Scuola universitaria professionale della Svizzera italiana
Dipartimento tecnologie innovative

SUPSI

Malware Analysis Framework and CTI Attribution Techniques

Studente/i
Gregori Nicolas

Relatore
Consoli Angelo

Correlatore
Nesurini Mariotti Alice

Committente
Consoli Angelo

Corso di laurea
Ingegneria Informatica

Codice progetto
C10463

Anno
2021/2022

Data 10.06.2022

Ringraziamenti

Vorrei ringraziare qui i miei genitori e miei nonni per il continuo supporto e sostegno espresso durante gli anni di università e in particolar modo per il lavoro di diploma finale.

Un pensiero va anche agli amici e colleghi che ho conosciuto in questi anni: i momenti di svago e di lavoro mi hanno aiutato sia a crescere che come persona che professionalmente.

Infine, dedico questo lavoro a mia nonna che mi guarda e mi veglia da lassù.

Indice generale

Indice delle figure

Abstract

Con la diffusione massiva di Internet, la circolazione di minacce informatiche in questi ultimi decenni è incrementata notevolmente. I malware, spesso generati da aggregati di codici malevoli, infettano sistemi informatici cercando di lasciare il più possibile un loro segno anche a causa degli utenti più inermi. L'unica metodologia di prevenzione è affidata all'analisi di file sospetti, nata allo scopo di mitigare sempre e più questo fenomeno. I processi di analisi statica e dinamica spesso vengono combinati in sistemi di analisi automatizzata, che permettono di analizzare il comportamento di un malware all'interno di un sistema isolato denominato sandbox. Ne esistono diversi in commercio: in particolare Cuckoo Sandbox, cui è stato case-study del presente lavoro e che ha fornito un buon spunto per lo sviluppo di un portale web con funzioni simili. Il sistema sviluppato è in grado di fornire un banco di analisi del malware, appoggiandosi a servizi di terze parti per l'esecuzione dell'analisi tramite l'utilizzo di API. Il presente lavoro ha permesso di sviluppare una maggior consapevolezza e sensibilità a queste tematiche, sperando che anche in futuro numerose persone prestino più attenzione per evitare la diffusione di queste insidie.

Abstract (en)

With the massive spread of the Internet, the circulation of malware has increased considerably in recent decades. Malware, often generated by aggregates of malicious codes, infect computer systems trying to leave their mark as much as possible on even the most defenceless users. The only prevention methodology is entrusted to the analysis of suspicious files, which was created with the aim of increasingly mitigating this phenomenon. Static and dynamic analysis processes are often combined in automated analysis systems, which make it possible to analyse the behaviour of malware within an isolated environment called sandbox. There are several of these on the market: in particular Cuckoo Sandbox, which was a case-study of the present work and which provided a good starting point for the development of a web portal with similar functions. The system developed can provide a malware analysis bank, relying on third-party services to perform the analysis using APIs. This work has made it possible to develop greater awareness and sensitivity to these issues, and it is hoped that more people will pay more attention in the future to avoid the spread of these pitfalls.

STUDENTSUPSI

Scheda progetto

Persone coinvolte

Proponente	Consoli Angelo
Relatore	Consoli Angelo
Correlatore	Mariotti Nesurini Alice

Dati generali

Codice	C10463
Anno accademico	2021/2022
Semestre	Semestre estivo
Corso di laurea	Ingegneria informatica (Informatica TP)
Opzione	O1 Architetture software di rete
Tipologia del progetto	diploma
Stato	proposta
Confidenziale	SI
Pubblicabile	NO

Descrizione

Negli ultimi anni si assiste a un proliferare di tecniche di infezione con nuovi malware polimorfi e sempre più risultanti da aggregazione di codici malevoli di origine diversa combinati da loro per realizzare exploit a più stadi (multistage attacks). Per attaccare un host o un server, possono venir usati trojan, bot, keyloggers, rootkit, worm, e diversi altri tipi di exploit. Per chi lotta contro il cybercrime, la guerra contro i malware negli ultimi anni si è complicata drasticamente. I malware sono sviluppati da veri professionisti e, sempre più sofisticati, si attivano sui sistemi infettati e fanno più danni possibili senza che l'utente si accorga di nulla. Il presente lavoro di tesi vuole realizzare un banco di analisi di malware a diversi livelli per poter identificare la natura dei programmi e il loro modus operandi.

Gli attaccanti sono consapevoli delle tecniche di detezione e analisi del malware, per cui sviluppano strategie sempre più ingegnose per riconoscere se una vittima è reale oppure se il malware è operato all'interno di un ambiente di analisi. Al fine di poter studiare e analizzare i malware risulta quindi indispensabile mascherare la sandbox utilizzata in modo da non venire riconosciuta e anche creare una configurazione di rete plausibile che inganni il malware stesso.

Il progetto si prefigge di analizzare i malware e in seguito permetterà di acquisire le conoscenze necessarie a mappare le strategie di attacco

a specifiche campagne di attacco facenti capo a gruppi malavitosi operanti nella rete.
Lo studio di tecniche di cyber threat intelligence permetterà lo sviluppo di moduli che possano interagire con soluzioni esistenti tramite standard tipici del settore quali ad es. STIX.

Compiti

- Studiare e documentare le varie tecniche di analisi del malware, sia di tipo statico sia dinamico.
- Studiare lo stato dell'arte nel settore specifico per definire un proprio ambiente di analisi e reporting secondo gli standard attivi nel settore della cyber threat intelligence.
- Realizzare un sistema di sandbox in grado di ingannare i principali malware in modo che questi credano di essere attivi in un sistema vittima nativo.
- Detettare e loggare tutti i comportamenti del malware in analisi.
- Sviluppare un'architettura software composta da un back-end e da un front-end applicando principi di security by design. Detettare la presenza di url/link malevoli in un log, a sua volta scaricare il contenuto di tale link e salvare tutte le informazioni possibili (data/ora dell'attacco e di download, dimensione del file, SHA256 e MD5, indirizzo IP e geolocalizzazione).
- Testare scrupolosamente l'infrastruttura sviluppata.
- Scegliere delle strategie di test adeguate alla verifica del sistema, eseguire, e documentare in modo preciso i risultati delle campagne di test effettuate, traendone conclusioni. Documentare il lavoro svolto.

Obbiettivi

- Disporre di un sistema completo in grado di soddisfare gli obiettivi del progetto.
- Disporre di una documentazione tecnica composta da manuale di deployment e manuale d'utilizzo, nonché da una descrizione tecnica che illustri come è strutturato il sistema sviluppato.

Tecnologie

Verranno discusse con il docente responsabile nel corso del progetto.

Contatto esterno

Nessun contatto esterno presente

Documenti allegati

Nessun allegato presente

Capitolo 1

Il malware

Grazie alla diffusione massiva di Internet, i malware sono diventati minacce più concrete nel mondo digitale. Abili e astuti, sono in grado di infettare i sistemi informatici sfruttando una moltitudine di tecniche da quelle praticamente "invisibili" a quelle più notabili. La loro diffusione è dovuta principalmente per effettuare danni irreversibili ai sistemi oppure rubare informazioni personali altrui. Sostanzialmente le metodologie impiegate per bloccarne la diffusione sono due: una maggior sensibilizzazione della tematica agli utenti e l'analisi di campioni sospetti di cui è oggetto il presente lavoro.

1.1 Contesto

Prima della diffusione in massa di Internet, la probabilità di infezione di un sistema era molto ridotta a oggi, in quanto i malware potevano solo circolare attraverso l'uso di supporti fisici, come ad esempio i floppy disk. I primi malware della storia iniziarono ad apparire tra gli anni '70 e '80, ma erano solo considerati dei prototipi creati a scopo dimostrativo e incapaci di diffondersi su larga scala. Il primo malware della storia è stato Creeper, un programma scritto per verificare la possibilità che un codice potesse replicare su macchine remote.
Alla fine degli anni '90 però, con l'esplosione del World Wide Web la metodologia di diffusione cambiò totalmente: l'utilizzo della rete per lo scambio di contenuti e allegati via e-mail incrementarono notevolmente lo scambio di questi software malevoli. Successivamente, verso l'inizio del nuovo millennio. nacque una nuova categoria di malware chiamata worm, la cui diffusione era causata da falle presenti nel sistema operativo e in altri programmi.
Particolarmente simbolico del periodo è la diffusione del worm SQL Slammer, che soltanto 15 minuti dopo il primo attacco, aveva infettato una moltitudine di server creando disagi non poco differenti tra cui: la crisi della Bank of America, il numero 911 americano per le emergenze e anche disservizi in alcune compagnie aeree del paese.
A partire dal 2010 è cominciato l'utilizzo di questi strumenti come armi per compiere vere e proprie guerre (in inglese *cyberwar*) . Lo dimostra il virus Stuxnet diffuso allo scopo di

colpire i sistemi informatici delle centrali nucleari iraniane. Nonostante l'apertura di questo fronte, il target preferito dagli hacker sono sicuramente gli utenti standard. Un esempio pratico di questi ultimi anni è la diffusione di CryptoLocker, comparso nel 2013 e ancora tutt'oggi presente anche se con nomi diversi, programmi atti a cifrare tutti i dati presenti sul disco rigido e chiedere un riscatto per ottenere il codice di blocco.

1.2 Definizione

In ambito della sicurezza informatica, per malware si intende un qualsiasi programma informatico usato per interferire nelle operazioni svolte da un utente di computer.
Di per sé, un malware non viene necessariamente creato per arrecare danni concreti ad un sistema informatico, ma può essere inteso anche come un mezzo per compiere operazioni losche senza essere rilevato per periodi di tempo più lunghi possibili.
Oltre a scovare informazioni nascoste, un malware viene anche creato con l'intento di arrecare danni a infrastrutture informatiche mediante tecniche tra cui: il sabotaggio, cifratura dei dati oppure estorcendo denaro tramite la cifratura dei file.
Più precisamente**,** malware è un termine generico che fa riferimento a varie tipologie di software intrusivo o malevolo. Inoltre, può apparire sotto varie forme: codice eseguibile, script o altro. [1]

1.3 Classificazione

I malware sono conosciuti maggiormente per i modi in cui si diffondono, piuttosto del loro effettivo comportamento. Nel linguaggio comune il termine virus viene usato come malware; ciò è dovuto al fatto che gli antivirus sono in grado di rilevare e rimuovere praticamente svariate categorie di software malevolo, oltre ai virus propriamente detti.
In sintesi, le principali categorie di malware conosciute sono le seguenti[1]:

- *Virus*: sono i più conosciuti e consistono in pezzi di codice che si diffondono copiandosi all'interno di altri programmi, in modo da essere eseguiti ogni volta che il file infetto viene aperto. Si diffondono tramite la condivisione di file infetti.
- *Worm*: a differenza dei più comuni conosciuti virus, non necessitano di altri file per diffondersi, ma tentano di replicarsi attraverso Internet modificando il SO della macchina ospite. Vengono eseguiti sfruttando delle vulnerabilità (bug) oppure da utenti inermi condizionati da tecniche di social engineering.
- *RAT*: sono programmi che consentono all'attaccante di accedere e eseguire comandi da remoto sul sistema ospite. Le sue funzionalità possono essere estese, impiegando altri moduli come ad esempio i keylogger.

