

EPICODE

# CS0424 S10 / L5

ANALISI STATICA E DINAMICA

Victoria M. Braile

Prof. Antonio Pozzi

# PANORAMICA

Traccia

Introduzione

Premessa

Buone Pratiche per un Ambiente Sicuro

01.Hashing e Confronto

02.Analisi delle Librerie Importate

03.Analisi delle Sezioni

04.Identificazione Costrutti Noti

05.Tabella esplicativa Codice Assembly

Conclusioni

# TRACCIA

Con riferimento al file **Malware U3 W2 L5.exe** nella cartella “**Esercizio Pratico U3 W2 L5**” sul desktop della macchina virtuale dedicata per l'analisi dei malware, rispondere ai seguenti quesiti:

- Quali **librerie** vengono importate dal file eseguibile? Fare anche una descrizione di tali librerie.
- Quali sono le **sezioni** di cui si compone il file eseguibile del malware? Fare anche una descrizione delle sezioni individuate.

Con riferimento alla figura in slide 3, risponde ai seguenti quesiti:

- Identificare i **costrutti noti** (creazione dello stack, eventuali cicli, altri costrutti).
- Ipotesizzare il **comportamento** della funzionalità implementata.
- Fare una **tabella** per spiegare il significato delle singole righe di **codice**.

# INTRODUZIONE

L'analisi del malware è un aspetto cruciale della cybersecurity che implica lo **studio di software malevolo** per comprendere il suo **comportamento**, le sue **capacità** e i potenziali **danni** che può arrecare.

Esistono principalmente due tipi di analisi:

- **Analisi statica:** É fare l'analisi del codice **senza eseguire il malware**.
- **Analisi dinamica:** Implica l'**esecuzione del malware** in un ambiente controllato per osservare il suo comportamento in tempo reale.

Questo esercizio pratico, basato sul file Malware\_U3\_W2\_L5, ha l'obiettivo di rispondere a domande specifiche riguardanti le librerie importate, le sezioni del file eseguibile e di analizzare una porzione di codice Assembly.

# PREMESSA

Prima di entrare nel vivo dell'esercizio, ecco alcuni concetti chiave utilizzati nel corso dell'analisi.

- **Malware:** Software progettato per danneggiare, disturbare o ottenere l'accesso non autorizzato a sistemi informatici.
- **Librerie:** Collezioni di funzioni e routine pronte all'uso che i programmi possono chiamare per eseguire specifiche operazioni.
- **Sezioni di un file eseguibile:** Parti distinte di un file binario, ognuna delle quali contiene tipi specifici di dati (codice eseguibile, dati statici, risorse, ecc.).
- **Assembly:** Linguaggio di programmazione a basso livello che rappresenta le istruzioni direttamente eseguibili dalla CPU.

# BUONE PRATICHE PER UN AMBIENTE SICURO

L'esercizio richiede di effettuare una analisi statica del malware, che dunque non prevede la sua esecuzione, ciononostante c'è sempre il rischio che il malware venga eseguito per sbaglio, è sempre meglio utilizzare buone pratiche da adottare per configurare un ambiente sicuro prima di effettuare l'analisi del malware, come di seguito:

- **Configurazione schede di rete:** Per la configurazione di rete della macchina virtuale è stata abilitata un'interfaccia di rete interna per monitorare il traffico che genera potenzialmente il malware.
- **Dispositivi USB:** al fine di evitare che un dispositivo USB collegato alla macchina fisica possa essere riconosciuto anche dall'ambiente di test, è stato disabilitato il controller USB. Infatti, il malware potrebbe utilizzare il dispositivo USB per propagarsi poi sulla macchina fisica.
- **Cartelle condivise:** allo stesso modo, le cartelle condivise tra la macchina reale ed il laboratorio virtuale potrebbero essere utilizzate dal malware per propagarsi al di fuori del laboratorio. Di conseguenza, è stata disabilitata anche la condivisione di cartelle tra host e guest.
- **Creazione istantanee:** analizzando i malware c'è il rischio di arrecare danni o compromettere l'ambiente di test. Sono state quindi create delle istantanee della macchina virtuale nel suo stato iniziale, prima di iniziare tutte le analisi, in modo tale da ripristinarlo qualora ce ne fosse bisogno.

