di errore. Questa è una buona pratica di gestione degli errori, assicurando che il programma non proceda se le risorse critiche non sono disponibili.

### 2. Preparazione dell'Ambiente di Esecuzione

- **Sezioni Critiche:** Viene inizializzata una sezione critica (CriticalSection), che è un meccanismo per la gestione sicura delle risorse condivise in ambienti multithreading. Ciò indica che il programma è progettato per funzionare in un ambiente con più thread, dove la concorrenza può causare problemi di accesso simultaneo alle risorse.
- **SeDebugPrivilege:** Viene tentata l'abilitazione del privilegio SeDebugPrivilege. Questo privilegio consente al programma di eseguire operazioni di debug su qualsiasi processo, incluso quelli di sistema. La richiesta di questo privilegio suggerisce che il programma potrebbe necessitare di accesso privilegiato per eseguire operazioni di monitoraggio o manipolazione a livello di sistema.

### 3. Gestione degli Argomenti della Riga di Comando

- **Processamento degli Argomenti:** Le funzioni `sub_41BB90` e `sub_41A380` processano i parametri passati alla riga di comando (`argv` e `argc`). Queste funzioni potrebbero essere responsabili della configurazione dell'esecuzione del programma in base agli argomenti forniti, come ad esempio avviare il programma in diverse modalità operative o configurare le connessioni di rete.
- **Verifica e Validazione:** Dopo il processamento degli argomenti, viene verificato se il programma deve continuare l'esecuzione o terminare. Questo passaggio è cruciale per prevenire esecuzioni non valide o non sicure basate su input non adeguati.

### 4. Terminazione del Programma

- **Pulizia e Chiusura:** Indipendentemente dal successo o dal fallimento delle operazioni iniziali, il programma esegue una routine di pulizia prima di terminare. Vengono rilasciate le risorse, e viene restituito un valore di ritorno (0 per successo, -1 per errore), che viene poi utilizzato dal sistema operativo per capire come il programma è terminato.

## 2.2. Conclusione

Questa parte di codice è responsabile dell'inizializzazione critica delle risorse di rete e dell'ambiente di esecuzione, processando i parametri della riga di comando per determinare il comportamento del programma. Inoltre, include una robusta gestione degli errori e delle routine di pulizia finali. Tutte queste operazioni indicano che il programma è progettato per essere resiliente e sicuro, specialmente in ambienti con accesso privilegiato e potenzialmente multi-threaded.

Il fatto che il codice tenti di abilitare il privilegio `SeDebugPrivilege` e utilizzi `Winsock` suggerisce che il programma potrebbe avere funzionalità avanzate, come il monitoraggio delle connessioni di rete o l'interazione con altri processi di sistema, operazioni che sono comuni nelle applicazioni di sicurezza (o in alcuni casi, nei malware).

Questa sezione di codice rappresenta quindi una fase di preparazione fondamentale del programma, assicurando che tutte le risorse necessarie siano correttamente inizializzate e che il programma sia pronto a operare in modo sicuro ed efficiente.