- *Trojan*: sono considerati i malware più pericolosi. Si diffondono nascondendosi in file utili al sistema. Una volta entrato nel sistema, i criminali ottengono l'accesso al computer della vittima. A sua volta, possono rubare i dati salvati sul sistema oppure installare altre minacce.
- Spyware: sono programmi che osservano segretamente le attività dell'utente sul computer inviando tutte le informazioni carpite al diretto interessato.
- Scareware: sono porte di accesso che maggiormente si nascondono dietro pop-up pubblicitari e infettano il sistema scaricando programmi che si fingono degli antivirus.
- Backdoor: sono programmi che consentono un accesso non autorizzato su cui sono in esecuzione. Tipicamente vengono diffusi in abbinamento con un trojan o con worm, oppure costituiscono una forma di accesso di emergenza al sistema, ad esempio per il recupero di una password dimenticata.
- *Malvertising o malicious advertising*: sono degli attacchi che originano dalle pubblicità delle pagine web.
- *Rootkit:* sono dei malware che forniscono al criminale i privilegi da amministratore del sistema infetto. Vengono progettati con lo scopo di rimanere nascosti agli occhi degli utenti, degli altri software e dal sistema operativo stesso.
- *Adware*: programmi software che presentano all'utente messaggi pubblicitari durante l'uso, a fronte di un prezzo ridotto o nullo. Possono causare danni quali rallentamenti del PC e rischi per la privacy in quanto comunicano le abitudini di navigazione ad un server remoto.
- *Hijacker:* sono programmi che prendono il controllo di un applicativi web, causando l'apertura di pagine web indesiderate.
- *Rabbit:* sono programmi il cui obiettivo è quello di saturare le risorse del sistema su cui risiede replicando con grande rapidità.
- *Keylogger:* sono programmi in grado di registrare tutto ciò che l'utente digita sulla tastiera oppure i testi copiati e incollati. Viene impiegato soprattutto per rubare password e dati sensibili.
- *File Batch:* non sono considerati veri e propri malware, ma lo possono diventare in base alla natura dello script.
- *A comando*: vengono attività secondo le volontà del cracker nel momento opportuno.
- *Automatici*: sono programmi che si avviano al boot del sistema oppure viene lanciato in esecuzione dall'utente.

Figura 1: Classificazione del malware

1.4. Scopo e propagazione

Il principale modo di propagazione del malware consiste di frammenti di codice parassita, che viene inserito in codice eseguibile già esistente. Tale frammento di codice può essere scritto in codice macchina ed inserito in un: applicativo esistente, codice di utility, programma di sistema o nel codice di boot del sistema operativo.
Un malware in sé è caratterizzato dal suo intento malevolo agendo contro la necessità dell'utente, e non include software che causi danni non voluti e può essere usato per vari scopi: rubare informazioni sensibili, accedere a sistemi informatici privati, o mostrare pubblicità indesiderata. In genere accedono ai dispositivi tramite Internet e i servizi di posta elettronica. Tuttavia, possono anche sfruttare siti Web violati, demo di videogiochi, file musicali, abbonamenti gratuiti o qualsiasi altro materiale scaricabile dal Web in dispositivi privi di protezione antivirus. [1]

1.5. Evitare le minacce

Ovviamente l'unico fattore di diffusione del malware è l'utente che, molte volte ignaro dei pericoli, commette grossi errori di valutazione nel riconoscere o meno se un URL oppure un file malevolo rappresenti una minaccia o meno. Quindi, diventa sempre più rilevante utilizzare semplici accorgimenti che se applicati possono sicuramente evitare spiacevoli situazioni. Sicuramente il primo passo è di installare e mantenere un antivirus scaricato soltanto da siti ufficiali, in quanto riconosco la maggior parte delle minacce presenti in circolazione.
Ancora per limitare il pericolo di infezione, bisogna essere molto cauti con link e allegati la cui provenienza è dubbia e dare agli account di sistema soltanto i permessi necessari per svolgere le loro mansioni. Le password sono una dei target che fanno più gola ai malware: perciò evitare di utilizzare la stessa password per più servizi e delegarne la gestione a un password manager. Infine, è opportuno mantenere tutto il lavoro sempre

aggiornato all'ultima versione, effettuare dei backup regolari e utilizzare un firewall a protezione del sistema.

Va infine aggiunto che oramai la maggior parte dei malware è sviluppata da veri e propri professionisti che conoscono molto bene le vulnerabilità dei sistemi sotto attacco e come coinvolgere gli utenti nell'attivare eventuali "trappole" nascoste. In questo contesto, l'analisi di malware è un'attività fondamentale per la ricerca di queste minacce con conseguente sviluppo di nuove contromisure di protezione.

Capitolo 2

Analisi del malware

I malware sono una minaccia sempre più presente nel mondo informatico; perciò, è opportuno utilizzare tecniche di prevenzione atte a estirpare il più possibile questo tipo di minacce. Una di queste è l'analisi di malware praticata da molti ricercatori e professionisti nell'ambito della cybersecurity. Le modalità di analisi si suddividono di norma in tre tipologie: statica, dinamica e ibrida, ognuna focalizzata su determinati aspetti. Però dato che il processo di analisi è lungo e richiederebbe numerosi tools, oggigiorno esistono dei tools automatizzati che forniscono un'analisi completa con tanto di reportistica dettagliata.

2.1. Definizione e obiettivi

Per analisi del malware si intende quel processo di analisi di un file allo scopo di estrarre più informazioni utili e possibili. Le informazioni estratte permettono di comprendere appieno le principali funzionalità del malware, come il sistema ospite viene infettato con il malware e quali misure adottare per rispondere a simili minacce nel futuro prossimo.

Un vantaggio principale dell'analisi del malware è quello di aiutare gli incidenti rispondere gli analisti della sicurezza per i seguenti scopi:

- Catalogare il tipo di malware, scovare le funzionalità e capire in quale contesto opera.
- Comprendere gli effetti che il malware produce sul sistema infetto e categorizzare l'attacco subito.
- Comprendere le modalità con la quale il malware comunica con l'attaccante.
- Classificare gli incidenti in base alla loro gravità.
- Filtrare più *IOC* (Indicatior of Compromise) possibili che permettano di identificare il comportamento del malware in attacchi futuri.
- Migliorare l'efficacia degli avvisi e delle notifiche degli IOC.

2.2 Casi d'uso

Oltre al contributo di minimizzare la "buona riuscita" degli attacchi, è altrettanto opportuno comprendere in quali contesti è necessario applicare questo processo. Di seguito sono riportati alcuni esempi di use case:

2.2.1 Rilevamento del malware

Oggigiorno i malware utilizzano tecniche sempre più sofisticate e robuste per bypassare i moderni sistemi di rilevamento. Di conseguenza, fornire un''analisi comportamentale sempre più approfondita e identificando il codice condiviso, le funzionalità o l'infrastruttura dannosa, le minacce possono essere rilevate in modo più efficace.
L' output dell'analisi in sé consiste nell'estrazione degli *IOC* (gli indicatori di compromissione) che inseriti in *SIEM* (piattaforme di intelligence sulle minacce), permettono di avvisare i team di minacce correlate in futuro.[2]

2.2.2 Incident response

Lo scopo dell'IR è quello di fornire un'analisi della causa principale del danno, determinare l'impatto e attuare un piano di riparazione e ripristino del sistema infetto; perciò, il processo di analisi favorisce efficienza ed efficacia anche in questo contesto.[2]

2.2.3 Caccia alle minacce

L'attività di analisi può esporre comportamenti e artefatti che i threat hunter possono utilizzare per trovare attività simili; ad esempio, cercando l'accesso a una particolare connessione di rete, porta o dominio. Cercando nei firewall e proxy o nei dati IEM, i team possono utilizzare questi dati per trovare minacce simili.[2]

2.2.4 Ricerca accademica

I ricercatori di malware accademici o industriali eseguono questi tipi di analisi per acquisire una maggior comprensione delle tecniche, exploit e strumenti usati da avversari. [2]

2.3 Fasi di un processo di analisi

2.3.1 Analisi delle proprietà statiche

Le proprietà statiche includono stringhe incorporate nel codice malware, dettagli dell'intestazione, hash, metadati e risorse. Questi tipi di dati sono acquisibili molto rapidamente, in quanto non è necessario eseguire il codice sorgente per vederli, ma basta utilizzare dei tools ad hoc. Le informazioni raccolte durante questa fase di analisi danno un'indicazione se è necessaria o meno un'indagine più approfondita, impiegando così tecniche più complete e determinare così i passi successivi da compiere.

2.3.2 Analisi comportamentale

L'analisi comportamentale viene utilizzata per osservare e interagire con un campione di malware in esecuzione in un laboratorio. Gli analisti cercando di comprendere il registro del campione, il file system, i processi coinvolti e i pacchetti scambiati in rete. Possono anche essere condotte analisi di memoria per scoprire come il malware utilizza la memoria. Se vi è il sospetto che la memoria venga compromessa, è opportuno impostare una simulazione per confermare la teoria.

A differenza dell'analisi statica, quella comportamentale richiede competenze avanzate. È un processo abbastanza lungo e perciò non può essere eseguito senza tools automatizzati.

2.3.3 Analisi automatizzata

L'analisi automatizzata valuta in modo rapido e semplice se un file sia sospetto o meno. Può determinare potenziali ripercussioni se il malware dovesse infiltrarsi nella rete e quindi produrre un report di facile lettura che fornisca informazioni rapide agli analisti. È sicuramente la metodologia migliore per analizzare un malware in tempi rapidi.