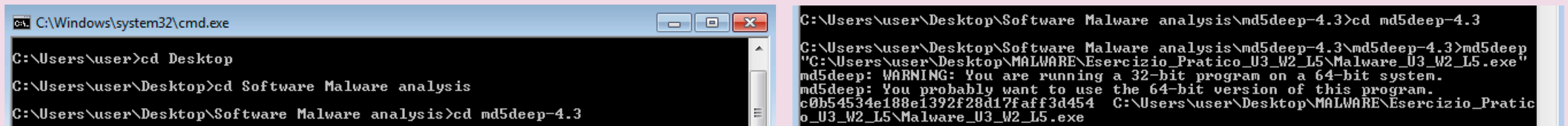
# 01. Hashing e Confronto

## 01.1. MD5DEEP

Prima di eseguire le richieste dell'esercizio, è importante fare una verifica per riconoscere se il file **Malware\_U3\_W2\_L5.exe** sia effettivamente un malware.

Per con l'analisi statica, come prima cosa viene **calcolato l'hash** del file attraverso il tool **md5deep**.

Dal **command prompt** della macchina Windows 7, dopo essersi spostati nella cartella contenente l'eseguibile di md5deep, viene dato il comando come nell'immagine.



```
C:\Windows\system32\cmd.exe
C:\Users\user>cd Desktop
C:\Users\user\Desktop>cd Software Malware analysis
C:\Users\user\Desktop\Software Malware analysis>cd md5deep-4.3

C:\Users\user\Desktop\Software Malware analysis\md5deep-4.3>cd md5deep-4.3
C:\Users\user\Desktop\Software Malware analysis\md5deep-4.3\md5deep-4.3>md5deep
"C:\Users\user\Desktop\MALWARE\Esercizio_Pratico_U3_W2_L5\Malware_U3_W2_L5.exe"
md5deep: WARNING: You are running a 32-bit program on a 64-bit system.
md5deep: You probably want to use the 64-bit version of this program.
c0b54534e188e1392f28d17faff3d454 C:\Users\user\Desktop\MALWARE\Esercizio_Pratic
o_U3_W2_L5\Malware_U3_W2_L5.exe
```

L'hash ottenuto è **c0b54534e188e1392f28d17faff3d454** ed è ora possibile confrontarlo con il database di malware conosciuti **Virus Total**.



# 01. Hashing e Confronto

## 01.2. VIRUS TOTAL

Si accede a <https://www.virustotal.com/gui/home/upload> e nella barra di ricerca viene inserito l’**hash** calcolato. Come si può vedere nelle immagini, svariate fonti hanno classificato il file come **trojan**, rafforzando la supposizione che **Malware\_U3\_W2\_L5.exe** consista effettivamente in un file **malware**.

40

/ 74

Community Score

40/74 security vendors flagged this file as malicious

Reanalyze Similar More

b71777edbf21167c96d20ff803cbcb25d24b94b3652db2f286dcd...

Size40.00 KB

Last Analysis Date14 days ago

Malware\_U3\_W2\_L5.exe

EXE

peexe

direct-cpu-clock-access

runtime-modules

armadillo

checks-network-adapters

Popular threat label

trojan.r002c0pdm21/ymacco

Threat categories

trojan

Family labels

r002c0pdm21

ymacco

Security vendors' analysis

Do you want to automate checks?

