



UNIVERSITÀ DEGLI STUDI DI  
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base  
Corso di Laurea Magistrale in Ingegneria Informatica

## Malware Analysis

Anno Accademico 2019/2020

Prof.  
**Simon Pietro Romano**

Autori  
**Forte Antonio**  
**matr. M63/845**

**Galdi Emanuele**  
**matr. M63/833**

**Maresca Fabio**  
**matr. M63/846**

# Indice

<b>Introduzione</b>	<b>1</b>
<b>1 Introduzione alla Malware Analysis</b>	<b>2</b>
1.1 I malware . . . . .	2
1.1.1 Definizione e classificazione dei malware . . . . .	2
1.1.2 Quanto possono essere pericolosi i malware . . . . .	6
1.2 La malware analysis . . . . .	7
<b>2 IOCs e Threat Intelligence</b>	<b>10</b>
2.1 Indicatori di compromissione . . . . .	10
2.1.1 Cosa sono gli IoC . . . . .	10
2.1.2 Esempi di IoC . . . . .	11
2.1.3 Standard di rappresentazione degli IoC . . . . .	13
2.2 Threat intelligence . . . . .	16
<b>3 Basic Static Analysis</b>	<b>20</b>
3.1 Determinare il tipo di file . . . . .	21
3.2 Generare un'impronta del malware . . . . .	23
3.3 Scansione del file binario sospetto con motori antivirus . . . . .	24
3.4 Identificare eventuali tecniche di packing e offuscamento . . . . .	26

3.5 Estrarre stringhe dal file sospetto . . . . .	27
3.6 Estrarre informazioni dalla struttura del file PE . . . . .	29
<b>4 Basic Dynamic Analysis</b>	<b>37</b>
4.1 Definizione e vantaggi della Basic Dynamic Analysis . . . . .	37
4.2 Tecniche di monitoring e tool . . . . .	38
<b>5 Advanced Malware Analysis e Forensics</b>	<b>44</b>
5.1 Advanced Malware Analysis . . . . .	44
5.2 Analisi forense e memory forensics . . . . .	46
<b>6 Automatic Malware Analysis</b>	<b>49</b>
6.1 Sandbox . . . . .	49
6.2 Proprietà delle Sandbox . . . . .	50
6.3 Classificazione delle Sandbox . . . . .	50
6.4 Tecniche anti-sandboxing . . . . .	52
6.5 Le Sandbox più utilizzate . . . . .	53
<b>7 Analisi del file <i>places.docx.exe</i></b>	<b>55</b>
7.1 Basic Static Analysis . . . . .	55
7.2 Basic Dynamic Analysis . . . . .	70
7.3 Esempio di regola YARA . . . . .	78
7.4 Automatic Malware Analysis tramite Sandbox . . . . .	79
7.4.1 Joe Sandbox . . . . .	79
7.4.2 Falcon Sandbox . . . . .	90
7.4.3 Any.Run . . . . .	96
<b>A Setup laboratorio</b>	<b>100</b>

## Bibliografia

112

# Introduzione

"Andy, sto solo facendo il mio lavoro, niente di personale, mi dispiace" è il messaggio che è possibile trovare all'interno del malware MyDoom, diffuso tramite mail phishing con l'obiettivo di installare una backdoor nel sistema infetto per consentirne il controllo da remoto. Ciò ci fa capire come i malware ci coinvolgano in prima persona, anche a noi comuni utilizzatori del web, e di come essi siano sempre più una realtà complessa e trasversale, che non comprende la sola informatica ma che abbraccia anche la psicologia e il social engineering.

Questo elaborato si pone l'obiettivo di sensibilizzare il lettore alla pericolosità di queste minacce con un primo capitolo quasi introduttivo, in cui sono illustrate le conseguenze di attacchi realmente avvenuti dopo aver catalogato i malware in maniera formale. Sempre in questo primo capitolo verrà introdotto il tema centrale dell'elaborato, ossia la malware analysis.

Nel secondo capitolo verranno trattati i concetti di indicatori di compromissione e threat intelligence. Nel terzo verrà approfondita la Basic Static Analysis, illustrando i suoi compiti ed obiettivi, ed i tool di cui essa si serve. Nel quarto capitolo verrà approfondita la Basic Dynamic Analysis (illustrando anche in questa occasione i tool necessari), analizzando il perché essa sia necessaria e complementare a quella statica. Si è scelto inoltre di riservare, per completezza, il quinto capitolo alla spiegazione generale di Advanced Malware Analysis e Forensics. Viene infine dedicato il sesto capitolo alla Automatic Malware Analysis, evidenziando anche i vantaggi e gli svantaggi del suo utilizzo.

Come ben sappiamo, però, per poter realmente comprendere tutti questi concetti teorici è necessario "sporcarsi le mani", quindi nel settimo capitolo è stato proposto un esempio di analisi di un malware reale attraverso l'utilizzo di un ambiente virtualizzato ad hoc (il cui setup è illustrato nell'appendice A) e l'utilizzo di alcune sandbox.

# **Capitolo 1**

## **Introduzione alla Malware Analysis**

### **1.1 I malware**

#### **1.1.1 Definizione e classificazione dei malware**

Il termine malware (contrazione di malicious software) è un termine generico che descrive un programma/codice dannoso che mette a rischio un sistema. Ostili, invasivi e volutamente maligni, i malware cercano di invadere, danneggiare o disattivare computer, sistemi, reti, tablet e dispositivi mobili, spesso assumendo il controllo parziale delle operazioni del dispositivo, interferendo con il suo normale funzionamento.

La diffusione dei malware può avvenire in tanti modi diversi, ad esempio tramite navigazione in rete, phishing, sfruttamento di vulnerabilità di software e servizi presenti su macchine connesse in rete, ma anche tramite chiavette USB, bluetooth, e canali di alimentazione (ad esempio le prese USB degli aeroporti o di altri luoghi pubblici).

I malware possono essere classificati in base a vari aspetti:

- La necessità di un programma host da parassitare
- La capacità di potersi replicare
- Il tipo di azioni compiute una volta raggiunto il sistema target

Nella seguente immagine (figura 1.1.1) possiamo vedere una rappresentazione ad albero della classificazione in base ai primi 2 aspetti.

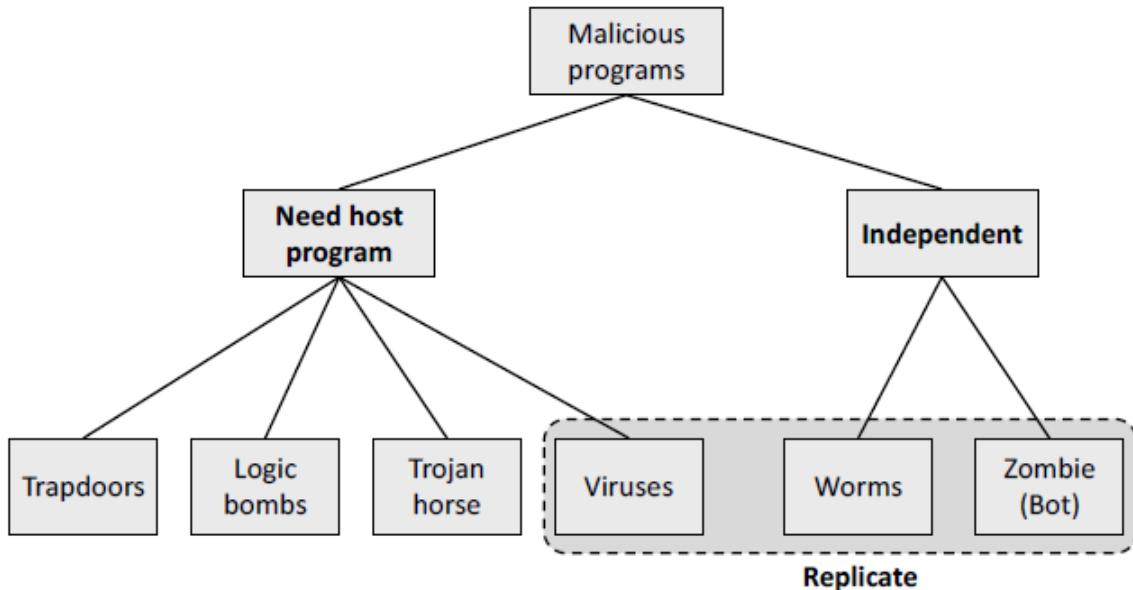


Figura 1.1.1: Classificazione dei malware

Hanno bisogno di un programma host e non si replicano:

- Trapdoors (noti anche come Backdoors): una backdoor (letteralmente "porta sul retro") ha l'obiettivo di prendere il controllo del sistema target. In particolare, essa permette ad un attaccante di accedere al sistema target ed eseguire comandi su di esso bypassandone i meccanismi di accesso.
- Logic bombs: una logic bomb è un programma con del codice malevolo che detona solo nel momento in cui determinate condizioni sono soddisfatte. Le condizioni possono essere le più svariate, come l'installazione di un programma, l'arrivo di una certa data o la creazione di un file.
- Trojans: il nome prende spunto dalla leggenda del Cavallo di Troia, proprio perché si presenta come un programma innocuo, ma che in realtà nasconde al suo interno un comportamento malevolo. Più nello specifico, fornisce all'utente le funzionalità che questi si aspetta da tale programma, ma compiendo anche delle azioni malevoli in background.

Hanno bisogno di un programma host e si replicano:

- Viruses: all’atto della sua esecuzione, un virus cerca di replicarsi ricopiando il suo codice in altri files, infettandoli. Il nome deriva dall’analogia con i virus biologici, i quali si replicano all’interno di cellule di altri organismi.

Non hanno bisogno di un programma host e si replicano:

- Worms: un worm è un programma indipendente che può replicarsi con l’obiettivo di diffondersi su altri computer tramite la rete, solitamente sfruttandone delle vulnerabilità oppure utilizzando vari stratagemmi. Un tipico stratagema potrebbe essere quello della diffusione tramite posta elettronica, dove il worm invia una copia di sé stesso agli indirizzi e-mail memorizzati nel sistema infettato. Essi possono effettuare azioni malevoli su tali sistemi, oppure semplicemente diffondersi. In quest’ultimo caso possono comunque far danni saturando la banda della rete.
- Zombies (Bots): un bot è un programma che, una volta infettata la macchina, viene attivato da remoto da un attaccante o da un centro di command and control (C&C) per lanciare, ad esempio, un attacco di tipo Denial of Service (DoS) verso altre macchine. Una macchina infettata da un bot diventa solitamente parte di una botnet, ossia un insieme di macchine infettate dallo stesso bot comandate tutte dallo stesso attaccante, in modo da poter sferrare un attacco di tipo Distributed DoS (DDoS).

Per quanto riguarda il tipo di azioni compiute, una possibile classificazione è la seguente:

- Adwares: programmi che mostrano messaggi pubblicitari indesiderati all’utente, solitamente attraverso dei pop-up o la redirezione di un browser verso pagine di tipo commerciale.
- Spywares: hanno l’obiettivo di raccogliere informazioni dal sistema target e trasmetterle all’attaccante. Tali informazioni possono essere ottenute ricercandole all’interno dei file presenti sul sistema, oppure monitorando le attività dell’utente tramite la registrazione dello schermo, dei tasti premuti e/o delle attività di rete.

- Keyloggers: sono particolari tipi di spyware specializzati nell’intercettazione e memorizzazione dei tasti premuti dall’utente del sistema target, in modo da rubargli informazioni sensibili. Spesso utilizzano dei filtri in modo da ritornare soltanto informazioni vicine a determinate parole chiave, come ad esempio "login" o "password".
- Ransomwares: hanno l’obiettivo di criptare i dati di un utente del sistema target, promettendo la chiave necessaria alla loro decriptazione solo a valle del pagamento di un riscatto. Possono criptare alcuni o tutti i files presenti sul sistema, oppure possono impedire del tutto l’accesso dell’utente al sistema, ad esempio criptando il MBR del disco.
- Downloaders: sono programmi che hanno la capacità di connettersi ad Internet e scaricare programmi o files maligni addizionali sul sistema target.
- Droppers: simili ai downloaders, ma hanno al proprio interno i files maligni addizionali da introdurre nel sistema target, senza aver bisogno di alcuna connessione ad Internet
- Advanced Persistent Threats (APTs): come si può intuire dal nome, sono attacchi che hanno l’obiettivo di permanere quanto più tempo possibile all’interno del perimetro target. Spesso questo tipo di attacchi è indirizzato a infrastrutture governative, multinazionali o grandi imprese, al fine di sottrarre informazioni sensibili, brevetti, segreti industriali e militari. Il fatto che siano “avanzati” è dimostrato dalle risorse necessarie alla loro attuazione: tecnologia avanzata, servizi di intelligence e una quantità di risorse economiche elevata.
- Rootkits: inizialmente i rootkits erano malware con l’obiettivo di ottenere i privilegi di root o amministratore sul sistema target, ossia i massimi privilegi possibili. I rootkit moderni non si preoccupano più solamente di elevare i privilegi, ma piuttosto mascherano il caricamento e l’esecuzione di un altro software, solitamente un malware.
- Fileless: sono malware che, come dice il nome, non memorizzano alcun file sull’hard disk. I fileless eseguono dalla memoria del sistema target, il che li rende molto complicati da rilevare. Per portare a termine le proprie attività malevoli essi sfruttano strumenti legittimi e trusted,

come ad esempio PowerShell o macro di Microsoft Office, e utilizzano come meccanismo di persistenza, ad esempio, il registro di sistema.

### 1.1.2 Quanto possono essere pericolosi i malware

Con il passare del tempo i malware stanno diventando una minaccia sempre più grave sotto più punti di vista: aumentano esponenzialmente in numero, evolvono le proprie funzioni in base alla psicologia umana, alcuni riescono ad aggirare i sistemi di monitoraggio. Basti pensare che negli anni '90 venivano prodotti un centinaio di malware al giorno, mentre attualmente ne vengono prodotti circa 300.000 (tra istanze totalmente nuove e copie che apportano leggere modifiche). Per quanto riguarda lo studio della psicologia umana un esempio lampante è notPetya, un'evoluzione del predecessore (Petya) che, invece di causare un arresto improvviso del sistema e insospettendo così la vittima, causa un legal reboot, fingendo così un aggiornamento di sistema programmabile con data e ora scelti dalla stessa vittima. Alcuni malware riescono ad aggirare i sistemi di monitoraggio utilizzando tecniche di evasione anti-sandboxing e anti-debugging.

Inoltre, essi sono diventati estremamente potenti, riuscendo a provocare danni gravissimi su scala nazionale. Alcuni esempi che lo dimostrano sono l'attacco ad una centrale nucleare in Iran causato dal malware Stuxnet, o il blackout che ha colpito ospedali, università, aziende e strade in Ucraina a causa di BlackEnergy. Ancora, NotPetya ha reso inaccessibili i dati degli ospedali del Regno Unito, creando poi anche problemi di sicurezza nazionale in UK e Asia colpendo strutture governative critiche; Mirai ha lanciato un attacco DDoS verso i server DNS di DynDNS, impedendo la fruizione dei servizi di Internet a una grande fetta della popolazione di Europa e Nord America.

Per arginare la criticità di questi problemi è necessario studiare il comportamento dei malware, in modo da trovare delle contromisure e cercare di prevenire attacchi futuri. Questo è quello che fa la malware analysis.

## 1.2 La malware analysis

La malware analysis è lo studio del comportamento dei malware. In particolare il file binario sospetto viene analizzato in ambiente controllato per identificarne le caratteristiche e le funzionalità. Il primo obiettivo della malware analysis è quello di appurare se un dato binario sospetto sia malevolo o meno. Nel caso in cui esso sia malevolo, bisogna capire a quale categoria di malware appartiene ed eventualmente se esso è una variante di un particolare malware già noto, evidenziandone eventuali similarità e differenze.

Grazie allo studio del comportamento dei malware è quindi possibile sia pensare a delle contromisure e sia determinare dei pattern utili a identificare e debellare più facilmente un malware simile in futuro. Inoltre, studiare il comportamento dei malware potrebbe rivelarsi utile a capire le intenzioni dell'attaccante. Infatti la malware analysis è sempre più utilizzata dai SOC (Security Operation Center) proprio perché non basta unicamente eliminare la minaccia, ma bisogna capire origine e motivo dell'attacco, e come esso sia avvenuto. Ad esempio un'azienda potrebbe non solo voler eliminare le minacce all'interno del proprio perimetro ma anche voler sapere in che modo sono state aggirate le proprie difese e da quanto tempo, chi sta utilizzando determinati dati sensibili e perché.

La malware analysis segue vari step (figura 1.2.1):

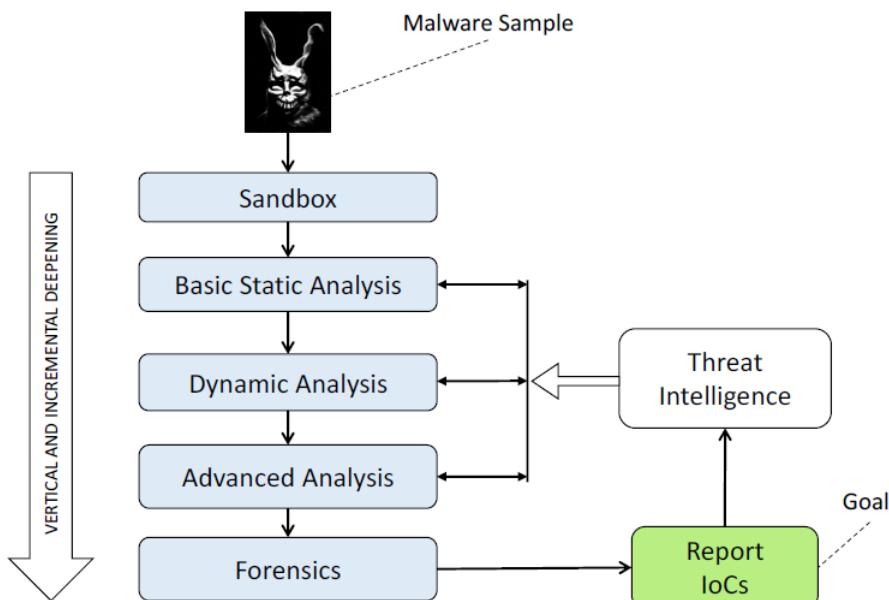


Figura 1.2.1: Processo della malware analysis

- Automatic analysis: il file binario viene usato come input di una sandbox, un’ambiente costruito ad hoc per analizzare file sospetti; questa restituirà un report sulle caratteristiche e il comportamento del file. In realtà un analista non ha come riferimento una singola sandbox ma ne utilizza diverse, perché queste possono presentare funzionalità differenti e raggiungere determinate tecniche di evasione in cui altre sandbox non riescono.
- Basic static analysis: analizza il file binario senza eseguirlo e permette di ricavarne delle informazioni. Utilizza ad esempio tool che permettono di identificare il tipo del file, creare e lavorare con l’hash crittografico di questo file, estrarre stringhe, controllare se il file sia offuscato oppure packed, ed altre informazioni utili a capire le sue intenzioni.
- Basic dynamic analysis: analizza il comportamento del file eseguendolo in un ambiente controllato e si concentra quindi sullo studiare quali processi vengono creati, quali connessioni vengono instaurate e verso chi, quali modifiche vengono apportate al file system (e quindi se determinati file vengono creati, copiati, spostati, eliminati) ed al registro di sistema.
- Advanced analysis: si divide in due approcci, advanced static analysis e advanced dynamic analysis, e studia nello specifico le azioni del malware attraverso il codice. In particolare la advanced static analysis studia il codice tramite l’utilizzo di disassemblatori, la advanced dynamic analysis studia le singole istruzioni tramite debugging. Essa è in grado di ottenere informazioni che non è possibile individuare nelle fasi precedenti poiché scende molto più nel dettaglio, ma è attuabile solo se si possiedono conoscenze avanzate di linguaggio assembly, architettura dei calcolatori e sistemi operativi.
- Forensics: il suo obiettivo è quello di individuare gli autori e le conseguenze di un incidente di sicurezza. In questo contesto si è però interessati alla memory forensics, ossia una tecnica basata sull’estrazione di informazioni dalla memoria centrale di un sistema che non sarebbero altrimenti reperibili in altro modo.
- Report IoCs: le informazioni raccolte dall’analisi vengono sintetizzate sotto forma di Indicators of Compromise (IoCs), che ne permettano il futuro rilevamento.

- Threat Intelligence: gli IoC e le informazioni ricavate dall’analisi vengono condivise e utilizzate per migliorare gli strumenti di analisi, in modo da riuscire a rilevare i nuovi malware.

Nonostante l’analisi automatica tramite sandbox sia il primo step nell’analisi di un malware, in quanto è un modo semplice e immediato per farsi un’idea generale sul comportamento del malware, in questo elaborato essa verrà approfondita in un secondo momento per evidenziare quanto il processo di analisi possa essere lungo e tedioso, soprattutto se si considera l’enorme quantità di malware con cui un analista possa entrare in contatto ogni giorno. Gli strumenti di analisi automatica si rivelano quindi una risorsa importantissima per l’analisi dei malware, però, a differenza di quanto si possa pensare, la professione dell’analista rimarrà sempre centrale e di primaria importanza. Tali strumenti infatti non possono sostituire la figura dell’analista, in quanto possono fornire un’idea generale del file analizzato, ma l’unico modo per scoprire nel dettaglio le funzionalità di un malware rimane quello di procedere manualmente. Questo approccio inoltre risulta essere necessario davanti a malware molto avanzati, o che semplicemente riescono ad attuare quelle tecniche di evasione anti-sandboxing e anti-debugging citate nel paragrafo precedente.

# Capitolo 2

## IOCs e Threat Intelligence

Prima di procedere con i vari passi dell’analisi, in questo capitolo verranno introdotti i concetti di Indicator of Compromise (IoC) e Threat Intelligence.

### 2.1 Indicatori di compromissione

#### 2.1.1 Cosa sono gli IoC

Gli Indicators of Compromise (IoC) sono artefatti osservabili in un sistema o in una rete che indicano la presenza di una compromissione con un alto grado di confidenza. Questi indicatori sono il risultato dell’analisi di piattaforme di threat intelligence, di sandbox e di tool e servizi che si occupano di descrivere le caratteristiche di un binario ricevuto come input. Questo perché, quando un malware esegue le sue attività, queste possono essere memorizzate nei file di log o essere tracciate attraverso appositi strumenti come verrà mostrato in seguito.

Gli IoC possono essere dei comportamenti anomali come la creazione (o modifica ed eliminazione) di determinate chiavi di registro, oppure la creazione (o download, copia, modifica ed eliminazione) di determinati file; allo stesso modo contattare URL e indirizzi IP sospetti o già etichettati come dannosi (blacklistati). Ancora, gli indicatori di compromissione possono essere una determinata sequenza di system call o di funzioni. Inoltre, gli IOC non necessariamente devono corrispondere a comportamenti

anomali, ma possono essere anche determinate caratteristiche del file, quali ad esempio dei metadati, o hash e signatures. Essi quindi si rivelano uno strumento utilissimo per la rilevazione dei malware.

Purtroppo, gli indicatori di compromissione presentano due limitazioni intrinseche:

- non sono universali, ovvero se analizziamo un file con più piattaforme di threat intelligence avremo più report che possono essere molto simili ma non identici, questo significa che gli indicatori possono differire a seconda dell'analisi effettuata e a seconda della cosa su cui si concentrano di più queste piattaforme.
- non c'è un unico formato standard per rappresentarli

### 2.1.2 Esempi di IoC

Di seguito è riportata una lista più dettagliata di alcuni esempi di IOC.

- Connessioni verso URL ed indirizzi IP sospetti: instaurare una connessione verso URL o indirizzi IP sospetti può essere un chiaro indicatore di compromissione, in quanto un malware potrebbe star contattando un command and control server, o cercando di scaricare file ed eseguibili aggiuntivi, oppure ancora inviare informazioni raccolte.
- Traffico di rete anomalo: uno dei più grandi campanelli d'allarme è il traffico dati anomalo all'interno del sistema (o perimetro), sia proveniente dall'esterno e sia dal sistema stesso. Un esempio di questo secondo caso è una connessione instaurata da un malware che vuole contattare un command and control server. Un buon approccio sarebbe quindi quello di monitorare entrambi i tipi di connessione.
- Anomalie legate ai privilegi degli utenti: bisogna tenere sotto controllo quanti e quali file vengono acceduti e da che tipo di utente, questo perché gli attaccanti sono soliti attuare una privilege escalation a partire da account già compromessi precedentemente, o risalire ad altri account a partire da quello attuale.

- Anomalie geografiche: si ha ad esempio quando avviene uno scambio di dati da/verso parti del mondo che non hanno niente a che vedere con il sistema, o anche quando viene effettuato un login da indirizzi IP di paesi diversi.
- Frequenza elevata nei tentativi di login: un tentativo di login errato non necessariamente significa qualcosa di anomalo, potrebbe essere un impiegato che ha digitato male la password; tuttavia, nel momento in cui si succedono più tentativi errati in poco tempo, è probabile che qualcuno stia cercando di accedere al sistema provando ad indovinare le credenziali.
- Numerose letture da database: i dati sensibili sono spesso conservati all'interno di database, quindi la lettura di una grande quantità di informazioni o numerosi tentativi di lettura possono essere un indice di compromissione.
- Dimensione delle risposte HTML: nel momento in cui si estorcono grandi quantità di informazioni le risposte HTML hanno una dimensione diversa dal solito.
- Numerose richieste di uno stesso file: un attaccante potrebbe provare più varianti di uno stessa richiesta prima di capire quale sia l'exploit vero e proprio.
- L'applicazione usa un port insolito: un'applicazione potrebbe utilizzare un port diverso da quello standard perché un attaccante sta comunicando con un command and control server, mascherando il suo traffico come quello dell'applicazione in questione.
- Cambiamenti sospetti al registro oppure a file di sistema: gli attaccanti spesso utilizzano il registro oppure accedono a file e configurazioni di sistema per stabilire la persistenza all'interno dell'host infetto.
- Traffico anomalo di richieste DNS: questo può essere indice di un'esfiltrazione di dati o di comunicazioni con un command and control server.
- Blocchi di dati in locazioni insolite: gli attaccanti spesso, prima di esportare grandi quantità di informazioni, le aggregano in una determinata locazione.

### 2.1.3 Standard di rappresentazione degli IoC

Come anticipato precedentemente, attualmente non c'è un formato standard per rappresentare gli indicatori di compromissione, piuttosto vi sono più modi per rappresentarli, tra cui OpenIOC, Yara, TAXII, STIX, CybOX. Questo è importante soprattutto perché, come verrà spiegato nel paragrafo successivo, uno degli aspetti principali della threat intelligence è quello di condividere le informazioni tra le organizzazioni in modo veloce e comprensibile.

Segue una rapida panoramica sui vari standard di rappresentazione menzionati e, in particolare, ci si focalizzerà su OpenIOC e sulle regole YARA. Queste ultime infatti sono ad oggi il metodo di rappresentazione più utilizzato.

#### TAXII, STIX, CybOX

Gli strumenti Structured Threat Information Expression (STIX), Trusted Automated Exchange of Indicator Information (TAXII), e Cyber Observable Expression (CybOX) sono uno sforzo guidato da una open community ed un insieme di specifiche che permettono la rappresentazione di informazioni di cyber threat intelligence in maniera standard, così da renderne più semplice lo scambio in maniera automatizzata.

#### OpenIOC

OpenIOC è un framework scritto in XML per la condivisione di threat intelligence e rappresentazione di IoC contenente un insieme base di più di 500 indicatori forniti da MANDIANT e stilati attraverso anni di esperienza in risposta ad alcuni dei più grandi e sofisticati attacchi osservati nella storia. Tale insieme di base può essere esteso per includere indicatori addizionali da diverse fonti, inoltre gli utenti e le organizzazioni che utilizzano OpenIOC possono aggiungere i propri set di indicatori.

In OpenIOC gli indicatori sono rappresentati tramite una espressione booleana che utilizza i classici operatori AND, OR e NOT, così come è possibile vedere nel seguente esempio (figura 2.1.1):

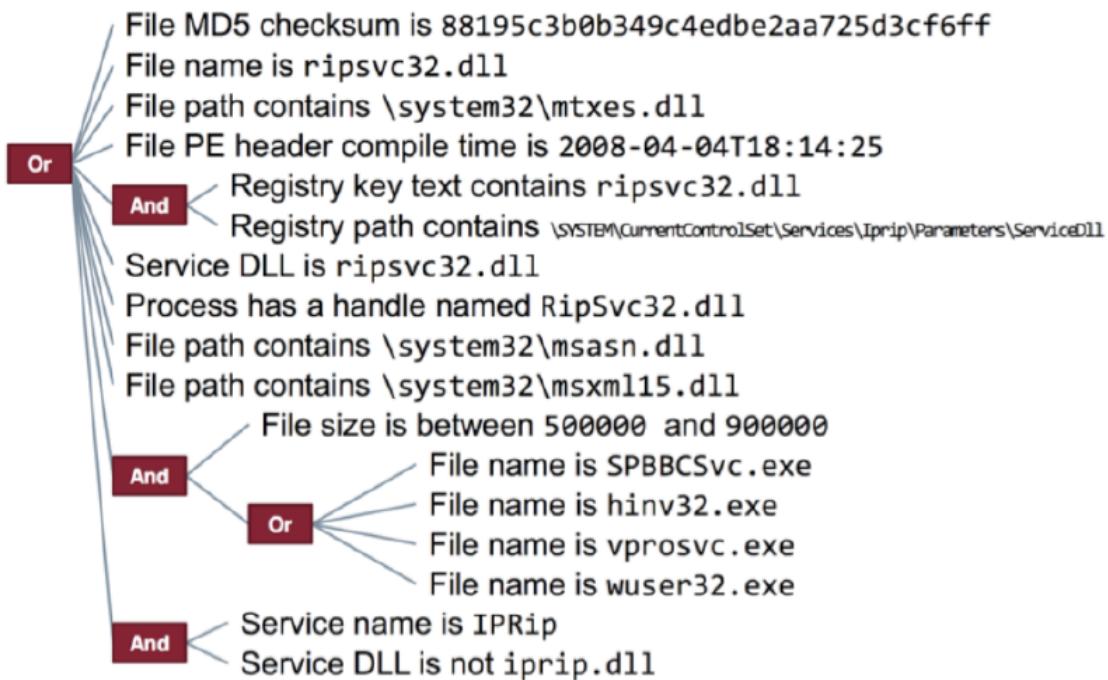


Figura 2.1.1: Esempio OpenIOC

## YARA

YARA è definito come “Swiss army knife for malware detection”. Esso è infatti un tool utilizzato per identificare e classificare i malware sulla base di determinate caratteristiche comuni a una stessa famiglia, cioè permette di eseguire una “signature-based detection”, in modo simile a come opera un qualsiasi antivirus. Infatti, moltissimi produttori di antivirus, come ad esempio VirusTotal, Kaspersky Lab, ed AlienVault, lo utilizzano come motore di analisi dei file. Inoltre è multiplattforma, compatibile con Windows, Linux e MacOS X, e può essere utilizzato sia da linea di comando che da script python grazie all'estensione yara-python. In particolare, con YARA è possibile creare descrizioni di famiglie di malware basate su pattern testuali o binari. Ad uno stesso file malevolo possono essere associati diversi pattern, poiché questi possono basarsi sull'identificazione di diversi aspetti del malware. I pattern possono essere descritti tramite regole YARA, le quali consistono essenzialmente in un insieme di stringhe ed un'espressione booleana che ne determina la logica.

Una regola YARA avrà quindi la seguente struttura (figura 2.1.2):

```
rule nome_regola
{
    meta:
        author = "Pippozzo"
        description = "sono un esempio di regola YARA"

    strings:
        $stringal = "malicious string"
        $stringa2 = "http://www.suspicious_url.org"

        //questo valore esadecimale indica la signature MZ di un file PE
        $stringa3_esadecimale = {4D 5A}

    condition:
        ($stringal or $stringa2) and $stringa3_esadecimale
}
```

Figura 2.1.2: Esempio regola YARA

Quindi una regola YARA è identificata da un nome, ed è generalmente composta da 3 sezioni:

- meta: questa sezione non è obbligatoria ma, se presente, riporta informazioni utili alla descrizione della regola.
- strings: questa è la sezione dove vengono definite le stringhe che faranno parte della regola. Esse saranno identificate da un identificatore che inizia con il carattere “\$” e che sarà utilizzato per farvi riferimento. Ci sono 3 tipi di stringhe in YARA:
  1. Stringhe testuali, che sono racchiuse tra doppi apici e sono di default codificate in ASCII e case-sensitive. Possono essere seguite da diverse keywords, alcune delle quali sono *wide* e *nocase*, che le rendono rispettivamente codificate in Unicode e case insensitive.
  2. Espressioni regolari, che sono definite allo stesso modo delle stringhe testuali ma racchiuse tra backslashes piuttosto che da doppi apici. Permettono di identificare un insieme di stringhe in un colpo solo tramite un insieme di metacaratteri.
  3. Stringhe esadecimali, che sono racchiuse tra parentesi graffe e sono utilizzate per definire sequenze grezze di bytes.

- condition: questa è la sezione dove risiede la logica della regola. Essa contiene un'espressione booleana funzione degli identificatori dichiarati nella sezione precedente e può contenere i tipici operatori booleani, relazionali e aritmetici. Possono essere anche utilizzati diversi operatori specifici, come ad esempio l'operatore *at* che consente di cercare una certa stringa ad un determinato offset nel file o a un certo indirizzo virtuale nello spazio di memoria di un processo. Inoltre, possono essere utilizzati identificatori che non siano delle stringhe definite nella sezione precedente, ma delle variabili speciali come ad esempio la variabile *filesize* che contiene la dimensione del file da analizzare espressa in bytes. Se l'espressione ha valore *true*, allora il file o processo soddisfa la regola, ossia si ha un match con tale regola.

È possibile inoltre estendere le funzionalità di YARA utilizzando dei moduli. Questi hanno la stessa utilità delle librerie per i linguaggi di programmazione in quanto permettono di richiamare nella sezione condition attributi o funzioni non definite dall'autore della regola ma messe a disposizione da tale modulo. La sintassi delle regole YARA è molto potente e si rimanda alla documentazione ufficiale per maggiori dettagli.

## 2.2 Threat intelligence

In ambito politico con il termine intelligence si intende la raccolta, il mantenimento, l'analisi e la diffusione di notizie e dati dalla cui elaborazione vengono ricavate informazioni utili alla tutela della sicurezza nazionale di uno Stato e alla prevenzione di attività destabilizzanti di qualsiasi natura. Vengono raccolte informazioni sugli avversari e analizzate per capire quali minacce aspettarsi da essi. Allo stesso modo la cyber threat intelligence prevede la raccolta di informazioni sulle minacce informatiche tramite sistemi di monitoraggio, queste vengono poi classificate in base a pertinenza e urgenza, e infine vengono analizzate in modo da poter prevenire incidenti futuri oppure identificare più facilmente quelli attuali.