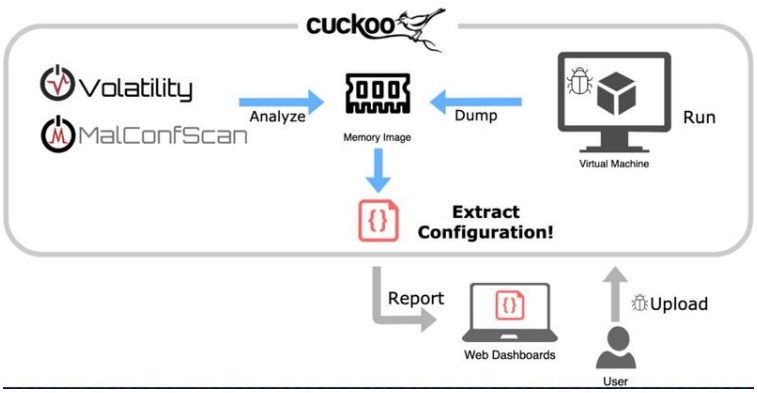


Figura 2: Esempio di un sistema di analisi automatizzato - Il caso Cuckoo

2.3.4 Inversione del codice

In questa fase gli analisti cercano di effettuare operazioni di reverse-engineering sul codice del campione analizzato utilizzando strumenti tra cui: debugger, disassembler e compilatori. Tali strumenti sono specializzati per decodificare i dati crittografati, determinare la logica dietro l'algoritmo del malware e comprendere eventuali funzionalità nascoste che il malware non ha ancora esibito. Effettuare reverse-engineering sul codice è un'attività molto onerosa in termini di tempo e soprattutto molto rara. Per questa motivazione spesso le indagini sul malware saltano questo passaggio con conseguente perdita di informazioni preziose sullo stesso.

2.4. Modalità di Analisi

2.4.1 Analisi statica

L'analisi statica di base non richiede un'esecuzione del codice sorgente, ma si concentra maggiormente sulla ricerca di indizi nocivi. Viene impiegata per identificare infrastrutture, librerie oppure file compressi dannosi.

Il punto di partenza è sempre quello di tutto occorre stabilire il tipo del target, in quanto fornisce già informazioni sul sistema operativo che potrebbe essere soggetto ad attacco. Anche se questa informazione è reperibile rapidamente, esistono comunque dei tool che forniscono una visione più completa; ad esempio, in ambiente Linux si può utilizzare il comando built-in nella shell *file* che fornisce una descrizione dettagliata del tipo (non si limita a mostrarne solo l'estensione).

Invece, l'utilizzo delle hash serve a identificare l'autenticità del file; quindi, se si combacia con quella del mittente si può venire a conoscenza se il campione posseduto sia effettivamente lo stesso o meno. Però non vi sono garanzie, di fatto ci sono algoritmi il cui valore di hash per file diversi potrebbe essere lo stesso: in questo caso si parla di collisione. È il caso dell'algoritmo MD5 ritenuto non sicuro, in quanto non è così scontato trovare delle collisioni, poiché la dimensione della stringa ritornata è di soli 128 bit. Dato che più la stringa di hash è lunga più l'algoritmo è sicuro, è necessario utilizzare altri algoritmi differenti da questo per garantire l'integrità dei dati. Per questa necessità ci sono altri algoritmi che utilizzano stringhe di dimensioni maggiori tra i quali:

- *SHA-1*: stringhe di 160 bit.
- *SHA-256*: stringhe di 256 bit.
- *SHA-512*: stringhe di 512 bit.

Diverso è invece l'approccio utilizzato da funzioni di hashing chiamate *fuzzy* come *ssdeep*. *ssdeep* è un algoritmo che si basa sulle similarità e crea rappresentazioni di file nel modo seguente: *chunksize: chunk:double_chunk***.** La differenza tra i due approcci di hash è abbastanza evidente: se si utilizzano signature tradizionali anche la minima variazione di un bit darà un risultato totalmente differente, mentre ciò non è influenzato minimamente nel calcolo del *ssdeep*.

Altre signatures utilizzate frequentemente sono l'import hash e l'authentihash. La prima è calcolata per tutte le librerie DLL e tutte le funzioni che vengono importate all'interno dell'eseguibile. Inoltre, a dipendenza dell'ordine in cui vengono importate il valore calcolato della funzione cambia radicalmente.

L'utilizzo di strumenti, quali debugger e disassembler, rappresenta un punto fondamentale in questa fase di analisi, soprattutto del formato PE.
Il formato PE è un formato proprietario di Windows, usato in particolar modo da file quali eseguibili e librerie dinamiche. Esso contiene tutte le istruzioni necessarie che, date in input al loader di Windows, permettono di eseguire il file. Eseguendo un'analisi è possibile ottenere diverse informazioni, tra cui diverse proprietà, il tipo di linking utilizzato e le procedure utilizzate dal file in analisi.
I campi analizzati nell' header sono i seguenti:

- *Image base:* è l'indirizzo prefissato dove l'immagine viene caricata in memoria. Solitamente il default è 0x400000. L'attaccante può cambiare questo indirizzo in base a determinati requisiti con un'opzione del tipo '-BASE:linker". [4]
- *Entry Point Name*: indica la prima sezione dell'applicativo caricata in memoria per l'esecuzione. [4]
- *Entry Point Address*: è l'indirizzo di memoria dove inizia l'esecuzione del codice sorgente dell'applicativo. Questo contiene l'indirizzo virtuale dell'entry point del modulo e solitamente si trova nella sezione *.text* . Per i driver dei dispositivi è l'indirizzo di inizializzazione della funzione, mentre per le DLL è opzionale. Se non è presente il valore vale 0. [4]
- *Entry Point Entropy*: è l'entropia dell'area di memoria dove inizia l'esecuzione del codice. [4]
- *Characterstics address*: l'indirizzo di memoria dove puntano i riferimenti ai flag descritti nel punto sottostante. [4]
- *Characterstics Readable*: sono dei flag che si riferiscono a un attributo dell'oggetto o del file immagine. [4]
- *Major Version Number***:** versione major dell'applicativo [5].
- *Minor Version Number***:** version minore dell 'applicativo [5].
- *Pointer to Symbol Table*: come dice il nome stesso, indica l'indirizzo dove si trova la tabella dei simboli usata dal compilatore. [5].
- *Number Of symbols***:** numero di righe contenuti nella tabella dei simboli. [5]
- *Compiler Flags*: flag di compilazione usati per la compilazione del binario [5]

Il formato PE presenta le seguenti sezioni [4]:

- *.text*: contiene il codice eseguibile dell'applicativo. Di norma è la prima sezione esistente e costituisce il punto di ingresso dell'applicazione, ovvero la prima istruzione che viene eseguita quando il programma viene caricato in memoria. Da notare che un applicativo potrebbe avere più punti di entrata.
- *.rdata*: contiene contenente le variabili globali accessibili in sola lettura e la tabella delle API che l'applicativo utilizza (compresi il nome e la DLL a cui appartengono)
- *.data***:** contiene le variabili globali accessibili da tutto il programma.
- *.rsrc*: contiene le risorse di cui l'eseguibile necessita per l'esecuzione.
- *.debug:* contiene informazioni di debugging per l'applicazione.

Non meno importanti sono le sezioni chiamate*. edata* e *.idata* . La sezione *.edata* indica quali funzioni la libreria esporta agli utenti. Al contrario la sezione *.idata* specifica quali funzioni e dati vengono importati e da quali DLL provengono.
Tuttavia, quelli riportati sono dei nomi standard di sezioni specifiche, ma non è detto che vengano sempre utilizzate con lo stesso nome o scopo.
Ogni sezione contiene le seguenti informazioni principali:

- *SizeOfRawData:* specifica la dimensione reale della sezione del file.
- *VirtualSize*: indica la dimensione della sezione caricata in memoria.
- *VirtualAddress*: indica l'RVA della sezione in memoria.
- *PointerToRawData*: l'offset relativo a dove inizia la sezione dati del file. Sommando tale valore all'indirizzo di memoria virtuale è possibile ottenere l'indirizzo da dove inizia la prossima sezione.
- *Characterstics:* specifica i privilegi di accesso che la sezione possiede in memoria mediante l'utilizzo di appositi flag. Questi flag specificano se la sezione possiede privilegi di lettura, scrittura, esecuzione o una combinazione di questi.

Le *version info properties*, definite come coppie chiave-valore, forniscono informazioni basandosi sulle risorse incorporate nel file. Tali proprietà sono spesso integrate in file binari; i file di testo invece non dispongono di queste informazioni. [6]

Il rich header è un header aggiuntivo proprietario non documentato e contenuto nei file PE compilati. È utile in quanto fornisce informazioni sull'ambiente di compilazione in cui è stato creato il file PE.

Un altro aspetto su cui porre attenzione è la ricerca delle stringhe, in quanto permette di ottenere dei suggerimenti riguardo le funzionalità del file. Per esempio, con questa analisi si può venire a conoscenza di tutte le chiamate a sistema fatte, le dipendenze utilizzate e l'eventuale utilizzo di URL o indirizzi IP., Microsoft possiede un tool di utilities chiamato "Strings". Quando il comando viene lanciato su un eseguibile, cerca in esso tutte le stringhe in formato ASCII e Unicode filtrando solamente quelle con almeno tre caratteri. Il comando però potrebbe anche detettare sequenze di byte (caratteri) che di sé non hanno utilità per l'analisi. [3]
Però, l'output di tutte queste ricerche non fornisce sempre un risultato di garanzia: poiché il codice effettivamente non viene eseguito, un malware sofisticato potrebbe includere comportamenti dannosi a runtime che rischiano così di non essere rilevati.
Ad esempio, se un file genera una stringa impiegata per il download di un file dannoso basandosi su questa stringa dinamica, questo comportamento potrebbe non essere identificato da un'analisi statica di base.
Questa è la motivazione cardine che ha spinto le aziende a rivolgersi all'analisi dinamica, per una comprensione più completa del comportamento del file.

2.4.2 Analisi dinamica

L'analisi dinamica del malware si basa su un approccio totalmente diverso rispetto al precedente: in quanto il codice viene eseguito in un ambiente sicuro chiamato sandbox. Questo sistema isolato consente ai professionisti in azione di osservare il malware in azione senza il rischio che possa infettare il proprio sistema e di sfuggire alla rete aziendale. L' analisi dinamica offre una visione più profonda consentendo di scoprire l'esatta natura di una minaccia, in quanto studia l'interazione del malware con l'ambiente circostante. In particolare, vengono monitorati:

- I processi messi in esecuzione dal malware, le loro proprietà ed eventuali anomalie.
- Le interazioni del malware con il file system.
- L'accesso e la modifica di chiavi e dati sul registry.
- Il traffico di rete generato.

In primo luogo, dato che la maggior parte dei malware comunica con host presenti sulla rete, è bene che ciò non avvenga, ma per far sì che il campione continui a funzionare, è necessario fornire tutti i servizi necessari. È necessario allora utilizzare dei tool di simulazione, tra cui:

- *INetSim*: progetto open source che gira sotto Linux e permette di simulare diversi servizi standard, tra cui DNS, HTTP e SMTP.
- *FakeNetG*: tool che gira sotto Windows è considerato uno dei migliori in circolazione per effettuare penetration testing e analisi di rete di malware.