Alibaba	Trojan:Win32/Generic.2cc376c1	AliCloud	Trojan:Win/Ymacco.AMH1
Antiy-AVL	Trojan:Win32.BTSGeneric	Avast	Win32:PUP-gen [PUP]
AVG	Win32:PUP-gen [PUP]	CrowdStrike Falcon	Win/malicious_confidence_100% (W)
Cylance	Unsafe	DeeplInstinct	MALICIOUS
DrWeb	Trojan.MulDrop7.63090	Elastic	Malicious (moderate Confidence)
ESET-NOD32	Win32/Agent.WOO	Fortinet	W32/Agent.WOO!tr
GData	Win32.Trojan.Agent.DZ3C1W	Google	Detected
Gridinsoft (no cloud)	Ransom.Win32.Wacatac.oa!s1	Ikarus	Trojan.Win32.Agent
Kingsoft	Win32.Troj.Undef.a	Ljonic	Trojan.Win32.Generic.4!c

Basic properties

MD5c0b54534e188e1392f28d17faff3d454

SHA-1bb6f01b1fef74a9cfc83ec2303d14f492a671f3c

SHA-256b71777edbf21167c96d20ff803cbcb25d24b94b3652db2f286dcd6efd3d8416a

Vhash044036651d1az2crz5bz

Authentihasha8ca4bd7f0eeca509d0cab34b123894bfb81ccfdadeafde39228f9e660522686

Imphash30146c4b4aedc56db4852f147883b9a0

Rich PE header hash684f866cc4786b48eecd2e55578ee968

SSDEEP384:5PvvWL94iMg9IVrpf6lXT2pCeea0dNDJXdhcYyfdyNugreAWoWv:ubvONpf6FT2QbvhDDuGeVoW

TLSH T1C0036C2779E14077C482C6B090B6CF2AFB7B663303528187CB542A5A3E319E5EA36357

File typeWin32 EXE executable windows win32 pe peexe

MagicPE32 executable (console) Intel 80386, for MS Windows

TrIDWin32 Executable MS Visual C++ (generic) (36.8%) | Microsoft Visual C++ compiled executable (generic) (19.5%) | V

DetectItEasyPE32 | Compiler: EP:Microsoft Visual C/C++ (6.0 (1720-9782)) [EXE32] | Compiler: Microsoft Visual C/C++ (12.00.978

MagikaPEBIN

File size40.00 KB (40960 bytes)

PEID packerMicrosoft Visual C++

Curioso notare come **Virus Total** identifichi, tra i **nomi comuni** di questo file, anche **Malware\_U3\_W2\_L5.exe**, evidentemente perché molti studenti Epicode lo hanno già analizzato prima di me.

Names

Malware\_U3\_W2\_L5.exe

Lab06-02.exe

Lab05-02.exe

MAL04.exe

HW-B-3.exe



# 02. Analisi delle Librerie Importate

Per identificare le librerie importate dal file eseguibile, si ricorre un tool di analisi statica **CFF Explorer**. Questo strumento permette di esaminare le funzioni importate ed esportare dal malware.

Una volta avviato CFF Explorer, si carica il file **Malware\_U3\_W2\_L5.exe** e ci si sposta su **import directory** dal menù a sinistra. Qui è possibile visualizzare informazioni sulle **librerie importate** dal file eseguibile, mentre per ciascuna libreria il pannello inferiore mostra una lista delle **funzioni richieste** nella libreria selezionata.

Le librerie importate sono **KERNEL32.dll** e **WININET.dll** come riportato nell'immagine di seguito.

Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

# 02. Analisi delle Librerie Importate

## KERNEL32.dll

KERNEL32.dll è una **libreria di sistema** fondamentale per i sistemi operativi Windows. Fa parte del core del sistema operativo e fornisce un insieme di **funzioni che gestiscono operazioni di basso livello** come gestione della memoria, gestione dei processi e dei thread, accesso al file system e input/output (I/O) di base.