Abbiamo 4 categorie di threat intelligence (figura 2.2.1):

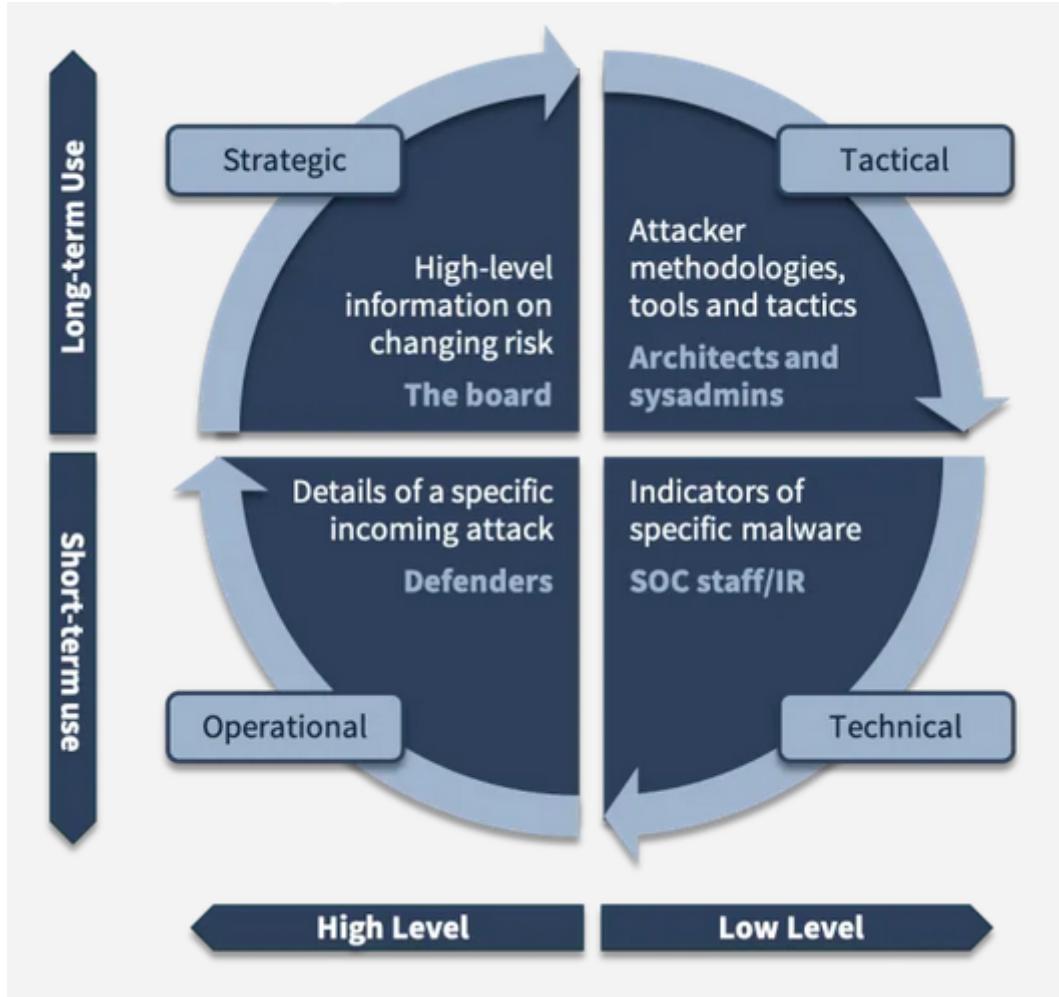


Figura 2.2.1: Tipologie di threat intelligence

- Strategic threat intelligence: riguarda le informazioni che possono aiutare le organizzazioni a capire il tipo di minacce contro le quali devono difendersi, le motivazioni e le capacità degli avversari e il potenziale impatto di tali attacchi, qualora andassero a segno. Tali informazioni permettono ai security teams la pianificazione delle risorse di cui avranno bisogno per proteggersi e mitigare le minacce correnti e future.
- Tactical threat intelligence: riguarda le informazioni relative alle metodologie e agli strumenti utilizzati dagli attaccanti. Tali informazioni sono utilizzate dalle organizzazioni per essere sempre preparate agli attacchi più recenti.
- Technical threat intelligence: riguarda le informazioni scambiate in maniera tecnica, come ad esempio tramite una lista di indirizzi IP blacklistati che possono essere importati in un firewall.

Tali informazioni hanno un periodo di vita breve in quanto gli attaccanti possono facilmente cambiare gli indirizzi IP, le checksum o altri parametri simili.

- Operational threat intelligence: riguarda le informazioni relative a uno specifico attacco imminente contro la propria organizzazione. Tali informazioni sono critiche affinché si possa reagire immediatamente ad un attacco.

La threat intelligence è utilissima in quanto gli attacchi informatici crescono drasticamente con il passare del tempo, ma molti di questi sono varianti di attacchi precedenti. Infatti vengono prodotti circa 300.000 malware al giorno ma di questi si può stimare che poche decine di migliaia siano istanze nuove mentre gli altri si rivelano essere copie che presentano leggere modifiche. Per studiare malware nuovi potrebbe quindi essere conveniente partire da informazioni di malware già noti, pertanto vengono raccolti, analizzati e condivisi sia indicatori di compromissione che indicatori di attacco. Mentre gli IoC hanno un approccio reattivo, in quanto rilevano le evidenze lasciate dal malware, gli indicatori di attacco o Indicators of Attack (IoA) hanno un approccio proattivo, in quanto si cercano i primi segni di un attacco attualmente in corso.

Vengono quindi collezionati IoC ed IoA provenienti da più sorgenti eterogenee sia di livello nazionale che internazionale, analizzati e correlati tra loro, condivisi con altre piattaforme di threat intelligence. La condivisione delle informazioni con altre strutture simili è una strategia vincente in quanto permette una raccolta di molte più informazioni e in maniera più veloce, e consente alle organizzazioni che vi partecipano di capire quali minacce aspettarsi e in che modo prevenirle o gestirle al meglio. Nella condivisione di informazioni vi può essere un'organizzazione centrale che funge da produttore di informazioni, detta Source, mentre le altre Subscribers le ricevono (modello Source-Subscriber), oppure tutte le organizzazioni giocano un ruolo paritetico secondo il modello peer-to-peer, oppure ancora tra le varie organizzazioni una lavora da hub e centralizza e coordina lo scambio di queste informazioni (modello Hub and Spoke).

Per la condivisione viene utilizzato il protocollo TLP (Traffic Light Protocol). Questo si occupa di garantire che le informazioni vengano diffuse in maniera controllata e non arrivino in mani sbagliate; in particolare queste informazioni sono classificate in più fasce ed associate a colori diversi: alcune pos-

sono essere trasmesse liberamente (quelle catalogate come bianche), altre possono essere condivise solo tra organizzazioni che si occupano di cyber security (verdi); poi, in maniera più restrittiva, informazioni più critiche (color ambra) possono essere condivise all'interno della stessa organizzazione, se non addirittura essere gestite unicamente da una minoranza della stessa organizzazione (rosse).

# Capitolo 3

## Basic Static Analysis

La basic static malware analysis è, come il nome stesso suggerisce, il processo di analisi di file sospetti attraverso l'utilizzo di tecniche che possono essere considerate statiche. Questo vuol dire che durante questa fase di analisi del file, quest'ultimo non sarà in nessun modo eseguito. È il primo step nel processo della malware analysis. La static analysis ha come obiettivo principale quello di estrarre quante più informazioni utili possibili dal file sospetto, il che vuol dire che l'analisi statica ci darà un'idea generale ma non rivelerà l'intera funzionalità del malware e di ciò che questo malware può fare. Questa fase dell'analisi, anche se non esaustiva, è tuttavia essenziale perché ci permette di prendere una decisione consapevole riguardo a dove focalizzare i nostri successivi sforzi di analisi del file binario sospetto, e quindi in definitiva ci permette di analizzare quest'ultimo in maniera più efficiente. L'analisi statica prevede i seguenti passi:

1. Determinare il tipo di file: questo ci aiuta a capire qual è il SO target e qual è l'architettura del processore (32-bit, 64-bit) per il quale è stato forgiato ad arte il file malevolo.
2. Generare un'impronta del malware: cioè generare un hash del malware. L'hash ci permette di identificare il malware in maniera univoca e inoltre ci permette di verificare se un file ha subito qualche cambiamento.
3. Scansionare il file binario sospetto con motori antivirus: in questa fase ad esempio è possibile utilizzare l'hash del file sospetto precedentemente generato per verificare in maniera immediata

se in rete o nel database di un servizio di scansione multi-antivirus (es. VirusTotal) quel file è stato già analizzato da qualcun altro. Se questo è il caso allora è possibile riutilizzare le informazioni già note evitando di dover scansionare nuovamente lo stesso file su diversi motori antivirus. Si tenga presente, però, che il fatto che il file non venga targato come malevolo da uno, alcuni o tutti gli antivirus che si sono utilizzati per la scansione non implica che il file sia non malevolo. Infatti, un malware può essere targato da un antivirus come non malevolo per il semplice fatto che non è presente nel suo database. Ora visto che diversi programmi antivirus utilizzano differenti firme ed euristiche, è chiaro che ha enormemente senso scansionare il malware con più programmi antivirus.

4. Identificare eventuali tecniche di packing e offuscamento: nel caso in cui queste tecniche siano state utilizzate, allora si necessita di tecniche di unpacking e deobfuscating per poter ottenere informazioni addizionali sul file malevolo.
5. Estrazione di stringhe dal file: ciò è utile per provare a trovare stringhe di testo che possano fornire informazioni utili sulle operazioni del malware.
6. Estrazione di informazioni dalla struttura del file PE: questo ci permette di ottenere molte informazioni sulle funzionalità del malware, grazie alle quali saremo capaci di avere una prima idea su quello che il malware può fare.

### 3.1 Determinare il tipo di file

Determinare qual è il tipo di file con il quale stiamo avendo a che fare ci aiuta a capire per quale sistema operativo e quale architettura (32-bit, 64-bit) quel file è stato pensato. Ad esempio, nel caso di un file eseguibile Windows il formato di file viene definito Portable Executable (PE), per cui quando ci imbattiamo in un file di questo tipo possiamo immediatamente dedurre che quel file è stato pensato per essere eseguito su un sistema operativo Windows.

Un errore da non commettere, però, è pensare di poter dedurre il tipo di file guardando esclusivamente all'estensione del file (nel caso di un eseguibile Windows: .exe, .dll, ecc...), in quanto un attaccante

potrebbe modificare l'estensione e realizzare la cosiddetta “double extension”, per indurre gli utenti a eseguire il file. Uno dei problemi relativi all'identificazione della double extension è che l'utente medio che usa un PC con SO Windows tipicamente non si accorge della doppia estensione a causa del fatto che in Windows di default l'estensione è nascosta per i tipi di file noti.

Per poter identificare accuratamente il tipo di file abbiamo bisogno invece di analizzare quella che prende il nome di file signature, ossia la firma del file. La file signature è una sequenza di bytes e si trova nell'intestazione del file. Per i file eseguibili Windows la file signature è rappresentata dai valori esadecimali *4D 5A* nei primi 2 bytes del file, che in codifica ASCII rappresentano i caratteri *MZ*. I file PE hanno tipicamente anche una stringa al loro interno con scritto: *“This program cannot be run in DOS mode”*. Tale stringa non è però sempre presente, perché ad esempio il file potrebbe essere stato offuscato oppure sottoposto a packing.

Esistono vari meccanismi per identificare la file signature di un file.

In ambiente Windows ad esempio esistono diversi tool che ci permettono di identificare il tipo di un file:

- HxD Hex Editor, che è un tool che ci permette di ispezionare tutti i byte di un file, inclusi ovviamente i byte relativi all'intestazione PE dove risiede la file signature.
- Exeinfo PE, che in realtà oltre al tipo del file sotto studio, recupera anche altre informazioni relative all'intestazione del file PE, come ad esempio l'entrypoint, la dimensione del file e il sottosistema.
- PE Studio, che oltre ad identificare il tipo di un file e ad offrire le funzionalità di base che la maggior parte dei tool per l'analisi statica di file PE mettono a disposizione, tenta anche di determinare se un file è dannoso sulla base di determinati indicatori. Quest'ultima caratteristica descrive appieno la natura di questo tool, il cui principale obiettivo è proprio quello di individuare artefatti sospetti all'interno di file eseguibili, al fine di facilitare e accelerare la valutazione iniziale del malware.

- CFF Explorer, che permette di identificare anch'esso il tipo di file, ma offre anche altre funzionalità come l'ispezione di file eseguibili (sia a 32 che a 64 bit), lo studio della struttura interna del file PE, modificare campi ed estrarre risorse.

## 3.2 Generare un'impronta del malware

Si definisce malware hashing il processo di generazione di valori hash crittografici che sono basati sul contenuto del file binario sospetto. Gli algoritmi di hashing più noti per fare l'identificazione dei malware sono Message-Digest Algorithm 5 (MD5) e Secure Hash Algorithm (SHA-1 e SHA-256). La generazione degli hash crittografici è essenziale, in quanto permette di avere un'impronta univoca per i malware, cosa che non sarebbe possibile se ci affidassimo per l'identificazione dei file al loro nome. I vantaggi dall'utilizzo degli hash crittografici sono molteplici:

- Il valore degli hash crittografici dipende dal contenuto dei file, per cui, se il contenuto resta invariato, possiamo identificare un file malevolo anche se viene cambiato il suo nome.
- Alcune tipologie di malware per loro propria natura possono, durante la loro esecuzione, replicarsi e copiarsi in differenti locazioni. Avere un hash crittografico del file malevolo ci permette di individuare eventuali copie e repliche dello stesso malware e capire se l'analisi può essere fatta su un solo file.
- Gli hash crittografici sono spesso usati come indicatori da condividere per permettere anche ad altri analisti di identificare il binario malevolo.
- Gli hash crittografici sono utilizzati per identificare i malware sui siti di analisi dei malware (es. VirusTotal, Hybrid Analysis...).
- Gli hash crittografici sono utilizzati per controllare online se quel file sospetto è stato già analizzato e individuato in passato da altri analisti o ricercatori, e questo ci permette eventualmente di riutilizzare informazioni già note che sono utili ai fini della nostra analisi.

In Windows esistono diversi tool per generare gli hash dei file, tra i quali ad esempio HashMyFiles, che permette di generare valori di hash crittografici attraverso diversi algoritmi per uno o anche più file contemporaneamente.

Tuttavia, le funzioni di hash crittografiche presentano anche degli svantaggi. In particolare, una minima variazione nel contenuto del file a cui esse sono applicate può portare ad un valore di hash completamente differente.

Un possibile modo per ovviare a questo problema è quello di utilizzare la tecnica del fuzzy hashing. Il fuzzy hashing è una tecnica che consiste nel comparare file sospetti tra loro in base alla loro similarità. Ciò è molto utile perché, a differenza del risultato dell'applicazione di una funzione di hash crittografico, il risultato ottenuto tramite il fuzzy hashing cambia in maniera direttamente proporzionale alle differenze tra i files. Questo permette di classificare un file sospetto come appartenente ad una famiglia specifica di malware.

Un tool che è possibile utilizzare per il calcolo del fuzzy hash di uno o più files è ssdeep. Il comando utilizzato per calcolare tale fuzzy hash è:

```
ssdeep [FILES]
```

La vera utilità del tool, però, la si può constatare con l'utilizzo della modalità “pretty matching”, che permette di determinare la percentuale di similarità tra due files. Per farlo, si utilizza l'opzione *-p*:

```
ssdeep -p [FILES]
```

### 3.3 Scansione del file binario sospetto con motori antivirus

Effettuare la scansione del file binario sospetto con diversi motori antivirus permette di determinare se effettivamente quel file può essere o meno considerato come malevolo.

VirusTotal è un ben noto servizio di scansione malware basato su interfaccia web. Esso permette di fare l'upload del file, dopodiché restituisce dei risultati ottenuti dalla scansione di molteplici scanner di antivirus. Tramite questo strumento è anche possibile fare delle ricerche direttamente nel database usando come parametri di ingresso per la ricerca un hash, un URL, un dominio o un indirizzo IP.

Molti strumenti di analisi di malware presentano un'integrazione con VirusTotal, come ad esempio PE studio, il quale automaticamente mostra i risultati ottenuti dalla query sul database di VirusTotal. Per effettuare la query viene utilizzato il valore dell'hash crittografico associato al file binario sotto studio.

Ci sono però alcuni problemi relativi alla scansione di file sospetti tramite scansioni antivirus. In generale, gli antivirus si basano su delle signatures e delle euristiche per definire se un file è o meno malevolo. Il punto debole delle signatures è che se la signature relativa al file sotto studio non è presente nel database dell'antivirus, il file viene reputato come benigno anche se potenzialmente potrebbe non esserlo. Per sopperire a questa mancanza vengono utilizzate delle euristiche in modo da permettere all'antivirus di rilevare anche malware ad esso non noti, però il loro punto debole è invece quello di portare alla segnalazione di falsi positivi. Inoltre, gli attaccanti utilizzano le tecniche di packing e di offuscamento per cui i motori degli antivirus potrebbero non riuscire ad identificare come malevolo il file sospetto.

Per quanto riguarda invece gli antivirus online, le problematiche sono principalmente le seguenti:

- Quando un file binario viene sottoposto ad analisi online i risultati della scansione sono memorizzati in un database e reperibili da chiunque in un secondo momento, questo vuol dire che un attaccante può venire a conoscenza del fatto che un proprio malware è stato già individuato, per cui poi potrà rendersi conto prima di dover ad esempio apportare dei cambiamenti al codice malevolo per evitare che quest'ultimo possa essere individuato.
- Bisogna tenere presente che quando viene caricato un file online per sotoporlo a scansione, il contenuto di questo file non è più privato ma sarà pubblico e potrà essere condiviso con terze parti, e questo ovviamente può essere un problema soprattutto se parliamo di file aziendali o file personali.

Per questi motivi, quando possibile, è sempre meglio utilizzare per prima cosa l'hash dei file sospetti invece che il file stesso, per capire istantaneamente se esso è stato già segnalato come malevolo.

### 3.4 Identificare eventuali tecniche di packing e offuscamento

Gli attaccanti o gli autori dei malware tipicamente utilizzano packer e cryptor per offuscare il contenuto dei propri gioielli così da evitare che il malware possa essere individuato da prodotti di sicurezza come ad esempio gli antivirus, e per fare in modo da rendere complicata l’analisi del suddetto malware agli analisti di malware, ai ricercatori e agli ingegneri che utilizzano tecniche di reverse engineering.

Un packer è un tool che può essere utilizzato anche in maniera legittima per comprimere il contenuto di un file. Nel caso di un attaccante ovviamente il file è tipicamente un binario malevolo per il quale si vuole ostacolare l’analisi.

Un programma packed, prima di essere eseguito, viene sottoposto ad unpacking grazie ad una routine. Esso infatti si presenta come un wrapper all’interno del quale sono presenti sia i dati relativi al file originario, sia la routine che verrà eseguita al momento dell’apertura del file e che farà l’unpacking in memoria. Se un programma è packed, la maggior parte dei tool che vengono usati per fare analisi statica non saranno in grado di scovare tutte le stringhe e le altre informazioni che sono state compresse.

Un cryptor lavora similmente ad un packer, con l’unica differenza che usa la crittografia per offuscare il contenuto del file malevolo. Ovviamente prima dell’esecuzione del binario viene invocata una routine che permette di fare la decriptazione del contenuto del file originario.

Un esempio di packer molto noto è UPX. Esso è un tool che fa compressione per offuscare il contenuto di un file, per cui la dimensione del file packed risulterà inferiore rispetto a quella del file non packed.

Per fare il packing di un file si utilizza il seguente comando:

```
upx -o malware_file_packed.exe malware_file_not_packed.exe
```

Per fare l’unpacking di un file packed con il packer UPX, invece, possiamo usare il comando:

```
upx -d malware_file_packed.exe
```

In Windows esistono 2 tool gratuiti che si chiamano Exeinfo PE e PEiD che ci permettono di capire se il contenuto di un file è stato offuscato e di base sono anche in grado di dirci quale compilatore è stato utilizzato per fare il build dell’applicazione. Nel caso specifico di file packed, Exeinfo PE

e PEiD possono dirci in alcuni casi anche il packer con il quale il file è stato sottoposto a packing, possono dirci l'autore del packer e sono anche in grado di consigliarci dei tool per poter fare l'unpacking del file. PEiD è però non più supportato e nelle versioni meno aggiornate presenta anche delle vulnerabilità ad attacchi di tipo buffer overflow.

### 3.5 Estrarre stringhe dal file sospetto

La string analysis è il processo di estrazione di stringhe, cioè di sequenze di caratteri, dal file malevolo al fine di ottenere quante più informazioni possibili per poter così avere un'idea sulla funzionalità del malware. Le stringhe di solito sono in formato ASCII (1 byte per carattere) e Unicode (2 bytes per carattere). Entrambi i formati memorizzano i caratteri in sequenze che terminano con un terminatore NULL che indica che la stringa è terminata.

Tipicamente potremmo essere interessati solo a stringhe di uno specifico formato, per cui alcuni tool che si occupano di estrarre le stringhe da un file permettono di applicare dei filtri per il formato di interesse per cui dobbiamo specificare, in base al nostro interesse, che tipo di stringhe e quale formato stiamo estraendo.

Sono diverse le stringhe verso le quali focalizziamo in maniera particolare la nostra attenzione:

- I nomi di file, ad esempio il nome di un file che viene creato dal binario malevolo.
- Gli URL, ossia i domini ai quali il malware si connette, per esempio l'URL di un centro di command and control.
- Gli indirizzi IP verso i quali e con i quali comunica il malware.
- Le chiavi di registro che possono essere usate per impostare la persistenza o per fare altre azioni malevoli agendo sulle configurazioni del SO.
- Eventuali comandi malevoli.

Pur non fornendo chiaramente un quadro completo del comportamento del malware, tuttavia la string analysis porta numerose informazioni ad un analista, motivo per il quale tipicamente un attaccante

include delle fake strings all'interno del file malevolo. Queste fake strings tipicamente sono generate in maniera randomica, e se quello è il caso allora prendono anche il nome di garbage strings.

Le fake strings sono incluse nel file malevolo dall'autore del malware per fuorviare un'eventuale analisi da parte di analisti o ricercatori. In particolare, possono essere utilizzate delle fake strings perfettamente sensate in modo da portare l'analista fuori strada, oppure delle fake strings senza alcun significato, che abbiamo chiamato garbage strings, utilizzate ad esempio per modificare il contenuto del file senza tuttavia cambiare le funzionalità del programma in esso contenuto, il che è molto utile per far cambiare l'hash crittografico associato al file, che non sarà più riconosciuto come malevolo dagli antivirus che si basano solo su signatures già note.

Si noti che i programmi legittimi tipicamente includono sempre molte stringhe, il che vuol dire che se ci imbattiamo in un file che include poche stringhe allora ci sono delle buone probabilità che quel file sia stato offuscato o sottoposto a packing, e di conseguenza ci sono delle buone probabilità che quel file sia malevolo.

Esistono diverse soluzioni per estrarre stringhe da un file sospetto. Alcune di queste sono:

- L'utility strings da linea di comando, presente su Linux ma della quale ne esiste il porting anche su Windows, che differisce leggermente nell'utilizzo. In questo elaborato ci concentriamo sull'utility in Windows.

Di default il comando strings estrae stringhe ASCII ed Unicode che sono lunghe almeno 4 caratteri. Esistono tuttavia diverse opzioni per il comando strings, ad esempio le opzioni `-a` e `-u` permettono rispettivamente di estrarre solo stringhe ASCII o solo Unicode, mentre l'opzione `-n` permette di specificare la lunghezza minima delle stringhe da estrarre. La sintassi del comando è la seguente:

```
strings [options] nome_file
```

Bisogna però prestare attenzione al fatto che l'utility Strings possa restituire anche delle stringhe che non sono delle stringhe reali, ma che sono solo il risultato dell'interpretazione del programma strings ad esempio di un indirizzo di memoria, di un'istruzione per la CPU o dati usati dall'eseguibile.

- Il tool PE studio, che permette di visualizzare sia stringhe in formato ASCII che stringhe in formato Unicode. Tra le tante caratteristiche di questo tool, abbiamo la possibilità di filtrare sulla colonna “blacklisted” in modo da prestare maggior attenzione a quelle che sono le stringhe che sono tipicamente associate ad un comportamento potenzialmente malevolo. Il filtro *hint* invece dà suggerimenti riguardo a quello che può essere particolarmente utile. Il filtro *group*, come suggerito dal nome, permette di filtrare in base ai gruppi, dove i gruppi sono tutti in riferimento alle funzionalità e alle librerie che sono state importate. I filtri *mitre-technique* e *mitre-tactic* permettono invece di filtrare in base a tecniche e tattiche che sono marcate come associate a comportamenti di attaccanti. Una tecnica o una tattica è marcata in PE studio come associata ad attacchi sfruttando la base di conoscenza ATT&CK (Adversarial Tactics, Techniques & Common Knowledge). ATT&CK è messa a disposizione dall’organizzazione MITRE Corporation, ed è accessibile a livello globale.
- Il tool PEiD, che ci dà una panoramica base di queste stringhe, nello specifico se si clicca da interfaccia grafica sulla freccia accanto a First Bytes e poi ancora su Strings, allora è possibile vedere le stringhe e il loro relativo offset.

### 3.6 Estrarre informazioni dalla struttura del file PE

Il formato Portable Executable (PE) è il formato usato dagli eseguibili in Windows (.exe, .dll, .sys, ecc...). Un file in tale formato è costituito da varie parti che contengono tutte le informazioni richieste dal SO affinché esso possa essere caricato in memoria. La descrizione di tale struttura è riportata nell’intestazione (header), che è chiamata intestazione PE, che viene posta all’inizio del file al momento della compilazione. Quando il binario viene eseguito, allora il loader del SO legge le informazioni presenti nell’intestazione PE del file e carica il contenuto del file binario in memoria.

Un file PE non è necessariamente un file malevolo, in quanto qualsiasi eseguibile in Windows ha questo formato. Nella nostra trattazione, però, ci concentreremo sui malware che hanno come obiettivo il sistema Windows e, in particolare, quelli sotto forma di file PE.

Fare l'analisi statica dell'intestazione PE risulta molto utile, visto che essa contiene tutte le informazioni che il SO richiede per eseguire il file malevolo e quindi ci dà un'idea di come il malware interagisce con il SO. Sulla base di ciò si possono inferire le funzionalità del malware. Infatti, abbiamo che il malware può interagire con il file system, con il registro e con la rete, e per poterlo fare ha chiaramente bisogno di funzioni esposte dal SO. La maggior parte di queste funzioni in ambiente Windows sono esportate in file DLL (Dynamic-Link Library).

La struttura di un file PE è la seguente (figura 3.6.1):

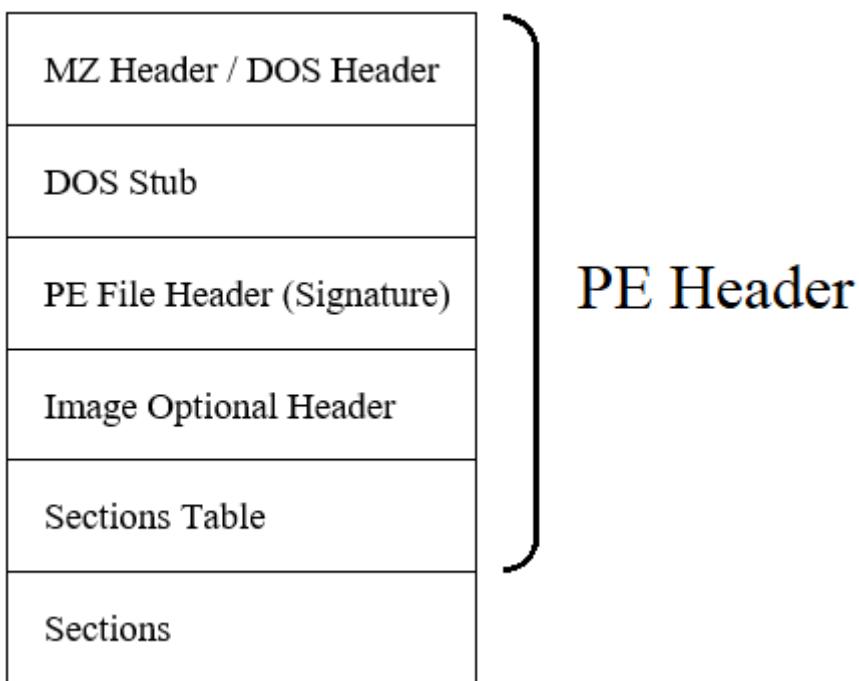


Figura 3.6.1: Struttura dei file Portable Executable

Quando parliamo di PE file, a prescindere dal fatto di stare parlando o meno di un malware, può risultare utile vedere il file come composto da un'intestazione (PE Header), che contiene metadati relativi all'eseguibile stesso, e da un insieme di sezioni, che contengono i dati e il codice dell'eseguibile. L'header PE è composto dalle seguenti parti:

- MZ Header/DOS Header definisce il file come un binario eseguibile, attraverso i primi 2 byte codificati in esadecimale come *4A 5D* e in ASCII come *MZ*.

- DOS Stub esiste fondamentalmente per motivi di compatibilità. Il suo scopo è quello di stampare il messaggio “*This program cannot be run in DOS mode*”.
- PE File Header (Signature) definisce l’eseguibile come un PE file. Tale signature è composta da 4 bytes, che in esadecimale sono *50 45 00 00*, dove i primi 2 bytes in ASCII significano proprio *PE*.
- Image Optional Header memorizza informazioni di interesse circa l’eseguibile, come il subsystem (Console, GUI, ecc...) e l’entry point, ma anche le data directories, contenenti tra le altre cose le informazioni relative alle funzioni importate ed esportate.
- La Section Table dà informazioni su come caricare l’eseguibile in memoria e dove sono localizzate le sezioni.

Le Sezioni possono essere sezioni eseguibili di codice o sezioni relative a dati utilizzati dall’eseguibile. Tipicamente analizzare le sezioni può essere interessante perché anche queste possono darci informazioni circa le funzionalità del file PE.

In generale le sezioni più comuni e di maggiore interesse sono le seguenti:

- *.code / .text* che contiene le istruzioni che sono eseguite dal processore. Se ci imbattiamo in questa sezione allora sappiamo che il file PE contiene codice eseguibile.
- *.data* che memorizza dati che possono essere sia letti che scritti. Ad esempio, in questa sezione troviamo le variabili globali.
- *.rdata* che contiene e memorizza dati che possono solo essere letti ma che non possono essere scritti o eseguiti.
- *.idata* che memorizza l’Import Table, ossia le informazioni sulle funzioni importate che sono anche le dipendenze del file eseguibile.
- *.edata* che contiene le informazioni sulle funzioni esportate
- *.rsc* che memorizza le risorse (stringhe, icone, dialogs, menu, file di configurazione...) che servono al binario o all’eseguibile.

- *.reloc* che contiene le informazioni per la rilocazione dei file di libreria
- *.pdata* che è presente solo negli eseguibili per architetture di processore a 64 bit. Questa sezione memorizza informazioni per la gestione delle eccezioni.

Non tutte le sezioni sono obbligatorie ovviamente, per esempio spesso la sezione *.idata* e la sezione *.edata* possono non essere presenti in quanto le loro informazioni possono essere inglobate nella sezione *.rdata*.

Le informazioni contenute nel file PE che sono per noi di maggiore interesse e alle quali ovviamente poniamo maggiore attenzione sono:

- Il timbro del compilatore, che ci dice quando e dove il malware è stato compilato. Tipicamente questa informazione non la si trova, in quanto gli autori dei malware sanno che gli analisti andranno alla sua ricerca, e quindi quello che tipicamente queste persone fanno è inserire delle date anteriori nel tempo. Tuttavia, quando quest'informazione è presente e non sembra essere del tutto inaccurata, allora può essere un grande indicatore su quando è stato compilato l'ultima volta il malware e quanto è vecchio il malware stesso.
- Il subsystem, che ci dà informazioni su quale sottosistema viene usato dal file malevolo. Tipicamente i malware usano un sottosistema GUI.
- La struttura delle sezioni, che ci fa capire se l'eseguibile è stato sottoposto a packing, ad esempio attraverso il loro nome. Inoltre possiamo fare un controllo sui permessi per le varie sezioni al fine di capire se questi permessi sono inconsistenti o comunque non sono quelli che ci aspetteremmo per le singole sezioni.
- Le librerie e gli import, che ci danno informazioni su quali librerie sono richieste dal malware e quali funzioni il malware importa da altri file (tipicamente file DLL), e questo ci dà informazioni circa la funzionalità del malware.

In base a tali informazioni, possono essere utilizzate delle tecniche di identificazione dei malware che possono dare buoni risultati anche nel caso in cui l'hash crittografico dell'intero file non risulti utile a causa di piccoli cambiamenti apportati al malware stesso.

Una prima tecnica, che viene per lo più utilizzata per identificare malware che potenzialmente sono stati realizzati dallo stesso gruppo di attaccanti, è l'import hash (anche noto come imphash). In alcuni casi, però, potremmo imbatterci in file che hanno lo stesso imphash, ma che in realtà sono stati generati da gruppi differenti. Ciò non è tanto raro, in quanto è abbastanza frequente che un malware possa essere generato con un builder kit comune che viene condiviso tra vari gruppi di hacker. L'imphash consiste nel generare valori di hash semplicemente basandosi sui nomi delle funzioni e delle librerie importate e sul loro ordine all'interno dell'eseguibile.

Un modo per calcolare in maniera automatica l'imphash di un file in ambiente Windows è quello di caricare il file in PE studio.

Una seconda tecnica per identificare sample che sono in qualche modo collegati tra loro è quella di usare l'hash delle sezioni. Ancora una volta PE studio fa il lavoro per noi presentando, sotto la categoria *sections*, l'MD5 delle singole sezioni che ha calcolato in maniera automatica. In questo modo, se viene cambiata una sola delle sezioni e non le altre, l'hash per la maggior parte delle sezioni risulterà identico.

Quando parliamo di file PE, il tool più efficiente per l'analisi statica in ambiente Windows è certamente PE studio. In PE studio abbiamo un albero ben strutturato e ben progettato che contiene tutte le categorie e informazioni di rilievo che ci aiuteranno a capire il malware e le sue funzionalità.

Le categorie di maggiore interesse sono:

- File-header: qui troviamo per esempio il timbro del compilatore. PE studio converte automaticamente per noi il timbro del compilatore, nel senso che ci dà direttamente il tempo di compilazione reale.
- Optional-header: qui troviamo per esempio il sottosistema utilizzato (es. GUI). Se il sottosistema è una GUI allora possiamo assumere che il malware utilizzerà la GUI per la maggior parte delle sue funzionalità.
- Sections: tra le informazioni che PE studio permette di visualizzare, le più rilevanti in questo contesto sono:

- Names, che è il nome della sezione. I nomi delle sezioni non sono usati dal sistema operativo, ma sono lì principalmente per una maggiore comprensione per l'essere umano. Questo permette ad un potenziale attaccante di utilizzare dei packer per rendere meno chiara l'analisi del file malevolo.