Per il monitoraggio del traffico di rete, generato dall' esecuzione del malware, si usa solitamente Wireshark: uno sniffer di pacchetti che consente di catturare il traffico di rete. L'analisi del traffico di rete risulta utile per comprendere il canale di comunicazione utilizzato dal malware e determinare così gli IOC basati sulla rete.
Invece per il monitoraggio delle risorse di sistema alcuni tool fondamentali sono:

- *Process Hacker*: esamina i processi in esecuzione sul sistema e per ispezionare i relativi attributi. Viene utilizzato, una volta lanciato il malware, per identificare il processo, eventuali sottoprocessi e tutti i loro attribuiti.
- *Process Monitor*: è uno strumento di monitoraggio molto avanzato che mostra l'interazione in tempo reale dei processi con il filesystem, il registry e altre attività di rete.

Anche l'analisi della memoria RAM è uno strumento molto potente per la ricerca dei malware, soprattutto quando adotta stratagemmi per rimanere nascosto. Però il punto di vista di quest'analisi è totalmente diverso da precedenti, in quanto viene avviata solamente nei casi in cui si sospetti la presenza di un malware nel sistema. Uno dei tool più famosi in questo ambito è certamente Volatility che presenta diversi plugin utili a tale

scopo. Una volta individuati processi malevoli, sarà possibile estrarli dalla memoria ottenendo degli eseguibili e analizzarli con le tecniche descritte precedentemente.
La vera sfida con questo tipo di analisi è che gli hacker astuti sono a conoscenza dei sistemi di sandbox, diventando anche bravi ad aggirare questo ambiente. Per ingannare un sistema di sandboxing, gli avversari nascondono al loro interno del codice che potrebbe rimanere inattivo fino a quando non vengono soddisfatte determinate condizioni.

2.4.3 Analisi ibrida

Come ribadito, l'analisi statica non è un modo affidabile per rilevare un codice malevolo e sofisticato e talvolta il malware può nascondersi dalla presenza del sandbox. Combinando sia tecniche di analisi statiche che dinamiche, l'analisi ibrida fornisce al team di sicurezza l'output migliore, principalmente perché è in grado di rilevare il codice dannoso che si cela dietro al malware e di conseguenza può estrarre molti più IOC dalla decodifica statica. L' analisi ibrida aiuta a rilevare le minacce sconosciute, anche quelle provenienti dai malware più insidi.
Un caso d'uso che dimostra l'utilità dell'analisi ibrida è il seguente: uno dei "pattern" dell'analisi ibrida è applicare un approccio statico ai dati generati da un'analisi comportamentale quando avvengono dei cambiamenti nella memoria. L'analisi dinamica lo rivelerebbe e gli analisti verrebbero avvisati di tornare indietro ed eseguire un'analisi statica di base su quel dump di memoria. Di conseguenza verrebbero generati più IOC e verrebbero esposti gli exploit zero-day.

2.5 Strumenti utilizzati

Nell'ambito dell'analisi del malware ci sono anche degli strumenti utili che vengono utilizzati costantemente da esperti del settore per rilevare eventuali minacce.

2.5.1. Yara

Yara è un tool sempre più indispensabile impiegato da molti professionisti per la ricerca e la detection di malware. *Yara* utilizza un approccio rule-based, ovvero permette di utilizzare descrizioni per la ricerca di malware della stessa famiglia basandosi su pattern testuali o binari. [8]

Una *Yara* rule è composta dalle seguenti sezioni:

- Metadati: informazioni aggiuntive sulla regola, ad esempio nome dell'autore, data e descrizione della regola. Sono opzionali per la scrittura della regola.
- Identificatori
- Stringhe: contiene l'insieme delle stringhe che l'analisi statica del malware potrebbe fornire.
- Condizioni: regole logiche usate per la detezione delle stringhe e degli indicatori forniti.

2.5.2 Mitre Attack

Mitre Attack un database di tattica di attacco, realizzato mediante l'osservazione di scenari di attacco reali. Tale database permette di progettare delle tecniche difensive adeguate a difendersi da ogni malware. Il framework è suddiviso in tattiche di attacco le quali, per essere completate, devono essere applicate diverse tecniche.

2.5.3. Virus Total

Virus Total è un sito online che permette di scovare malware all'interno di file o URL. Utilizza più di 70 software antivirus, tra cui i più noti: Kaspersky, BitDefender, Avira e Malwarebytes. Inoltre, per ogni motore di antivirus utilizzato è possibile consultare un report dettagliato riguardo ad ogni singolo oggetto controllato. *Virus Total* permette di

analizzare un campione sia modalità statica che dinamica. Nella modalità statica il tool non esegue il campione ma piuttosto effettua un confronto delle varie signatures dei diversi file caricati e li confronta con i database. Invece, nella modalità dinamica *Virus Total* utilizza proprio Cuckoo Sandbox per analizzare il file caricato.

Capitolo 3

Cuckoo Sandbox

Il primo passo da compiere è quello di scegliere il sistema di sandboxing dove fare girare i campioni. Uno di questi presenti sul mercato è Cuckoo che fornisce veramente un quadro di analisi completo per ogni campione caricato al suo interno. Nel presente capitolo verranno analizzate le principali peculiarità del sistema, così come anche alcune sue criticità. Il sistema è composto da due parti: l'host principale e le macchine Guests. Il sistema installato è stato completamente virtualizzato tramite l'utilizzo di VMware ESXi, con sistema sottostante Linux Ubuntu Versione 20.04 LTS. Le macchine guest invece sono state create utilizzando come sistema operativo Windows 7, che sebbene antiquato, è considerato uno dei migliori per la semplicità di configurazione. Inoltre, per motivazioni "storiche" Windows 7 è la piattaforma per la quale sono stati realizzati e pensati più malware.

3.1 Concetto di Sandbox

Un sandbox è un meccanismo di sicurezza utilizzato per testare codice sospetto o programmi forniti da utenti terze parti, utenti e URL inaffidabili [11].
Questo concetto si applica in modo particolare nell'ambito dell'analisi del malware: di fatto l'obiettivo è di eseguire applicativi sconosciuti e sospetti in un ambiente isolato per ottenere più informazioni possibili [10].
Come ribadito nel capitolo precedente, il sandboxing è una pratica comune dell'analisi dinamica di un malware: invece di analizzare staticamente il target, viene eseguito e monitorato in tempo reale [10]. Anche questo approccio ha sicuramente dei lati positivi e negativi, ma permette di ottenere dettagli sul comportamento del file in analisi.
Però, la buona pratica consiglia di combinare sia analisi statica che dinamica, allo scopo di ottenere più dettagli possibili del target analizzato.

3.2 Introduzione a Cuckoo Sandbox

Cuckoo Sandbox è un software open-source multipiattaforma automatizzato per l'analisi del malware, composto da diversi moduli scritti in Python, consente l'esecuzione di file e URL sospetti in ambiente isolato. Attualmente il progetto è stato archiviato su GitHub, in quanto una sua riscrittura è in corso.
Tra le sue peculiarità, Cuckoo svolge le seguenti funzionalità [9]:

- *Registrazione delle chiamate di sistema effettuate dal processo.*
- *Analisi dei file creati durante l'esecuzione del processo.*
- *Dump di memoria.*
- *Analisi del traffico di rete.*

Figura 3: Cuckoo Logo

3.3. Casi d'uso

Cuckoo Sandbox è una piattaforma che può essere utilizzata per analizzare diverse tipologie di file tra cui [9]:

- *Eseguibili di Windows.*
- *Script PHP e Python.*
- *URL e file HTML.*
- *Archivi ZIP e RAR.*
- *Eseguibili in Java.*

3.4 Architettura

L'architettura di Cuckoo consiste principalmente in un software di gestione centralizzato, che si occupa di lanciare l'esecuzione dell'analisi di file e URL.
Ogni analisi viene lanciata in un ambiente pulito e isolato che può essere virtuale o fisico.
Come da immagine sottostante, i componenti principali dell'architettura sono:

- *Cuckoo Host*: è responsabile dell'esecuzione delle analisi sulle macchine guests, nonché di generare i report e i dump risultanti del processo.
- *Guest machines*: ambienti puliti e isolati dove viene lanciato il malware o l'url da analizzare.

L' host e le macchine guest comunicano mediante una rete virtuale isolata e dedicata.

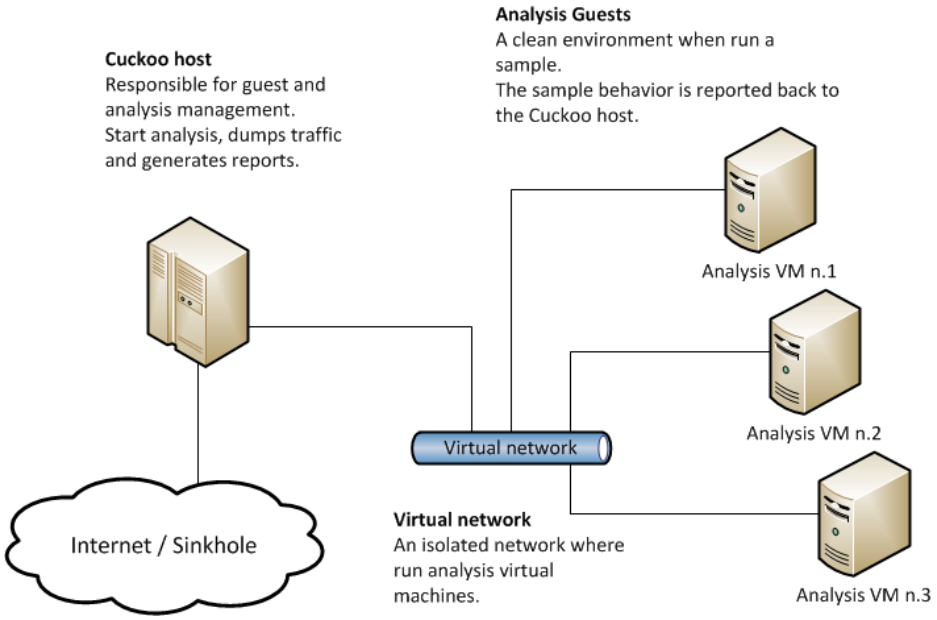


Figura 4: Architettura Cuckoo

3.5 Installazione

Sebbene sia uno dei più buoni sistemi di sandboxing presenti sul mercato, Cuckoo non è così semplice e intuitivo da installare, in quanto richiede diverse dipendenze aggiuntive per funzionare. Una volta completata l'installazione è necessario configurare le macchine host e configurare l'ambiente per le prime analisi.
Nel caso del presente lavoro, Cuckoo è stato installato su un sistema Linux Ubuntu 20.04 LTS virtualizzato. Le seguenti dipendenze sono necessarie per eseguire Cuckoo correttamente:

```
sudo apt-get install libjpeg-dev zlib1g-dev swig
sudo apt-get install libffi-dev libssl-dev
```