Alcune funzioni di KERNEL32.dll riportate in questa tabella includono:

- **Sleep**: Sospende l'esecuzione del thread corrente per un intervallo di tempo in millisecondi.
- **SetStdHandle**: Imposta un handle per un dispositivo standard (input, output, o errori)
- **GetStringTypeW**: Determina i tipi di carattere di una stringa wide-character (Unicode).
- **GetStringTypeA**: Determina i tipi di carattere di una stringa ANSI.
- **LCMapStringW**: Mappa una stringa Unicode a una nuova in base a info sulla localizzazione.
- **LCMapStringA**: Mappa una stringa ANSI a una nuova stringa in base a infor sulla localizzazione.
- **MultiByteToWideChar**: Converte una stringa multi-byte in una stringa wide-character.
- **GetCommandLineA**: Recupera il comando usato per avviare il processo corrente in formato ANSI.
- **GetVersion**: Recupera il numero di versione del sistema operativo Windows.
- **ExitProcess**: Termina il processo chiamante e tutti i suoi thread.
- **TerminateProcess**: Termina immediatamente un processo specificato.
- **GetCurrentProcess**: Recupera un pseudo handle per il processo chiamante.
- etc

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
000065E4	000065E4	0296	Sleep
00006940	00006940	027C	SetStdHandle
0000692E	0000692E	0156	GetStringTypeW
0000691C	0000691C	0153	GetStringTypeA
0000690C	0000690C	01C0	LCMapStringW
000068FC	000068FC	01BF	LCMapStringA
000068E6	000068E6	01E4	MultiByteToWideChar
00006670	00006670	00CA	GetCommandLineA
00006682	00006682	0174	GetVersion
00006690	00006690	007D	ExitProcess
0000669E	0000669E	029E	TerminateProcess
000066B2	000066B2	00F7	GetCurrentProcess
000066C6	000066C6	02AD	UnhandledExceptionFilter
000066E2	000066E2	0124	GetModuleFileNameA
000066F8	000066F8	00B2	FreeEnvironmentStringsA
00006712	00006712	00B3	FreeEnvironmentStringsW
0000672C	0000672C	02D2	WideCharToMultiByte
00006742	00006742	0106	GetEnvironmentStrings
0000675A	0000675A	0108	GetEnvironmentStringsW
00006774	00006774	026D	SetHandleCount
.....	.....	.....	.....

Nota: le informazioni qui riportate sono addizionali, quindi non ho fornito una istantanea di tutta la tabella delle funzioni della libreria, e di conseguenza anche la descrizione delle funzioni è limitata.

# 02. Analisi delle Librerie Importate

## WININET.dll

WININET.dll è una libreria di sistema che **fornisce un'API per le applicazioni Windows** per interagire con **protocolli Internet** come **HTTP** e **FTP**. È molto utilizzata per **operazioni di rete** come il download di file, l'invio di richieste HTTP, la gestione delle sessioni e la manipolazione dei cookie.

Di seguito una descrizione delle funzioni esportate da WININET.dll:

- **InternetOpenUrlA**: Apre una URL specificata. Utilizzata per iniziare una connessione a una risorsa specificata da una URL.
- **InternetCloseHandle**: Chiude un handle internet. Utilizzata per chiudere un handle che è stato precedentemente aperto utilizzando una funzione di WinINet.
- **InternetReadFile**: Legge i dati da un handle internet aperto.
- **InternetGetConnectedState**: Determina lo stato della connessione a Internet.
- **InternetOpenA**: Inizia l'utilizzo delle funzioni WinINet.

La presenza di KERNEL32.dll e WININET.dll nel file indica che il malware potrebbe eseguire operazioni di sistema di basso livello e interagire con Internet. **KERNEL32.dll** suggerisce che il malware potrebbe manipolare processi, memoria e file, mentre **WININET.dll** indica la possibilità di comunicazioni di rete, come il download di payload aggiuntivi o l'esfiltrazione di dati.

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00006640	00006640	0071	InternetOpenUrlA
0000662A	0000662A	0056	InternetCloseHandle
00006616	00006616	0077	InternetReadFile
000065FA	000065FA	0066	InternetGetConnectedState
00006654	00006654	006F	InternetOpenA

# 03. Analisi delle Sezioni

Per controllare le **sezioni** del file **Malware\_U3\_W2\_L5.exe** bisogna spostarsi nella sezione **section headers**.