Quando viene usato un packer, questo tipicamente cambierà il nome delle sezioni con un nome abbreviato che può far riferimento al packer stesso. Per esempio, nel caso di UPX possiamo avere che le sezioni avranno come nome (UPX1, UPX2...), e questo chiaramente è un buon indicatore che il file è stato sottoposto a packing.
  - Entry-point, che è il Relative Virtual Address al quale inizia il codice eseguibile. Normalmente l'entry point dovrebbe essere nella sezione *.code / .text*.
  - Virtual-Size, che indica la dimensione della sezione quando caricata in memoria.
  - Raw-size, che indica la dimensione della sezione su disco. Normalmente Raw-Size e Virtual-Size dovrebbero assumere valori soltanto leggermente diversi a causa dell'allineamento della sezione. Se però la differenza tra i valori è rilevante allora questa è una grande indicazione che il binario è stato sottoposto a packing.
  - Writable, executable, ecc, cioè i permessi associati alle varie sezioni. I permessi per la sezione *.code / .text* dovrebbero essere quelli di esecuzione. La sezione *.rdata* dovrebbe essere leggibile e inizializzata, mentre la sezione *.data* dovrebbe essere leggibile e scrivibile, ma entrambe non dovrebbero essere eseguibili.
- Libraries: sono le librerie di funzioni offerte dal SO che vengono utilizzate. In ambiente Windows ci riferiamo alle DLL che sono usate dall'eseguibile. Da esse ovviamente l'eseguibile dipende, e questo ci dà una panoramica base della funzionalità del campione di malware. Queste librerie sono caricate in memoria quando il file malevolo viene eseguito.

PE studio ci permette di ordinare le librerie in base a quelle che sono blackliste. Le librerie blackliste non sono malevoli per loro natura, ma sono flaggate come tali perché tipicamente sono state riscontrate in file malevoli, o comunque perché contengono funzioni che permettono

a un file di fare operazioni che possono essere sfruttate per fare qualcosa di malevolo. Inoltre PE studio ci permette di avere un’idea sulla quantità di funzioni importate dalle singole DLL.

- Imports: sono le funzioni importate dalle singole DLL. Se non conosciamo cosa fa una di queste funzioni, allora PE studio ci permette con un semplice click di fare una ricerca nella MSDN (Microsoft Developer Network) library. Se il numero di funzioni importate è esiguo, allora questa è una forte indicazione che il binario è stato sottoposto a packing.

Alcune funzioni importate come *LoadLibrary( )*, *LdrLoadDLL( )*, *GetProcAddress( )* o *LdrGetProcAddress( )*, dovrebbero farci storcere il naso, in quanto queste funzioni permetteranno al file malevolo di caricare DLL a runtime invece che al momento del caricamento, il che vuol dire che le conclusioni derivanti dalla nostra analisi sulle librerie potrebbero essere del tutto inaccurate, visto che i tool di analisi statica delle librerie non saranno in grado di presentare in maniera esaustiva le librerie utilizzate dal malware. Queste tuttavia non sono le uniche funzioni dalle quali mettersi in guardia, in quanto bisogna anche comprendere ad esempio tutte quelle funzioni che permettono di agire sul file system, sul registro di sistema e con la rete.

- Exports: sono le funzioni esportate dall’eseguibile o dalla DLL, il che fa sì che queste funzioni possano essere utilizzate da altri programmi. Tipicamente in un file malevolo gli export sono funzionalità malevole messe a disposizione di altri programmi.

Si noti che le funzioni esportate non sono presenti nei *.exe*, in quanto questi non sono progettati per fornire funzionalità ad altri *.exe*, mentre sono più frequenti nelle DLL che invece per loro propria natura sono implementate per fornire funzionalità ad altri eseguibili.

- Strings: sono le stringhe contenute nell’eseguibile. PE studio ci permette di filtrare, come già detto, in base a quelle che sono considerate stringhe potenzialmente malevole.

Come accennato, la sezione delle risorse (*.rsc*) contiene file necessari all’eseguibile come icone, dialogs e file di configurazione.

È importante far rientrare nell’analisi statica anche l’analisi della sezione delle risorse, perché gli attaccanti possono inserire in questa sezione info sulla configurazione o memorizzare altri dati e file

malevoli, come fanno ad esempio i malware di tipo dropper. Questa sezione è utile anche perché può fornire informazioni sull'origine del malware.

I tool in ambiente Windows più noti per effettuare l'analisi della sezione delle risorse sono:

- Resource Hacker, che ci permette di visualizzare le varie risorse. Esse possono essere visualizzate nella editor view in forma grafica (solo quelle che lo permettono), sia nella binary view in forma binaria. Alcune tipologie di risorse possono essere:
  - Bitmap, che fornisce solitamente l'icona del file.
  - Manifest, che ci dà informazioni sulle versioni utilizzate, l'encoding e l'ID del sistema operativo supportato.
  - Dialog, che mostra le finestre di dialogo, e questo ci può confermare per esempio che il malware usa una GUI.
  - Binary, che ci indica che come risorsa c'è un file binario che può essere un eseguibile malevolo.
- PE studio può anch'esso essere usato per ispezionare le risorse che esistono per un particolare eseguibile. In PE studio nella categoria resources possiamo vedere le stesse tipologie di risorse visualizzate con Resource Hacker, dove per ognuna sono fornite varie informazioni, tra cui:
  - Type, che è il tipo della risorsa.
  - Name, che è un nome binario associato alla risorsa.
  - Signature, che dà informazioni riguardo alla risorsa.
  - Language, che ci dà un'idea su dove questo particolare pezzo del malware è stato compilato.

# **Capitolo 4**

## **Basic Dynamic Analysis**

### **4.1 Definizione e vantaggi della Basic Dynamic Analysis**

La basic dynamic analysis è il secondo step nel processo della malware analysis. Essa viene effettuata solo dopo che si è riusciti ad ottenere tutte le informazioni possibili con la basic static analysis. Essa infatti consiste nell'esecuzione del malware in un ambiente isolato e opportunamente controllato, in modo da monitorarne il comportamento, ragion per cui tale tipo di analisi è anche chiamata "behavioral".

Si tenga presente che tale tipo di analisi è molto più pericolosa di quella discussa nel capitolo precedente, in quanto per portarla a termine, oltre che ottenere di proposito il malware, bisogna eseguirlo e quindi infettare la propria macchina. Ciò è esattamente quello che normalmente un utente vorrebbe evitare ad ogni costo. Per tale motivo la cosa più importante da fare prima di partire con questo tipo di analisi è quella di assicurarsi di fare tale operazione in un ambiente isolato, come può essere una VM che non comunica né con il SO host, né verso Internet.

Tuttavia, la dynamic analysis è molto importante perché permette di ottenere molte più informazioni della static analysis, in quanto quando un malware viene eseguito, esso può interagire con il sistema in vari modi e fare varie attività, come ad esempio creare processi, leggere/creare/modificare/eliminare files, directories e chiavi di registro, oppure comunicare in rete. In base all'analisi di queste attività ci si può rendere conto di quale sia la tipologia di malware che ci si trova davanti e il suo scopo.

Le attività fondamentali da monitorare sono quindi quelle relative ai processi, al file system, al registro e alla rete. Ognuna di queste verrà analizzata in dettaglio nei paragrafi seguenti.

Come già detto in precedenza, ci focalizzeremo sui malware che hanno come target il sistema operativo Microsoft Windows, in quanto sono quelli più diffusi. Analizzeremo solo i file nel formato Portable Executable (PE) e gli esempi che prenderemo in considerazione saranno relativi agli executables (.exe), malgrado anche altri tipi di files utilizzino il formato PE, come ad esempio le Dynamic-Link Libraries (DLL), cioè librerie che esportano funzioni e che vengono caricate dinamicamente da altri eseguibili e possono essere condivise da più processi. Gli EXE, a differenza delle DLL, non esportano funzioni e i 2 tipi di file sono riconosciuti dal SO proprio grazie all'header PE.

Le DLL necessitano di un eseguibile host che le possa lanciare, quindi analizzare un malware che è sotto forma di DLL può essere più complicato. Per farlo si possono lanciare tramite l'eseguibile *rundll32.exe*, oppure modificare l'header PE per far sì che il sistema tratti la DLL come se fosse un EXE, avviando la funzione *DLLMain* che costituisce l'entry point della DLL nella quale è solitamente posto il codice malevolo.

Un file EXE, invece, può essere semplicemente eseguito tramite un double-click.

## 4.2 Tecniche di monitoring e tool

Come abbiamo già detto, le operazioni di un malware possono riguardare files, chiavi di registro e rete, ma tali operazioni vengono effettuate da dei processi la cui creazione è associata al malware stesso. Quindi, per costruirsi un quadro generale delle azioni svolte dal malware, è necessario combinare l'utilizzo di vari strumenti di monitoraggio.

In particolare, bisogna per prima cosa avere visibilità dei processi che sono in esecuzione nel sistema in tempo reale, in modo da accorgersi repentinamente di processi anomali dovuti all'esecuzione del malware. Bisogna inoltre monitorare ogni attività fatta da ciascun processo in termini di chiamate di sistema, modifiche al registro, interazione con il file system e con altri processi. Si deve, infine, correlare tali attività eseguite sulla macchina con quelle eseguite in rete.

In seguito verranno descritti alcuni dei più famosi tool utilizzati per la basic dynamic analysis, tutti necessari in quanto ciascuno di essi permette di avere una vista differente su tutte o parte delle informazioni necessarie a questa fase dell’analisi.

## Process Hacker

Process Hacker è uno dei tool più utilizzati per quanto riguarda lo studio dei processi. In particolare è uno strumento gratuito ed open source che permette di monitorare i processi attivi nel sistema in tempo reale. Esso è molto utile per fare attività di troubleshooting e debugging, ma è anche utile per identificare processi malevoli e per capire cosa stanno tentando di fare.

La schermata principale del programma mostra l’elenco dei processi ordinati secondo una struttura ad albero gerarchica, ciascuno dei quali associato con varie informazioni come ad esempio il PID, l’utilizzo delle risorse, l’username dell’utente sotto il quale esso gira e una descrizione. Possono inoltre essere aggiunte altre colonne a quelle di base, per avere più informazioni, ad esempio si possono vedere le invocazioni da command line dei processi, per avere un’idea di come sono stati creati.

Facendo un mouseover su uno di tali processi è possibile ottenere informazioni aggiuntive, inoltre con un right-click è possibile sosponderlo o killarlo, ma soprattutto è possibile accedere alle sue proprietà, dalle quali è possibile accedere a molte informazioni relative al processo, come ad esempio le stringhe caricate in memoria.

Oltre alla tab principale relativa ai processi, il programma presenta anche una tab relativa ai servizi e driver presenti attualmente nel sistema. Ci sono, inoltre, altre due tab relative all’utilizzo di disco e rete da parte dei vari processi.

Per quanto riguarda il suo utilizzo nella malware analysis, esso può essere avviato appena prima di avviare l’eseguibile malevolo, in modo tale da osservare quali sono i processi che vengono da esso generati e quindi per procedere con ulteriori analisi.

Si noti che i processi creati sono visualizzati in verde e quelli killati in rosso, e potrebbe accadere che non tutti i processi permangano in memoria a lungo in quanto sono solo processi ausiliari creati dal malware. In tali casi è importante riuscire a cogliere l’attimo per evitare di perdersi qualche attività.

## RegShot

RegShot è uno strumento gratuito ed open source per il monitoraggio del registro di sistema. In particolare questo tool permette di creare una istantanea (snapshot) del registro di sistema, in modo da poterla comparare con una seconda istantanea creata in seguito e visualizzare i cambiamenti.

È molto utilizzato per verificare quali modifiche sono state apportate al registro di sistema dopo l'installazione di nuovi software, ma può essere molto utile anche per osservare i cambiamenti apportati al registro da un software malevolo. È possibile inoltre fare controlli anche sulle cartelle e relative sottocartelle specificate.

Per utilizzarlo basta creare uno snapshot appena prima dell'esecuzione del malware e uno dopo la sua esecuzione, per poi compararli.

## Process Monitor

Process Monitor, detto anche Procmon, è un tool molto utilizzato per il monitoraggio dei processi in esecuzione nel sistema. Esso permette inoltre di monitorare tutte le operazioni effettuate da tali processi in termini di accessi al file system, così come anche al registro e alla rete. Per ogni evento rilevato sono fornite numerose informazioni, tra cui:

- Il timestamp di generazione dell'evento;
- Il processo che lo ha causato con il relativo PID;
- Il tipo di operazione effettuata;
- Il path sul quale è compiuta l'operazione con il relativo esito;
- Il dettaglio dell'operazione compiuta.

Inoltre, le informazioni che non riescono ad entrare nelle colonne possono essere visualizzate facendo un double-click sulla particolare attività. È possibile anche cliccare sul path relativo all'operazione svolta e aprire explorer in quel punto per esaminare eventuali files acceduti.

Procmon include un potente sistema non distruttivo di filtri che consente di lavorare al meglio con l'enorme mole di dati che il programma genera. Infatti, anche in un sistema pulito possono esserci decine di migliaia di eventi al minuto, per questo è fondamentale l'utilizzo dei filtri per poter visualizzare solo le attività desiderate.

Si noti che tali attività vengono loggate in memoria RAM, quindi si consiglia di farlo girare solo per il tempo necessario per evitare di saturare la memoria. Il logging inoltre avviene su tutti gli eventi anche se vengono applicati dei filtri, in quanto essi servono solo per filtrarne la visualizzazione.

Il filtraggio in Procmon è molto potente, consente di filtrare su determinati eseguibili, processi, ma anche sulle singole operazioni effettuate sia sul registro che sul filesystem filtrando sulle varie system calls (ad esempio *RegSetValue*, *CreateFile*, *WriteFile*, ecc...). Il filtraggio permette inoltre sia di filtrare solo su determinati eventi, sia di escludere alcuni eventi.

Per quanto riguarda il suo utilizzo nella malware analysis, si possono utilizzare:

- Le attività sui processi, per individuare eventuali altri processi creati dal malware.
- Le attività nel file system, per capire quali file va ad utilizzare o se ad esempio si sta replicando in altre locazioni.
- Le attività sul registro, per capire se un malware sta provando ad esempio a fare persistenza.
- Le attività sulla rete, per visualizzare le porte sulle quali il malware potrebbe essere in ascolto.

Utilizzare le attività svolte dai vari processi in combinazione con quanto osservato con Process Hacker e RegShot, oltre che a quanto scoperto con la basic static analysis, permette di ottenere un migliore filtraggio e quindi una più rapida individuazione delle attività sospette.

## Wireshark

Wireshark è il tool più utilizzato per quanto riguarda il monitoraggio della rete. Esso è un packet sniffer ampiamente utilizzato per la risoluzione di problemi di rete, per l'analisi e lo sviluppo di protocolli o di software di comunicazione e per la didattica.

Le funzionalità di Wireshark sono molto simili a quelle di tcpdump, ma con un'interfaccia grafica e maggiori funzionalità di ordinamento e filtraggio.

Il software trae vantaggio dall'utilizzo della cosiddetta modalità promiscua di una scheda di rete, la quale diventa in grado di "intercettare" tutto il traffico, compreso quello destinato verso altri dispositivi connessi alla rete locale.

Grazie a Wireshark è quindi possibile verificare in tempo reale che cosa avviene sulla rete locale e individuare eventuali credenziali di accesso trasmesse in chiaro, attività sospette e così via, al fine di attuare eventuali contromisure. Esso può però essere utilizzato anche con intenzioni meno buone, come ad esempio sniffing di password, reverse-engineering di protocolli di rete, rubare informazioni sensibili e mettersi in ascolto di conversazioni tenute da altri utenti della rete.

All'apertura di Wireshark è possibile scegliere l'interfaccia di rete da utilizzare per lo sniffing dei pacchetti e quindi è possibile farla partire.

Possono essere utilizzati vari filtri per eliminare le informazioni non necessarie, in quanto, come per Procmon, le informazioni restituite sono molte più di quelle realmente necessarie.

Nella finestra superiore vengono visualizzati i vari pacchetti con determinate informazioni, la cui visualizzazione può essere modificata a piacimento, mentre nella finestra inferiore è mostrato il dettaglio del pacchetto selezionato. È presente, inoltre, una ulteriore finestra che permette di visualizzare il contenuto del pacchetto selezionato in esadecimale.

Questo strumento può essere utilizzato per capire quali sono le operazioni di rete che un malware sta eseguendo, come ad esempio la risoluzione di determinati domini che possono essere dei Command and Control (C&C), oppure dei domini contattati per mandare le informazioni esfiltrate dalla macchina, o ancora dei domini dai quali scaricare ulteriori file o software malevoli.

## **INetSim**

Per monitorare le attività di rete sarebbe necessario che la macchina sulla quale è eseguito il malware fosse connessa ad Internet. Il problema è che, come è stato già detto, anche se il malware è eseguito su una VM, la connessione di tale VM ad Internet comporta comunque dei rischi.

Il malware infatti potrebbe diffondersi sulla rete infettando gli altri sistemi sulla stessa rete, potrebbe diventare parte di una botnet, oppure potrebbe contattare un eventuale C&C.

Durante la basic dynamic analysis è fondamentale riuscire a determinare il comportamento del malware senza permettergli di contattare il C&C, quindi per monitorare le sue attività di rete è necessario simulare i servizi offerti dalla rete Internet. Per fare ciò, una possibile soluzione è quella di emulare i servizi offerti dalla rete, e un tool molto utilizzato per fare ciò è INetSim.

INetSim è una suite software Linux-based gratuita che permette di simulare i servizi Internet standard, come ad esempio DNS, HTTP, FTP, ecc, ma anche dei dummy services che gestiscono connessioni verso porte non standard. Esso è altamente configurabile e può essere utilizzato per fornire dei dummy files in base all'estensione del file richiesto al posto di fornire un errore, così da potersi rendere conto di cosa il malware tenti di fare una volta ricevuto tale file dal C&C. Si potrebbe addirittura fornire esattamente il file che esso richiede se si riuscisse a capire qual è il file richiesto. Esso permette inoltre di loggare le richieste e le connessioni in arrivo, così che ci si possa rendere conto delle attività del malware.

Per utilizzarlo è possibile creare una VM con SO Linux che è collegata alla VM sulla quale è eseguito il malware, impostando una configurazione di rete interna tra le due e dirottando tutto il traffico della macchina vittima verso la macchina sulla quale è eseguito INetSim.

Utilizzando sulla macchina Linux INetSim in combinazione con Wireshark è possibile ricavare informazioni preziose.

# **Capitolo 5**

## **Advanced Malware Analysis e Forensics**

### **5.1 Advanced Malware Analysis**

Le tecniche di advanced malware analysis si basano sull’analisi del codice per capire il funzionamento interno del binario. Esse vengono utilizzate in quanto, malgrado le tecniche di basic malware analysis siano utili per capire le funzionalità di base del malware, queste ultime non sono in grado di fornire tutte le informazioni necessarie riguardo le sue funzionalità. Potremmo ad esempio trovare tramite basic static analysis che una particolare funzione sia importata, ma non possiamo sapere se e come venga effettivamente utilizzata; oppure potremmo capire come un malware risponde quando riceve un determinato pacchetto tramite una basic dynamic analysis, ma per capire effettivamente il formato del pacchetto bisogna scavare più a fondo, guardando nel codice del binario.

Effettuare il reversing del codice del malware può permettere la decodifica dei dati criptati al suo interno, determinandone la logica e osservando altre sue capacità che non sono state rivelate durante la basic analysis.

Gli autori di malware scrivono codice malevolo in un linguaggio di alto livello, ad esempio C o C++, il quale è compilato in un eseguibile attraverso un compilatore. Tuttavia, in questa fase dell’analisi tutto ciò che l’analista ha a disposizione è l’eseguibile malevolo, senza avere il suo codice sorgente. In questo caso, quindi, per capire più a fondo il funzionamento interno del malware e capirne gli aspetti critici, l’unica speranza è l’utilizzo dell’analisi avanzata, che si basa sull’analisi del codice del

malware ottenuto a partire dal suo binario tramite reverse-engineering.

L'advanced malware analysis può essere suddivisa, come per la basic malware analysis, in analisi statica e dinamica:

- **Advanced static analysis:**

consiste nel fare un reverse-engineering del malware caricando il suo eseguibile all'interno di un disassembler, che è un programma che traduce codice macchina in codice assembly. In questo modo è possibile interpretare il codice per capire il comportamento del programma senza eseguire il binario.

- **Advanced dynamic analysis:**

consiste nell'eseguire il malware in maniera controllata all'interno di un debugger, il quale è anch'esso un programma che disassembra il codice, però permette anche di eseguire sia le istruzioni una per una, sia intere funzioni in una sola volta, invece di eseguire l'intero programma.

Durante tale processo di esecuzione controllato è possibile visualizzare i valori memorizzati nelle varie locazioni di memoria, in modo da avere visione di ciò che accade in ogni momento.

Un malware però può contenere migliaia di linee di codice e centinaia di funzioni, per cui è difficile tenere traccia di tutte le variabili e le funzioni in gioco. Per questo ci vengono in aiuto tool avanzati con funzionalità di disassembling e debugging.

In questo elaborato non si tratterà a fondo questo tipo di analisi, in quanto essa ha una curva di apprendimento molto più ripida rispetto alla basic malware analysis. È infatti richiesta la conoscenza approfondita di linguaggi assembly e di alto livello, architettura dei calcolatori e concetti avanzati dei SO, in particolare del SO Windows. Verranno di seguito comunque introdotti due tra i tool più famosi e utilizzati per l'analisi avanzata, ossia IDA Pro e OllyDbg.

## **IDA Pro**

Interactive Disassembler Professional (IDA Pro) è un disassembler molto potente ed è quello più utilizzato nella malware analysis e nel reverse-engineering. Esso può girare su Windows, Linux e Mac OS e supporta l'analisi di vari formati di file, compreso i file in formato PE.

Oltre alla versione commerciale, ne esiste anche una versione gratuita con alcune limitazioni, come ad esempio la mancata possibilità di effettuare anche debugging, cosa invece permessa nella versione commerciale.

IDA Pro permette di disassemblare un intero programma e consente di fare varie attività, come la function discovery, lo stack analysis, la local variable identification e tanto altro.

Un’importante caratteristica di questo disassembler è la sua interattività, cioè tutti gli aspetti del processo di disassemblaggio possono essere modificati, manipolati, riordinati e ridefiniti.

È anche possibile salvare i progressi della propria analisi all’interno di un database.

### OllyDbg

OllyDbg è un debugger per sistemi Windows a 32 bit. Esso traccia i registri, riconosce le funzioni e i parametri passati alle principali librerie standard, insieme ad eventuali salti condizionali, tabelle, costanti, variabili e stringhe. È uno dei software più utilizzati dagli specialisti del settore poiché è gratuito, è semplice da usare ed ha la possibilità di essere esteso tramite plugin.

Uno dei primi utilizzi fu quello di craccare programmi ed effettuare Reverse engineering, mentre tutt’oggi è sfruttato soprattutto per la sua capacità di analizzare i malware.

Si noti che questo tool permette solo il debugging di binari compilati per processori a 32 bit, come sono la maggior parte dei malware in formato PE. Se ci fosse la necessità di analizzare un binario a 64 bit, allora bisognerebbe utilizzare altri tools, come ad esempio x64dbg.

## 5.2 Analisi forense e memory forensics

Con il termine analisi forense ci si riferisce ad una dettagliata investigazione con l’obiettivo di rilevare e documentare il corso, i motivi, i colpevoli e le conseguenze di un incidente di sicurezza o una violazione delle regole di un’organizzazione o di leggi di stato. Essa implica l’uso di un ampio insieme di tecnologie e procedure investigative.

Gli specialisti dell’analisi forense raccolgono differenti tipi di informazioni lavorando sia in maniera convenzionale con informazioni cartacee, sia con dispositivi elettronici. Nell’ultimo caso si parla nello specifico di digital forensics.

La digital forensics può però essere utilizzata anche nella malware analysis. In particolare, quella utilizzata è una sua parte chiamata memory forensics.

La memory forensics è una tecnica investigativa che mira a trovare ed estrarre artefatti forensi dalla memoria RAM di un computer, la quale contiene importanti informazioni riguardo allo stato di esecuzione di un sistema, come le applicazioni che girano nel sistema, gli oggetti ai quali accedono (in termini di files, registro, ecc), le connessioni di rete attive, i moduli e i driver caricati, ma anche chiavi di cifratura, password, clipboard e tanto altro. Per questo motivo la memory forensics può essere utilizzata nelle azioni di risposta agli incidenti di sicurezza (incident response) e nella malware analysis.

In particolare, nella incident response molto spesso ci si trova a non avere accesso al campione del malware, ma si può solamente ottenere l’immagine della memoria del sistema sospetto. Tale immagine può essere analizzata per confermare l’infezione e per trovare gli artefatti malevoli. Nella malware analysis, invece, il campione del malware è accessibile ma può essere comunque utile fare questa operazione per ottenere informazioni aggiuntive riguardo il comportamento del malware nella fase post-infection. Può essere inoltre utile per determinare le eventuali capacità di evasione del malware, oppure particolari malware che non scrivono alcun componente malevolo sul disco, ma solo in memoria (malware fileless).

Gli step della memory forensics sono principalmente due:

1. Memory acquisition, ossia l’acquisizione (il dumping) della memoria della macchina target su un supporto non volatile separato.
2. Memory analysis, cioè l’analisi del dump ottenuto nella fase precedente per estrarre artefatti forensi o, nel nostro caso, evidenze di infezione.

Per quanto riguarda la memory acquisition esistono numerosi tools, come ad esempio DumpIt, Memoryze e WinPmem, l’ultimo dei quali è open source.

Invece, per quanto riguarda la memory analysis il tool maggiormente utilizzato è Volatility, un framework open source per la memory forensics avanzato scritto in Python e che permette di estrarre artefatti digitali dall'immagine della memoria.

# Capitolo 6

## Automatic Malware Analysis

Come è possibile intuire dai capitoli precedenti, il processo manuale da seguire per portare a termine l’analisi di un malware è ripetitivo e time-consuming. Esso è stato per anni l’unico metodo possibile, però ciò richiedeva personale altamente qualificato a livello teorico e tecnico e che fosse armato di grande pazienza. Ovviamente, con l’aumentare in maniera vertiginosa del numero di malware che vengono creati ogni giorno, questo approccio diventa assolutamente non scalabile. Se, infatti, un analista potesse analizzare un malware ogni 30 minuti e lavorare, senza pause, per 8 ore al giorno, riuscirebbe ad analizzarne solo 16 nell’arco dell’intera giornata e, considerando 250 giorni lavorativi all’anno, riuscirebbe a completare l’analisi di soli 4000 malware all’anno. Purtroppo, però, 4000 è un numero che non è neanche lontanamente vicino al numero di malware che vengono creati attualmente ogni giorno. Per questo motivo sono sempre più popolari strumenti che permettono l’automazione della malware analysis, che permette ai team di sicurezza di reagire più rapidamente alle potenziali minacce e consentire loro di dedicare maggior tempo ad attività come la risk-reduction. Uno degli strumenti più utilizzati oggigiorno per l’analisi automatica dei malware sono le sandbox.

### 6.1 Sandbox

Una sandbox è un meccanismo di sicurezza utilizzato per avviare programmi sospetti in un ambiente isolato e controllato, senza rischiare di infettare un sistema reale. Le sandbox forniscono degli

ambienti virtualizzati che simulano la maggior parte dei servizi di un sistema reale, in modo che il software in essi eseguito possa funzionare normalmente. Il loro obiettivo, quindi, è quello di registrare ogni operazione associata al file avviato, fornendo un report che include varie informazioni, come attività in rete, file system e registro, i processi creati, le funzioni importate ed esportate e tanto altro. Inoltre, utilizzando pattern comportamentali precedentemente sintetizzati, esse possono anche avvertire l'analista della pericolosità di un file (o in alcuni casi anche bloccarlo direttamente) se questo mostra un comportamento inconsueto per la natura del file stesso.

Tuttavia, come già detto all'inizio di questo elaborato, la figura dell'analista non può essere sostituita totalmente dagli strumenti di analisi automatica come le sandbox, in quanto l'unico modo di capire nel dettaglio le funzionalità dei malware rimane quello di procedere manualmente.

## 6.2 Proprietà delle Sandbox

Una buona sandbox deve raggiungere tre obiettivi:

1. Visibilità, cioè deve vedere il più possibile dell'esecuzione di un programma, altrimenti potrebbe non rendersi conto delle attività svolte dal malware e potrebbe non trarre solide deduzioni sulla presenza o assenza di comportamenti dannosi.
2. Resistenza al rilevamento, ovvero deve eseguire il monitoraggio in modo tale da rendere difficile al malware l'identificazione della propria presenza e, in risposta, alterare il proprio comportamento per eludere il rilevamento.
3. Scalabilità, cioè deve poter eseguire molti campioni contemporaneamente, garantendo che l'esecuzione di un campione non interferisca con l'esecuzione degli altri.

## 6.3 Classificazione delle Sandbox

Esistono di base 3 modi per lavorare con una sandbox:

- creare un laboratorio personalizzato con il proprio insieme di tool;
- acquistare una sandbox preconfezionata e installarla (preferibilmente in loco);
- utilizzare una sandbox disponibile pubblicamente.

Avere un laboratorio personalizzato o acquistare una sandbox preconfezionata non sono soluzioni percorribili per tutti, perché sono soluzioni troppo complesse, richiedono troppo tempo e/o sono troppo costose. In questi casi le sandbox disponibili pubblicamente possono essere d'aiuto.

Tuttavia, ci sono alcuni contro legati all'utilizzo di sandbox pubbliche:

- le sandbox pubbliche sono ovviamente disponibili e accessibili a tutti, inclusi i malintenzionati;
- se si carica un campione di malware che è stato specificamente pensato per il proprio ambiente, allora si sta essenzialmente dando la possibilità ai propri avversari di capire che le loro operazioni sono state rilevate;
- non è una buona idea caricare campioni che contengono informazioni specifiche sul proprio ambiente, come password o impostazioni di configurazione. È inoltre opportuno astenersi dalla condivisione di campioni contenenti dati riservati degli utenti o dei clienti o materiale protetto da copyright.

Un altro parametro in base al quale è possibile fare una classificazione delle sandbox è l'interazione con l'operato dell'analista. In base a questo parametro distinguiamo 2 tipologie di sandbox:

1. Sandbox con interazione umana, che prevedono l'interazione dell'analista durante la detonazione del malware. Questa proprietà permette all'analista di bloccare il funzionamento del file sospetto e/o di interagire manualmente con esso rispondendo, per esempio, a specifiche richieste che il malware stesso potrebbe effettuare per garantire una corretta esecuzione. Questa capacità permette all'analista di avere un punto di vista preferenziale, ma delega la complessità dell'individuazione comportamentale all'essere umano. Ovviamente tali sandbox hanno capacità di scalabilità ridotte e, se lasciate eseguire senza l'intervento umano, sono più semplici da

eludere, in quanto non implementano logiche di automatismo e di auto-hiding (tecnica per la quale una sandbox cerca di assomigliare il più possibile ad un sistema vittima).

2. Sandbox senza interazione umana, che hanno capacità di scalare molto più ampie (cioè hanno la capacità di analizzare molti più file) e funzionalità di auto-hiding (ossia di apparire come un normale sistema vittima). Tuttavia, esse non offrono all'analista un punto di vista flessibile, non abilitando l'analista all'interazione con il malware in esecuzione.

## 6.4 Tecniche anti-sandboxing

Alcuni malware riescono a capire quando vengono eseguiti all'interno di una VM o di una sandbox, ad esempio controllando quanti e quali file sono presenti nel file system, la cronologia dei browser o se nel sistema in cui si sono insediati esiste un client di posta elettronica. Inoltre, possono controllare la frequenza con cui arrivano i comandi dall'esterno, in quanto i software e gli esseri umani producono input con frequenze completamente diverse. Qualora uno di questi malware dovesse accorgersi di essere studiato, potrebbe cambiare il proprio comportamento o addirittura restare dormiente.

Nell'immagine 6.4.1 sono riassunte le principali tecniche di evasione utilizzate dai malware.

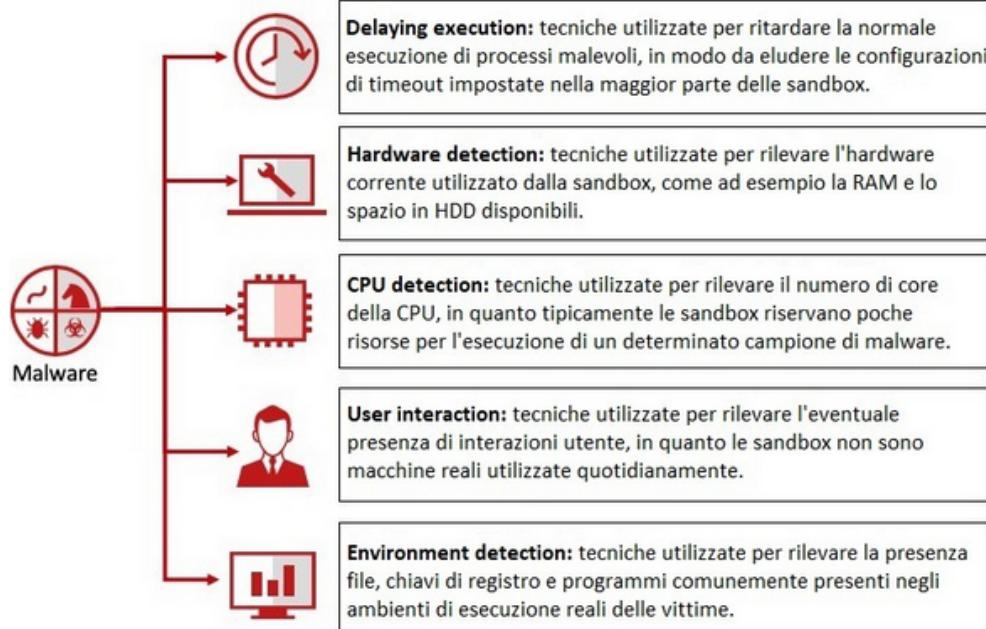


Figura 6.4.1: Principali tecniche di evasione

## 6.5 Le Sandbox più utilizzate

### Joe Sandbox

Joe Sandbox è una piattaforma per la malware analysis messa a disposizione da Joe Security. Essa è disponibile in due versioni: cloud e locale. La versione Cloud permette di eseguire file e analizzare URL in maniera automatica in un ambiente controllato e, inoltre, monitora il comportamento delle applicazioni e del sistema operativo alla ricerca di attività sospette. Tutte le attività sono raccolte in report di analisi che contengono informazioni rilevanti circa le minacce rilevate. Joe Sandbox Cloud consente l'analisi di qualsiasi malware destinato ai sistemi operativi Windows, Android, macOS, Linux e iOS. Joe Security garantisce che Joe Sandbox Cloud è una soluzione pensata per mantenere privati i file caricati e i report delle analisi, il che vuol dire che nessun dato è condiviso con terze parti.

Joe Sandbox Cloud Basic è raggiungibile al seguente link:

<https://www.joesandbox.com/>

Per poter sottomettere i file da analizzare, è richiesta una registrazione.

### Falcon Sandbox

Falcon Sandbox è una soluzione per l'analisi automatizzata dei malware messa a disposizione da Payload Security. Falcon Sandbox esegue un'analisi approfondita delle minacce evasive e sconosciute, arricchisce i risultati tramite l'utilizzo della threat intelligence e fornisce anche dei possibili indicatori di compromissione. Hybrid-Analysis.com è, invece, una pagina web alimentata da Falcon Sandbox e messa a disposizione sempre da Payload Security per offrire un servizio gratuito per l'analisi di malware. Inoltre, Hybrid-Analysis.com permette di cercare report di malware già analizzati e permette di scaricare campioni di malware. Tuttavia, fornisce solo un sottoinsieme delle funzionalità messe a disposizione da Falcon Sandbox. Hybrid Analysis combina i dati relativi all'esecuzione dei malware con un'analisi approfondita dei dump della memoria al fine di individuare IOC, come stringhe o sequenze di chiamate API, che permettono di rilevare anche minacce sconosciute. Tutti i dati estratti dal motore di Hybrid Analysis sono automaticamente elaborati e integrati nei report di Falcon Sandbox.