Per funzionare Cuckoo ha bisogno necessariamente della versione 2.7 di Python per funzionare e di pip2; perciò, occorre procedere come segue[11]:

```
sudo apt-add-repository universe
sudo apt install python2-minimal
sudo apt update
sudo apt install curl
curl https://bootstrap.pypa.io/pip/2.7/get-pip.py --output get-pip.py
sudo python2 get-pip.py

pip2 install -U virtualenv setuptools dev-tools
```

Per utilizzare l'interfaccia web basata su Django, è necessario installare MongoDB [12]:

```
sudo apt-get install mongodb
```

Per memorizzare tutti i vari report di analisi è consigliato l'utilizzo di Postgresql, installabile mediante il seguente comando[12]:

```
sudo apt-get install postgresql libpq-dev psycopg2
```

In modo tale da ottenere un dump dell'analisi di rete svolta dal malware in esecuzione, è opportuno installare un packet sniffer appropriato configurato per catturare il traffico e registrarlo su un file. Per default Cuckoo utilizza `tcpdump`:

```
sudo apt-get install tcpdump apparmor-utils
sudo aa-disable /usr/sbin/tcpdump
```

Per funzionare `tcpdump` richiede i privilegi dell'utente root, perciò a meno che non si voglia eseguire Cuckoo come amministratore eseguire i seguenti comandi:

```
 sudo groupadd pcap
 sudo usermod -a -G pcap cuckoo
 sudo chgrp pcap /usr/sbin/tcpdump
 sudo setcap cap_net_raw,cap_net_admin=eip /usr/sbin/tcpdump
```

Per testare l'esito dei comandi precedenti eseguire il seguente:

```
getcap /usr/sbin/tcpdump
/usr/sbin/tcpdump = cap_net_admin,cap_net_raw+eip
```

Se `setcap` non dovesse essere installato, eseguire il seguente comando:

```
sudo apt-get install libcap2-bin
```

Oltre queste dipendenze ritenute essenziali, è consigliabile installare altre librerie per migliorare la qualità del report di analisi. La prima è `M2Crypto`, libreria wrapper di `OpenSSL` e comprende diversi algoritmi di cifratura. Viene installata allo scopo di fornire maggior sicurezza alle applicazioni in Python e fornisce un servizio di crittografia. Dipende da `swig` e dalla libreria `libssl1.0-dev`, per installarla utilizzare il seguente comando:

```
pip install -u m2crypto == 0.24.0.
```

È importante mantenere la versione di questa libreria per assicurarsi che il tutto funzioni correttamente.

Volatility invece è una libreria impiegata per eseguire analisi sui dump di memoria del campione dato. È in grado di individuare molto in profondità i cambiamenti fatti al sistema così come è in grado di individuare la presenza di rootkit. Supponendo già di avere **git** installato sulla macchina, eseguire i seguenti comandi per installarla:

```
git clone https://github.com/volatilityfoundation/volatility.git
```

Entrare nella directory appena clonata ed eseguire:

```
sudo python setup.py install
```

Distorm3 è un decomposer: prende in input delle istruzioni e ritorna una struttura binaria che descrive il malware, piuttosto che semplice testo statico. Per installarla eseguire i seguenti comandi:

```
git clone  https://github.com/gdabah/distorm/releasese
tar -zxvf distorm-3.4.1.tar.gz
```

Entrare nella directory appena scompattata ed eseguire i seguenti comandi:

```
sudo python setup.py install
sudo apt-get install libjansson-dev libmagic-dev
sudo apt-get install libtool-bin
```

PyCrypto (Python Cryptography) è una collezione di varie implementazioni di algoritmi di cifratura e di hash crittografiche sicure. Nuovamente, i comandi per installarla sono i seguenti:

```
sudo -H pip install pycrypto
sudo -H pip install ansible –upgrade
sudo -H pip install IPython==5.0
sudo -H pip install jupyter
sudo -H pip install openpyxl
sudo -H pip install ujson
```

Cuckoo può essere eseguito dall'utente corrente oppure può esserne creato uno nuovo per effettuare il setup. Per creare un nuovo utente digitare il seguente comando:

```
sudo adduser cuckoo
```

Per utilizzare VirtualBox assicurarsi che l'utente appena creato appartiene al gruppo `vboxusers`:

```
sudo usermod -a -G vboxusers cuckoo
```

Dopo tutta questa fase iniziale di setup si può finalmente installare Cuckoo; è consigliabile installarlo in un ambiente virtuale per i seguenti motivi:

- È considerata una buona pratica.
- Le dipendenze di Cuckoo potrebbero non essere aggiornate completamente, ma invece utilizzano una versione compatibile con quella del sistema installato.
- Usando un ambiente virtuale permette anche agli utenti non aventi privilegi di amministratore di installare dipendenze addizionali oppure di aggiornare Cuckoo.
- Le dipendenze di altri software potrebbero essere in conflitto con quelle usate da Cuckoo (soprattutto per problematiche di versione).

Quindi digitare:

```
virtualenv venv
. venv/bin/activate
pip install -U pip setuptools
pip install -U cuckoo
```

Dopo aver concluso la lunga procedura di installazione, eseguendo il comando `cuckoo init` per la prima volta, allo scopo di controllare lo stato. Oltre a fornire un feedback dell'installazione, il comando creerà nella directory utente una directory nascosta denominata `.cuckoo` in `$CWD/conf .` Nella sottodirectory `.conf` si trovano tutti i file di configurazione che rendono Cuckoo operativo.

Infine, è necessario installare il software di virtualizzazione (nel nostro caso è VirtualBox) nel modo seguente:

```
sudo apt-get install virtualbox
```

3.6. Configurazione

3.6.1 Configurazione Utenti Guest

Come ribadito nel capitolo precedente, come software di virtualizzazione si è deciso di utilizzare Virtualbox con un sistema operativo Windows 7. Dopo l'installazione, è opportuno effettuare i seguenti passaggi per trasformare una macchina virtuale in un sistema di sandboxing:

- Disabilitare tutte le protezioni di Windows tra cui l'UAC (User Access Control) e Windows Defender.
- Disabilitare Windows Update.
- Installare programmi di utility generale tra cui: Mozilla Firefox, 7zip, MS Office, Adobe Reader PDF e Adobe Flash Player.
- Installare Python versione 2.7 (quella utilizzata da Cuckoo) e la libreria Python pillow utilizzata per il salvataggio degli screenshot di analisi.
- Definire alla macchina un indirizzo IP statico in modo tale che possa comunicare con l'host di Cuckoo, agganciando una delle interfacce di rete virtuali della macchina a una di quelle create su VirtualBox.
- Installare lo script agent.py in modo tale che esso si avvia al boot della macchina ed è necessario per far comunicare le due parti.
- Effettuare uno snapshot della macchina in questo stato, che sarà utilizzato da Cuckoo per ripristinare il sistema di sandboxing allo stato originale dopo l'analisi.

Ogni macchina virtuale utilizzata deve essere registrata in un file di configurazione apposito a dipendenza del sistema di virtualizzazione utilizzato. In tal caso, dato che il sistema usato è VirtualBox, registrare le informazioni nel file `virtualbox.conf` con il seguente formato come da esempio:

```
[virtualbox]
mode = gui
machines = win7 # lista di vm usate

[win7] # nome della virtual machine registrata

label = win7 # nome della virtual machine registrata

platform = windows
ip = 192.168.56.1 #IP statico della macchina
snapshot = ready # nome dello snapshot usato
```

3.6.2 Configurazione Cuckoo Host

Cuckoo necessita la configurazione di una base di dati per funzionare correttamente:

```
sudo -u postgres psql
CREATE DATABASE cuckoo;
CREATE USER cuckoo WITH ENCRYPTED PASSWORD 'PASSWORD';
GRANT ALL PRIVILEGES ON DATABASE cuckoo to cuckoo.
\q
```

Dopo aver creato il database è necessario aprire il file `cuckoo.conf` e inserire la stringa riporta qui sotto nella sezione indicata:

```
[database]
```

```
connection = postgresql://cuckoo:@password@localhost/cuckoo
```

Rispetto alla configurazione delle macchine guests, la configurazione del server è più semplice e richiede di cambiare i seguenti parametri:

auxiliary.conf

```
[sniffer]
enabled = yes
```

processing.conf

```
[memory]
enabled = yes
```

3.7 Utilizzo

Per utilizzare Cuckoo basta digitare il comando `cuckoo` sul terminale bash di Linux. Dopo aver cercato eventuali aggiornamenti, il sistema si mette in attesa di file da analizzare.
Però prima di iniziare ad effettuare le analisi è buona pratica utilizzare il comando `cuckoo community --force,` che scarica tutte le signatures presenti nel server di Cuckoo. Inoltre, digitando il comando `cuckoo --help` è possibile dare un'occhiata a una breve documentazione contenente i comandi principali. Per eseguire le analisi si possono utilizzare due approcci differenti:

- Utilizzare il comando `submit,` però prima è necessario far partire il vero motore di Cuckoo, ovvero il suo daemon lanciabile con il comando `cuckoo -d.`
- Caricare il malware utilizzando l'interfaccia web di Cuckoo che verrà descritta nel paragrafo successivo.

3.8. Interfaccia Web di Cuckoo

Cuckoo presenta anche in sé un portale basato su Django utilizzato per caricare un file da analizzare e/o un link malevolo. Inoltre, è anche possibile visualizzare tutti i report delle analisi fatte.
Per abilitare l'interfaccia web il procedimento è abbastanza semplice:

- Avviare MongoDB nel seguente modo: sudo `service mongodb start`
- Nel file di configurazione `reporting.conf` abilitare il modulo di reporting di MongoDB semplicemente sostituendo non con yes e pois salvare le modifiche
- Eseguire l'applicazione con questo comando: `cuckoo web runserver` [16].

reporting.cong

```
[mongodb]
 enabled = yes
```

Di default l'applicazione gira in localhost sulla porta 8000, ma è possibile cambiare queste impostazioni con il seguente comando:

```
cuckoo web runserver IP: PORT
```

Rimpiazzare IP e PORT con i valori desiderati.