Il pannello principale a destra mostra le informazioni circa le sezioni di cui si compone l'eseguibile.

Come riportato di seguito, l'eseguibile in esame è composto da tre sezioni: **.text**, **.rdata** e **.data**.

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations N...	Linenumbers ...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00004A78	00001000	00005000	00001000	00000000	00000000	0000	0000	60000020
.rdata	0000095E	00006000	00001000	00006000	00000000	00000000	0000	0000	40000040
.data	00003F08	00007000	00003000	00007000	00000000	00000000	0000	0000	C0000040

- **.text**: contiene le istruzioni che la CPU eseguirà una volta che il software sarà avviato. Generalmente questa è l'unica sezione di un file eseguibile che viene eseguita dalla CPU, in quanto tutte le altre sezioni contengono dati o informazioni a supporto.
- **.rdata**: include le informazioni circa le librerie e le funzioni importate ed esportate dall'eseguibile.
- **.data**: contiene i dati e le variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma.



# 04. Identificazione Costrutti Noti

Il codice assembly fornito fa uso di diverse istruzioni comuni per la gestione dello stack e il controllo del flusso.

```
push    ebp
mov     ebp, esp
push    ecx
push    0          ; dwReserved
push    0          ; lpdwFlags
call    ds:InternetGetConnectedState
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jz      short loc_40102B
```

**Creazione dello Stack:** `push ebp` e `mov ebp, esp` sono istruzioni tipiche per inizializzare un nuovo frame dello stack. Il registro base **pointer** (`ebp`) viene **salvato** nello **stack** e poi **impostato** sullo **stack pointer** (`esp`).

Queste sono istruzioni che servono a **posizionare valori sullo stack**, che saranno usati come **parametri** per la **chiamata** a funzione `InternetGetConnectedState` che verifica lo stato della connessione Internet

**Controllo del Flusso:** `cmp` e `jz` vengono usati per il **controllo condizionale** del **flusso**. Formano un **ciclo if** che controlla che sia **zero** il risultato di `InternetGetConnectedState`. Se la condizione di **comparazione** è **soddisfatta**, si esegue un **salto** a un'altra parte del codice.

# 04. Identificazione Costrutti Noti

```
push    offset a$SuccessInterne ; "Success: Internet Connection\n"  
call    sub_40117F  
add     esp, 4  
mov     eax, 1  
jmp     short loc_40103A
```

Il registro **esp** viene **incrementato** di **4**, per **liberare** lo **spazio** occupato dal messaggio nello stack.

Viene impostato il valore di **eax** a **1**, valore di ritorno per indicare che la connessione è attiva.

Il programma salta poi all'indirizzo **loc\_40103A**.

Gestione della connessione avvenuta con successo:  
Viene inserito nell'indirizzo di memoria **[ebp+var\_4]** il messaggio **Success: Internet Connection\n** per indicare che la **connessione** è **attiva**.

Viene poi **chiamata** la funzione **sub\_40117F**, che gestisce il messaggio di **successo**.

# 04. Identificazione Costrutti Noti

```
loc_40102B:                ; "Error 1.1: No Internet\n"  
push    offset aError1_1NoInte  
call    sub_40117F  
add     esp, 4  
xor     eax, eax
```

Viene **chiamata** la funzione **sub\_40117F**, una funzione che **gestisce** il **messaggio di errore**.  
Il registro **esp** viene **incrementato** di **4**, per liberare lo spazio occupato dal messaggio nello stack.  
Il **registro eax** viene **azzerato**.

Gestione dell'errore di connessione:  
Questo blocco inizia con il messaggio **Error 1.1: No Internet\n**, che indica che la **connessione è assente**.  
Viene inserito nell'indirizzo di memoria l'indirizzo della funzione **aError1\_1NoInte**.