La pagina di Hybrid Analysis è raggiungibile al seguente link: <https://www.hybrid-analysis.com/>.

## Any.Run

Any.Run è una sandbox interattiva i cui servizi sono utilizzabili online. Essa permette sia di assistere al processo di ricerca automatica di una minaccia e sia di intervenire manualmente, proprio come se fossimo in un sistema reale. Inoltre, Any.Run mette a disposizione degli utenti registrati la possibilità di scaricare campioni di malware già analizzati.

Any.Run è raggiungibile al seguente link:

<https://any.run/>.

Per poter sottomettere i file da analizzare, è anche qui richiesta una registrazione.

## Cuckoo Sandbox

Cuckoo Sandbox è un sistema open source per l'analisi automatizzata di malware. E' un software gratuito che permette di analizzare qualsiasi file sospetto che ha come target un SO Windows, macOS, Linux o Android. Di default Cuckoo Sandbox è in grado di:

- analizzare diverse tipologie di file come eseguibili, documenti Office, pdf e email.
- analizzare e scaricare il traffico di rete, anche se crittografato con SSL/TLS. Inoltre, grazie al supporto nativo al routing di rete, è in grado di eliminare il traffico di rete oppure instradarlo tramite INetSim, un'interfaccia di rete o una VPN.
- eseguire analisi avanzate della memoria del sistema virtualizzato infetto.

Per tali motivi Cuckoo Sandbox permette di recuperare informazioni come file creati, cancellati e scaricati dai malware durante la loro esecuzione. Fornisce inoltre una traccia del traffico di rete in formato PCAP, dei dump della memoria dei processi malevoli e degli screenshots per offrire ulteriori informazioni relative all'esecuzione dei malware.

Una versione di Cuckoo Sandbox messa a disposizione as-a-service dal CERT (Computer Emergency Response Team) è reperibile al seguente URL: <https://cuckoo.cert.ee/>.

# Capitolo 7

## Analisi del file *places.docx.exe*

Per mostrare in pratica le nozioni illustrate a livello teorico, è stata condotta un’analisi utilizzando un campione di malware, *places.docx.exe*, recuperato in rete. Tale file è stato fornito in allegato a questo documento in formato compresso (password “infected”). Gli autori non si assumono alcuna responsabilità della mancata osservanza delle misure precauzionali descritte in dettaglio nell’appendice A. Il file compresso è infatti un MALWARE, quindi bisogna maneggiarlo con attenzione.

Si procederà quindi con le fasi di basic static analysis e basic dynamic analysis, per poi terminare con l’analisi automatica tramite sandbox.

### 7.1 Basic Static Analysis

Il file utilizzato si presenta con il nome *places.docx* con un’icona associabile al programma Microsoft Office Word, ma inizialmente tale programma non era installato all’interno del sistema Windows, per cui il fatto che abbia tale icona risulta molto strano. Ciò fa pensare immediatamente di essere davanti ad un caso di double extension. Togliendo infatti la spunta dall’opzione *Hide extensions for known file types*, il file si rivela per ciò che è realmente, ossia un file *.exe*, in accordo con il sospetto iniziale (figura 7.1.1).



Figura 7.1.1: Double extension

Per determinare il tipo di file, il campione *places.docx.exe* è stato aperto con il programma Exeinfo PE, il quale ha presentato il seguente risultato (figura 7.1.2):

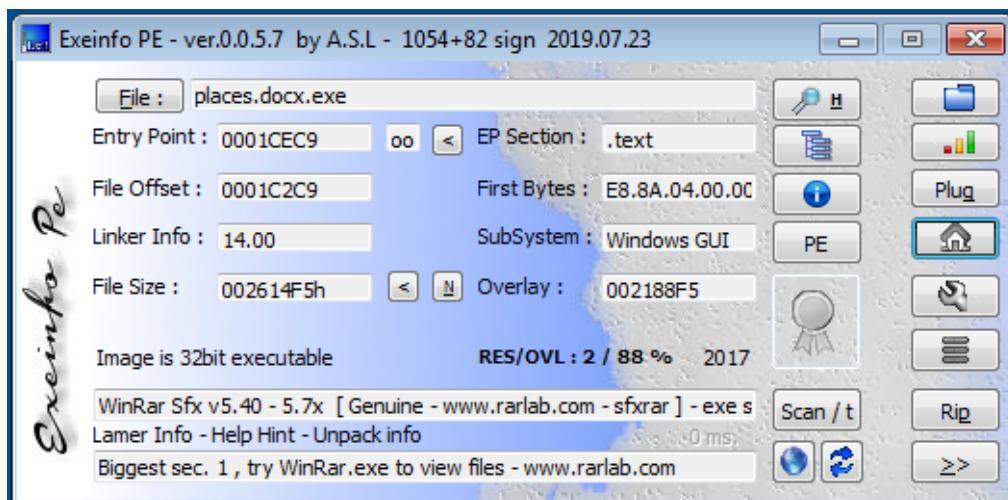


Figura 7.1.2: Suggerimento di estrazione da Exeinfo PE

Dall’analisi in Exeinfo PE risulta che il file è un *32bit executable* e presenta l’utilizzo del subsystem *Windows GUI*. Inoltre il tool ci suggerisce di estrarne il contenuto con WinRar.

Si noti che, se il file fosse stato packed, il tool sarebbe probabilmente riuscito ad individuare lo strumento utilizzato per effettuare tale packing, in modo da poterlo reperire per effettuare l’operazione inversa. Il fatto che ci venga suggerito di aprirlo con WinRar, ci porta a pensare che esso sia un archivio auto-estraente, ossia un archivio che non necessita di un programma di gestione archivi per estrarre le informazioni contenute al suo interno, in quanto viene generato un file .exe con il quale, cliccandoci sopra, è possibile avviare la procedura guidata di estrazione. Osservando le proprietà del file, infatti, questo è ciò che ci si trova davanti (figura 7.1.3):



Figura 7.1.3: Proprietà di archivio auto-estraente

Aprendo il file con WinRAR, sono stati trovati i seguenti files (figura 7.1.4):

Nome oggetto	Dimensione	Compresso	Tipo	Modificato il	CRC32
File folder					
audiqdq.exe	202.752	90.477	Application	01/01/2018 22:49	98DE1E08
places.docx	2.140.085	2.106.826	Office Open XML Document	05/04/2018 02:31	4F460834

Figura 7.1.4: Estrazione con WinRAR

Come era possibile intuire, il file conteneva al suo interno un file *.docx*, utilizzato probabilmente solo come operazione di facciata, mentre altre azioni potrebbero essere condotte dal file *audiqdq.exe*.

Si è quindi analizzato anche il file *audiqdq.exe* con il programma Exeinfo PE (figura 7.1.5).

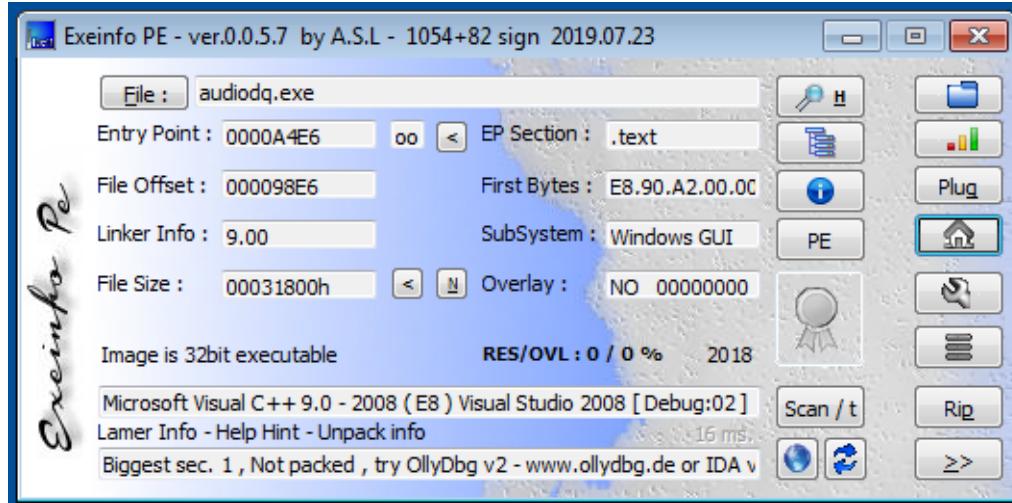


Figura 7.1.5: Exeinfo PE - File not packed

Anch'esso viene identificato come *32bit executable* con subsystem *Windows GUI*, ma questa volta il tool ci restituisce anche l'IDE utilizzato e, soprattutto, la dicitura “*Not packed*”, il che vuol dire che non è stata effettuata alcuna azione di offuscamento.

Per accertarsi del formato del file, è stato utilizzato anche il tool HxD hex editor (figura 7.1.6):

Offset(h)	00 01 02 03 04 05 06 07	Decoded text
00000000	E4 5A 90 00 03 00 00 00	MZ.....
00000008	04 00 00 00 FF FF 00 00	....ÿÿ..
00000010	B8 00 00 00 00 00 00 00	.....
00000018	40 00 00 00 00 00 00 00	@.....
00000020	00 00 00 00 00 00 00 00	.....
00000028	00 00 00 00 00 00 00 00	.....
00000030	00 00 00 00 00 00 00 00	.....
00000038	00 00 00 E8 00 00 00 00	...è....
00000040	0E 1F BA 0E 00 B4 09 CD	..°..í..
00000048	21 B8 01 4C CD 21 54 68	!.Lí!Th
00000050	69 73 20 70 72 6F 67 72	is progr
00000058	61 6D 20 63 61 6E 6E 6F	am canno
00000060	74 20 62 65 20 72 75 6E	t be run
00000068	20 69 6E 20 44 4F 53 20	in DOS
00000070	6D 6F 64 65 2E OD OD OA	mode....
00000078	24 00 00 00 00 00 00 00	\$.....
00000080	AA C3 11 B9 EE A2 7F EA	Ã.:ic.é
00000088	EE A2 7F EA EE A2 7F EA	ic.éic.é

Figura 7.1.6: HxD - Analisi formato file

Esso riporta la signature distintiva *MZ* che ci dice che questo è un file di tipo *portable executable PE*, così come anche la stringa “*This program cannot be run in DOS mode*”.

Tali informazioni, insieme a tante altre informazioni utili, possono essere ricavate analizzando il file con il tool PE studio (figura 7.1.7):

property	value
md5	FD42D02EA05AF684C7523D5D5719E55C
sha1	8999E312A6058F6EFC8B8A0360D195425D0A8535
sha256	4398615F0A498124B498ABB261150F11CE4778AFDE18D2F4E6717F2826A9D65D
md5-without-overlay	n/a
sha1-without-overlay	n/a
sha256-without-overlay	n/a
first-bytes-hex	4D 5A 90 00 03 00 00 04 00 00 FF FF 00 00 B8 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00
first-bytes-text	M Z .....
file-size	202752 (bytes)
size-without-overlay	n/a
entropy	6.580
imphash	n/a
signature	Microsoft Visual C++ 8
entry-point	E8 90 A2 00 E9 79 FE FF FF 8B 4C 24 04 F7 C1 03 00 00 00 74 24 8A 01 83 C1 01 84 C0 74 4E F7 C1
file-version	6, 1, 7600, 12786
description	Windows Audio Device Graph Isolation
file-type	executable
cpu	32-bit
subsystem	GUI
compiler-stamp	0x5A4B2B71 (Mon Jan 01 22:49:21 2018)
debugger-stamp	0x5A4B2B71 (Mon Jan 01 22:49:21 2018)
resources-stamp	empty
exports-stamp	n/a
version-stamp	empty
certificate-stamp	n/a

Figura 7.1.7: PE studio

Come è possibile vedere nello screenshot in alto, nella schermata principale di PE studio è possibile visualizzare, oltre alle informazioni già evidenziate in precedenza, anche il compiler-stamp e una descrizione. In particolare, il compiler-stamp riporta l'anno 2018, il che è una data plausibile al tempo della nostra analisi, mentre la descrizione riporta “*Windows Audio Device Graph Isolation*”, il che vuol dire che questo eseguibile probabilmente vuole fingersi il processo di Windows *audiodg.exe*. Il vero processo *audiogd.exe* permette alle applicazioni di terze parti di poter utilizzare l'audio del computer. Il nome del file *audiogd.exe* è stato quindi pensato ad arte, visto che esso differisce di una sola lettera rispetto al nome del programma legittimo.

Vengono inoltre restituiti gli hash MD5, SHA-1 e SHA-256, i quali possono essere ottenuti anche tramite il tool HashMyFiles, che in aggiunta permette di mostrare l'hash di più files contemporaneamente. Sono infatti stati aperti con HashMyFiles sia il file originario, sia quelli derivati dalla sua estrazione (figura 7.1.8).

Filename	Identical	Extension	File Size	MD5	SHA1	SHA-256
places.docx.exe		exe	2.495.733	62bb4224d8e8ec5c3495090b09b52e1c	ce478936f245647b557cd782a48c071fb7545e...	d18a7cd2bacd43e9a69f25018a3e26bca7f5a...
audiodq.exe		exe	202.752	fd42d02ea05af684c7523d5d5719e55c	8999e312a6058f6efc8b8a0360d195425d0a85...	4398615f0a498124b498abb261150f11ce4778...
places.docx		docx	2.140.085	7f4802eaeeae7c29c3cf60e247fb9626	b505f26476d19b6a974b112b061326fc7af1ca...	3b052247207a0e7227cf7c07039b8a37ba392e...

Figura 7.1.8: HashMyFiles

Tramite tali hash è possibile consultare il database di VirusTotal per controllare se tali file sono stati già analizzati e identificati come malevoli.

Dai seguenti screenshot possiamo notare come sia il file *places.docx.exe* (figura 7.1.9) che il file *audiodq.exe* (figura 7.1.10) siano stati identificati come malevoli:

Figura 7.1.9: VirusTotal - places.docx.exe

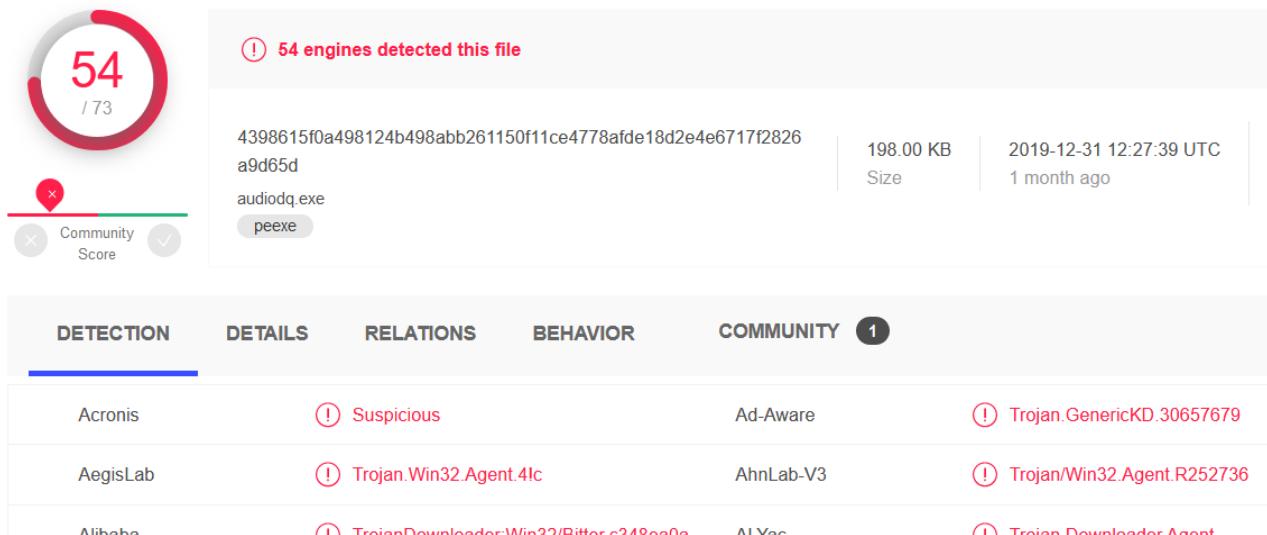


Figura 7.1.10: VirusTotal - audiodq.exe

Il file *places.docx*, invece, è stato targato come non malevolo (figura 7.1.11):

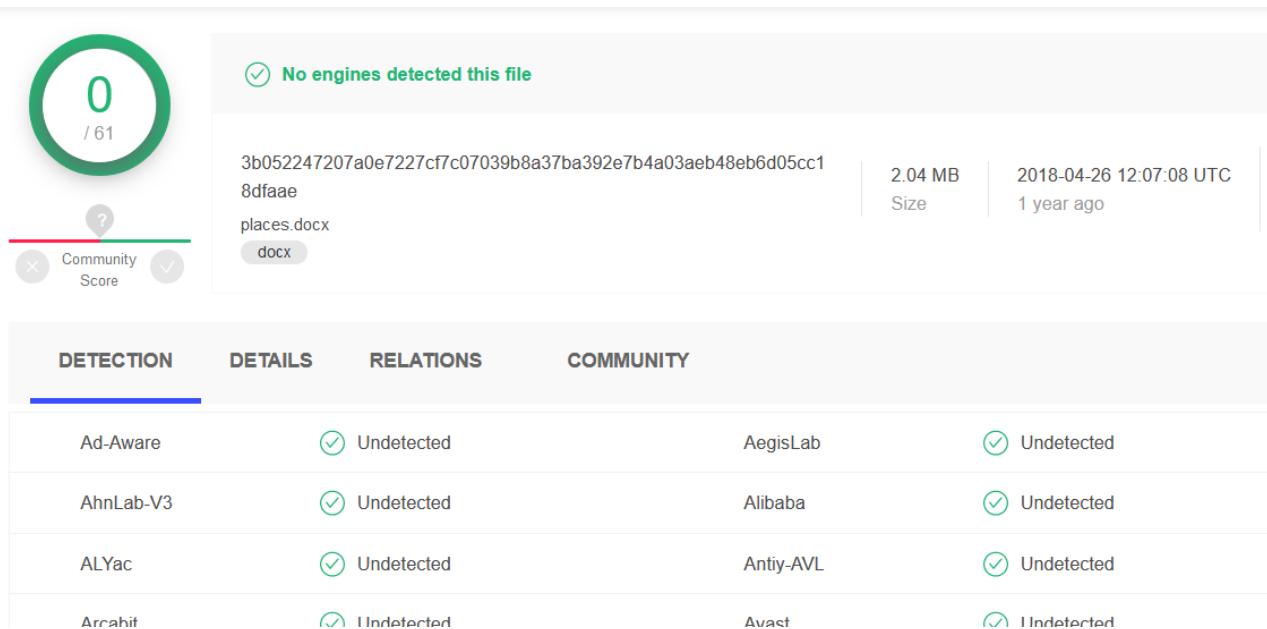


Figura 7.1.11: VirusTotal - places.docx

Grazie a PE studio è possibile visualizzare anche dei possibili indicatori di compromissione relativi al file sospetto (figura 7.1.12):

xml-id	indicator (32)	detail	level
1430	The file references string(s) tagged as blacklist	count: 44	1
1269	The file references blacklist library(ies)	count: 1	1
1266	The file imports symbol(s) tagged as blacklist	count: 32	1
1124	The file references MITRE Technique(s)	count: 5	2
1262	The file imports anonymous function(s)	count: 12	2
1036	The file checksum is invalid	checksum: 0x00000000	2
1424	The original name of the file has been detected	name: audiodq.exe	3
1215	The file ratio of the section(s) has been determined	ratio: 00.10%	3

Figura 7.1.12: PE studio - Indicatori

Da tale immagine possiamo notare la presenza di stringhe, librerie e funzioni importate che il tool riconosce come “blacklistate”, ossia presenti spesso all’interno di file malevoli. Viene inoltre rilevato il fatto che la checksum è invalida, il che è spesso legato al fatto di trovarsi di fronte ad un eseguibile malevolo. Infatti, alcuni compilatori utilizzati dagli autori di malware potrebbero non supportare la generazione della checksum, oppure tali autori potrebbero modificare l’eseguibile dopo la compilazione, il che invaliderebbe tale checksum. Ad esempio, l’applicazione di tecniche di packing ed offuscamento possono comportare la modifica di un file dopo la sua compilazione e sono spesso effettuate tramite tool che tipicamente non si preoccupano di aggiornare la checksum.

Il passo successivo è stato quello di analizzare in dettaglio le stringhe contenute nel file tramite l’utilizzo di PE studio (figura 7.1.13):

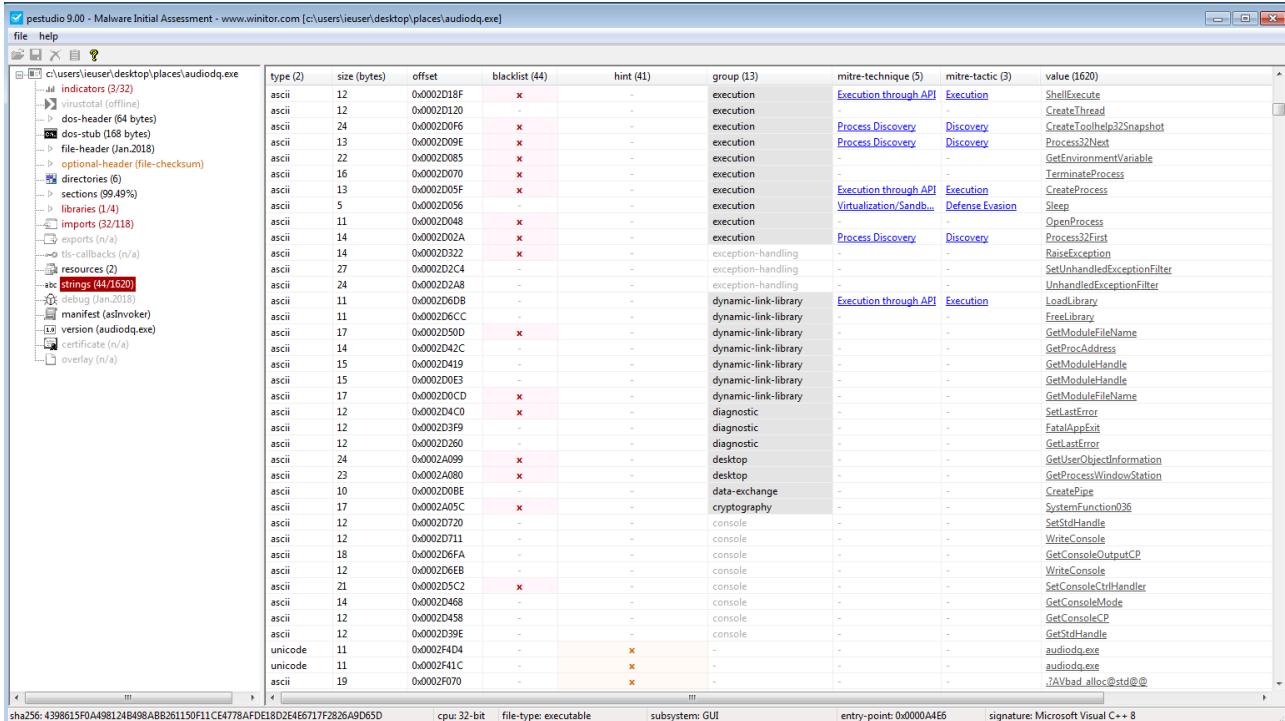


Figura 7.1.13: PE studio - Stringhe

Tale screen non è esaustivo, ma mette in risalto le potenzialità offerte dallo strumento. In particolare, le stringhe vengono marcate sia come “blacklist”, ossia stringhe associate solitamente a comportamenti malevoli, sia come “hint”, ossia stringhe che possono dare degli indizi sul comportamento dell’eseguibile. Se una stringa è marcata, però, ciò non vuol dire per forza che essa sia legata ad un comportamento malevolo e, allo stesso modo, possono esserci alcune stringhe non marcate ma che in realtà possono essere legate a comportamenti malevoli.

Alcune delle stringhe più interessanti che sono state trovate sono le seguenti:

- “GET %s”, “GET /”, “HTTP/1.0”, “HTTP/1.1\n”, “https://”, “Host: %s”: queste stringhe suggeriscono che questo eseguibile potrebbe fare attività su internet, in particolare utilizzando il protocollo http/https, probabilmente per connettersi con un C&C o per scaricare ulteriori files.
- “audiodq.exe”: questa stringa rappresenta il nome dell’eseguibile. Essa è probabilmente usata per poter salvare un file con lo stesso nome in un’altra locazione.
- “d:\new\_downloader\_healthnewsone\Release\audiodq.pdb”: questa stringa contiene un path che fornisce varie informazioni, come ad esempio la parola *downloader*, che potrebbe far pen-

sare al fatto che l'eseguibile sia proprio un malware di categoria downloader, mentre la parola *healthnewsone* potrebbe dare indicazioni ad esempio sul nome del malware o sul dominio al quale si connette. Il file specificato, inoltre, è in formato *.pdb* (Program DataBase), cioè un formato relativo a files contenenti informazioni di debugging relative ad un certo programma, solitamente generati durante il processo di compilazione.

- Sono state inoltre rilevate stringhe che fanno riferimento a funzioni di libreria di Windows. Queste saranno brevemente trattate nella sezione relativa all'analisi delle funzioni di libreria importate dall'eseguibile.

Si è poi passati all'analisi delle informazioni contenute nell'header PE.

Per prima cosa sono state analizzate le sezioni tramite il tool PE studio (figura 7.1.14):

property	value	value	value	value	value
name	.text	.rdata	.data	.rsrc	.reloc
md5	2FCF81D28A6EDE26544DEA...	C632C2753218D9382C8B412...	A369D4957AC0079D9A1518...	CE007EF93132DFB97275B97...	81D168486715FE72286CE27...
entropy	6.622	5.185	3.059	3.913	6.541
file-ratio (99.49%)	80.56 %	10.86 %	3.28 %	0.76 %	4.04 %
raw-address	0x00000400	0x000028200	0x00002D800	0x00002F200	0x00002F800
raw-size (201728 bytes)	0x00027E00 (163328 bytes)	0x00005600 (22016 bytes)	0x00001A00 (6656 bytes)	0x00000600 (1536 bytes)	0x00002000 (8192 bytes)
virtual-address	0x00401000	0x00429000	0x0042F000	0x00435000	0x00436000
virtual-size (215620 bytes)	0x00027C66 (162918 bytes)	0x000055D6 (21974 bytes)	0x00005310 (21264 bytes)	0x00000560 (1376 bytes)	0x00001F98 (8088 bytes)
entry-point	0x0000044E6	-	-	-	-
writable	-	-	x	-	-
executable	x	-	-	-	-
shareable	-	-	-	-	-
discardable	-	-	-	-	x
initialized-data	-	x	x	x	x
uninitialized-data	-	-	-	-	-
unreadable	-	-	-	-	-
self-modifying	-	-	-	-	-
blacklisted	-	-	-	-	-
virtualized	-	-	-	-	-

Figura 7.1.14: PE studio - Sezioni

Da questa immagine è possibile vedere che le sezioni sono quelle solite, con l'entry point correttamente piazzato nella sezione *.text*, che è anche l'unica ad essere eseguibile. Per quanto riguarda la *raw size* e la *virtual size*, esse sono praticamente uguali, cosa che invece non sarebbe stata vera se il file fosse stato sottoposto a packing, in quanto in quel caso lo spazio occupato su disco sarebbe stato molto minore di quello occupato in memoria a valle della procedura di unpacking. Vengono inoltre mostrati anche gli hash MD5 delle singole sezioni.

Si è poi passati all'analisi delle librerie importate (figura 7.1.15):

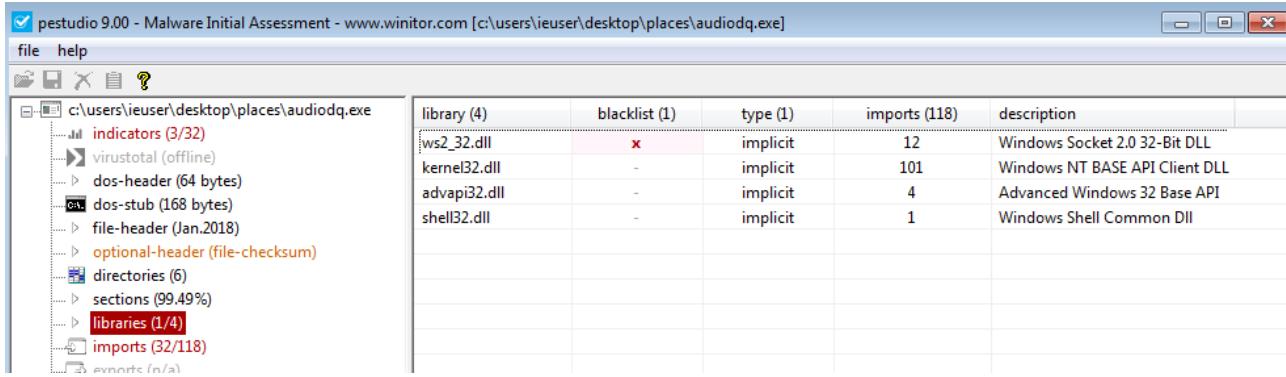


Figura 7.1.15: PE studio - Librerie

Le librerie mostrate sono librerie molto comuni, importate dalla maggior parte degli eseguibili Windows. In particolare, *Kernel32.dll* contiene funzionalità di base del SO come accesso e manipolazione di memoria, file e hardware, mentre *advapi32.dll* è relativa alla gestione dei servizi e all'accesso al registro di sistema. La libreria *shell32.dll*, invece, è relativa alle API della shell di Windows. La libreria *ws2\_32.dll* è una libreria relativa al networking e all'utilizzo delle socket. Essa è blacklisted in quanto un programma che la utilizza probabilmente esegue task relativi alla rete, il che ovviamente non deve essere per forza qualcosa di malevolo. Tuttavia, dato che in questo caso sospettiamo che questo sia un malware di tipo downloader, questa libreria risulta sospetta.

Per scendere più nel dettaglio e capire quali siano effettivamente le funzioni importate da tali librerie, è possibile visualizzare gli import (figura 7.1.16):

name (118)	group (10)	mitre-technique (5)	mitre-tactic (3)	type (1)	anonymous (12)	blacklist (32)	library (4)
GetTimeZoneInformation	system-information	-	-	implicit	-	x	kernel32.dll
MoveFileA	file	-	-	implicit	-	x	kernel32.dll
CreateProcessA	execution	Execution through API	Execution	implicit	-	x	kernel32.dll
TerminateProcess	execution	-	-	implicit	-	x	kernel32.dll
GetEnvironmentVariableA	execution	-	-	implicit	-	x	kernel32.dll
Process32Next	execution	Process Discovery	Discovery	implicit	-	x	kernel32.dll
CreateToolhelp32Snapshot	execution	Process Discovery	Discovery	implicit	-	x	kernel32.dll
OpenProcess	execution	-	-	implicit	-	x	kernel32.dll
Process32First	execution	Process Discovery	Discovery	implicit	-	x	kernel32.dll
GetCurrentThreadId	execution	-	-	implicit	-	x	kernel32.dll
GetCurrentThread	execution	-	-	implicit	-	x	kernel32.dll
GetEnvironmentStringsW	execution	-	-	implicit	-	x	kernel32.dll
GetCurrentProcessId	execution	-	-	implicit	-	x	kernel32.dll
SetEnvironmentVariableA	execution	-	-	implicit	-	x	kernel32.dll
ShellExecuteA	execution	Execution through API	Execution	implicit	-	x	shell32.dll
RaiseException	exception-handling	-	-	implicit	-	x	kernel32.dll
GetModuleFileNameA	dynamic-link-library	-	-	implicit	-	x	kernel32.dll
GetModuleFileNameW	dynamic-link-library	-	-	implicit	-	x	kernel32.dll
SetLastError	diagnostic	-	-	implicit	-	x	kernel32.dll
SetConsoleCtrlHandler	console	-	-	implicit	-	x	kernel32.dll
115 (WSAStartup)	network	-	-	implicit	x	x	ws2_32.dll
11 (inet_addr)	network	-	-	implicit	x	x	ws2_32.dll
9 (htons)	network	-	-	implicit	x	x	ws2_32.dll
116 (WSACleanup)	network	-	-	implicit	x	x	ws2_32.dll
4 (connect)	network	-	-	implicit	x	x	ws2_32.dll
23 (socket)	network	-	-	implicit	x	x	ws2_32.dll
3 (closesocket)	network	-	-	implicit	x	x	ws2_32.dll
51 (gethostbyaddr)	network	-	-	implicit	x	x	ws2_32.dll
52 (gethostbyname)	network	-	-	implicit	x	x	ws2_32.dll
19 (send)	network	-	-	implicit	x	x	ws2_32.dll
57 (gethostvalue)	network	-	-	implicit	x	x	ws2_32.dll
16 (recv)	network	-	-	implicit	x	x	ws2_32.dll
GetSystemInfo	system-information	-	-	implicit	-	-	kernel32.dll
GetComputerNameA	system-information	System Information ...	Discovery	implicit	-	-	kernel32.dll
IsDebuggerPresent	system-information	System Information ...	Discovery	implicit	-	-	kernel32.dll
QueryPerformanceCounter	system-information	-	-	implicit	-	-	kernel32.dll
GetTickCount	system-information	System Time Discov...	Discovery	implicit	-	-	kernel32.dll
EnumSystemLocalesA	system-information	-	-	implicit	-	-	kernel32.dll
GetUserNameA	system-information	-	-	implicit	-	-	advapi32.dll
InitializeCriticalSectionAndSpi...	synchronization	-	-	implicit	-	-	kernel32.dll

sha256: 4398615F0A498124B498ABB261150F11CE4778AFDE18D2E4E6717F2826A9D65D    cpu: 32-bit    file-type: executable    subsystem: GUI    entry-point: 0x0000A4E6    signature: ...

Figura 7.1.16: PE studio - Imports

Così come nel caso delle stringhe, tale screenshot non contiene tutte le funzioni che sono state reputate essere le più interessanti, tuttavia queste sono di seguito elencate e descritte per macrocategorie:

- *GetTimeZoneInformation*, *GetSystemTimeAsFileTime*, *GetTickCount*, *GetSystemInfo*, *GetComputerName*, *GetUserName*, *GetStartupInfo*, *GetLocaleInfo*, *EnumSystemLocales*, *QueryPerformanceCounter*: sono tutte relative all’ottenimento di informazioni dal sistema, come data, area geografica e nome dell’utente e del computer. Tali informazioni possono essere utilizzate da un software malevolo per trasmetterle all’attaccante.
- *IsDebuggerPresent*: un processo che chiama questa funzione può utilizzarla per capire se sta eseguendo all’interno di un debugger. Ciò potrebbe essere utilizzato da un malware per cambiare il proprio comportamento se scoprissse di essere sotto analisi.