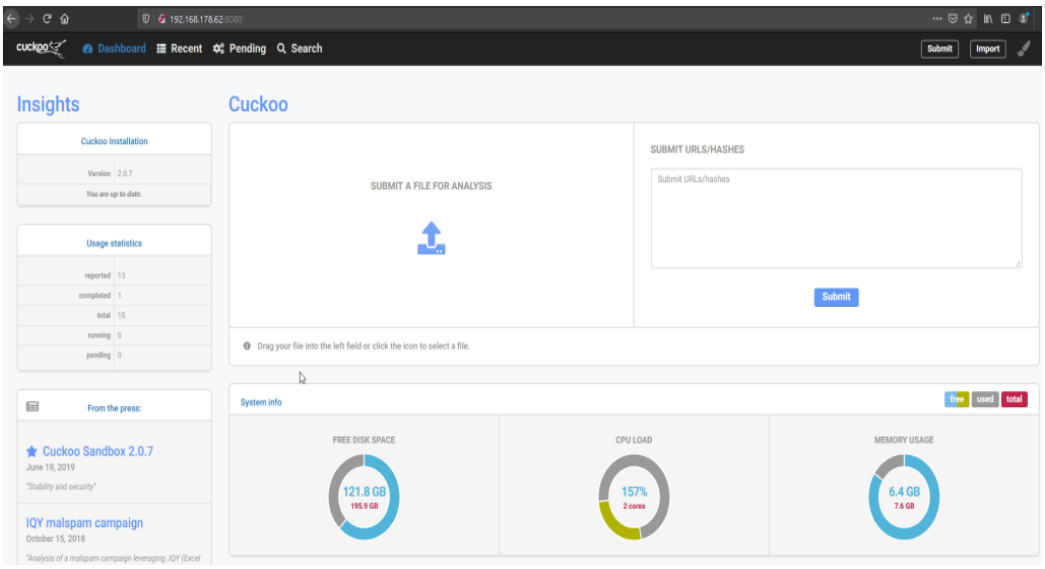


Figura 5: Interfaccia Web di Cuckoo

3.9 Cuckoo Rest API

Cuckoo fornisce anche un'API Rest, semplice e leggera, che può essere utilizzata da applicazioni di terze parti per analizzare i malware e URL malevoli. Sotto il cofano di questa API gira un server utilizzando Flask.
Per far partire il server API è sufficiente digitare il seguente comando: `cuckoo api`. Di default il servizio viene lanciato in localhost sulla porta 8090. Se si vogliono cambiare questi valori utilizzare il comando seguente:

```
cuckoo api --host IP --port PORT
```

Rimpiazzare i campi `IP` e `PORT` con i valori desiderati.
Per utilizzare la rest API, è necessario conoscere una chiave segreta che si trova nel campo `api_token` nel file `cuckoo.conf`. Quindi ogni volta che si effettua una richiesta presso un endpoint remoto, è necessario utilizzare il seguente header `"Authorization Bearer: token"`, rimpiazzando token con quello trovato nel file di configurazione.

Infine, è opportuno tener presente che si utilizza il servizio attraverso una rete non sicura bisogna utilizzarlo dietro un proxy server come `nginx` e abilitare il protocollo HTTPS. I comandi principali della REST API di Cuckoo sono i seguenti:

- *POST/tasks/create/file*: aggiunge un file alla lista dei tasks da analizzare e processare. Ritorna l'ID del corrispettivo task.
- *POST/tasks/create/url*: aggiunge un url alla lista dei tasks da analizzare e processare. Ritorna l'ID del corrispettivo task.
- *POST/tasks/create/submit*: Aggiunge uno o più file e/o file incorporati negli archivi o un nuovo elenco separato di URL/hash all'elenco delle attività in sospeso. Restituisce il submission ID e un task ID per ogni target da analizzare.
- *GET /tasks/sample/{id}:* restituisce una lista di tasks per target.
- *GET /tasks/view/{id}*: ritorna il dettaglio del task associato al rispettivo ID.
- *GET /tasks/delete/{id}*: rimuove il task dal database e ne elimina il risultato.
- *GET /tasks/reschedule/{id}/{priority}*: riprogramma l'esecuzione del task con il rispettivo ID e con una certa priorità. La priorità è opzionale e il valore di default fissato è 1.
- *GET /tasks/report/{id}/{format}*: ritorna un report dettagliato del task associato al rispettivo ID in uno specifico formato. Se il formato non viene specificato, l'endpoint ritorna un file JSON.
- *GET/tasks/summary/{id}:* ritorna un report conciso in formato JSON del task associato al rispettivo ID.
- *GET/tasks/rereport/{id}:* Eseguire nuovamente per il task associato al relativo ID

3.10 Cuckoo distribuito

Sebbene non sia stato sperimentato per i fini del presente lavoro, Cuckoo supporta anche la possibilità di essere usato come sistema distribuito. L'esigenza di utilizzare un sistema distribuito nasce dall'esigenza di avere più nodi per la raccolta e analisi di malware che possono essere distribuiti tramite la sua Rest API. Mediante questa configurazione una sola macchina esegue Cuckoo in modalità distribuita, mentre diversi nodi eseguono il daemon Cuckoo. Per eseguire Cuckoo distribuito, basta utilizzare il seguente comando:

```
cuckoo distributed server -H IP e - -p PORT.
```

Rimpiazzare HOST e IP con i valori desiderati.

A differenza della normale configurazione di Cuckoo, in questa è obbligatorio specificare il formato di output del report, poiché una volta recuperati dal nodo master essi vengono scartati dal nodo worker.

3.11 Considerazioni su Cuckoo

Provando ad effettuare qualche analisi sottoponendo a Cuckoo sia malware che file innocui, ha permesso di comprendere che il sistema utilizza una sua metrica interna denominata score, impiegata per dare una valutazione della pericolosità del file.
Tale score era anche influenzato da alcuni processi, che essendo in esecuzione durante lo snapshot della macchina, venivano scambiati da Cuckoo come malevoli e andavano a influenzare lo score. Per evitare ciò il PID di questi processi è stato inserito in un'apposita white list.
Uno degli "svantaggi" principali di Cuckoo è la lunga e tediosa procedura di installazione e configurazione, con cui ho sperimentato nel corso del lavoro. Un'altra difficoltà non indifferente è stata la scelta di virtualizzare completamente il sistema, a causa di problematiche di prestazione dovute al setup di un ambiente di virtualizzazione nested (macchine guest). Perciò il sistema è stato installato utilizzando una macchina più potente rispetto a un comune laptop. Tutto sommato è un ottimo case-study per affacciarsi all'analisi del malware e ha permesso di capire con quale approccio venga sviluppato un tool di questo tipo.

Capitolo 4

Benchmark e Stato dell'Arte

Dopo aver analizzato un case-study tipico quale Cuckoo, si definisce l'obiettivo principale del seguente lavoro: sviluppo di un portale web che raccoglie informazioni sui file in analisi. Ne esistono già numerosi in circolazione, tra cui ad esempio: Hybrid Analysis, File Scan IO e Virus Total. Dato il grande investimento di tempo necessario per sviluppare un motore di analisi malware, nel portale i dati verranno raccolti mediante delle chiamate a endpoint remoti di diversi servizi. Inoltre, vengono discusse anche le principali tecnologie impiegate per lo sviluppo del portale di analisi dei malware

4.1 Obiettivo

L'obiettivo "massiccio" del presente lavoro è quello di sviluppare un'applicazione che permetta di raccogliere più informazioni utili e possibili da un campione caricato. Per l'impiego sostanziale di tempo per sviluppare un motore di analisi, le analisi verranno raccolte utilizzando delle API di diversi servizi remoti. In particolare, il portale supporta:

- L'analisi di file sospetti.
- La ricerca di minacce mediante l'utilizzo di signatures conosciute, in particolare SHA-256.
- L'analisi di URL sospetti.

4.2 Benchmark

Facendo una rapida ricerca online ha permesso di capire se effettivamente fossero già presenti applicativi di questa natura. Se ne trovano veramente molti, da quelli in grado di fornire una panoramica completa a quelli che si limitano soltanto all'analisi. Tra i più famosi si annoverano: Any Run, Hybrid Analysis, FileScanIO e JoeSandbox.

4.2.1 AnyRun

AnyRun è un tool completo che permette di analizzare qualsiasi tipo di file. Possiede un'interfaccia utente abbastanza semplice: basta semplicemente identificare il target, scegliere il sistema operativo della sandbox, impostare le opzioni di connettività e scegliere per quanto tempo debba durare l'analisi. Dopo aver lanciato l'analisi, sarà possibile vedere il campione in esecuzione in real-time attraverso l'interfaccia di monitoraggio. Inoltre, il sandbox è in grado di registrare tutte le richieste di rete, le chiamate relative ai processi, e tutte le interazioni con il file system e i registri del processore.

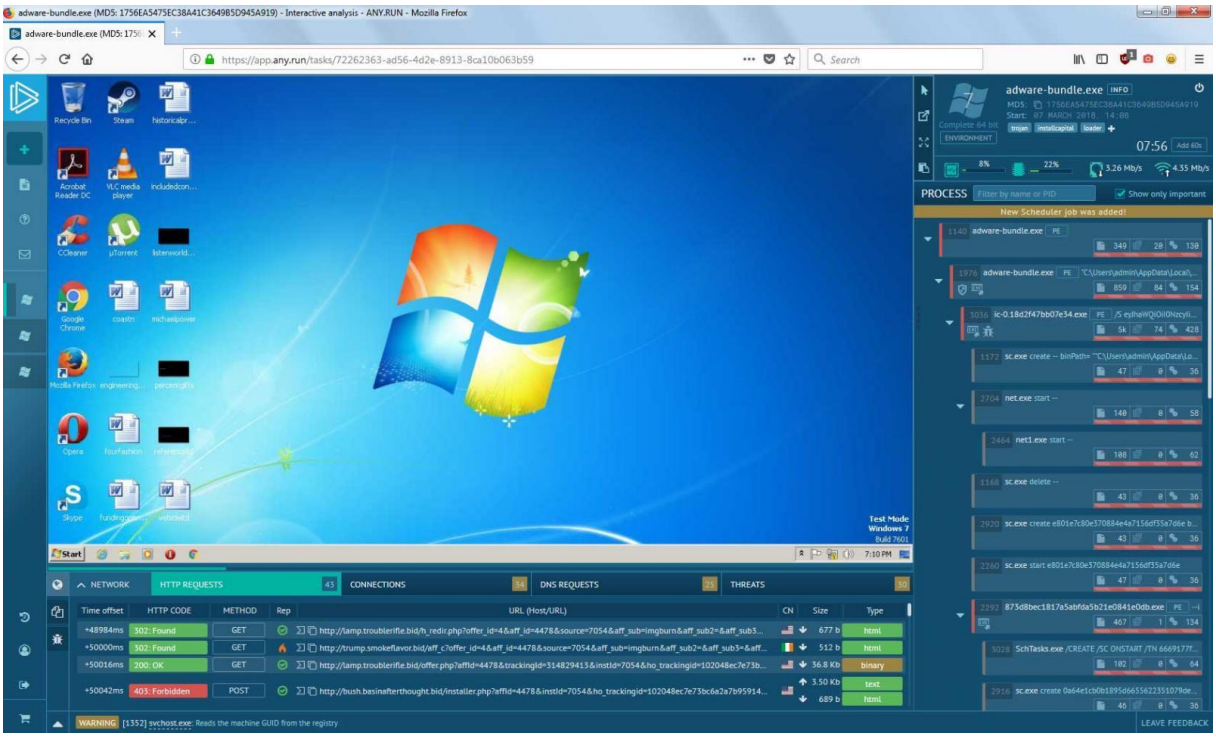


Figura 6: Interfaccia di AnyRun

AnyRun possiede un servizio API da utilizzare in applicazioni terze parti, ma è riservato soltanto a utenti premium. Inoltre, è possibile effettuare un numero massimo di 250 chiamate mensili.