# 04. Identificazione Costrutti Noti



```
loc_40103A:  
mov     esp, ebp  
pop     ebp  
retn  
sub_401000 endp
```

## Fine del programma:

Questo blocco ripristina il **valore originale di esp e ebp**. Viene eseguita l'istruzione **retn**, che **termina la funzione corrente**.

**sub\_401000 endp** potrebbe essere il **punto di inizio di un'altra funzione o semplicemente il punto di terminazione del programma**.

## Comportamento della Funzionalità Implementata

Il codice **verifica** lo stato della **connessione Internet** e stampa un **messaggio** appropriato in base all'**esito** della **verifica**.

Se **c'è una connessione Internet**, viene stampato "**Success: Internet Connection**", altrimenti viene stampato "**Error 1.1: No Internet**".

# TABELLA ESPLICATIVA CODICE ASSEMBLY

INDIRIZZO	ISTRUZIONE	SIGNIFICATO
-	<b>push ebp</b>	Salva il registro ebp sullo stack
-	<b>mov ebp, esp</b>	Inizializza il frame dello stack con il valore di esp
-	<b>push ecx</b>	Salva il registro ecx sullo stack
-	<b>push 0</b>	Spinge 0 (dwReserved) sullo stack
-	<b>push 0</b>	Spinge 0 (lpdwFlags) sullo stack
-	<b>call ds:InternetGetConnectedState</b>	Chiama la funzione per verificare la connessione Internet
-	<b>mov [ebp+var_4], eax</b>	Salva il valore di ritorno in var_4
-	<b>cmp [ebp+var_4], 0</b>	Confronta var_4 con 0 (nessuna connessione)
-	<b>jz short loc_401028</b>	Se var_4 è 0, salta a loc_401028
-	<b>push offset aSuccessInterne</b>	Spinge l'offset della stringa "Success: Internet Connection"

# TABELLA ESPLICATIVA CODICE ASSEMBLY

INDIRIZZO	ISTRUZIONE	SIGNIFICATO
-	<b>call sub_40117F</b>	Chiama la funzione per stampare il messaggio
-	<b>add esp, 4</b>	Ripulisce lo stack rimuovendo l'argomento passato alla funzione
-	<b>mov eax, 1</b>	Imposta il registro eax a 1 (indicando successo)
-	<b>jmp short loc_40103A</b>	Salta a loc_40103A
loc_401028	<b>push offset aError1_1NoInte</b>	Spinge l'offset della stringa "Error 1.1: No Internet"
-	<b>call sub_40117F</b>	Chiama la funzione per stampare il messaggio
-	<b>add esp, 4</b>	Ripulisce lo stack rimuovendo l'argomento passato alla funzione
-	<b>xor eax, eax</b>	Imposta il registro eax a 0 (indicando fallimento)
loc_40103A	<b>mov esp, ebp</b>	Ripristina esp al valore di ebp
-	<b>pop ebp</b>	Ripristina ebp dallo stack
-	<b>retn</b>	Ritorna alla funzione chiamante

# CONCLUSIONI

L'esercizio ha permesso di approfondire diverse **tecniche di analisi di un file eseguibile malevolo**, identificando le **librerie importate**, le **sezioni** del file e analizzando dettagliatamente il **codice Assembly**.

L'analisi ha evidenziato l'**importanza di comprendere il funzionamento interno dei malware per sviluppare contromisure efficaci**.

Ulteriori passaggi potrebbero includere l'**analisi dinamica del malware** in un ambiente controllato per **osservare il comportamento** in tempo reale e l'**implementazione** di tecniche di **rilevamento**.

Per mitigare le vulnerabilità, è fondamentale **mantenere aggiornati i sistemi**, utilizzare **software di sicurezza avanzati** e **formare gli utenti** sulle buone pratiche di sicurezza informatica.

Questo esercizio ha fornito un'esperienza pratica e dettagliata nell'analisi di un file eseguibile malevolo, rafforzando la **comprensione delle tecniche di reverse engineering** e migliorando le competenze nella **difesa contro i malware**.