- *MoveFile, WriteFile, CreateFile, ReadFile, GetFileType*: sono le funzioni utilizzate per la manipolazione dei file, che possono essere usate da un malware per replicarsi o per far danni sul sistema vittima.
- *RegOpenKey, RegQueryValue, RegCloseKey*: sono funzioni che permettono di leggere informazioni dal registro, per poterle magari inviare all'attaccante o per rendersi conto dell'ambiente su cui sta girando il software. Possono inoltre essere usate per impostare nuove chiavi nel registro, ad esempio al fine di rendere persistente l'infezione.
- *OpenProcess, CreateProcess, TerminateProcess*: sono funzioni utilizzate per fare operazioni sui processi, che possono essere utilizzate dai malware per creare processi in background al fine di portare a termine azioni malevoli.
- *CreateToolhelp32Snapshot, Process32First, Process32Next*: la prima funzione permette di ottenere uno snapshot di un gruppo di processi, più altre informazioni come l'heap, i moduli e i thread che questi utilizzano. Un malware potrebbe utilizzare tali informazioni per rendersi conto di quali processi sono attivi nel sistema ed agire di conseguenza. Le altre due funzioni servono rispettivamente per prendere il primo processo e per scorrere i processi in tali snapshot.
- *GetEnvironmentVariable, SetEnvironmentVariable*: servono per leggere le variabili d'ambiente e per settarne di nuove. Possono ad esempio essere utilizzate dai malware per bypassare l'UAC o per sfruttare il fatto che alcuni eseguibili "whitelisted" localizzano le DLL da cui dipendono tramite variabili d'ambiente, potendo così caricare una DLL malevola in modo da poter consentire l'esecuzione di codice malevolo con privilegi di amministratore in maniera silente.
- *ShellExecute*: utilizzata per eseguire vari tipi di operazioni su un file, come ad esempio operazioni di apertura, modifica o, se il file è un eseguibile, di esecuzione. Anche questa può essere utilizzata da un malware per fare operazioni su file e lanciare eseguibili.
- *LoadLibrary*: funzione utilizzata per caricare librerie a run-time. Essa può essere utilizzata da un malware per evitare di caricare le librerie all'atto del caricamento, allo scopo di eludere una

futura analisi. Infatti, tali librerie non compaiono all'interno della sezione relativa agli import, così da rendere trasparente alla fase di basic static analysis l'utilizzo di determinate funzioni.

- *Sleep*: funzione utilizzata per postporre determinate azioni nel tempo. Spesso è utilizzata dai malware come tecnica di evasione anti-sandboxing, in quanto spesso questo timer di sleep è settato ad un tempo che va oltre il timeout della maggior parte delle sandbox, che potrebbero non rilevare alcuna attività malevola.
- *inet\_addr, WSASStartup, WSACleanup, recv, send, socket, closesocket, connect, gethostbyaddr, gethostbyvalue, gethostvalue, htons*: funzioni relative all'utilizzo delle socket per le comunicazioni in rete. Possono essere usate da un malware, come già detto, per connettersi a un C&C o per scaricare ulteriori file malevoli.

Inoltre, vengono mostrati i campi relativi a *mitre tactics e techniques* associate alle varie funzioni.

- *Defense Evasion - Virtualization/Sandbox Evasion*: comprendenti funzioni spesso utilizzate per tecniche di evasione (Sleep).
- *Discovery - System Time Discovery, System Information Discovery, Process Discovery*: comprendenti funzioni spesso utilizzate per la scoperta di informazioni sul sistema (*GetTickCount, GetSystemTimeAsFileTime, GetComputerName, IsDebuggerPresent, Process32Next, CreateToolhelp32Snapshot, Process32First*).
- *Execution - Execution through API*: comprendenti funzioni che permettono di lanciare l'esecuzione di altri binari (*CreateProcess, ShellExecute, LoadLibrary*).

Si è poi passati all'analisi della sezione *.rsrc* relativa alle risorse. Nel file *audiiodq.exe* le risorse che sono state trovate non sono state di particolare interesse, mentre sono state trovate risorse molto più interessanti all'interno del file *places.docx.exe*.

La risorsa più interessante è certamente quella relativa all'icona utilizzata dal file, che è quella di Microsoft Office Word. In particolare, come è possibile vedere dalla figura 7.1.17, abbiamo diverse icone con varie risoluzioni e per diverse versioni del programma.

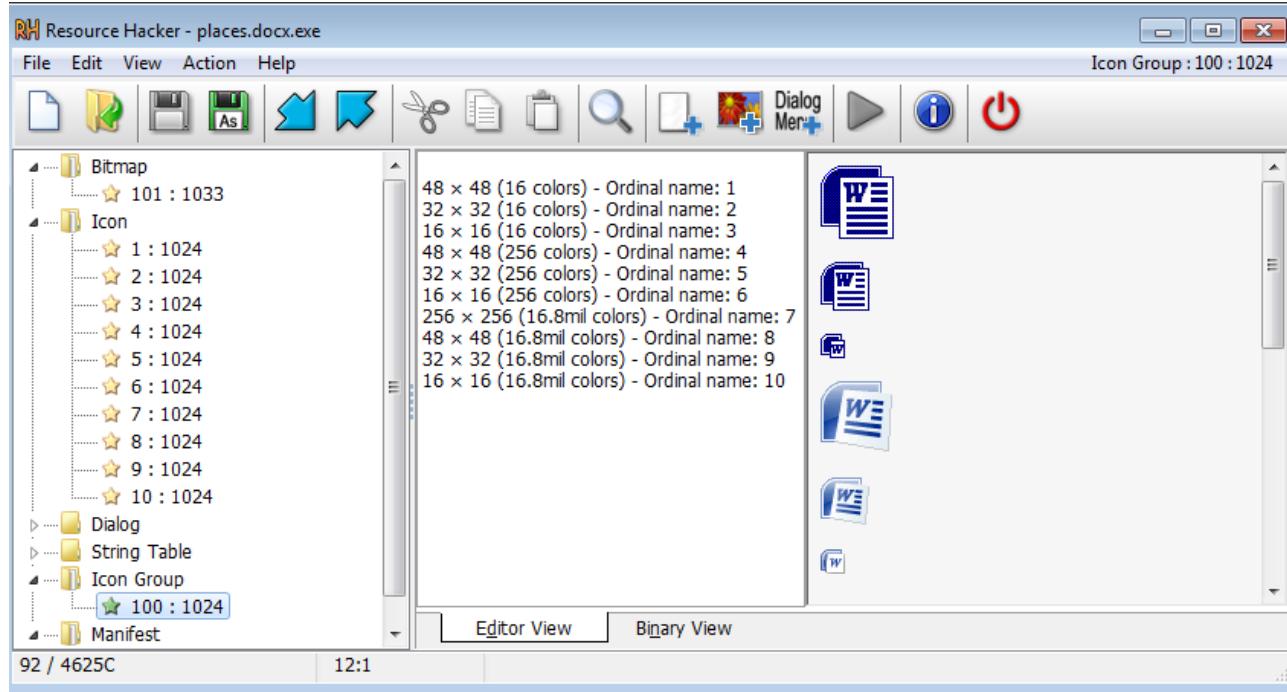


Figura 7.1.17: Resource Hacker - Icône

Un'altra risorsa interessante riguarda il manifest, che contiene metadati che sono però stati già ottenuti tramite l'analisi con i tool precedenti.

Le altre risorse contenute in questo file, come ad esempio i seguenti dialogs, sono relative alla procedura di estrazione guidata dell'archivio auto-estraente (figura 7.1.18):

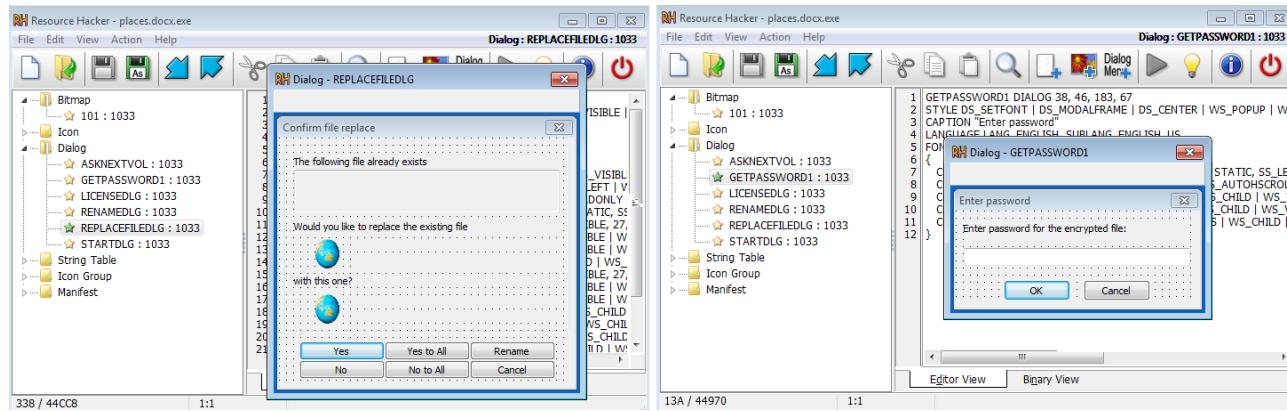


Figura 7.1.18: Resource Hacker - Dialogs

## 7.2 Basic Dynamic Analysis

Terminata la fase di basic static analysis, si è proceduto all'esecuzione del campione di malware, in modo da analizzarne il comportamento tramite l'utilizzo di vari strumenti.

Per prepararsi all'esecuzione del malware, sono state avviate le due VM connesse rete interna, e sono stati avviati i vari tool di monitoraggio. In particolare, sulla VM Ubuntu si è avviato Wireshark per il monitoraggio della rete, mentre su quella Windows sono stati avviati Process Hacker, Process Monitor e RegShot.

All'avvio del malware, come sospettato in precedenza, viene mostrato un documento Word contenente i migliori luoghi da visitare in Cina, in accordo con il nome *places.docx.exe* del binario (figura 7.2.1):

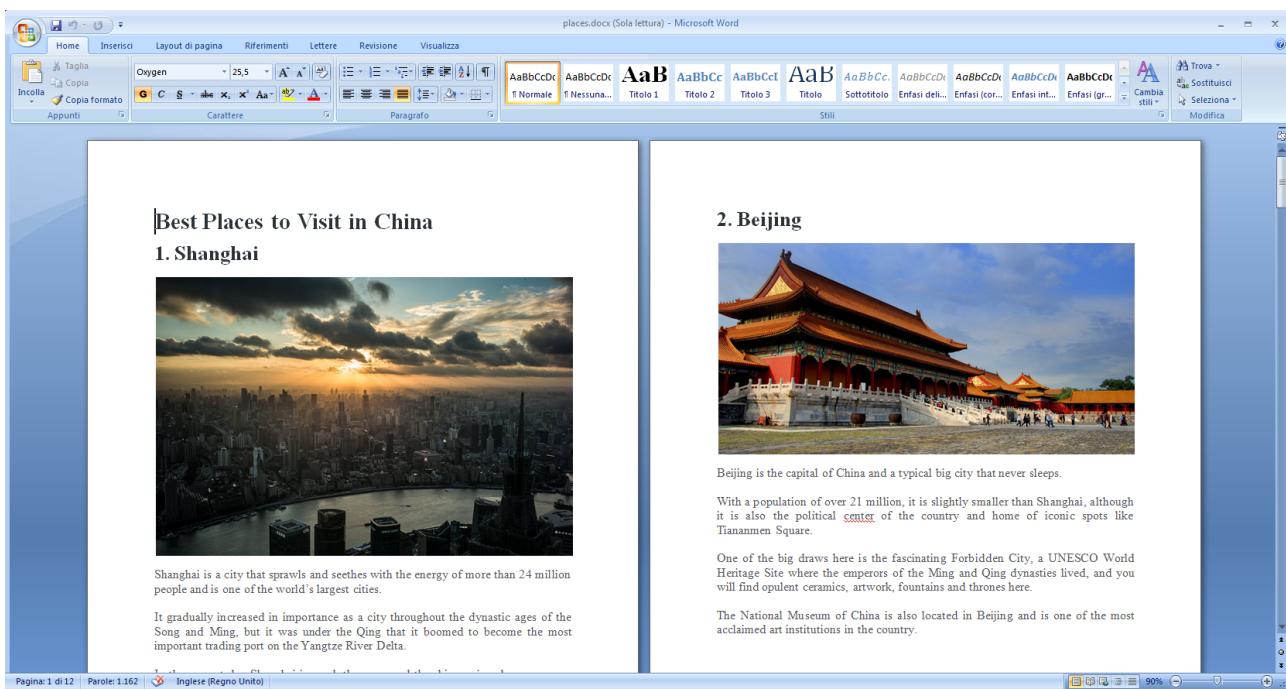


Figura 7.2.1: File docx mostrato dal malware

Tuttavia, vengono avviati anche altri processi in background, come evidenziato da Process Hacker nella figura 7.2.2:

explorer.exe	2156	2,86	2,93 kB/s	33,02 MB	IE10WIN7\IEUser	Windows Explorer	C:\Windows\Explorer.EXE
VBoxTray.exe	2268	0,04	64 B/s	1,34 MB	IE10WIN7\IEUser	VirtualBox Guest Additions Tr...	"C:\Windows\System32\VBox...
GrooveMonitor.exe	2276			1,56 MB	IE10WIN7\IEUser	GrooveMonitor Utility	"C:\Program Files\Microsoft ...
Regshot-x86-ANSI.exe	2840			55,05 MB	IE10WIN7\IEUser	Regshot 1.9.0 x86 ANSI	"C:\Users\IEUser\Desktop\Reg...
ProcessHacker.exe	2868	1,53		8,12 MB	IE10WIN7\IEUser	Process Hacker	"C:\Program Files\Process Ha...
Procmon.exe	2964	9,46	3,75 MB/s	14,76 MB	IE10WIN7\IEUser	Process Monitor	"C:\Users\IEUser\Desktop\Pro...
places.docx.exe	144			2,24 MB	IE10WIN7\IEUser		"C:\Users\IEUser\Desktop\pla...
audiodq.exe	3248			580 kB	IE10WIN7\IEUser	Windows Audio Device Graph...	"C:\intel\logs\audiodq.exe"
cmd.exe	3304			1,22 MB	IE10WIN7\IEUser	Windows Command Processor	"C:\Windows\system32\cmd....
SearchIndexer.exe	2528	0,11	3,04 kB/s	16,92 MB	NT AUTHORITY\SYSTEM	Microsoft Windows Search In...	C:\Windows\system32\Search...
WmiPrvSE.exe	3096			5,12 MB	NT AUTHORITY\SYSTEM	WMI Provider Host	C:\Windows\system32\wbem...
svchost.exe	3692			1,09 MB	NT A...\\LOCAL SERVICE	Host Process for Windows Ser...	C:\Windows\system32\svcho...
dllhost.exe	3204			1,02 MB	IE10WIN7\IEUser	COM Surrogate	C:\Windows\system32\DIHos...
conhost.exe	3312			632 kB	IE10WIN7\IEUser	Console Window Host	\??\C:\Windows\system32\co...



explorer.exe	2156	1,68	996 B/s	33,02 MB	IE10WIN7\IEUser	Windows Explorer	C:\Windows\Explorer.EXE
VBoxTray.exe	2268			1,34 MB	IE10WIN7\IEUser	VirtualBox Guest Additions Tr...	"C:\Windows\System32\VBox...
GrooveMonitor.exe	2276	0,02		1,56 MB	IE10WIN7\IEUser	GrooveMonitor Utility	"C:\Program Files\Microsoft ...
Regshot-x86-ANSI.exe	2840			55,05 MB	IE10WIN7\IEUser	Regshot 1.9.0 x86 ANSI	"C:\Users\IEUser\Desktop\Reg...
ProcessHacker.exe	2868	10,84	7,5 kB/s	8,41 MB	IE10WIN7\IEUser	Process Hacker	"C:\Program Files\Process Ha...
Procmon.exe	2964	24,86	8,16 MB/s	16,23 MB	IE10WIN7\IEUser	Process Monitor	"C:\Users\IEUser\Desktop\Pro...
places.docx.exe	144	4,49	35,29 kB/s	2,29 MB	IE10WIN7\IEUser		"C:\Users\IEUser\Desktop\pla...
audiodq.exe	3248			580 kB	IE10WIN7\IEUser	Windows Audio Device Graph...	"C:\intel\logs\audiodq.exe"
cmd.exe	3304	0,64	465 B/s	1,66 MB	IE10WIN7\IEUser	Windows Command Processor	"C:\Windows\system32\cmd....
WINWORD.EXE	3340			9,92 MB	IE10WIN7\IEUser	Microsoft Office Word	"C:\Program Files\Microsoft ...
SearchIndexer.exe	2528	0,03	712 B/s	16,92 MB	NT AUTHORITY\SYSTEM	Microsoft Windows Search In...	C:\Windows\system32\Search...
WmiPrvSE.exe	3096			5,12 MB	NT AUTHORITY\SYSTEM	WMI Provider Host	C:\Windows\system32\wbem...
svchost.exe	3692			1,09 MB	NT A...\\LOCAL SERVICE	Host Process for Windows Ser...	C:\Windows\system32\svcho...
dllhost.exe	3204			1,02 MB	IE10WIN7\IEUser	COM Surrogate	C:\Windows\system32\DIHos...
conhost.exe	3312	1,14	60 B/s	836 kB	IE10WIN7\IEUser	Console Window Host	\??\C:\Windows\system32\co...



explorer.exe	2156	5,30	5,68 kB/s	33,07 MB	IE10WIN7\IEUser	Windows Explorer	C:\Windows\Explorer.EXE
VBoxTray.exe	2268			1,34 MB	IE10WIN7\IEUser	VirtualBox Guest Additions Tr...	"C:\Windows\System32\VBox...
GrooveMonitor.exe	2276	0,92	1,46 kB/s	2,56 MB	IE10WIN7\IEUser	GrooveMonitor Utility	"C:\Program Files\Microsoft ...
Regshot-x86-ANSI.exe	2840			55,05 MB	IE10WIN7\IEUser	Regshot 1.9.0 x86 ANSI	"C:\Users\IEUser\Desktop\Reg...
ProcessHacker.exe	2868	3,04	544 B/s	8,42 MB	IE10WIN7\IEUser	Process Hacker	"C:\Program Files\Process Ha...
Procmon.exe	2964	11,00	6,21 MB/s	16,23 MB	IE10WIN7\IEUser	Process Monitor	"C:\Users\IEUser\Desktop\Pro...
places.docx.exe	144	4,49	35,29 kB/s	2,29 MB	IE10WIN7\IEUser		"C:\Users\IEUser\Desktop\pla...
audiodq.exe	3248			580 kB	IE10WIN7\IEUser	Windows Audio Device Graph...	"C:\intel\logs\audiodq.exe"
cmd.exe	3304			1,66 MB	IE10WIN7\IEUser	Windows Command Processor	"C:\Windows\system32\cmd....
WINWORD.EXE	3340	61,47	608,13 kB...	14,09 MB	IE10WIN7\IEUser	Microsoft Office Word	"C:\Program Files\Microsoft ...
SearchIndexer.exe	2528	0,78	11,27 kB/s	16,92 MB	NT AUTHORITY\SYSTEM	Microsoft Windows Search In...	C:\Windows\system32\Search...
WmiPrvSE.exe	3096			5,12 MB	NT AUTHORITY\SYSTEM	WMI Provider Host	C:\Windows\system32\wbem...
svchost.exe	3692			1,09 MB	NT A...\\LOCAL SERVICE	Host Process for Windows Ser...	C:\Windows\system32\svcho...
dllhost.exe	3204			1,02 MB	IE10WIN7\IEUser	COM Surrogate	C:\Windows\system32\DIHos...
conhost.exe	3312			836 kB	IE10WIN7\IEUser	Console Window Host	\??\C:\Windows\system32\co...

Figura 7.2.2: Process Hacker - Processi

L'immagine precedente rappresenta l'evoluzione nel tempo dei processi appena dopo l'avvio del malware. In verde vengono evidenziati i processi appena creati e in rosso quelli appena eliminati. Quindi, viene avviato il processo *places.docx.exe* che avvia il processo *audiodq.exe*. Quest'ultimo, a sua volta, avvia il processo *cmd.exe*. Come si può notare nel secondo passo, il processo *places.docx.exe*

avvia anche il processo *WINWORD.EXE* relativo a Microsoft Office Word. In seguito il processo *places.docx.exe* viene eliminato, lasciando attivo solo il processo *audiodq.exe*.

Le informazioni ottenute tramite Process Hacker sono state utilizzate per un filtraggio intelligente all'interno di Process Monitor. In particolare, è stato effettuato il filtraggio sui processi *places.docx.exe*, *audiodq.exe* e *WINWORD.EXE*, dapprima su operazioni relative ai processi creati e al file system, e poi su operazioni relative al registro di sistema.

Per quanto riguarda processi creati e file system, dall'enorme quantità di informazioni fornite da Process Monitor sono stati evidenziati alcuni degli eventi più interessanti.

Nella seguente immagine (figura 7.2.3) è possibile osservare come il processo *places.docx.exe* crei, sotto “*C:\*”, la cartella *intel* e poi la cartella *logs*, per poi creare al suo interno i files *audiodq.exe* e *places.docx*. Questa è probabilmente l'operazione di estrazione automatica dell'archivio auto-estraente, che utilizza come percorso di estrazione di default “*C:\intel\logs*”, che è un percorso che potrebbe non destare sospetti alla vittima. Si noti, inoltre, come il processo *places.docx.exe* vada a leggere alcune informazioni personali, come ad esempio i file temporanei e i cookies.

Process Name	PID	Operation	Path	Result	Detail	Command Line
W places.docx.exe	2900	ReadFile	C:\Windows\System32\riched20.dll	SUCCESS	Offset: 230.400, Le...,"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	CreateFile	C:\intel	SUCCESS	Desired Access: R...,"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	ReadFile	C:\\$Directory	SUCCESS	Offset: 94.208, Len...,"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	CloseFile	C:\intel	SUCCESS	"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	CreateFile	C:\intel	SUCCESS	Desired Access: W...,"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	CloseFile	C:\intel	SUCCESS	"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	CreateFile	C:\intel	SUCCESS	Desired Access: W...,"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	CloseFile	C:\intel	SUCCESS	"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	CreateFile	C:\intel\logs	SUCCESS	Desired Access: R...,"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	CloseFile	C:\intel\logs	SUCCESS	"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	CreateFile	C:\intel\logs	SUCCESS	Desired Access: W...,"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	CloseFile	C:\intel\logs	SUCCESS	"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	CreateFile	C:\intel\logs	SUCCESS	Desired Access: E...,"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	CloseFile	C:\Users\IEUser\Desktop\places	SUCCESS	"C:\Users\IEUser\Desktop\places\places.docx.exe"	
[...]						
W places.docx.exe	2900	ReadFile	C:\Users\IEUser\Desktop\places\places.docx.exe	SUCCESS	Offset: 298.196, Le...,"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	CreateFile	C:\intel\logs\audiodq.exe	NAME NOT FOUND	Desired Access: R...,"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	CreateFile	C:\intel\logs\audiodq.exe	NAME NOT FOUND	Desired Access: R...,"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	CreateFile	C:\intel\logs\audiodq.exe	SUCCESS	Desired Access: G...,"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	ReadFile	C:\Users\IEUser\Desktop\places\places.docx.exe	SUCCESS	Offset: 298.235, Le...,"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	ReadFile	C:\Users\IEUser\Desktop\places\places.docx.exe	SUCCESS	Offset: 331.003, Le...,"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	3132	CreateFile	C:\intel\logs\audiodq.exe	SUCCESS	Desired Access: Write Attributes, Synchron...,"C:\Users\IEUser\Desktop\place...	
W places.docx.exe	3132	CreateFile	C:\intel\logs\places.docx	NAME NOT FOUND	Desired Access: Read Attributes, Dispositio...,"C:\Users\IEUser\Desktop\place...	
W places.docx.exe	3132	CreateFile	C:\intel\logs\places.docx	NAME NOT FOUND	Desired Access: Read Attributes, Dispositio...,"C:\Users\IEUser\Desktop\place...	
W places.docx.exe	3132	CreateFile	C:\intel\logs\places.docx	SUCCESS	Desired Access: General Write, Read Attr...,"C:\Users\IEUser\Desktop\place...	
W places.docx.exe	3132	CreateFile	C:\intel\logs\places.docx	SUCCESS	Desired Access: Write Attributes, Synchron...,"C:\Users\IEUser\Desktop\place...	
[...]						
W places.docx.exe	2900	CreateFile	C:\Windows\System32\secur32.dll	SUCCESS	Desired Access: R...,"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	CloseFile	C:\Windows\System32\secur32.dll	SUCCESS	"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	CreateFile	C:\Users\IEUser\AppData\Local\Microsoft\Windows\Temporary Internet Files	SUCCESS	Desired Access: R...,"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	CloseFile	C:\Users\IEUser\AppData\Local\Microsoft\Windows\Temporary Internet Files	SUCCESS	"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	CreateFile	C:\Users\IEUser\AppData\Roaming\Microsoft\Windows\Cookies	SUCCESS	Desired Access: R...,"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	CloseFile	C:\Users\IEUser\AppData\Roaming\Microsoft\Windows\Cookies	SUCCESS	"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	CreateFile	C:\intel\logs\audiodq.exe	SUCCESS	Desired Access: R...,"C:\Users\IEUser\Desktop\places\places.docx.exe"	
W places.docx.exe	2900	CreateFile	C:\intel\logs\audiodq.exe	00000000	"C:\Users\IEUser\Desktop\places\places.docx.exe"	
[...]						

Figura 7.2.3: Process Monitor

Dalla seguente immagine (figura 7.2.4), invece, è possibile notare la creazione del processo *audiodq.exe* e anche che il processo *places.docx.exe* ricerchi i files *SwDRM.dll* e *speedfan.exe* nel percorso da lui creato in precedenza, senza riuscire a trovarli. Essi potrebbero essere creati in seguito all’interazione in rete avvenuta con successo, che purtroppo non è possibile osservare a causa della rete simulata. Vengono inoltre creati i processi *cmd.exe* e *WINWORD.EXE*.

			[...]	
places.docx.exe	2900	CreateFile	C:\intel\logs\audiodq.exe	SUCCESS
places.docx.exe	2900	CreateFile	C:\Windows\System32\en-US\setupapi.dll.mui	SUCCESS
places.docx.exe	2900	ReadFile	C:\Windows\System32\setupapi.dll	SUCCESS
places.docx.exe	2900	Process Create	C:\intel\logs\audiodq.exe	SUCCESS
audiodq.exe	2300	Process Start		SUCCESS
places.docx.exe	2900	CreateFile	C:\Windows\System32\apphelp.dll	SUCCESS
places.docx.exe	2900	CloseFile	C:\Windows\System32\apphelp.dll	SUCCESS
places.docx.exe	2900	CreateFile	C:\Windows\System32\apphelp.exe	SUCCESS
			[...]	
places.docx.exe	2900	HeadFile	C:\intel\logs\audiodq.exe	SUCCESS
places.docx.exe	2900	CloseFile	C:\intel\logs\audiodq.exe	SUCCESS
places.docx.exe	2900	CreateFile	C:\intel\logs\ui\SwDRM.dll	PATH NOT FOUND
places.docx.exe	2900	CreateFile	C:\Program Files\speedfan\speedfan.exe	PATH NOT FOUND
places.docx.exe	2900	CloseFile	C:\Windows\AppPatch\sysmain.sdb	SUCCESS
places.docx.exe	2900	CloseFile	C:\intel\logs\audiodq.exe	SUCCESS
places.docx.exe	2900	CreateFile	C:\intel\logs\places.docx	SUCCESS
			[...]	
places.docx.exe	2900	CloseFile	C:\Windows\System32\vrml.dll	SUCCESS
audiodq.exe	2300	CreateFile	C:\Windows\System32\cmd.exe	SUCCESS
audiodq.exe	2300	ReadFile	C:\Windows\System32\cmd.exe	SUCCESS
audiodq.exe	2300	Process Create	C:\Windows\System32\cmd.exe	SUCCESS
audiodq.exe	2300	CreateFile	C:\Windows\System32\apphelp.dll	SUCCESS
audiodq.exe	2300	CloseFile	C:\Windows\System32\apphelp.dll	SUCCESS
audiodq.exe	2300	CreateFile	C:\Windows\System32\annhttp.dll	SUCCESS
			[...]	
places.docx.exe	2900	CloseFile	C:\Program Files\Microsoft Office\Office12\WINWORD.EXE	SUCCESS
places.docx.exe	2900	CreateFile	C:\intel\logs	SUCCESS
places.docx.exe	2900	CloseFile	C:\intel\logs	SUCCESS
places.docx.exe	2900	CreateFile	C:\Program Files\Microsoft Office\Office12\WINWORD.EXE	SUCCESS
places.docx.exe	2900	Process Create	C:\Program Files\Microsoft Office\Office12\WINWORD.EXE	SUCCESS
WINWORD.EXE	3164	Process Start		SUCCESS
places.docx.exe	2900	CreateFile	C:\Windows\AppPatch\sysmain.sdb	SUCCESS

Figura 7.2.4: Process Monitor

Nel seguente screenshot (figura 7.2.5), invece, viene mostrato l’utilizzo delle librerie di sistema da parte del processo *audiodq.exe*. In particolare, vengono mostrate quelle utilizzate per le interazioni in rete, anche se in realtà sono moltissime le librerie utilizzate da tale processo, tra le quali sono probabilmente presenti le funzioni di libreria evidenziate nella fase di basic static analysis.

audiodq.exe	2300	CreateFile	C:\Windows\System32\mswsock.dll	SUCCESS	Desired Access: Read Attribute..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CloseFile	C:\Windows\System32\mswsock.dll	SUCCESS	"C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CreateFile	C:\Windows\System32\mswsock.dll	SUCCESS	Desired Access: Read Data/Li..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CloseFile	C:\Windows\System32\mswsock.dll	SUCCESS	"C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CreateFile	C:\Windows\System32\WSHTCPIP.DLL	SUCCESS	Desired Access: Read Attribute..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CloseFile	C:\Windows\System32\WSHTCPIP.DLL	SUCCESS	"C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CreateFile	C:\Windows\System32\WSHTCPIP.DLL	SUCCESS	Desired Access: Read Data/Li..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CloseFile	C:\Windows\System32\WSHTCPIP.DLL	SUCCESS	"C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CreateFile	C:\Windows\System32\vlaapi.dll	SUCCESS	Desired Access: Read Attribute..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CloseFile	C:\Windows\System32\vlaapi.dll	SUCCESS	"C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CreateFile	C:\Windows\System32\vlaapi.dll	SUCCESS	Desired Access: Read Data/Li..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CloseFile	C:\Windows\System32\vlaapi.dll	SUCCESS	"C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CreateFile	C:\Windows\System32\vlaapi.dll	SUCCESS	Desired Access: Read Attribute..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CloseFile	C:\Windows\System32\vlaapi.dll	SUCCESS	"C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CreateFile	C:\Windows\System32\NapiNSP.dll	SUCCESS	Desired Access: Read Attribute..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CloseFile	C:\Windows\System32\NapiNSP.dll	SUCCESS	"C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CreateFile	C:\Windows\System32\NapiNSP.dll	SUCCESS	Desired Access: Read Data/Li..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CloseFile	C:\Windows\System32\NapiNSP.dll	SUCCESS	"C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CreateFile	C:\Windows\System32\pnprnsp.dll	SUCCESS	Desired Access: Read Attribute..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CloseFile	C:\Windows\System32\pnprnsp.dll	SUCCESS	"C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CreateFile	C:\Windows\System32\pnprnsp.dll	SUCCESS	Desired Access: Read Data/Li..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CloseFile	C:\Windows\System32\pnprnsp.dll	SUCCESS	"C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CreateFile	C:\Windows\System32\DNSAPI.dll	NAME NOT FOUND	Desired Access: Read Attribute..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CreateFile	C:\Windows\System32\dnsapi.dll	SUCCESS	Desired Access: Read Attribute..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CloseFile	C:\Windows\System32\dnsapi.dll	SUCCESS	"C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CreateFile	C:\Windows\System32\dnsapi.dll	SUCCESS	Desired Access: Read Data/Li..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CloseFile	C:\Windows\System32\dnsapi.dll	SUCCESS	"C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CreateFile	C:\Windows\System32\winmr.dll	SUCCESS	Desired Access: Read Attribute..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CloseFile	C:\Windows\System32\winmr.dll	SUCCESS	"C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CreateFile	C:\Windows\System32\winmr.dll	SUCCESS	Desired Access: Read Data/Li..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CloseFile	C:\Windows\System32\winmr.dll	SUCCESS	"C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CreateFile	C:\Windows\System32\wshbth.dll	SUCCESS	Desired Access: Read Attribute..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CloseFile	C:\Windows\System32\wshbth.dll	SUCCESS	"C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CreateFile	C:\Windows\System32\wshbth.dll	SUCCESS	Desired Access: Read Data/Li..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CloseFile	C:\Windows\System32\wshbth.dll	SUCCESS	"C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CreateFile	C:\Windows\System32\IPHLPAPI.DLL	NAME NOT FOUND	Desired Access: Read Attribute..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CloseFile	C:\Windows\System32\IPHLPAPI.DLL	SUCCESS	Desired Access: Read Attribute..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CreateFile	C:\Windows\System32\IPHLPAPI.DLL	SUCCESS	"C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CloseFile	C:\Windows\System32\IPHLPAPI.DLL	SUCCESS	Desired Access: Read Data/Li..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CreateFile	C:\Windows\System32\IPHLPAPI.DLL	SUCCESS	"C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CloseFile	C:\Windows\System32\IPHLPAPI.DLL	SUCCESS	Desired Access: Read Attribute..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	CreateFile	C:\intel\Logs\WINNSI.DLL	NAME NOT FOUND	Desired Access: Read Attribute..."C:\Intel\Logs\audiodq.exe"

Figura 7.2.5: Process Monitor

Per quanto riguarda le operazioni sul registro di sistema, tra le tante informazioni è stata evidenziata quella relativa alla persistenza, come mostrato nel seguente screenshot (figura 7.2.6):

places.docx.exe	2900	RegQueryValue	HKL\Software\Microsoft\Windows\CurrentVersion\Run\vgfmsw	SUCCESS	Type: REG_DWORD, Length: ... "C:\Users\lEUser\Desktop\places\places.docx.exe"
audiodq.exe	2300	RegOpenKey	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run	SUCCESS	Desired Access: Maximum Alloc..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	RegOpenKey	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\vgfmsw	SUCCESS	Desired Access: All Access "C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	RegOpenKey	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\vgfmsw	NAME NOT FOUND	Length: 144 "C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	RegOpenKey	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\vgfmsw	NAME NOT FOUND	Desired Access: Query Value..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	RegOpenKey	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\vgfmsw	REPARSE	Desired Access: Query Value..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	RegOpenKey	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\vgfmsw	NAME NOT FOUND	Desired Access: Query Value..."C:\Intel\Logs\audiodq.exe"
audiodq.exe	2300	RegOpenKey	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\vgfmsw	REPARSE	Desired Access: Query Value..."C:\Intel\Logs\audiodq.exe"

Figura 7.2.6: Process Monitor

Per monitorare le operazioni sul registro è stato utilizzato anche il tool RegShot, che si presenta in questo modo (figura 7.2.7):