4.2.2 Hybrid Analysis

Hybrid Analysis è un altro tool di analisi automatizzato basato sulla Falcon Sandbox di Crowdstrike. Come il servizio precedente, fornisce una panoramica completa dell'analisi

di un file, includendo però anche degli screenshot che mostrano l'interazione del file con l'ambiente circostante. Il sistema utilizza anche servizi di terze parti come Meta Defender e VirusTotal in modo tale da effettuare confronti sulla bontà delle analisi svolte.

Hybrid Analysis possiede un API leggera e in grado di tornare i risultati abbastanza rapidamente ed è raggiungibile al seguente URL: https://hybrid-analysis.com/api/v2/ . Ogni chiave dispone di quattro livelli di privilegio: *restricted, default, elevated e super.* In base al privilegio ottenuto più endpoint diventano disponibili per l'uso. Per effettuare delle chiamate occorre autenticarsi passando come header della richiesta i parametri seguenti all'endpoint desiderato:

- *api-key*: è la chiave generata dall'utente al momento della registrazione.
- *user-agent*: indica il software impiegato per effettuare la richiesta. Per evitare problemi con la blacklist di Hybrid Analysis, si può utilizzare la stringa 'Falcon Sandbox'.

Il servizio mette a disposizione le seguenti funzionalità:

- analisi di file e URL.
- analisi rapida di file e URL.
- ricerca di report basata su signatures oppure per stringhe
- reporting delle analisi avviate
- info su l sistema e sui sistemi di sandbox disponibili
- gestione di file collection che permettono di raggruppare in "container" dei file o URL da analizzare.

Per effettuare l'analisi basta semplicemente caricare il file o URL ai corrispettivi endpoint POST *submit/file* oppure POST *submit/url.* Effettuando il submit del file, esso deve essere passato nel corpo della richiesta.
Tali endpoint necessitano inoltre, come parametro aggiuntivo, di conoscere environment di analisi. Tale parametro è specificato mediante un ID e può avere i seguenti valori:

- *Windows 7 32 bit:* ID 100
- *Windows 7 32 bit (HWP Support)*: ID 110
- *Windows 7 64 bit: ID 120*
- *Android Static Analysis: – ID 200*
- Linux (Ubuntu 16.04 – 64 bit): ID 300

Ogni richiesta effettuata torna un identificativo chiamato *job_id* impiegato per ottenere i dettagli dell'analisi. Passando questo identificativo a endpoint differenti è possibile ottenere:

- *GET /report/{id}/summary:* un report JSON condensato che descrive tutte le proprietà del target analizzato.

- *GET/report{id}/pcap*: un dump di un file PCAP come risultato dell'analisi dei pacchetti rete.
- *GET /report/{id}/memory-strings :* un dump zip contenente i risultati dell'analisi delle stringhe.
- *GET /report/{id}/report/{type}:* un report di analisi in formato JSON, XML o HTML
- *GET /report/{id}/sample:* scarica il target analizzato.
- *GET /report/{id}/screenshot:* screenshot dell'analisi effettuata all'interno della sandbox in formato base64.
- *GET /report/{id}/state*: ritorna lo stato di processamento di un'analisi.

4.2.3 FileScanIO

FileScan.IO è un servizio gratuito di analisi malware che offre valutazioni rapide e approfondite dei file, intelligence sulle minacce ed estrazione di indicatori di compromissione (IOC) per una vasta gamma di file eseguibili, documenti e script.

FleScanIO possiede un sistema di API completamente gratuito in grado di fornire una panoramica veramente completa riguardo all'analisi statica di un file. L'API è raggiungibile al seguente URL: https://www.filescan.io/api/. Per generare una richiesta è sufficiente passare nell' header i seguenti argomenti:

- *X-Api-Key*: è la chiave generata dall'utente al momento della registrazione.

Attraverso i seguenti endpoint remoti è possibile:

- *POST /api/scan/file*: utilizzato per scansionare un file
- *POST /api/scan/url:* utilizzato per scansionare un URL.

Ogni richiesta sottoposta torna un identificativo denominato *flow_id*, utilizzato per ritornare informazioni sull'analisi effettuata. Gli endpoint remoti principali sono i seguenti:

- *GET /api/scan/{flow_id}/report*: ritorna in formato JSON il dettaglio di tutta l'analisi effettuata.
- *GET /api/scan/reports/{flow_id}/download*: scarica il report di analisi in uno dei seguenti formati: HTML. PDF, STIX e MISP.

L' API di FileScanIO permette di ottenere veramente moltissime informazioni e si focalizza principalmente su:

- Ricerca di informazioni quali signatures, data di compilazione e packers utilizzati
- Analisi del formato PE.
- Analisi delle stringhe
- Ricerca di possibili matching con qualche Yara Rules.

- Ricerca delle dipendenze utilizzate eseguibili, così come le API utilizzate e raggruppate per libreria.

4.2.4 Virus Total

Sebbene già descritto nel capitolo 2, Virus Total mette a disposizione un'API completa per effettuare analisi di file e URL ed effettuare ricerche di file malevoli basandosi sulla signatures di un file. L'API possiede due livelli privilegi: free e premium, ma la prima è più che sufficiente per effettuare ricerche di base. Come altre API, anche in questo caso è necessario avere una chiave di autenticazione per usufruire del servizio. È sufficiente impiegare nell'header della richiesta i seguenti argomenti:

- *x-apikey*: chiave generata dall'utente

L'unica limitazione della versione free è riferita al divieto di utilizzare questa API per prodotti commerciali. L'API è raggiungibile mediante il seguente base URL: https://www.virustotal.com/api/v3

Gli endpoint fondamentali per l'analisi sono:

- *POST /files*: permette di caricare ed analizzare un file. Il file deve essere passato come parametro del corpo della richiesta. L'endpoint torna un identificativo *id* scovabile nella sezione *data* nella risposta in formato JSON.
- *POST /url*: permette di analizzare un url. L'endpoint funziona in modo analogo al precedente, con l'unica differenza dettata dall'URL che deve essere passato come payload del messaggio.
- *GET /files/{id}:* ritorna delle informazioni su un file passando come ID una signature tra le seguenti: SHA-256, SHA-1 oppure MD-5
- *GET /analyses/{id}:* ritorna il dettaglio dell'analisi passando nell'URL l'identificativo della submission
- *GET /url/report/{id}/* ritorna l'analisi di un url basandosi su un URL identifier. In tal caso per URL identifier si definisce una stringa avente uno dei seguenti formati: la signature SHA256 canonizzata dell'UR oppure la stringa risultante dalla codifica dell'URL in base64 (senza il padding "=").

4.3 Tecnologie utilizzate

In questa sezione vengono illustrati tutti linguaggi utilizzati per lo sviluppo del portale web.

4.3.1 Python

Python è un linguaggio di programmazione di “alto livello “interpretato adatto allo sviluppo di applicazioni distribuite, scripting, computazione numerica e system testing. Ideato all’inizio degli anni ’90, grazie alla sua flessibilità viene anche impiegato per scrivere il back-end di web apps utilizzando framework come Flask o Django. Tra le peculiarità del linguaggio si annoverano caratteristiche quali: dinamicità, flessibilità e semplicità. Supporta diversi paradigmi di programmazione tra cui: object-oriented, strutturata e funzionale. Le caratteristiche più comuni del linguaggio sono: l’uso dell’indentazione per la sintassi delle specifiche e lo strong typing, in quanto il controllo dei tipi delle variabili viene fatto a runtime.

Figura 7: Python Logo

4.3.2 HTML

HTML è un linguaggio di markup sviluppato da Tim-Berners Lee nel 1991 ed è lo standard con la quale vengono costruite le pagine web. È definito come linguaggio di formattazione in quanto descrive le modalità di impaginazione o visualizzazione grafica del contenuto, testuale e non di una pagina web attraverso opportuni tag di formattazione. Le pagine HTML possiedono una struttura ad albero, in quanto ogni elemento padre può avere più figli e a sua volta essere figlio di un altro elemento. Pertanto, il linguaggio definisce soltanto la struttura delle pagine web; abbellimenti grafici ed elementi dinamici possono essere aggiunti mediante l’utilizzo combinato di fogli di stile (CSS) e script (JavaScript). Questi tre linguaggi sono la base per lo sviluppo di un front-end.

Figura 8: HTML Logo

4.3.3 CSS

CSS è un linguaggio impiegato per definire il layout di documenti scritti in un linguaggio di markup, quali HTML e CSS. I fogli di stile permettono di definire: la posizione, il font, bordi e colore di una pagina web. Nei fogli di stile le varie regole vengono applicate a cascata: quella più specifica viene applicata.

Figura 9: CSS Logo

4.3.4 JavaScript

JavaScript è un linguaggio di scripting lato client impiegato per rendere dinamiche le pagine web. Esso si occupa di gestire il comportamento degli elementi di una pagina; infatti, è impiegato per cambiare il codice HTML e CSS reagendo a determinati eventi o condizioni. Introdotto a partire dal 1995, noto prima con i nomi Mochan e successivamente LiveScript, fu standardizzato per la prima volta nel 1997 con il nome di ECMAScript.

Figura 10: JavaScript Logo

4.4 Librerie e framework utilizzati

Dopo aver definito i linguaggi utilizzati nel corso del progetto, qui si riportano le principali librerie e framework, che integrano dei pattern ricorrenti per lo sviluppo di una web app, rendendo così anche il lavoro più agevole.

