ASSEMBLY COSTRUTTI NOTI

1) Push ebp
2) Mov ebp, esp
3) Push ecx
4) Push 0 ; dwReversed
5) Push 0 ; lpdwFlags
6) Call ds: InternetGetConnectedState
7) Mov [ebp+var_4], eax
8) Cmp [ebp+var_4], 0
9) Jz short loc_40102B
10) Push offset aSuccessInterne ; “Success: Internet Connection\n”
11) Call sub_40105F
12) Add esp, 4
13) Mov eax, 1
14) Jmp short loc_40103A

LEGGENDA:

- **IN ROSSO CREAZIONE DELLO STACK**
- **IN BLU FUNZIONE INTERNETGETCONNECTEDSTATE**
- **IN GIALLO IF STATEMENT**
- **IN VERDE FUNZIONE PRINTF**

PRIMO COSTRUTTO NOTO: CREAZIONE DELLO STACK

1) Push ebp
2) Mov ebp, esp

Le prime due righe di codice servono a creare uno stack per le variabili locali: notiamo la presenza dei due puntatori **EBP (Extended Base Pointer)** ed **ESP (Extended Stack Pointer)** che puntano rispettivamente alla base ed alla cima dello stack. Notiamo anche che non viene immediatamente definito lo spazio dedicato alle variabili locali.

Lo **stack è una parte della memoria che viene utilizzata per gestire variabili locali e parametri delle funzioni durante l'esecuzione di un programma**. Puoi pensare allo stack come a **una pila di piatti**, dove puoi aggiungere un piatto in cima (operazione chiamata "**push**") o rimuovere un piatto dalla cima (operazione chiamata "**pop**"). Quindi, l'uso dello stack segue una struttura **LIFO (last in, first out),** il che significa che l'ultimo elemento aggiunto è il primo elemento rimosso.

SECONDO COSTRUTTO NOTO: FUNZIONE INTERNETGETCONNECTEDSTATE

3) **Push ecx**
4) **Push 0 ; dwReversed**
5) **Push 0 ; lpdwFlags**
6) **Call ds: InternetGetConnectedState**

Le prime tre istruzioni "**push**" vengono utilizzate per mettere in cima allo stack tre parametri **(ecx, 0 e 0) che saranno successivamente passati alla funzione InternetGetConnectedState**. Questa funzione ha il compito di verificare se il computer ha accesso a Internet.

TERZO COSTRUTTO NOTO: IF STATEMENT

8) **Cmp [ebp+var_4], 0**
9) **Jz short loc_40102B**
14) **Jmp short loc_40103A** (indirizzo di memoria del salto nel caso in cui non si verifichi la condizione ZF=1

Notiamo per prima cosa la presenza di un'istruzione **cmp**, che confronta (facendo la differenza) la variabile scritta nel registro **EBP+var_4 e 0**.

- **Se il risultato di questa operazione dà come valore 0, la ZF (Zero Flag) assume valore 1**, e la condizione **jz (Jump Zero) viene confermata** e pertanto viene fatto un **salto alla locazione di memoria con indirizzo 40102B.**
- **Se il risultato dell'operazione è diverso da 0, la ZF assume valore 0** ed **il programma continuerebbe ad eseguire le righe di codice successive**, **fino ad arrivare al salto non condizionale all'indirizzo di memoria 40103A.**

Instruction	Description	signed-ness	Flags	short jump opcodes	near jump opcodes
JE JZ	Jump if equal Jump if zero		ZF = 1	74	0F 84

QUARTO COSTRUTTO: FUNZIONE PRINTF

10) **Push offset aSuccessInterne ; "Success: Internet Connection\n"**
11) **Call sub_40105F**

Ad occhio si può pensare che in queste due righe si richiama la funzione PRINTF.

SPIEGAZIONE DI OGNI SINGOLA RIGA

1) push ebp

Si **"pusha" l'extended base pointer** sulla cima dello stack

2) mov ebp, esp

Si assegna il valore del registro dell'**Extended Stack pointer al registro dell'Extended Base Pointer**

3) push ecx

Tramite l'istruzione **push**, viene posto il valore inserito nel registro "**ecx**" in cima allo stack

4) Push 0 ; dwReserved

Viene posto un valore vuoto di 4 byte sullo stack.

5) push 0 ; lpdwFlags

Come sopra, questa istruzione pone un valore vuoto di 4 byte sullo stack.

6) call ds: InternetGetConnectedState

Viene chiamata la funzione **InternetGetConnectedState** che controlla se una determinata macchina ha accesso ad Internet.

7) mov [ebp+var_4], eax

Viene copiato il valore contenuto nel registro "eax" nel registro "ebp+var_4"

8) cmp [ebp+var_4], 0

Tramite **cmp viene effettuata una sottrazione** tra il parametro contenuto nel registro **ebp+4_var** e **0** la quale modifica la **ZF (zero flag)**.

9) jz short loc_40102B

Tramite **jump viene controllata la zero flag ottenuta dalla precedente istruzione "cmp".** Se questa risulterà uguale a 1 verrà effettuato uno "**short jump**" all'indirizzo di memoria "**40102B**", **altrimenti verranno eseguite normalmente le successive righe di codice**

10) push offset aSuccessInterne ; "Success: Internet Connection\n"

Con questa istruzione viene inserita la stringa "**aSuccessInterne**" in un registro in cima allo stack

11) call sub_40105F

Viene chiamata una funzione inserita nell' indirizzo di memoria "40105F"

12) add esp, 4

Viene effettuata una somma tra il valore inserito all'interno del registro "esp"

13) mov eax, 1

Viene sostituito il valore contenuto all'interno del registro "eax" con il valore 1

14) jmp short loc_40103A

Viene effettuato un salto all'indirizzo di memoria 40103A

PROBABILE FUNZIONALITÀ IMPLETATA NEL CODICE ASSEMBLY

Dal frammento di codice proposto non possiamo determinare con certezza quale sia la funzionalità del codice proposto. In linea di massima, potrebbe trattarsi di una **backdoor**, in quanto il programma verifica la connessione ad Internet della macchina target.