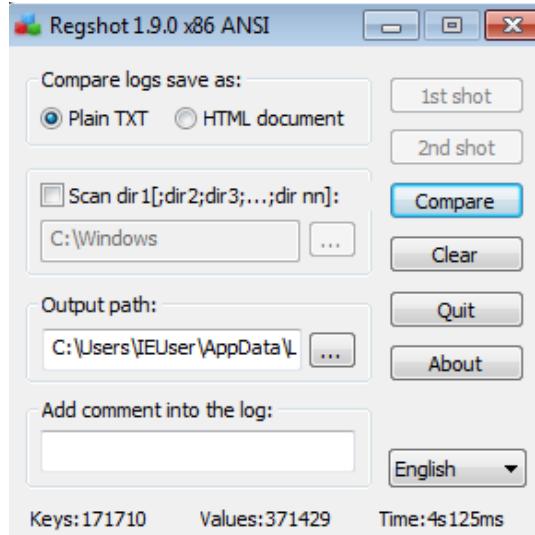


Figura 7.2.7: RegShot

È stato fatto il primo shot prima dell'esecuzione del malware, mentre il secondo è stato fatto a valle della sua esecuzione. Cliccando il tasto *Compare* è stato generato un report contenente tutte le differenze all'interno del registro. Il seguente screenshot (figura 7.2.8) mostra, ancora una volta, la chiave utilizzata per la persistenza:

```
C:\Users\IEUser\Desktop\regshot.txt - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
File Open Save Save As Find Replace Go Tools Plugins Window Help
regshot.txt
55 HKU\S-1-5-21-1716914095-909560446-1177810406-1000_Classes\Local Settings\Software\Microsoft\Windows\Shell\MuiCache\C:\Users\IEUser\Downl...
56 HKU\S-1-5-21-1716914095-909560446-1177810406-1000_Classes\Local Settings\Software\Microsoft\Windows\Shell\MuiCache\C:\Users\IEUser\Desktop...
57 HKU\S-1-5-21-1716914095-909560446-1177810406-1000_Classes\Local Settings\Software\Microsoft\Windows\Shell\MuiCache\C:\Program Files\NTCo...
58
59 -----
60 Values added: 50
61 -----
62 HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Installer\UserData\S-1-5-18\Products\00002109F1007040000000000F01FEC\Usage\SpellingAndGra...
63 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Office\12.0\Common\General\OfficeMenuDiscovered: 0x00000001
64 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Office\12.0\Common\ReviewCycle\ReviewToken: "(D09DCE0B-D94A-43F7-9F9...
65 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Office\12.0\Word\MTT: OC 0D 00 10 35 EO 29 86 DA D5 01 00 00 0C
66 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Office\12.0\Word\File MRU\Item 1: "[F00000002]\{T01D5DA862A84B270]*C...
67 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Shared Tools\Proofing Tools\1.0\Custom Dictionaries\1: "CUSTOM.DIC"
68 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Shared Tools\Proofing Tools\1.0\Custom Dictionaries\UpdateComplete:
69 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\.docx\OpenWithList\@: "WINW...
70 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\.docx\OpenWithList\MRUList:
71 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\DECache\WinWord\System\Prc...
72 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\DECache\WinWord\System\Win...
73 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\DECache\WinWord\System\Win...
74 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBF5SCD-ACE2-4F4F-9178-...
75 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\Extensible Cache\
76 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\Extensible Cache\
77 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\Extensible Cache\
78 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\Extensible Cache\
79 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\Extensible Cache\
80 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Run\igfxmssw: "C:\intel\logs\audiocdq.exe"
81 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\IMEMIP\0x0410\Input: "Plain,Dotted,AppText,AppWindow,AppText"
82 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\IMEMIP\0x0410\TargetConverted: "Plain,Single,White,DkBlue,DkBlue"
83 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\IMEMIP\0x0410\Converted: "Plain,Dotted,AppText,AppWindow,AppText"
84 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\IMEMIP\0x0410\TargetNotConverted: "Plain,Single,DkBlue,AppWindow,App...
85 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\IMEMIP\0x0410\InputError: "Plain,Thick,AppText,AppWindow,Red"
86 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Microsoft\IMEMIP\0x0410\FixedConverted: "Plain,AppText,AppWindow,AppText"
87 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\BagMRU\11: 50 00 31 0
88 HKU\S-1-5-21-1716914095-909560446-1177810406-1000\Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\BagMRU\11\NodeSlot: 0x -
```

Figura 7.2.8: RegShot - Risultato comparazione

A questo punto è possibile analizzare le attività di rete. Per prima cosa si è utilizzato ancora una volta il tool Process Hacker, che mette in evidenza come il processo *audiodq.exe* faccia partire una connessione TCP verso la porta 80 e poi la chiuda. Ciò avviene ad intervalli regolari, probabilmente a causa del fatto che non ottiene la risposta che si aspetta, ma una risposta di default fornita da INetSim. Nella figura 7.2.9 è mostrata l’evoluzione temporale di quanto detto, con la connessione che si apre (verde) e si chiude (rosso):

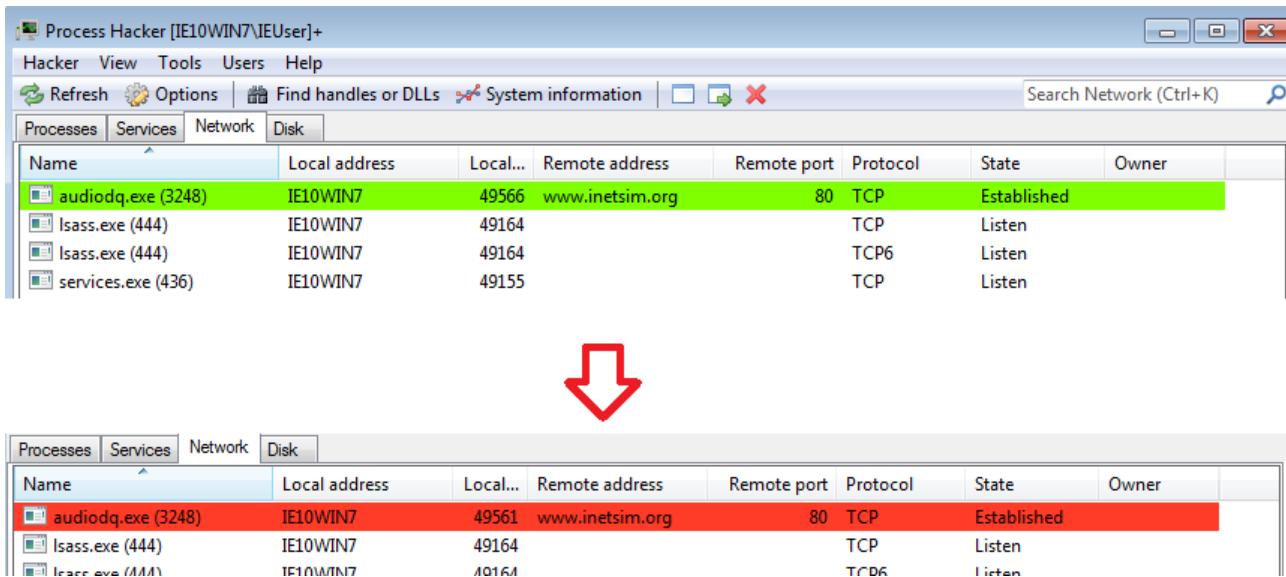


Figura 7.2.9: Process Hacker - Attività di rete

Per analizzare più in dettaglio il traffico di rete è stato utilizzato lo strumento Wireshark. La prima informazione importante ricavata in Wireshark è la richiesta DNS per risolvere l’URL “*healthnewsone.com*”, dove *healthnewsone* è una parola identificata all’interno di un path trovato nell’analisi statica delle stringhe del file. Nella seguente immagine (figura 7.2.10) è mostrata la richiesta con la relativa risposta, dove viene risolto tale URL con l’IP della VM Ubuntu sulla quale è in esecuzione INetSim:

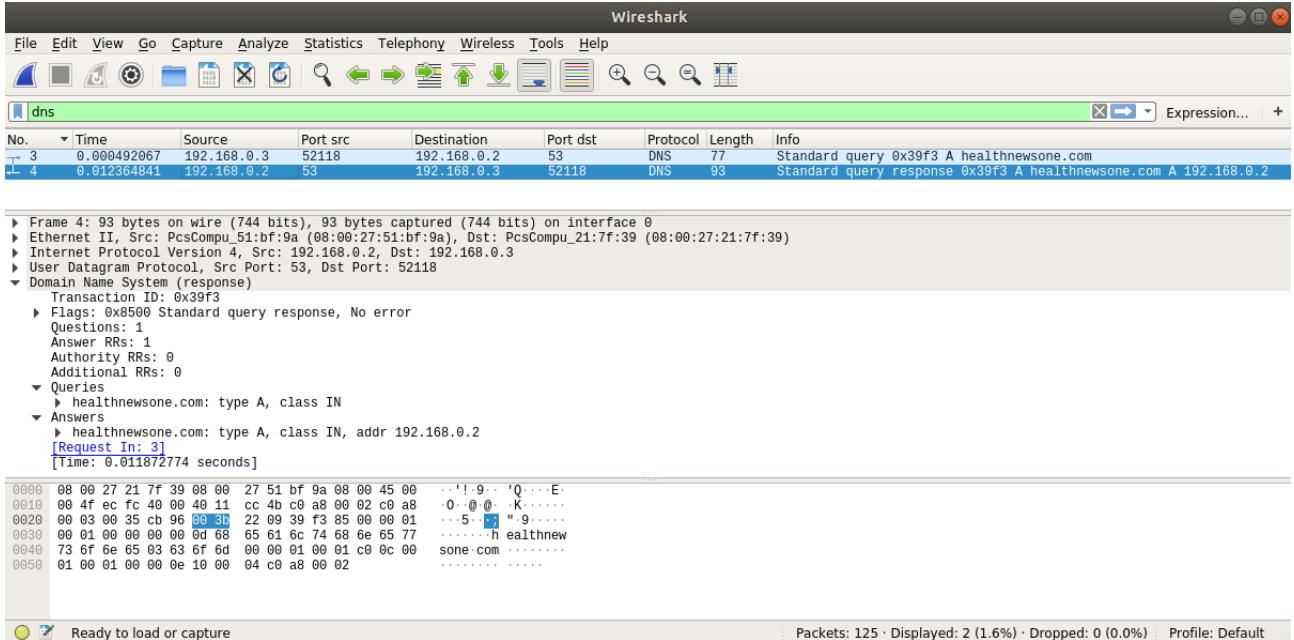


Figura 7.2.10: Wireshark - Risoluzione DNS

L'altra informazione importante è quella relativa alla richiesta HTTP GET, dove viene richiesta una pagina *accept.php* per la quale si riceve come risposta un 200 OK dalla VM Ubuntu, che fornisce la pagina di default. In particolare, nella richiesta è possibile notare come nella query string vengano passati dei parametri contenenti informazioni riguardanti la macchina della vittima, come il nome del PC, la versione di Windows e il nome utente. Ciò è visibile nella figura 7.2.11:

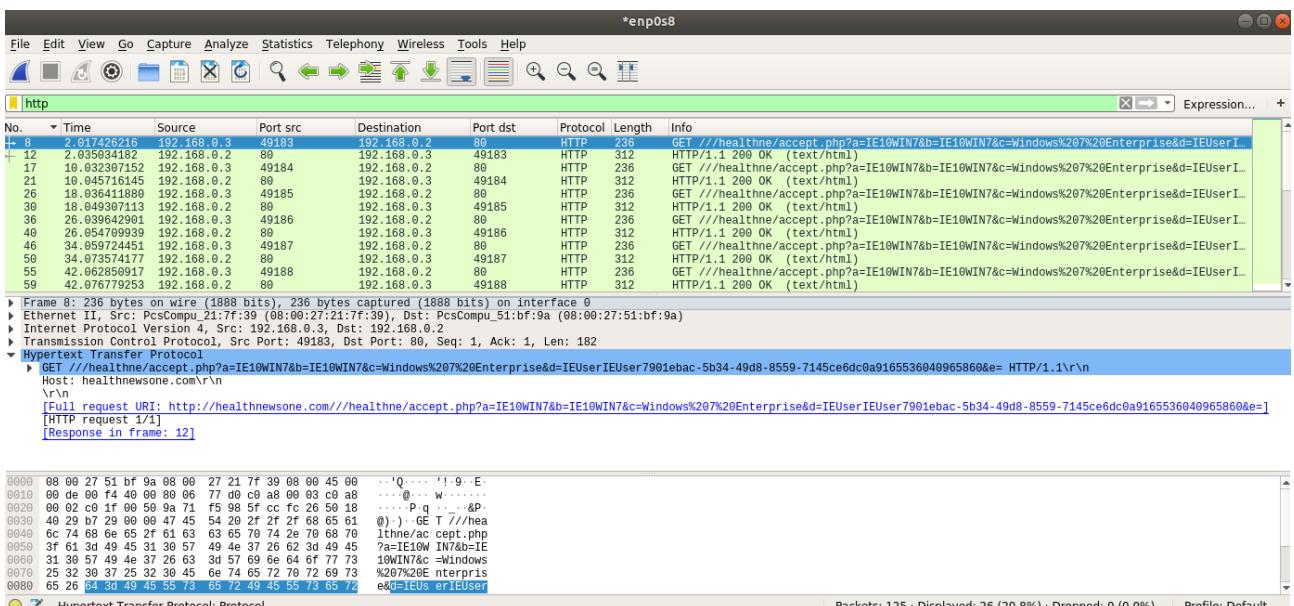


Figura 7.2.11: Wireshark - Richieste HTTP

## 7.3 Esempio di regola YARA

Per mostrare brevemente il funzionamento delle regole YARA per la rilevazione di file malevoli, è stata scritta una semplicissima regola (figura 7.3.1) basata sulle informazioni ricavate dalla basic static analysis:

```
rule places_malware
{
    meta:
        author = "Emanuele Galdi, Antonio Forte, Fabio Maresca"
        description = "Questa semplice regola serve ad identificare il malware dell'esempio"

    strings:
        $a = {4D 5A}
        $b = "audiodq.exe" wide
        $c = "IsDebuggerPresent"
        $d = "Sleep"

    condition:
        $a at 0x0 and $b and $c and $d
}
```

Figura 7.3.1: Regola YARA

La prima stringa è relativa alla signature distintiva dei file in formato PE in esadecimale, mentre le altre 3 sono state trovate durante la fase di estrazione delle stringhe. Si noti che per la stringa *audiodq.exe* si è dovuto utilizzare la keyword *wide* in quanto la stringa da ricercare è in formato Unicode. Nella condition è stato utilizzato il costrutto *at* per far sì che la condizione sulla signature sia verificata solo se essa si trova all'inizio del file, ossia con offset nullo (0x0). Le 4 stringhe sono state poste in *and* tra loro, in quanto devono essere verificate tutte affinché il file sia rilevato.

Tale regola Yara è stata utilizzata per verificare che all'interno del percorso *C:\intel\logs* ci sia effettivamente il file malevolo cercato. Per utilizzarla è stato utilizzato il seguente comando:

```
yara32 -s -r yara_rule_malware.yara C:\intel\logs
```

Dove:

- *yara32* è l'eseguibile di yara per Windows
- *-s* è l'opzione utilizzata per mostrare le stringhe rilevate

- `-r` è l'opzione per effettuare una ricerca ricorsiva nelle cartelle
- `yara_rule_malware.yara` è il nome della regola
- `C:\intel\logs` è il path dal quale far partire la verifica

Il risultato del suo utilizzo è rappresentato in figura 7.3.2:

```
C:\Users\IEUser\Desktop>yara32 -s -r yara_rule_malware.yara C:\intel\logs
places_malware C:\intel\logs\audiiodq.exe
0x0:$a: 4D 5A
0x2f410:$b: a\x00u\x00d\x00i\x00o\x00d\x00q\x00.\x00e\x00x\x00e\x00
0x2f4c8:$b: a\x00u\x00d\x00i\x00o\x00d\x00q\x00.\x00e\x00x\x00e\x00
0x2d2e2:$c: IsDebuggerPresent
0x2d056:$d: Sleep
```

Figura 7.3.2: Esecuzione regola YARA

Il file `audiiodq.exe` ha matchato la regola in quanto conteneva tutte le stringhe secondo quanto specificato dalla condition.

## 7.4 Automatic Malware Analysis tramite Sandbox

In questo paragrafo analizzeremo il file `places.docx.exe` utilizzando alcune tra le sandbox più utilizzate.

### 7.4.1 Joe Sandbox

La pagina principale di Joe Sandbox Cloud Basic si presenta come in figura 7.4.1:

Figura 7.4.1: Joe Sandbox - Pagina principale

In questo caso è stata utilizzata la versione Basic, in quanto essa è risultata più che sufficiente per i nostri scopi. Sono stati selezionati i parametri da utilizzare per l'analisi, tra cui il tipo di interazione in rete e il sistema operativo, come mostrato in figura 7.4.2:

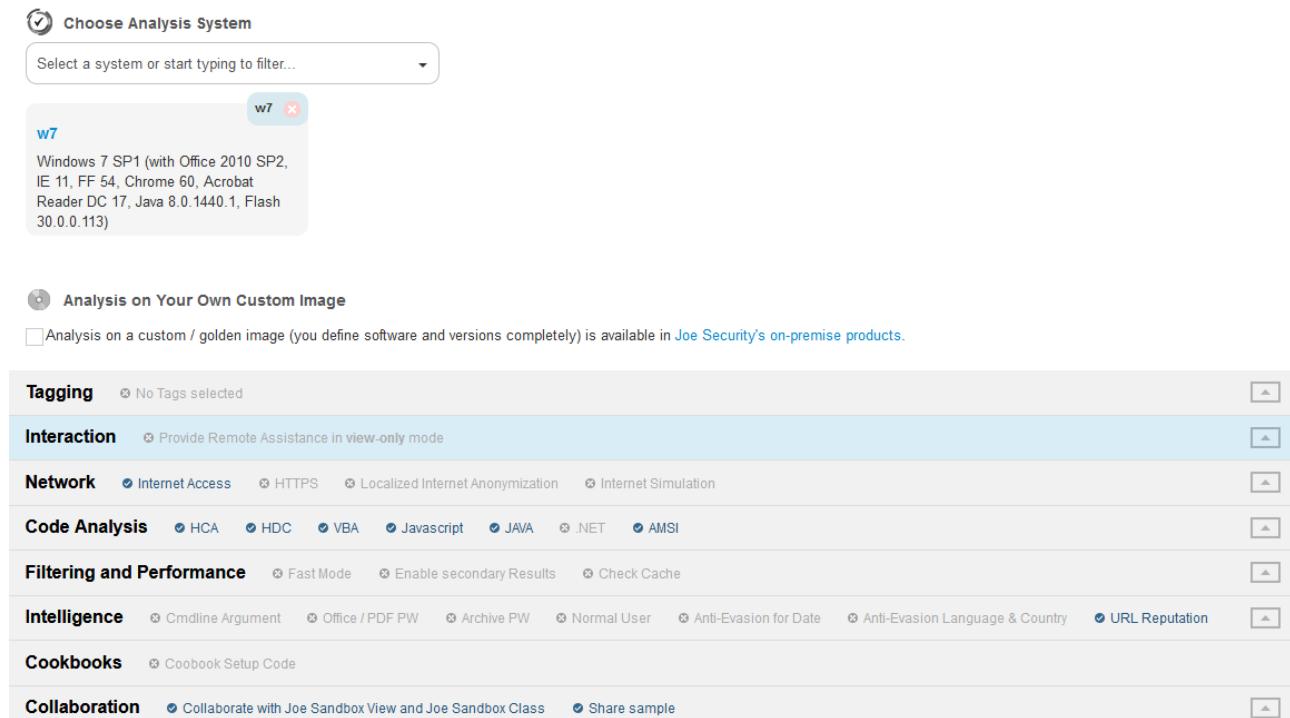


Figura 7.4.2: Joe Sandbox - Parametri per l'analisi

Una volta terminata l'analisi, viene presentata a video la schermata di overview contenente una panoramica sul file analizzato. In particolare, nella seguente immagine (7.4.3) viene per prima cosa mostrato il risultato generale, ossia il fatto che il file sia targato come malevolo e sia riconosciuto come un particolare tipo di Trojan chiamato *Artra*. Vengono mostrati inoltre i verdetti di JoeSandbox, VirusTotal, Metadefender e Avira, tutti concordanti sul fatto che il file sia malevolo.

Engine	Info	Verdict	Score	Reports
Joe Sandbox	System: Windows 7 SP1 (with Office 2010 SP2, IE 11, FF 54, Chrome 60, ...)	MALICIOUS	100/100	<a href="#">View Report</a>
VirusTotal		MALICIOUS	54/73	
OPSWAT Metadefender		MALICIOUS	19/35	
Avira		MALICIOUS		

Figura 7.4.3: Joe Sandbox - Panoramica

È inoltre possibile visualizzare una panoramica sugli IOCs, come ad esempio gli IP e i domini contattati, gli URL e i files droppati. Come è possibile osservare nelle figure 7.4.4 e 7.4.5, essi sono gli stessi rilevati durante l'analisi manuale, fatta eccezione per l'indirizzo IP che è stato possibile risolvere contattando un vero servizio DNS, a differenza del DNS simulato offerto da INetSim:

Engines	IOCs	
IPs		
IP	Country	Detection
45.35.182.76 	United States	
Domains		
Name	IP	Detection
www.healthnewsone.com 	45.35.182.76 	
healthnewsone.com 	0.0.0.0 	

Figura 7.4.4: Joe Sandbox - IOCs

URLs			
Name	Detection		
http://healthnewsone.com//healthne/accept.php?a=688098&b=688098&c=Windows%207%20Professional&d=useruser0f4f5130-48fa-4204-b1c4-585fb81cd251565536040965860&e= 			
http://healthnewsone.com/healthne/healthne/http://healthnewsone.com/healthne/healthne/ 			
http://healthnewsone.com/healthne/healthne/ 			
Dropped files			
Name	File Type	Hashes	Detection
C:\intel\logs\audiodq.exe 	PE32 executable (GUI) Intel 80386, for MS Windows 	#	
C:\intel\logs\places.docx 	Microsoft Word 2007+ 	#	
C:\Users\user\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.MSO18220431.jpeg 	"cmp3.10.3.2Lq4 0xf4621759", baseline, precision 8, 1024x633, frames 3 	#	

Figura 7.4.5: Joe Sandbox - IOCs

Oltre a questa panoramica generale, JoeSandbox mette a disposizione un report dettagliato. Tale report si presenta fornendo per prima cosa le informazioni di base già viste, per poi proseguire con un grafico che classifica il malware in base alla sua tipologia (figura 7.4.6):

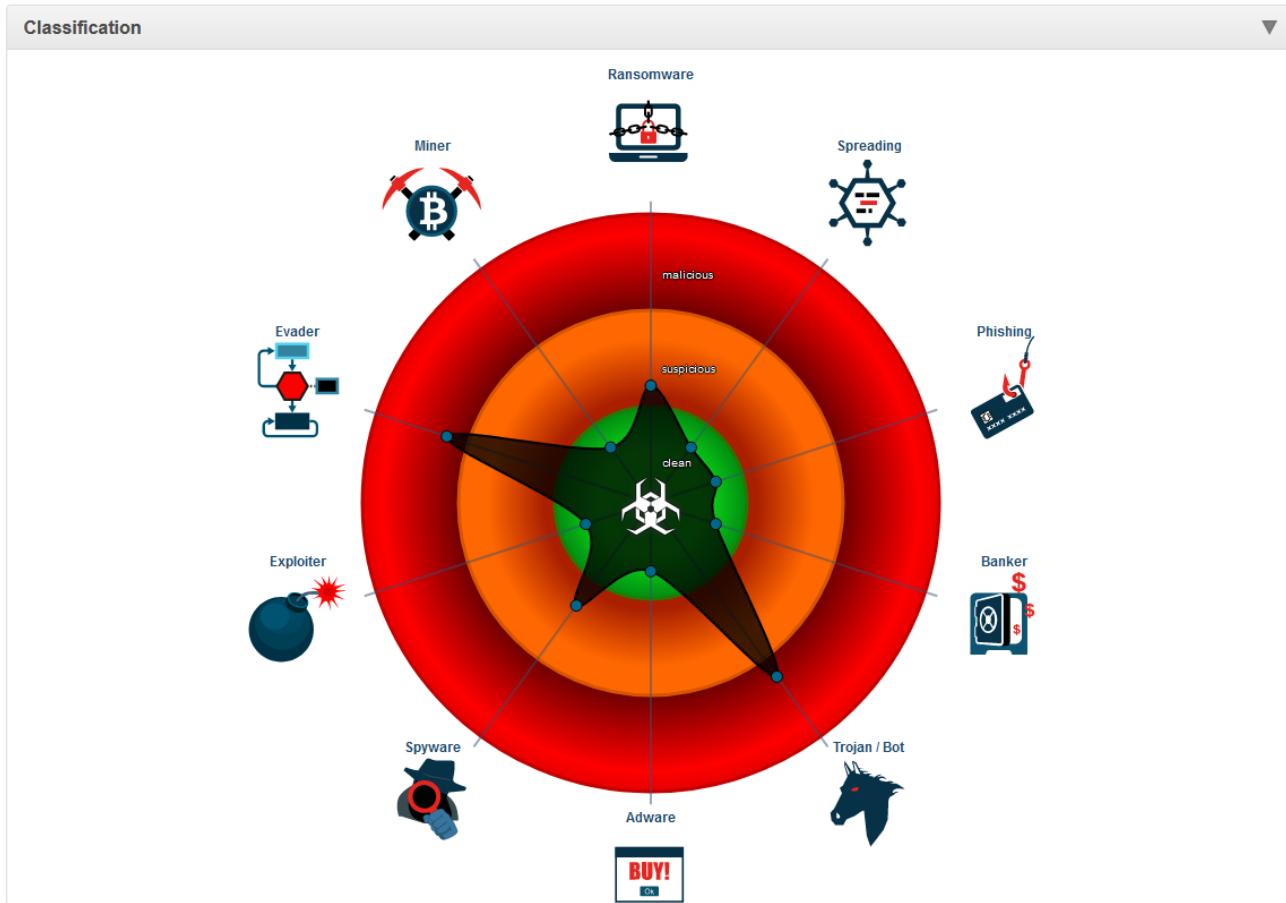


Figura 7.4.6: Joe Sandbox - Grafico di classificazione

Da tale immagine è possibile riconoscere come esso sia classificato come Trojan ed Evader, dove per Evader si intende un malware che mette in campo tecniche di evasione. Esso è sospettato essere anche in parte uno Spyware (in quanto legge informazioni sensibili) e un Ransomware (a causa dell'utilizzo di funzioni legate alla crittografia).

Vengono poi mostrati gli indicatori di compromissione (IOCs) in dettaglio, marcati con colori differenti e divisi per categorie. Più il colore è vicino al rosso e più l'indicatore di compromissione è degno di attenzione.

Nella seguente immagine (figura 7.4.7) sono rappresentati in rosso gli IOCs relativi al rilevamento tramite altri motori antivirus:

AV Detection:	
Antivirus detection for dropped file	Show sources
Antivirus detection for sample	Show sources
Multi AV Scanner detection for dropped file	Show sources
Multi AV Scanner detection for submitted file	Show sources

Figura 7.4.7: Joe Sandbox - IOCs AV Detection

Nella seguente immagine (figura 7.4.8) sono rappresentati gli IOCs relativi alle attività di rete. In questo caso l'indicatore più preoccupante è relativo ad una quantità di traffico anomalo segnalato tramite l'IDS Snort, in quanto vengono fatte tante richieste http una dietro l'altra. Vengono inoltre evidenziate in arancione alcune anomalie relative alla well-formedness dei messaggi http e alla compromissione dell'IP contattato. Vengono infine mostrati indicatori in azzurro relativi alla capacità di scaricare files da Internet e alle risoluzioni via DNS:

Networking:	
Snort IDS alert for network traffic (e.g. based on Emerging Threat rules)	Show sources
HTTP GET or POST without a user agent	Show sources
Internet Provider seen in connection with other malware	Show sources
Contains functionality to download additional files from the internet	Show sources
Downloads files	Show sources
Downloads files from webservers via HTTP	Show sources
Performs DNS lookups	Hide sources
Source: unknown	DNS traffic detected: queries for: healthnewsone.com
Tries to download or post to a non-existing http route (HTTP/1.1 404 Not Found / 503 Service Unavailable)	Show sources
Urls found in memory or binary data	Show sources

Figura 7.4.8: Joe Sandbox - IOCs Networking

Nell'immagine 7.4.9 invece sono mostrati alcuni IOC relativi alle attività sul sistema vittima. In particolare, viene rappresentato in rosso il fatto che il campione inviato per l'analisi è già noto e classificato come malware. In arancione, invece, vengono mostrati vari indicatori sospetti relativi ad esempio ai files droppati, alle risorse contenute nell'eseguibile ed i match con delle regole YARA. Infine in azzurro e in verde ci sono indicatori relativi alla lettura di varie informazioni dal sistema (le

quali potrebbero essere esfiltrate) e la creazione di files al suo interno, oltre che indicatori relativi al fatto che viene controllato se è presente Microsoft Office, che sono presenti delle modifiche alle GUI e che il file è più grande dei file solitamente identificati come malware (a causa del fatto che contiene un documento in formato .docx al suo interno).

**System Summary:**

<b>Submitted sample is a known malware sample</b>	<a href="#">Show sources</a>
Contains functionality to communicate with device drivers	<a href="#">Show sources</a>
Detected potential crypto function	<a href="#">Show sources</a>
Dropped file seen in connection with other malware	<a href="#">Show sources</a>
Found potential string decryption / allocating functions	<a href="#">Show sources</a>
PE file contains strange resources	<a href="#">Show sources</a>
Sample file is different than original file name gathered from version info	<a href="#">Show sources</a>
Tries to load missing DLLs	<a href="#">Show sources</a>
Yara signature match	<a href="#">Show sources</a>
Classification label	<a href="#">Show sources</a>
Contains functionality to enum processes or threads	<a href="#">Show sources</a>
Creates files inside the user directory	<a href="#">Show sources</a>
Creates temporary files	<a href="#">Show sources</a>
Might use command line arguments	<a href="#">Show sources</a>
PE file has an executable .text section and no other executable section	<a href="#">Show sources</a>
Reads ini files	<a href="#">Show sources</a>
Reads software policies	<a href="#">Show sources</a>
Reads the hosts file	<a href="#">Show sources</a>
Sample is known by Antivirus	<a href="#">Show sources</a>
Sample reads its own file content	<a href="#">Show sources</a>
Spawns processes	<a href="#">Show sources</a>
Uses an in-process (OLE) Automation server	<a href="#">Show sources</a>
Found graphical window changes (likely an installer)	<a href="#">Show sources</a>
Checks if Microsoft Office is installed	<a href="#">Show sources</a>
Submission file is bigger than most known malware samples	<a href="#">Show sources</a>

Figura 7.4.9: Joe Sandbox - IOCs System

Gli indicatori di maggiore interesse sono anche rappresentati sotto forma di grafo interattivo, che mostra le azioni più importanti compiute dai vari processi (figura 7.4.10):

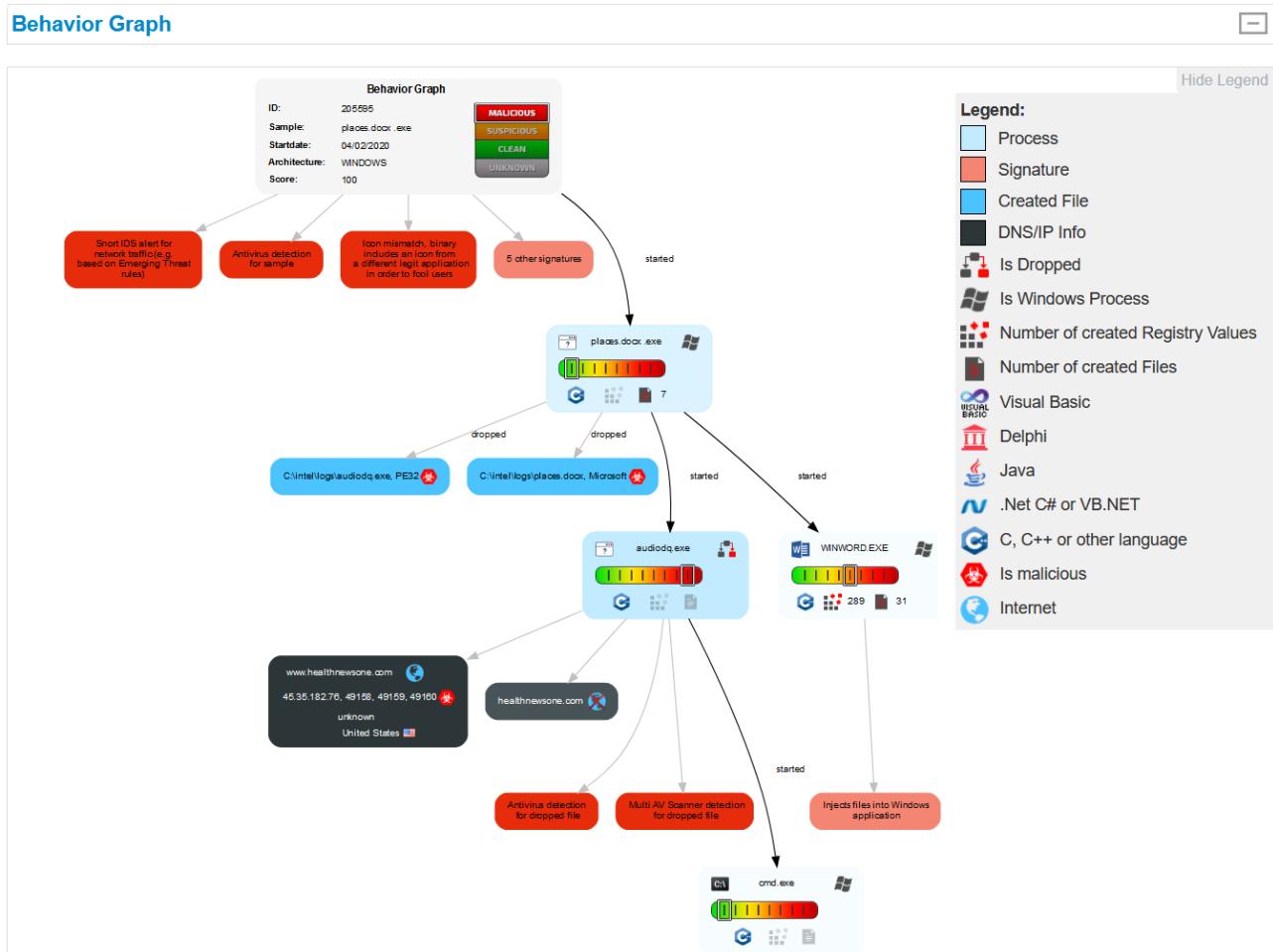


Figura 7.4.10: Joe Sandbox - Behavior Graph

È possibile visualizzare anche gli screenshot catturati sulla macchina durante l'esecuzione del malware, assieme ad un'animazione temporizzata (figura 7.4.11):

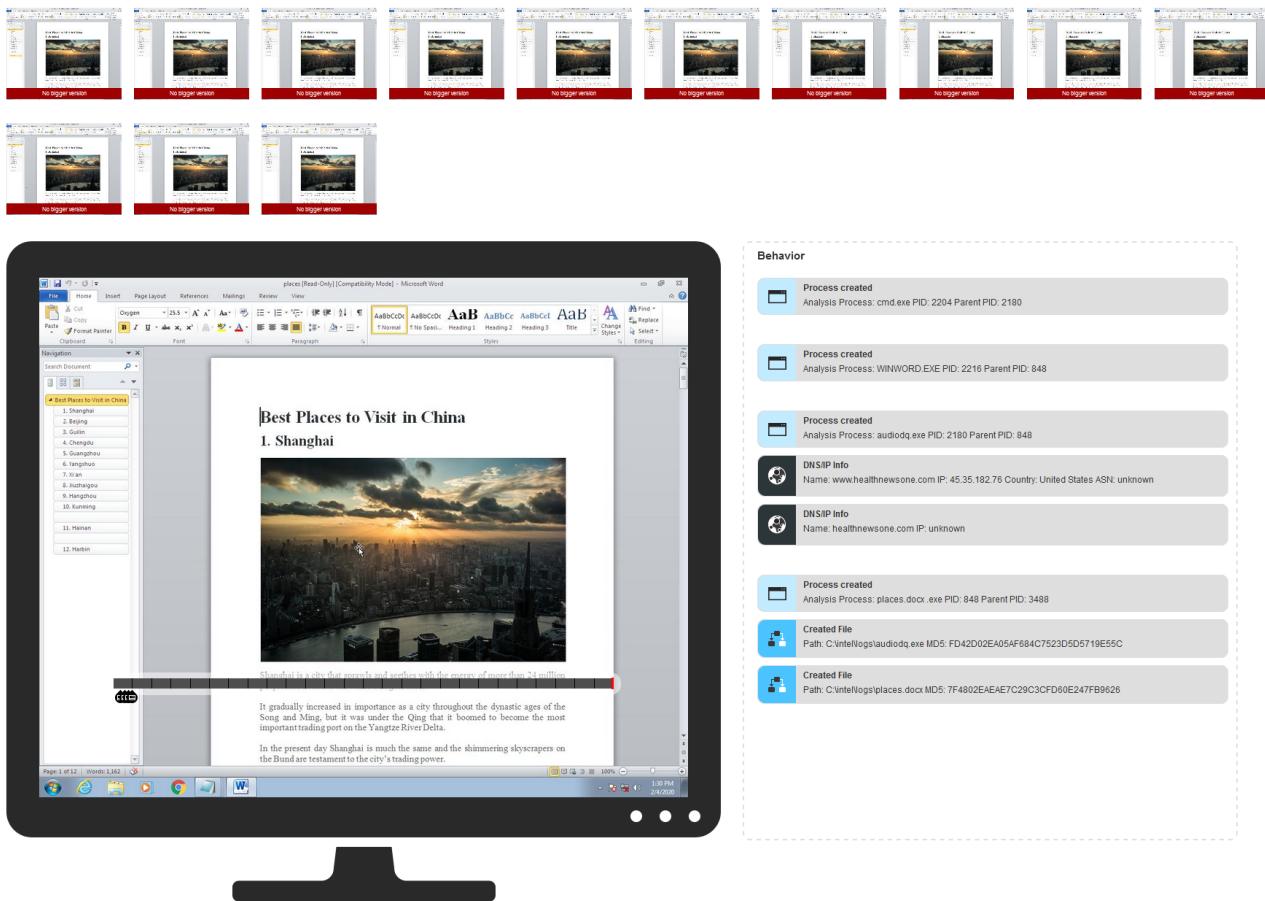


Figura 7.4.11: Joe Sandbox - Screenshots e animazione

Il report continua fornendo, oltre ad altri IOCs, anche altre informazioni riguardanti l'analisi statica del malware e di tutti i files droppati e/o scaricati da esso, ma anche le loro interazioni in rete e nel sistema.

Viene infine fornita una sezione relativa alla advanced malware analysis tramite ispezione approfondita del codice, per ottenere ancora più informazioni sul comportamento del malware, insieme alle regole YARA generate in base a tale analisi con possibilità di scaricarle (figura 7.4.12):

**Disassembly****Code Analysis**

Analysis Process: places.docx.exe PID: 848 Parent PID: 3488 places.docx.exe COMMON

Analysis Process: audiodq.exe PID: 2180 Parent PID: 848 audiodq.exe COMMON

**Execution Graph**

Execution Coverage



Dynamic/Packed Code Coverage



Signature Coverage



Execution Coverage:

4.5%

Dynamic/Decrypted Code Coverage:

0%

Signature Coverage:

5.3%

Total number of Nodes:

2000

Total number of Limit Nodes:

31

**Executed Functions**

Function 00322F10, Relevance: 54.6, APIs: 16, Strings: 15, Instructions: 303 CFG HDD NETWORK SLEEP COMMON

Download Yara Rule

Function 003341DD, Relevance: 1.5, APIs: 1, Instructions: 4 HDD COMMON

Download Yara Rule

Function 00322D30, Relevance: 64.9, APIs: 4, Strings: 33, Instructions: 128 CFG HDD REGISTRY COMMON

Download Yara Rule

Function 003235F0, Relevance: 42.2, APIs: 14, Strings: 10, Instructions: 156 CFG HDD SLEEP PIPE PROCESS COMMON

Download Yara Rule

**Non-executed Functions**

Function 00334FA2, Relevance: 66.4, APIs: 44, Instructions: 432 HDD COMMON LIBRARYCODE

Download Yara Rule

Function 0033A6ED, Relevance: 9.2, APIs: 6, Instructions: 181 HDD COMMON CRYPTO

Download Yara Rule

Function 0032ABF6, Relevance: 7.6, APIs: 5, Instructions: 58 HDD COMMON LIBRARYCODE

Download Yara Rule

Figura 7.4.12: Joe Sandbox - Disassembly

In particolare, a scopo esemplificativo è stata scaricata la regola Yara relativa alla prima function, che si presenta in questo modo (figura 7.4.13):

Figura 7.4.13: Joe Sandbox - Regola YARA

Essa contiene una stringa che rappresenta tutti i codici operativi necessari affinché la funzione selezionata sia presente nel file analizzato. Si noti che i caratteri ? all'interno della sintassi di Yara rappresentano delle wildcards.

Il report è molto lungo e pieno di informazioni e quindi di queste ne sono state estratte solo alcune.

Il report dettagliato è visibile al seguente link: <https://www.joesandbox.com/analysis/205595/0/html>

## 7.4.2 Falcon Sandbox

Anche questa volta sono stati selezionati i parametri da utilizzare per l'analisi (figura 7.4.14):

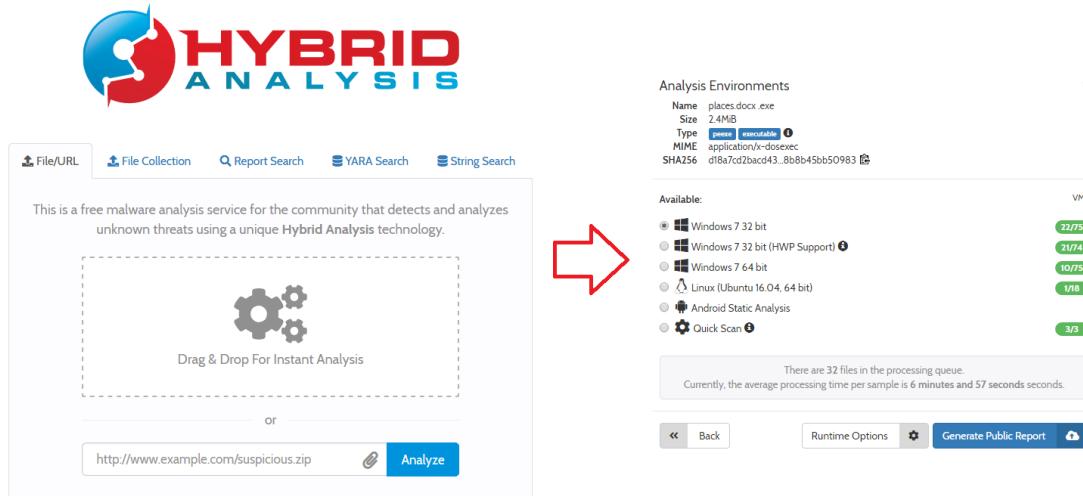


Figura 7.4.14: Falcon Sandbox - Parametri per l'analisi

Ad analisi terminata ci si trova anche in questo caso davanti ad una pagina di overview che marca il file come malevolo e fornisce i risultati di Falcon Sandbox, Metadefender e VirusTotal, assieme ad un'anteprima del risultato dell'analisi fornita da Falcon Sandbox, come mostrato nelle seguenti immagini (7.4.15 e 7.4.16):

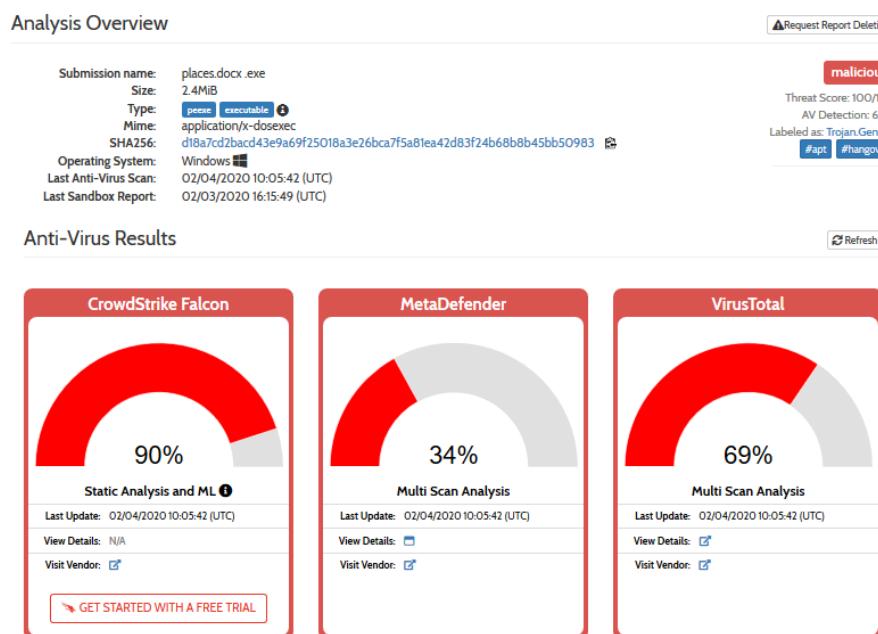


Figura 7.4.15: Falcon Sandbox - Panoramica generale

## Falcon Sandbox Reports

The screenshot shows the Falcon Sandbox interface. On the left, a red-bordered box displays a summary for a file named "places.docx.exe". The summary includes:

- MALICIOUS**
- File Name:** places.docx.exe
- Analyzed on:** 02/03/2020 16:15:49 (UTC)
- Environment:** Windows 7 32 bit
- Threat Score:** 100/100
- AV Detection:** 69% Trojan.Broskod
- Indicators:** 9 (red), 22 (orange), 30 (green)
- Network:** USA

Below this summary are two small screenshots of the Windows desktop environment within the sandbox.

On the right, a larger box titled **FALCON SANDBOX TECHNOLOGY** highlights features:

- Strong Hybrid Analysis:** Powered by Falcon Sandbox
- Easily Deploy and Scale:** Process up to 25,000 files per month with Falcon Sandbox Private Cloud or select an unlimited license with the On-Prem Edition.
- Extensive Coverage:** Expanded support for file types, operating systems and export file formats.
- Unparalleled Customization:** Import custom virtual machine images and customize settings to mirror your real-world environment.

A blue button at the bottom right says **» Learn more**.

Figura 7.4.16: Falcon Sandbox - Panoramica generale

Cliccando sul nome del file viene mostrato il report dettagliato, che inizia con le informazioni generali sul processo (come mostrato nella figura 7.4.17):

The screenshot shows the detailed report for the file "places.docx.exe". At the top right, there is a summary box with the following information:

- malicious**
- Threat Score: 100/100
- AV Detection: 64%
- Labeled as: Trojan.Generic
- #apt #hangover

Below this are social media sharing buttons for LinkedIn, Twitter, and E-Mail.

The main content area is titled **Incident Response**. It includes a section for **Risk Assessment** with the following details:

<b>Persistence</b>	Spawns a lot of processes Writes data to a remote process
<b>Fingerprint</b>	Queries process information Queries sensitive IE security settings Reads the active computer name Reads the cryptographic machine GUID
<b>Evasive</b>	Marks file for deletion
<b>Network Behavior</b>	Contacts 1 domain and 1 host. <a href="#">View all details</a>

Figura 7.4.17: Falcon Sandbox - Informazioni generali

Vengono poi mostrati gli indicatori di compromissione, raggruppati per grado di pericolosità in base al colore in malicious (rossi) e suspicious (gialli) e informative (azzurri), ma anche in base alla categoria.

In questo primo screenshot (figura 7.4.18) è possibile vedere gli indicatori in rosso, i quali forniscono indicazioni circa il fatto che il file sia stato identificato come malevolo da altri motori antivirus o che i files estratti e i processi generati sono stati identificati come malevoli.

## Indicators

<p> ⓘ Not all malicious and suspicious indicators are displayed. Get your own cloud service or the full version to view all details.</p>	
<b>Malicious Indicators</b>	9
<b>External Systems</b>	
Sample was identified as malicious by a large number of Antivirus engines	▼
Sample was identified as malicious by at least one Antivirus engine	▼
<b>General</b>	
The analysis extracted a file that was identified as malicious	▼
The analysis spawned a process that was identified as malicious	▼

Figura 7.4.18: Falcon Sandbox - Malicious IOCs

Nel secondo screenshot (immagine 7.4.19) è possibile vedere gli indicatori in giallo. Essi forniscono informazioni relative ad esempio alle tecniche di evasione o al fatto che vengono lette alcune informazioni riguardanti il sistema vittima.

<b>Suspicious Indicators</b>	22
<b>Anti-Detection/Stealthyness</b>	
Queries process information	▼
<b>Cryptographic Related</b>	
Found a cryptographic related string	▼
<b>Environment Awareness</b>	
Reads the active computer name	▼
Reads the cryptographic machine GUID	▼

Figura 7.4.19: Falcon Sandbox - Suspicious IOCs

Nel terzo screenshot (figura 7.4.20) è possibile infine vedere gli indicatori in azzurro, che forniscono informazioni aggiuntive come i domini e i server contattati e il fatto che il malware ha l'abilità di leggere ulteriori informazioni sul sistema vittima.

Informative	
<b>Anti-Reverse Engineering</b>	
Contains ability to register a top-level exception handler (often used as anti-debugging trick)	▼
<b>Environment Awareness</b>	
Contains ability to query machine time	▼
Contains ability to query the machine timezone	▼
Contains ability to query the system locale	▼
Makes a code branch decision directly after an API that is environment aware	▼
Possibly tries to detect the presence of a debugger	▼
Reads the registry for installed applications	▼
<b>General</b>	
Contacts domains	▼
Contacts server	▼

Figura 7.4.20: Falcon Sandbox - Informative IOCs

Vengono mostrati poi (figura 7.4.21) degli screenshot catturati sulla macchina durante l'esecuzione del malware:

#### Screenshots

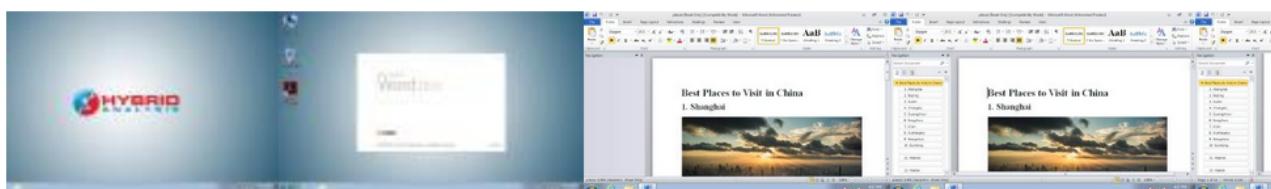


Figura 7.4.21: Falcon Sandbox - Screenshots

Inoltre vengono mostrate in dettaglio l'analisi dei processi, delle attività di rete e dei files estratti.

Nell'immagine 7.4.22 è possibile vedere l'albero dei processi generati e, cliccandoci sopra, è possibile visualizzare tutte le informazioni ad essi relative. In particolare, così come evidenziato nell'analisi manuale, possiamo osservare come il processo *reg.exe* modifichi una chiave di registro notoriamente utilizzata per motivi di persistenza.

## Hybrid Analysis

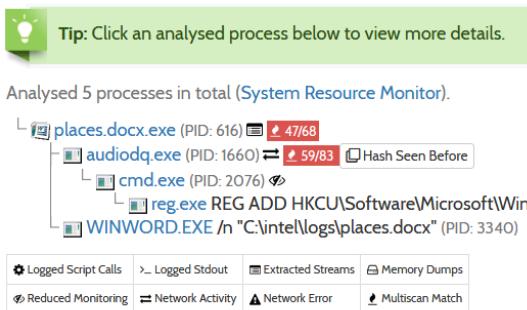


Figura 7.4.22: Falcon Sandbox - Albero dei processi

In questa seconda immagine (7.4.23) è possibile visualizzare le richieste DNS, gli host contattati e il traffico HTTP generato.

## Network Analysis

### DNS Requests

[Login to Download DNS Requests \(CSV\)](#)

Domain	Address	Registrar	Country
healthnewsone.com	45.35.182.76 TTL: 599	NameSilo, LLC Organization: See PrivacyGuardian.org Name Server: NS1.QHOSTER.NET Creation Date: Thu, 15 Feb 2018 05:49:41 GMT	United States

### Contacted Hosts

[Login to Download Contacted Hosts \(CSV\)](#)

IP Address	Port/Protocol	Associated Process	Details
45.35.182.76 OSINT	80 TCP	audiodq.exe PID: 1660	United States

### HTTP Traffic

Endpoint	Request	URL	Data
45.35.182.76.80 (healthnewsone.com)	GET	//healthne/accept.php?a=HAPUBWS-P	GET //healthne/accept.php?a=HAPUBWS-P&b=cZCDEplPTK&c=Windows%207%20Professional&d=UnbyHRqUnbyHRq7C&b=cZCDEplPTK&c=Windows%207%20Professional&d=UnbyHRqUnbyHRq73c94c5-cebb-4f98-a75f-22a797d1d50b365536040965860&e=HTTP/1.1 Host: healthnewsone.com <a href="#">More Details</a>

Figura 7.4.23: Falcon Sandbox - Network Analysis

In questa terza immagine (7.4.24) è invece possibile vedere i file estratti dall’archivio auto-estraente con le relative informazioni.

## Extracted Files

Displaying 10 extracted file(s). The remaining 2 file(s) are available in the full version and XML/JSON reports.

Malicious	
	audiодq.exe
	Download Disabled
	Extended File Details
	VirusTotal Report
	Metadefender Report
	Extracted Streams
	Hash Seen Before
Size	198KiB (202752 bytes)
Type	pe32 executable
Description	PE32 executable (GUI) Intel 80386, for MS Windows
AV Scan Result	Labeled as "Trojan.Generic" (59/83)
Runtime Process	places.docx.exe (PID: 616)
MD5	fd42d02ea05af684c7523d5d5719e55c
SHA1	8999e312a6058f6efc8b8a0360d195425d0a8535
SHA256	4398615f0a498124b498abb261150f11ce4778afde18d2e4e6717f2826a9d65d

Clean	
	places.docx
	Download Disabled
	VirusTotal Report
	Hash Not Seen Before
Size	2MiB (2140085 bytes)
Type	docx office
Description	Microsoft Word 2007+
AV Scan Result	0/61
Runtime Process	places.docx.exe (PID: 616)
MD5	7f4802eaeae7c29c3cf60e247fb9626
SHA1	b505f26476d19b6a974b112b061326fc7af1ca5b
SHA256	3b052247207a0e7227cf7c07039b8a37ba392e7b4a03aeb48eb6d05cc18dfaee

Figura 7.4.24: Falcon Sandbox - Extracted files

Il report completo è reperibile al seguente link:

<https://www.hybrid-analysis.com/sample/d18a7cd2bacd43e9a69f25018a3e26bca7f5a81ea42d83f24b68b8b45bb50983>

### 7.4.3 Any.Run

Anche in questo caso, come è possibile osservare dalla figura 7.4.25, sono stati selezionati i parametri da utilizzare per l’analisi:

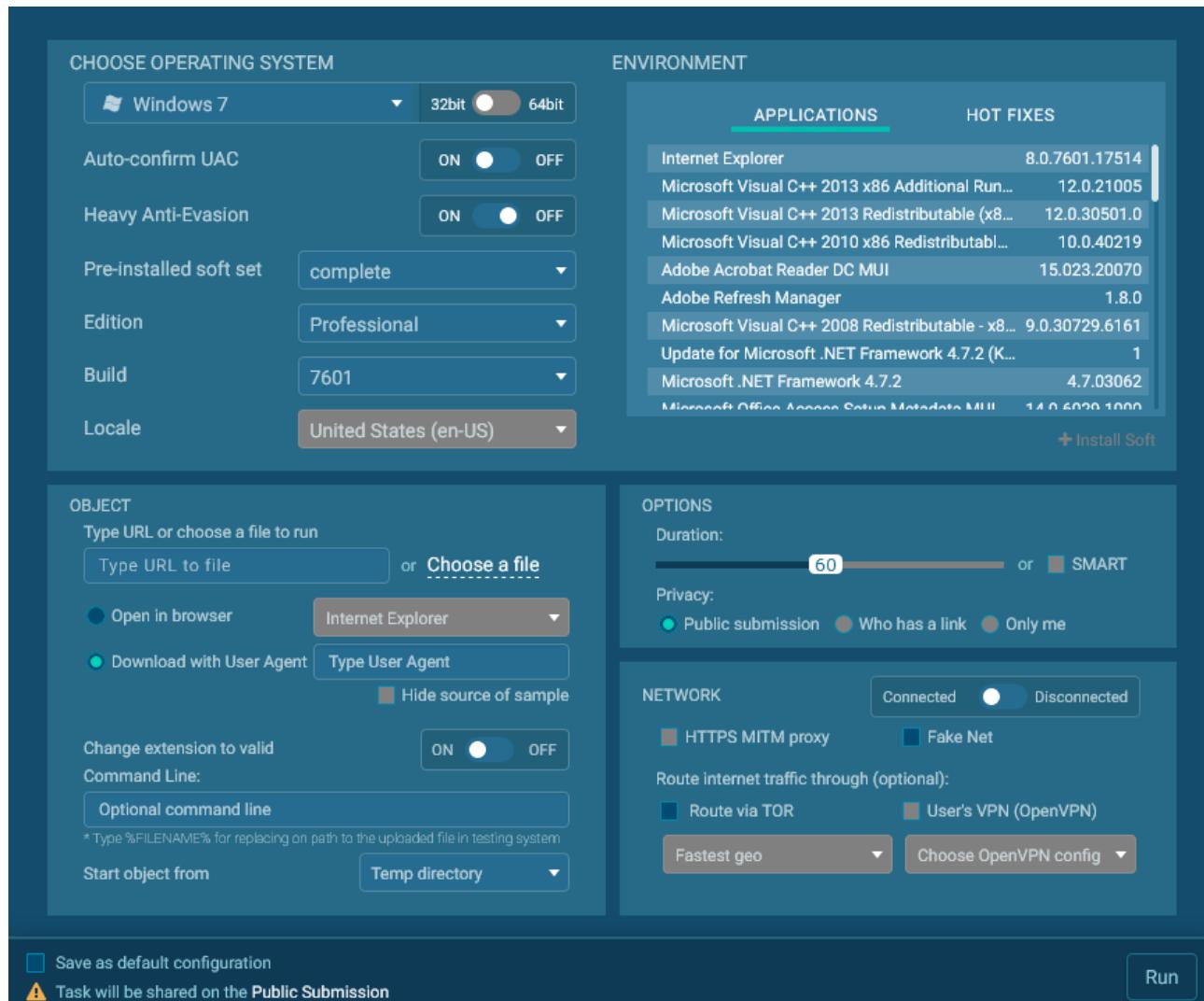


Figura 7.4.25: Any.Run - Parametri per l’analisi

Una volta terminata l’analisi viene visualizzata la seguente pagina (figura 7.4.26), nella quale è possibile vedere gli screenshot catturati sulla macchina durante l’esecuzione dell’analisi, mentre in basso e a destra sono visibili informazioni dettagliate sui risultati dell’analisi stessa. In particolare, in basso è possibile osservare le attività compiute sulla rete e quelle compiute sul file system, mentre sulla destra è possibile visualizzare le attività dei processi generati dal malware.

Cliccando su ciascuno dei processi, così come anche sulle voci relative ad attività su rete e file system, è possibile visualizzare ulteriori dettagli.

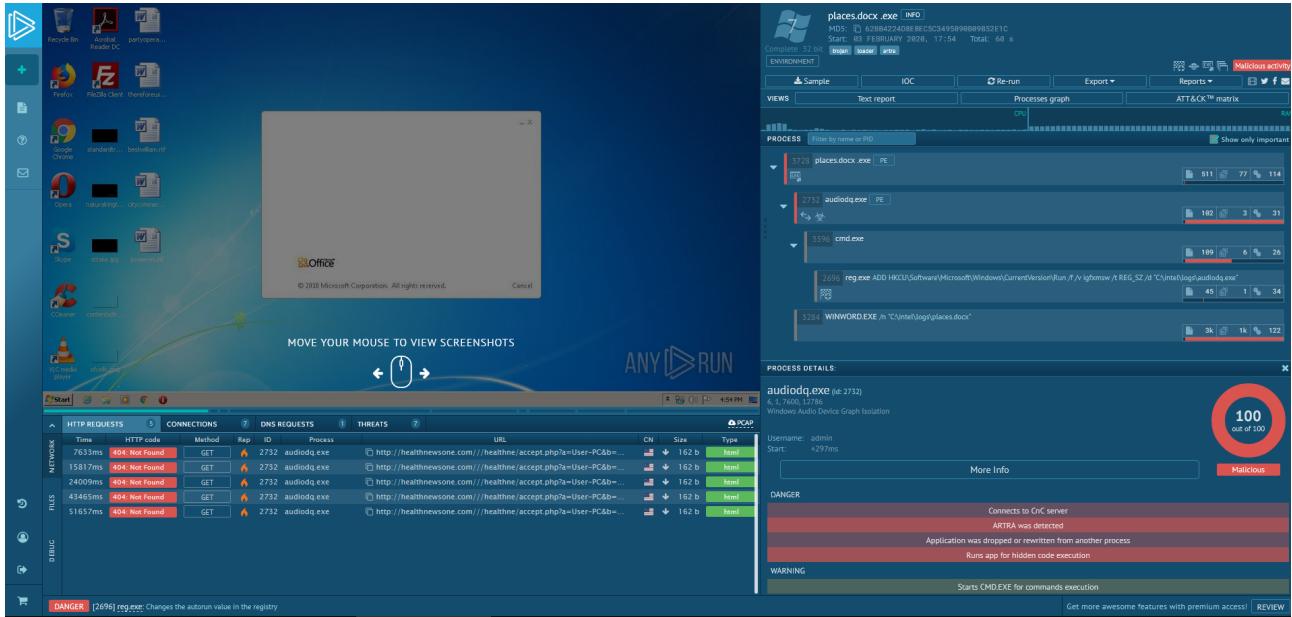


Figura 7.4.26: Any.Run - Panoramica

Per quanto riguarda i processi, è anche possibile visualizzare la gerarchia di generazione di tali processi sotto forma di grafo. Ciò è illustrato nell'immagine 7.4.27:



Figura 7.4.27: Any.Run - Gerarchia processi

Sulla destra è anche possibile cliccare per visualizzare gli IOCs, come mostrato nella seguente immagine (7.4.28):

TITLE	TYPE	IOC	REP	ACTION
Main object - "places.docx.exe"				<input type="checkbox"/> <input checked="" type="checkbox"/>
SHA256		D18A7CD2BACD43E9A69F25018A3E26BCA7F5A81EA42D83F24B68B8B45BB50983	<input type="checkbox"/>	<input checked="" type="checkbox"/>
SHA1		CE478936F245647B557CD782A48C071FB7545E97	<input type="checkbox"/>	<input checked="" type="checkbox"/>
MD5		62BB4224D8E8EC5C3495090B09B52E1C	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Dropped executable file				<input type="checkbox"/> <input checked="" type="checkbox"/>
SHA256		C:\intel\logs\audiodq.exe 4398615F0A498124B498ABB261150F11CE4778AFDE18D2E4E6717F2826A9D65D	<input type="checkbox"/>	<input checked="" type="checkbox"/>
DNS requests				<input type="checkbox"/> <input checked="" type="checkbox"/>
DOMAIN		healthnewsone.com	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Connections				<input type="checkbox"/> <input checked="" type="checkbox"/>
IP		45.35.182.76	<input type="checkbox"/>	<input checked="" type="checkbox"/>
HTTP/HTTPS requests				<input type="checkbox"/> <input checked="" type="checkbox"/>
URL		http://healthnewsone.com:///healthne/accept.php?a=User-PC&b=USER-PC&c=Windows%20%20Professional&d=adminadmin90059c37-1320-41a4-b58d-2b75a9850d2f1565536040965860&e-	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Figura 7.4.28: Any.Run - IOCs

Gli IOCs principali trovati, come ad esempio i file droppati e le richieste DNS e HTTP, sono quelli già osservati con l'analisi manuale e le sandbox precedenti.

E' possibile anche ottenere un report dettagliato in formato testuale, nel quale è possibile visionare tutte le informazioni già presenti dall'interfaccia grafica interattiva, presentate in maniera dettagliata e lineare.

Anche in questo caso, è possibile ad esempio ottenere la suddivisione degli IOCs in base al colore (figura 7.4.29):

## Behavior activities



MALICIOUS	SUSPICIOUS	INFO
<b>Application was dropped or rewritten from another process</b> <ul style="list-style-type: none"><li>audiodq.exe (PID: 2732)</li></ul>	<b>Starts Microsoft Office Application</b> <ul style="list-style-type: none"><li>places.docx.exe (PID: 3728)</li></ul>	<b>Creates files in the user directory</b> <ul style="list-style-type: none"><li>WINWORD.EXE (PID: 3284)</li></ul>
<b>Runs app for hidden code execution</b> <ul style="list-style-type: none"><li>audiodq.exe (PID: 2732)</li></ul>	<b>Executable content was dropped or overwritten</b> <ul style="list-style-type: none"><li>places.docx.exe (PID: 3728)</li></ul>	<b>Reads Microsoft Office registry keys</b> <ul style="list-style-type: none"><li>WINWORD.EXE (PID: 3284)</li></ul>
<b>Changes the autorun value in the registry</b> <ul style="list-style-type: none"><li>reg.exe (PID: 2696)</li></ul>	<b>Starts CMD.EXE for commands execution</b> <ul style="list-style-type: none"><li>audiodq.exe (PID: 2732)</li></ul>	
<b>ARTRA was detected</b> <ul style="list-style-type: none"><li>audiodq.exe (PID: 2732)</li></ul>	<b>Uses REG.EXE to modify Windows registry</b> <ul style="list-style-type: none"><li>cmd.exe (PID: 3596)</li></ul>	
<b>Connects to CnC server</b> <ul style="list-style-type: none"><li>audiodq.exe (PID: 2732)</li></ul>		

Figura 7.4.29: Any.Run - Suddivisione IOCs

Il report dettagliato dell'analisi è disponibile al seguente link:

<https://any.run/report/d18a7cd2bacd43e9a69f25018a3e26bca7f5a81ea42d83f24b68b8b45bb50983/c1924b93-c0f8-4ee7-87cd-5cbe9011b592#registry>

# Appendice A

## Setup laboratorio

Prima di iniziare a "divertirci" con la malware analysis, la prima cosa da fare è mettere in piedi un ambiente sicuro ed isolato dalle macchine che utilizziamo tutti i giorni. Se eseguissimo un malware sul nostro amato PC, staremmo non solo mettendo a rischio tutto ciò che è in esso contenuto, ma anche tutti gli altri dispositivi vulnerabili sulla rete. Per fare ciò, il modo più semplice e veloce è utilizzare delle Virtual Machines (VM), che in questo caso fungono da "sandbox", in quanto il malware che viene eseguito all'interno di una VM (nel SO guest) non danneggerà il SO host se stiamo sufficientemente attenti. Stare sufficientemente attenti vuol dire seguire alcuni piccoli consigli che ridurranno al minimo la probabilità di incidenti di percorso. I principali sono i seguenti:

- Utilizzare sempre l'ultima versione del software di virtualizzazione, in quanto un malware potrebbe potenzialmente sfruttare una vulnerabilità di tale software per scappare dalla VM.
- Utilizzare una configurazione di rete per le VM tale per cui i SO guest non possano in alcun modo comunicare con il SO host e la rete Internet.
- Non collegare alcun media rimovibile al SO guest per poi riutilizzarlo sul SO host, in quanto il malware potrebbe diffondersi anche tramite tali media.
- Disabilitare la possibilità di avere cartelle condivise tra il SO host e i SO guest, in quanto il malware potrebbe diffondersi anche attraverso di esse.

- Non mantenere alcun dato sensibile all'interno dei SO guest nei quali si sta portando avanti l'analisi, in quanto tali informazioni potrebbero essere sottratte dal malware nel caso in cui riuscisse a comunicare con il suo obiettivo.
- Assicurarsi di scaricare soltanto campioni di malware compressi e protetti da password, per evitare un'esecuzione accidentale (il famoso effetto "Oh sh!t").

In questo elaborato è stato utilizzato un PC con il SO Windows 10, sul quale è stato installato il software di virtualizzazione Oracle VM VirtualBox (attualmente alla versione 6.1.2), il cui installer è reperibile a questo link: <https://www.virtualbox.org/wiki/Downloads>.

Sono state utilizzate 2 VMs. La prima con SO Windows 7 SP1, sulla quale verranno installati la maggior parte degli strumenti di analisi e sulla quale verrà eseguito il malware, mentre la seconda con SO Ubuntu 18.04.3 LTS, sulla quale verranno installati gli strumenti di simulazione dei servizi di rete e di monitoraggio del traffico di rete. Tali VM saranno configurate in modo da far parte della stessa rete interna, la quale sarà isolata sia dalla rete del SO host, sia dalla rete Internet.

## Setup Windows VM

Una volta creata la VM con la procedura guidata di VirtualBox e installato il SO Windows guest, bisogna seguire i passi illustrati di seguito:

1. Installare le Guest Additions nel SO guest, in modo da poter utilizzare appieno le funzionalità di VirtualBox, come ad esempio la modifica della risoluzione, gli appunti condivisi e il trascina e rilascia. Per farlo andare su *Dispositivi -> Inserisci l'immagine del CD delle Guest Additions* (figura A.0.1).

A questo punto avviare l'installer e seguire la procedura guidata.

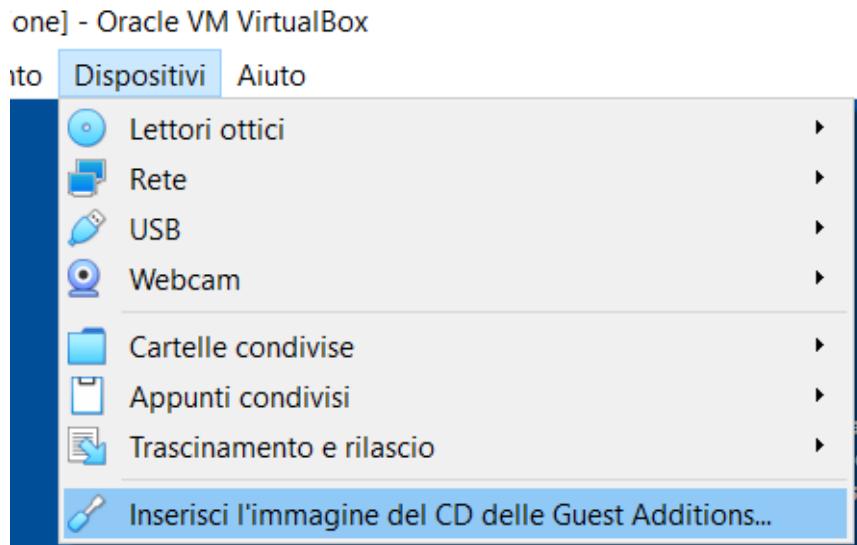


Figura A.0.1: VirtualBox - Guest Additions

Per abilitare gli *appunti condivisi* e il *trascina e rilascia*, aprire le impostazioni della VM e cliccare su *Generale* -> tab *Avanzate* -> impostare *Appunti condivisi* su *Bidirezionale* e *Trascina e rilascia* su *Da host a guest*.

2. Scaricare (e laddove serve, installare) tutti i programmi necessari connettendosi ad Internet. Per fare questo bisogna momentaneamente lasciare la configurazione di rete con l'opzione "NAT". I programmi sono i seguenti:

- Mozilla Firefox (<https://www.mozilla.org/it/firefox/new/>)
- WinRAR (<https://www.winrar.it/prelievo.php>)
- Visual C++ Redistributable for Visual Studio 2015  
(<https://www.microsoft.com/en-us/download/confirmation.aspx?id=48145>)
- HxD hex editor (<https://mh-nexus.de/en/hxd/>)
- HashMyFiles ([https://www.nirsoft.net/utils/hash\\_my\\_files.html](https://www.nirsoft.net/utils/hash_my_files.html))
- ssdeep (<https://ssdeep-project.github.io/ssdeep/>)
- Strings (<https://docs.microsoft.com/en-us/sysinternals/downloads/strings>)
- PE Studio (<https://www.winitor.com/get.html>)

- UPX (<https://upx.github.io/>)
- Exeinfo PE (<http://exeinfo.atwebpages.com/>)
- Resource Hacker (<http://www.angusj.com/resourcehacker/>)
- YARA (<https://virustotal.github.io/yara/>)
- Process Hacker (<https://processhacker.sourceforge.io/>)
- RegShot (<https://sourceforge.net/projects/regshot/>)
- Process Monitor (<https://docs.microsoft.com/it-it/sysinternals/downloads/procmon>)

3. Cambiare la configurazione di rete della VM in modo tale che non sia più connessa ad Internet, bensì ad una rete interna totalmente isolata dall'esterno, alla quale saranno attaccate solo le due VM. Per fare ciò bisogna effettuare alcune operazioni:

- A VM spenta, aprire le impostazioni della VM e andare nella scheda *Rete*. A questo punto modificare le impostazioni della scheda di rete come mostrato nella figura A.0.2:

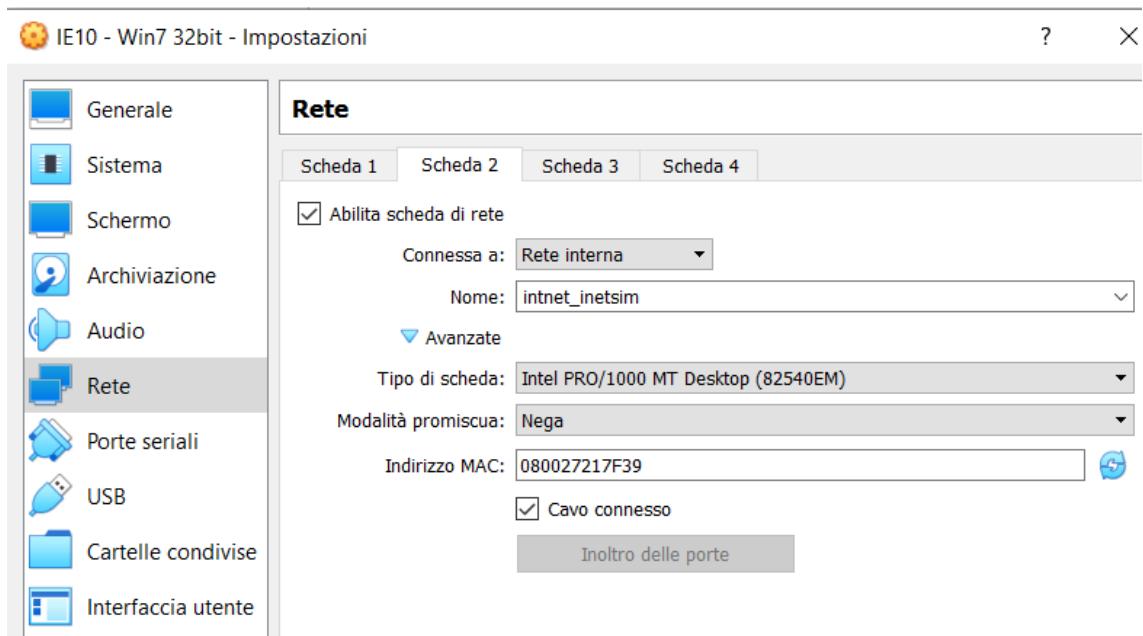


Figura A.0.2: VirtualBox - Configurazione rete interna

- Avviare la macchina e andare su *Start -> Control Panel -> Network and Internet -> Network and Sharing Center* -> Cliccare sulla connessione *Local Area Connection* associata

alla scheda di rete utilizzata per la rete interna. A questo punto cliccare su *Properties* e, dalla finestra che si apre, cliccare su *Internet Protocol Version 4 (TCP/IP)* e ancora una volta su *Properties*. Modificare quindi le informazioni contenute nella finestra che è appena apparsa impostando staticamente indirizzo IP, subnet mask, default gateway e preferred DNS server, come mostrato di seguito (figura A.0.3):

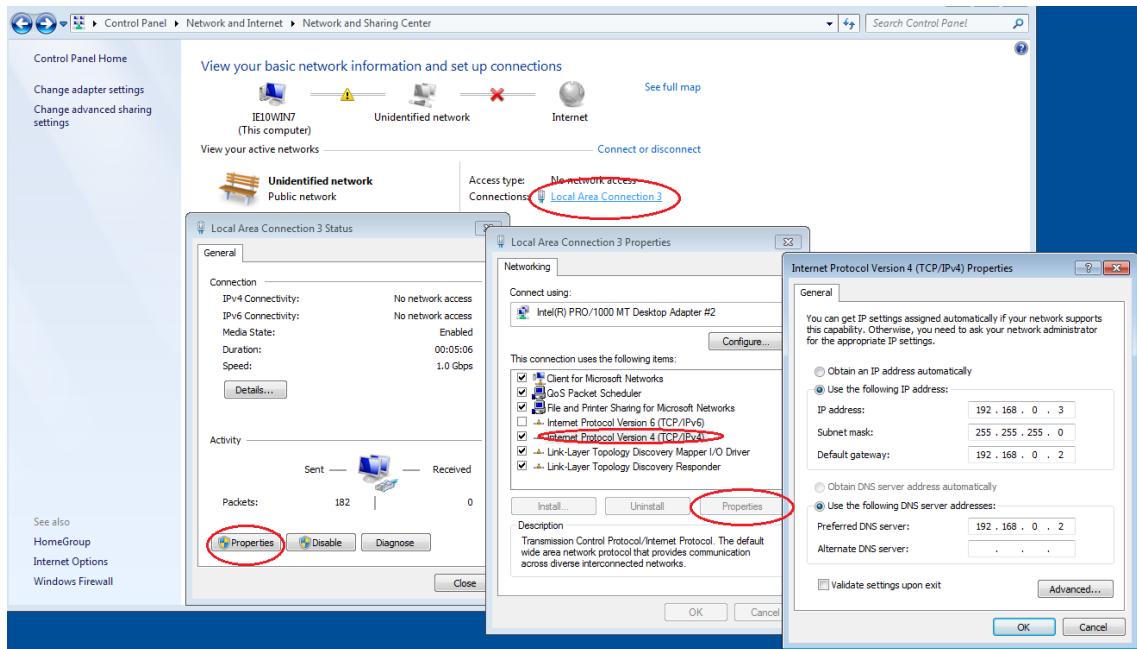


Figura A.0.3: Modifica delle impostazioni di rete

L’indirizzo inserito nei campi *default gateway* e *preferred DNS server* è l’indirizzo IP che verrà assegnato in seguito alla VM Ubuntu, in modo che la VM Ubuntu possa intercettare tutte le richieste fatte dalla VM Windows e simularle, come vedremo, tramite INetSim. Si noti che, una volta effettuato questo cambiamento, non sarà più possibile accedere ad internet, neanche per scaricare i campioni di malware. Per ottenere tali campioni, quindi, si può utilizzare la funzionalità di trascinamento e rilascio di VirtualBox, oppure utilizzare una seconda scheda di rete configurata con NAT da attivare solo per scaricare il campione e poi, una volta ottenuto, disattivarla. Altre alternative possono essere quelle di utilizzare il file manager di VirtualBox, oppure la creazione di una cartella condivisa tra host e guest, la quale dovrà anch’essa essere disabilitata prima di eseguire il malware.

4. Disabilitare i servizi di Windows Defender e Windows Update, i quali potrebbero interferire con l'analisi. Per farlo, andare su *Start -> Run ->* Digitare *services.msc* e confermare. A questo punto trovare i due servizi da disabilitare e farci doppio click. Nella scheda *General* impostare *Startup type* a *Disabled*, poi cliccare su *Stop* e confermare (figura A.0.4).

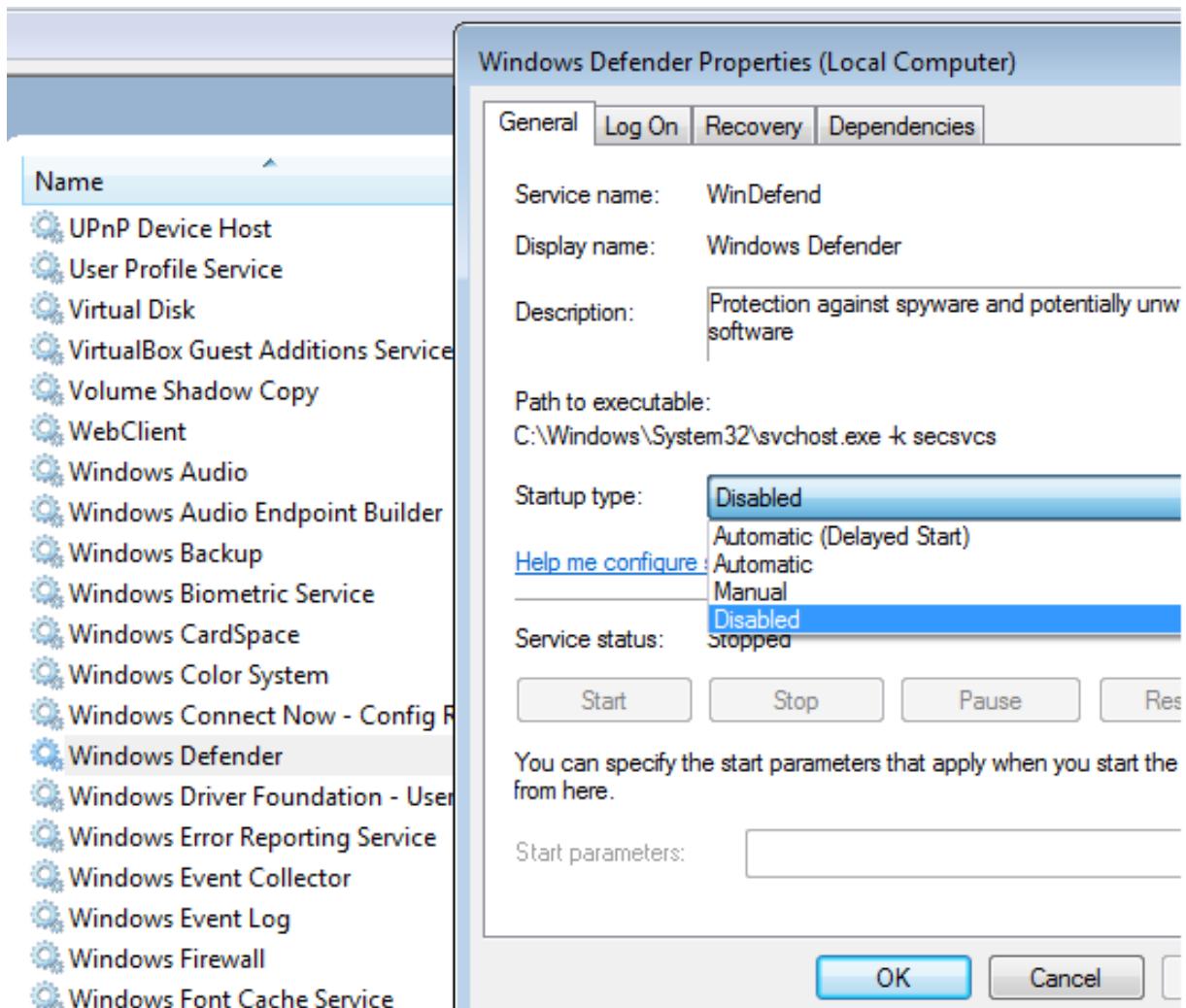


Figura A.0.4: Disabilitazione di Windows Defender e Windows Update

5. Disabilitare *Windows Firewall*, in quanto anch'esso potrebbe interferire con l'analisi. Per farlo andare su *Start -> Control Panel -> System and Security -> Windows Firewall -> Turn Windows Firewall on or off*. A questo punto disabilitare *Windows Firewall*, come mostrato nell'immagine A.0.5.

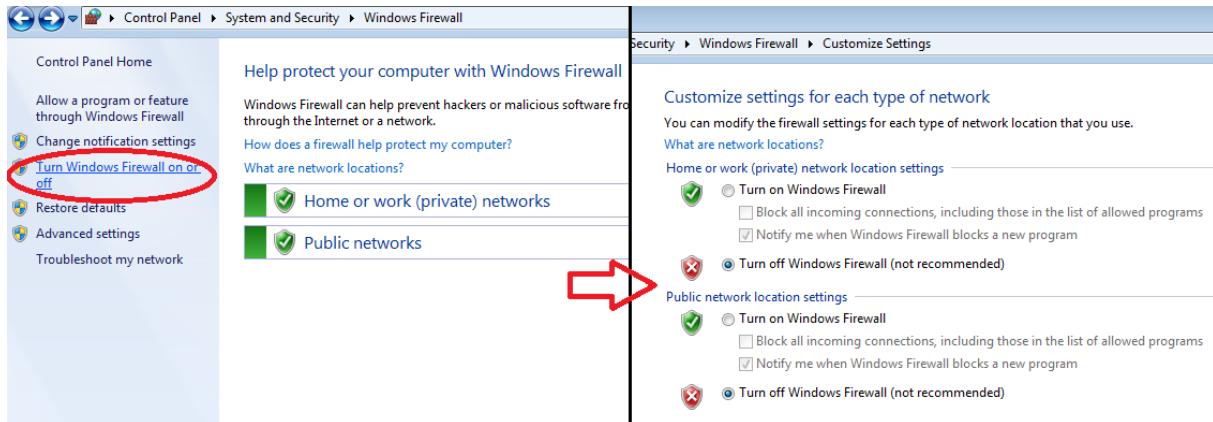


Figura A.0.5: Disabilitazione di Windows Firewall

6. Creare un’istantanea (snapshot) della VM nello stato corrente, ossia dopo aver completato la configurazione iniziale. Ciò risulta utilissimo in questo caso, in quanto dopo l’esecuzione di un malware, per poter ripetere l’analisi o per poter analizzare un malware diverso, bisogna prima riportarsi ad uno stato precedente all’attivazione del malware senza dover necessariamente rifare tutta la configurazione. Per farlo, a VM spenta, cliccare sull’icona con l’elenco a destra del nome della VM e selezionare la scheda *Istantanee* (figura A.0.6). A questo punto cliccare il pulsante *Crea*.

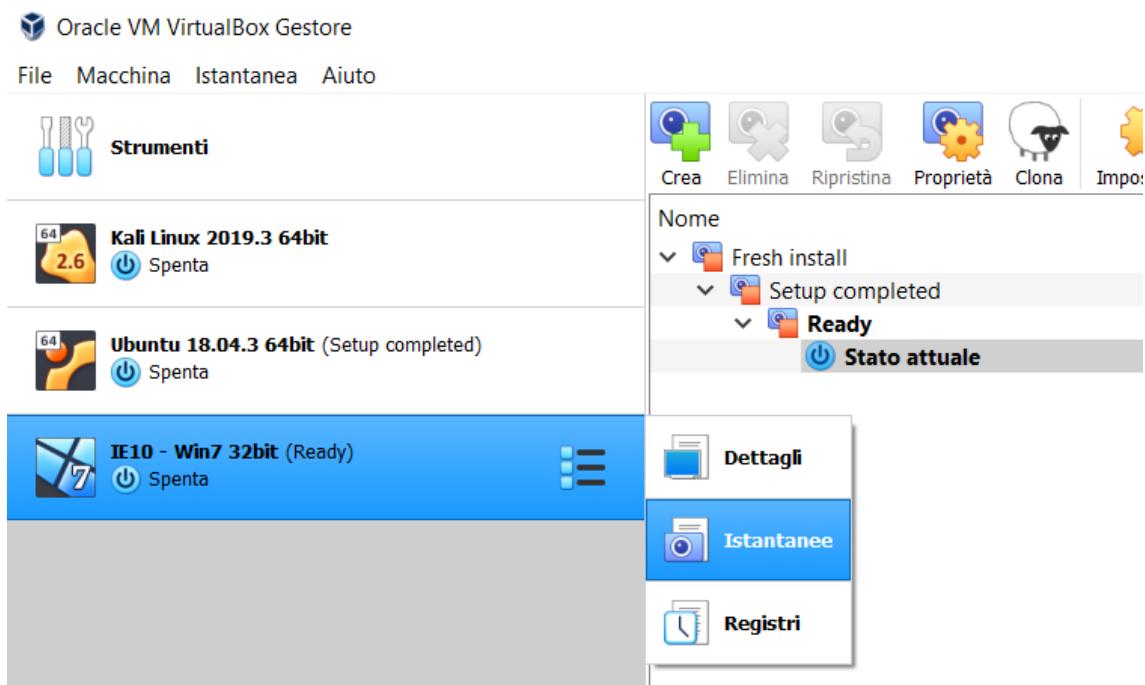


Figura A.0.6: VirtualBox - Creazione istantanee

È in generale possibile creare più snapshot di vari stati della macchina seguendo la stessa procedura, in modo da poter sempre tornare allo stato desiderato cliccando su tale snapshot e selezionando *Ripristina*.

## Setup Ubuntu VM

Una volta creata la VM con la procedura guidata di VirtualBox e installato il SO Ubuntu guest, bisogna seguire i passi illustrati di seguito:

1. Installare le Guest Additions nel SO guest seguendo gli stessi passi visto con la VM Windows.
2. Scaricare e installare tutti i programmi necessari connettendosi ad Internet. Per fare questo bisogna momentaneamente lasciare la configurazione di rete con l'opzione "NAT". Prima di tutto lanciare il comando:

```
$ sudo apt-get update
```

A questo punto bisogna installare i seguenti programmi: net-tools, Wireshark, INetSim. Per farlo, lanciare i seguenti comandi:

```
$ sudo su  
# echo "deb http://www.inetsim.org/debian/ binary/" > /etc/apt/sources.list.d/inetsim.list  
# wget -O - http://www.inetsim.org/inetsim-archive-signing-key.asc | apt-key add -  
# apt update  
# apt-get install inetsim  
# apt-get install wireshark  
# apt-get install net-tools
```

3. Cambiare la configurazione di rete della VM in modo tale che non sia più connessa ad Internet, bensì alla rete interna creata in precedenza e alla quale è già attaccata l'altra VM. Per fare ciò bisogna effettuare alcune operazioni:

- A VM spenta, aprire le impostazioni della VM e andare nella scheda *Rete*. A questo punto modificare le impostazioni della scheda di rete come mostrato di seguito (immagine A.0.7):

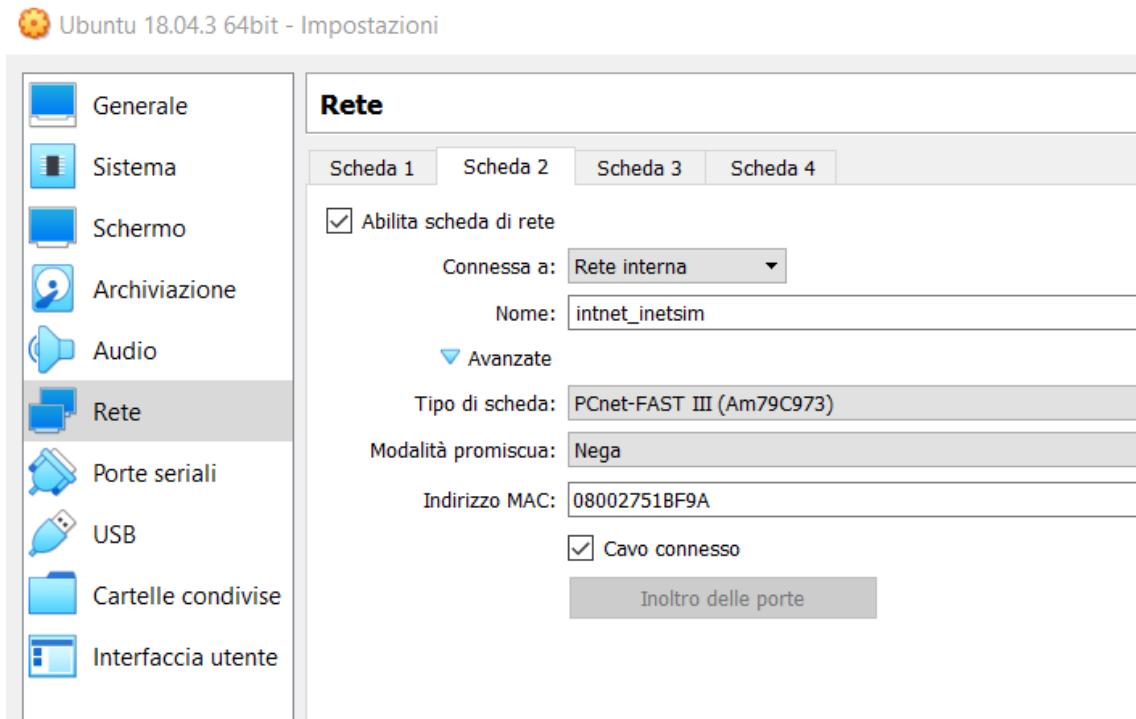


Figura A.0.7: VirtualBox - Configurazione rete interna

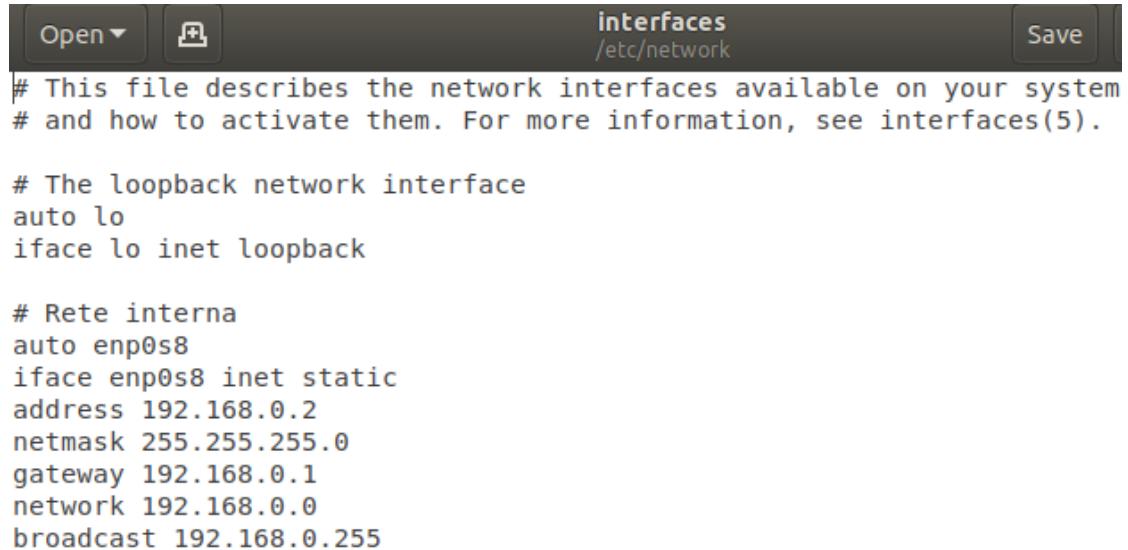
- Avviare la macchina, aprire il terminale e visualizzare il nome associato all’interfaccia di rete precedentemente creata digitando il comando:

```
$ ifconfig
```

- Aprire un’altra istanza di terminale e digitare il seguente comando:

```
$ sudo gedit /etc/network/interfaces
```

- Modificare il file utilizzando il nome dell’interfaccia ottenuto dal punto precedente, che per noi era *enp0s8* (figura A.0.8).



The screenshot shows a text editor window with the title "interfaces /etc/network". The interface "lo" is configured with an IP of 192.168.0.2 and a netmask of 255.255.255.0. The interface "enp0s8" is also configured with an IP of 192.168.0.2 and a netmask of 255.255.255.0.

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

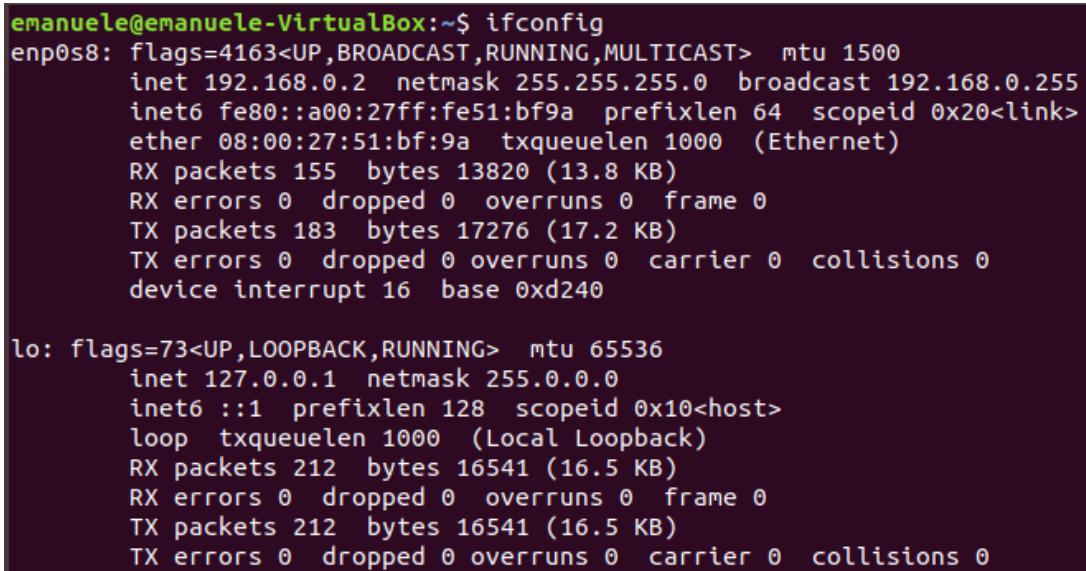
# The loopback network interface
auto lo
iface lo inet loopback

# Rete interna
auto enp0s8
iface enp0s8 inet static
    address 192.168.0.2
    netmask 255.255.255.0
    gateway 192.168.0.1
    network 192.168.0.0
    broadcast 192.168.0.255
```

Figura A.0.8: Configurazione interfaccia di rete

- Controllare che i parametri appena settati corrispondano (figura A.0.9) lanciando di nuovo il comando:

```
$ ifconfig
```



The terminal output shows the configuration of network interfaces. The interface "enp0s8" is up and has an IP of 192.168.0.2, a netmask of 255.255.255.0, and a broadcast address of 192.168.0.255. The interface "lo" is up and has an IP of 127.0.0.1, a netmask of 255.0.0.0, and a broadcast address of 127.0.0.1.

```
emanuele@emanuele-VirtualBox:~$ ifconfig
enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.0.2 netmask 255.255.255.0 broadcast 192.168.0.255
        inet6 fe80::a00:27ff:fe51:bf9a prefixlen 64 scopeid 0x20<link>
              ether 08:00:27:51:bf:9a txqueuelen 1000 (Ethernet)
              RX packets 155 bytes 13820 (13.8 KB)
              RX errors 0 dropped 0 overruns 0 frame 0
              TX packets 183 bytes 17276 (17.2 KB)
              TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
              device interrupt 16 base 0xd240

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
              loop txqueuelen 1000 (Local Loopback)
              RX packets 212 bytes 16541 (16.5 KB)
              RX errors 0 dropped 0 overruns 0 frame 0
              TX packets 212 bytes 16541 (16.5 KB)
              TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figura A.0.9: Output di ifconfig

4. A questo punto, per verificare se la configurazione è andata a buon fine, è consigliabile eseguire il ping dalla VM Windows a quella Ubuntu e viceversa.

5. Configurare INetSim affinché possa stare in ascolto e simulare tutti i servizi sull'indirizzo IP statico 192.168.0.2 configurato precedentemente, in quanto di default esso è in ascolto sull'indirizzo di loopback 127.0.0.1. Per farlo, aprire il file di configurazione di INetSim digitando nel terminale il seguente comando:

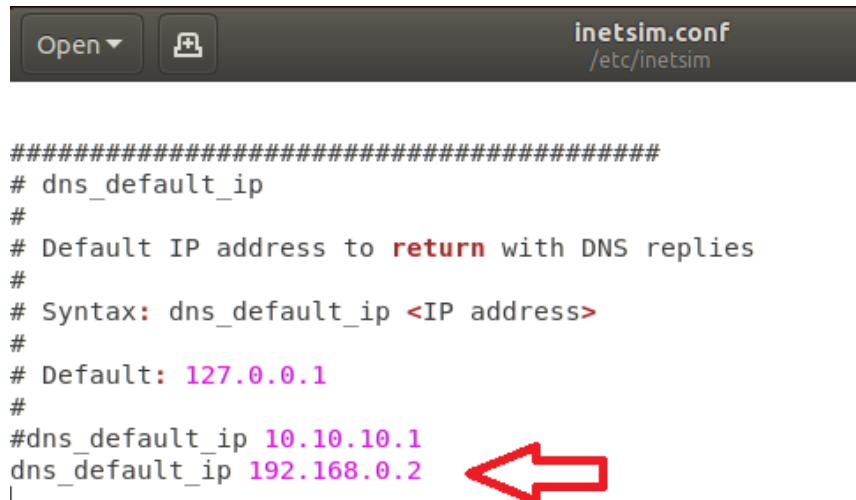
```
$ sudo gedit /etc/inetsim/inetsim.conf
```

A questo punto portarsi nella sezione *service\_bind\_address* e modificare il file affinché tale pezzo appaia in questo modo (figura A.0.10):

```
#####
# service_bind_address
#
# IP address to bind services to
#
# Syntax: service_bind_address <IP address>
#
# Default: 127.0.0.1
#
#service_bind_address 10.10.10.1
service_bind_address 192.168.0.2
```

Figura A.0.10: Configurazione INetSim

6. Configurare INetSim affinché possa risolvere tutti i nomi di dominio con l'indirizzo 192.168.0.2 configurato precedentemente, in quanto di default verrebbero risolti a 127.0.0.1. Per farlo, nello stesso file di configurazione di prima, portarsi nella sezione *dns\_default\_ip* e modificare il file affinché tale pezzo appaia come in figura A.0.11:



```
#####
# dns_default_ip
#
# Default IP address to return with DNS replies
#
# Syntax: dns_default_ip <IP address>
#
# Default: 127.0.0.1
#
#dns_default_ip 10.10.10.1
dns_default_ip 192.168.0.2
```

Figura A.0.11: Configurazione INetSim

## 7. Per verificare il corretto funzionamento di INetSim:

- Avviare INetSim con il comando:

```
$ sudo inetsim
```

- Andare sulla VM Windows, aprire Firefox e digitare il seguente URL: <http://gitlab.comics.unina.it>
- Verificare che la pagina mostrata sia quella fornita da INetSim (come in figura A.0.12):



Figura A.0.12: Pagina di default di INetSim

# Bibliografia

- [1] Michael Sikorski, Andrew Honig - Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software
- [2] Monnappa K A - Learning Malware Analysis: Explore the concepts, tools, and techniques to analyze and investigate Windows malware
- [3] William Stallings - Computer Security: Principles and Practice
- [4] Corrado Aaron Visaggio - Malware Analysis
- [5] Cos'è un malware? -  
<https://it.malwarebytes.com/malware/>
- [6] What are Indicators of Compromise? | Digital Guardian -  
<https://digitalguardian.com/blog/what-are-indicators-compromise>
- [7] Indicators of attack versus indicators of compromise -  
<http://index-of.es/z0ro-Repository-3/Indicators%20Of%20Attack%20Versus%20Indicators%20Of%20Compromise.pdf>
- [8] Top 15 Indicators Of Compromise -  
<https://www.darkreading.com/attacks-breaches/top-15-indicators-of-compromise/d/d-id/1140647>
- [9] STIX, TAXII, and CYBOX -  
<https://www.isao.org/resource-library/tools/stix-taxii-and-cybox/>
- [10] What is Open Indicators of Compromise (OpenIOC) Framework -  
<https://cyware.com/educational-guides/cyber-threat-intelligence/what-is-open-indicators-of-compromise-openioc-framework-ed9d>
- [11] Welcome to YARA's documentation! -  
<https://yara.readthedocs.io/en/latest/>
- [12] Cyber Threat Intelligence -  
[https://www.securnite.com/index.php/onepress\\_service/cyber-threat-intelligence/](https://www.securnite.com/index.php/onepress_service/cyber-threat-intelligence/)
- [13] Cyber Threat Intelligence e condivisione delle informazioni: conoscere le minacce per prevenirle -  
<https://www.cybersecurity360.it/soluzioni-aziendali/cyber-threat-intelligence-e-condizione-delle-informazioni-conoscere-le-minacce-per-prevenirle/>

[14] Wireshark -

<https://it.wikipedia.org/wiki/Wireshark>

[15] Wireshark, cos'è e come funziona il packet sniffer più famoso -

[https://www.ilsoftware.it/articoli.asp?tag=Wireshark-cos-e-e-come-funziona-il-packet-sniffer-piu-famoso\\_18842](https://www.ilsoftware.it/articoli.asp?tag=Wireshark-cos-e-e-come-funziona-il-packet-sniffer-piu-famoso_18842)

[16] What is Forensic Analysis? -

<https://enterprise.comodo.com/blog/what-is-forensic-analysis/>

[17] Evolution of Malware Sandbox Evasion Tactics - A Retrospective Study -

<https://www.mcafee.com/blogs/other-blogs/mcafee-labs/evolution-of-malware-sandbox-evasion-tactics-a-retrospective-study/>

[18] Comparing Free Online Malware Analysis Sandboxes -

<https://securityintelligence.com/comparing-free-online-malware-analysis-sandboxes/>

[19] Malware Analysis Automation using Public and Private Sandboxes -

<https://tines.io/blog/malware-analysis-automation-using-public-and-private-sandboxes/>

[20] Come fanno le sandbox a far esplodere i malware senza fare danni -

[https://www.agi.it/blog-italia/cybersecurity/sendbox\\_malware-4823305/post/2019-01-11/](https://www.agi.it/blog-italia/cybersecurity/sendbox_malware-4823305/post/2019-01-11/)

[21] Learn More About CrowdStrike Falcon Sandbox - FAQ -

<https://www.crowdstrike.com/endpoint-security-products/falcon-sandbox-malware-analysis/falcon-sandbox-faq/>

[22] Malware Analysis - Joe Sandbox Cloud -

<https://www.joesecurity.org/joe-sandbox-cloud>

[23] What is Cuckoo? -

<https://cuckoosandbox.org/>