4.4.1 Django

Django è un web framework impiegato per lo sviluppo di applicazioni web. È scritto in linguaggio Python e possiede una licenza open-source. Come altri framework dello stesso tipo, si basa sul paradigma di programmazione "Model-Template-View". Tra le peculiarità del framework troviamo:

- Astrazione di un database relazionale ad oggetti.
- Installazione di funzionalità aggiuntive tramite plugins.
- Sistema di "view generiche" basato su tag con ereditarietà dei template.
- Sistema di template basato sulla ereditarietà dei template.
- API robusta per la gestione del database.
- Supporto per l'internazionalizzazione.
- Gestione degli utenti e dell'autenticazione.
- Gestione di URL basata su espressioni regolari.

Figura 11: Django Logo

4.4.2 Bootstrap

Bootstrap è una raccolta di strumenti liberi per la creazione di siti e applicazioni per il web. Essa contiene modelli di progettazione basati su HTML e CSS, sia per la tipografia, che per le varie componenti dell'interfaccia quali: moduli, pulsanti e navigazione, così come alcune estensioni opzionali di JavaScript. Inoltre, è compatibile con tutte le versioni di tutti i principali browser supportando anche la responsiveness. Ciò significa che il layout delle pagine si adotta automaticamente in base alle caratteristiche del dispositivo utilizzato.

Figura 12: Bootstrap Logo

4.4.3 JQuery

JQuery è una libreria JavaScript per applicazioni web distribuita come software libero. È nata con l'obiettivo di semplificare la selezione, la manipolazione, la gestione degli eventi e l'animazione di elementi DOM in pagine HTML, oltre che per semplificare l'utilizzo di funzionalità AJAX, la gestione degli eventi e la manipolazione del CSS.

Figura 13: jQuery Logo

4.5 Database

4.5.1 PostgreSQL

PostgreSQL è un completo DBMS a oggetti rilasciato con licenza libera ed è un prodotto che offre caratteristiche uniche nel suo genere che lo pongono per alcuni aspetti all'avanguardia nel settore delle basi di dati. PostgreSQL utilizza il linguaggio SQL per eseguire delle query su dei dati. La sua programmabilità è il suo punto di forza, in quanti permette agli utenti la definizione di nuovi tipi basati su normali tipi di dato, permettendo alla base dati stessa di comprendere dati complessi.

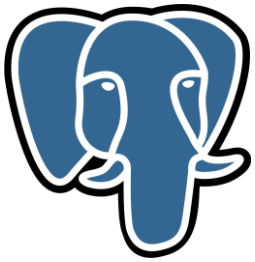


Figura 4.9: PostgreSQL Logo

Capitolo 5

Sviluppo del sistema

In questo capitolo viene presentata la struttura di massima del sistema sviluppato e di come le API si interfacciano con l'applicazione sviluppata. L'applicativo cerca di fornire una panoramica delle proprietà del file in analisi. L' applicativo è stato sviluppato utilizzando come backend il framework Django basato su Python e come database PostgreSQL.

5.1. Introduzione al sistema

Nella figura sottostante si riporta lo schema del sistema realizzato. L'applicazione in questione è un motore di analisi di malware in grado di utilizzare servizi di terze parti via Internet (API) per raccogliere più informazioni possibili sul malware da analizzare. In più vengono anche impiegate le API di Cuckoo, la cui installazione è stata descritta dettagliatamente nel capitolo 3. Per motivi di sicurezza, la comunicazione tra l'applicazione e il server che gira su Cuckoo viene fatta mediante un proxy server opportunamente configurato. Come proxy server è stato utilizzato Nginx.

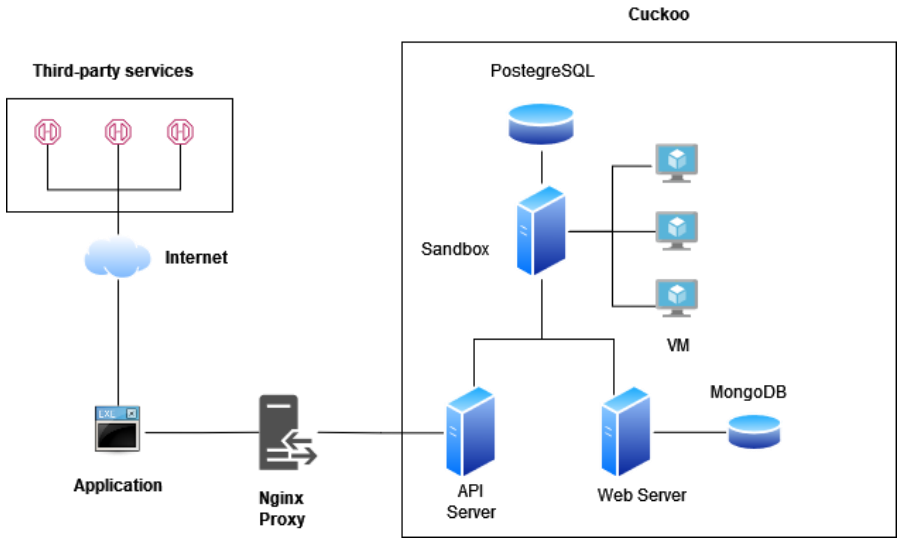


Figura 14: Schema del sistema realizzato

5.2 Introduzione all'applicativo

L'applicativo presenta un layout semplice realizzato interamente con Bootstrap; questo per garantire la responsiveness. Le parti comuni ad ogni pagina sono la navigation bar che presenta i seguenti menù: History, Sandbox e About. Invece, il footer contiene semplicemente il nominativo di chi ha sviluppato l'applicazione con relativo collegamento al repository di GitHub.

5.2.1 Index

La maschera principale permette di caricare un file, un signature di tipo SHA-256 oppure un URL per l'analisi. In base al campo riempito il sistema effettuerà le opportune chiamate API per le analisi e successivamente, una volta finita l'elaborazione, farà un redirect sulla pagina di dettaglio del report di analisi. Lo start del processo di analisi è delegato all'endpoint *POST /submit,* mentre il pulsante Reset permette di cancellare tutto il contenuto dei campi compilati. È importante comprendere che il redirect non avviene ad effetto immediato, in quanto i risultati delle analisi non vengono tornati in un tempo determinato. Ciò dipende da molti fattori tra cui: la priorità assegnata al task oppure dal numero di tasks in coda. Per ovviare a questa problematica l'unica soluzione possibile è quella di effettuare chiamate ripetute agli endpoint che tornano il report dell'analisi finché lo status del report non è completo. Infatti, molto spesso i JSON ritornati da queste API contengono un campo status che indica se il task è stato accodato, in esecuzione oppure completata. La pagina è raggiungibile mediante il seguente endpoint *GET/*.
La gestione della richiesta agli endpoint remoti è gestita da classi che fungono da wrapper: uno per ogni servizio utilizzato. Per completezza si è deciso di ricoprire tutti i possibili endpoint presenti nelle documentazioni delle rispettive API. Le richieste HTTP vengono generate mediante l'utilizzo della libreria Python *request*. Inoltre, onde evitare di specificare le chiavi di autenticazione delle API per ogni richiesta, esse vertono sempre sulla stessa sessione inizializzata alla creazione della classe.

A livello di database è stato utilizzato un modello Target che contiene le relative informazioni:

- name: contiene il nome del target da analizzare. In caso il target sia un file il sistema esegue un upload del file in una cartella dedicata denominata media.
- type: specifica il tipo di target in analisi. È stato utilizzato un enumerativo che può assumere i seguenti valori: FILE, SHA-256 e URL.

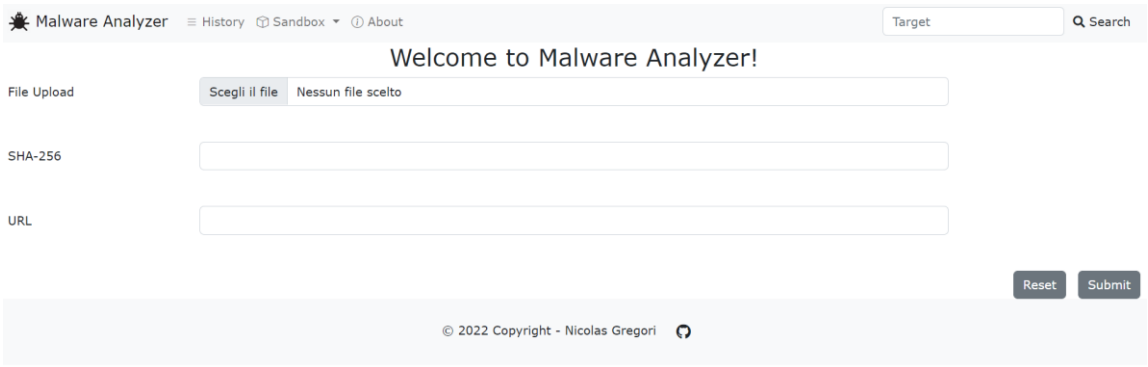


Figura 15: Submission Analisi

5.2.2 History

La voce del menù History sulla barra di navigazione ritorna lo storico di tutte le analisi effettuate. Le righe sono ordinate cronologicamente dalla più recente alla più vecchia. Come si può notare nella seguente tabella sono mostrate solamente le informazioni essenziali del report quale: Timestamp, il nome del target e la relativa tipologia. In fondo alla pagina si può notare come stia stato implementato un sistema di paginazione, il quale permette di scorrere le righe della tabella senza continuamente ricaricare la pagina.
Non è stato necessario implementare la paginazione mediante JavaScript, in quanto Django possiede già un componente built-in chiamato *Paginator* in grado di gestire direttamente la numerazione e gli elementi da mostrare su ogni pagina. La numerazione della pagina viene riportata come querystring nel modo seguente: *page={num_page}.* A livello di template, è stato solamente necessario gestire il codice per la generazione dinamica dei bottoni per far scorrere le pagine.
A livello di database una riga di questa tabella è mappata come una classe di nome *Report che* contiene:

- La data di inoltro della submission.
- Una chiave esterna ad un oggetto di tipo Target
- Un campo che contiene delle referenze alle varie analisi svolte dai vari servizi remoti di ricerca. Per l'ordine di memorizzazione delle referenze è stata utilizzata la seguente convenzione:
    - Indice 0: FileScanIO + Hybrid Analysis.
    - Indice 1: VirusTotal.
    - Indice 2: Cuckoo.

In alto a destra è presente una search bar che permette di filtrare le righe della tabella effettuando una ricerca per target e inserendo una keyword.
La pagina è raggiungibile mediante il seguente endpoint *GET /analysis*. In caso di paginazione l'URL diventa nel modo seguente:

- *GET /analysis/?page=<num_page>*: in caso di paginazione.
- *GET /analysis/?q=<keyword>:* in caso di ricerca per target.

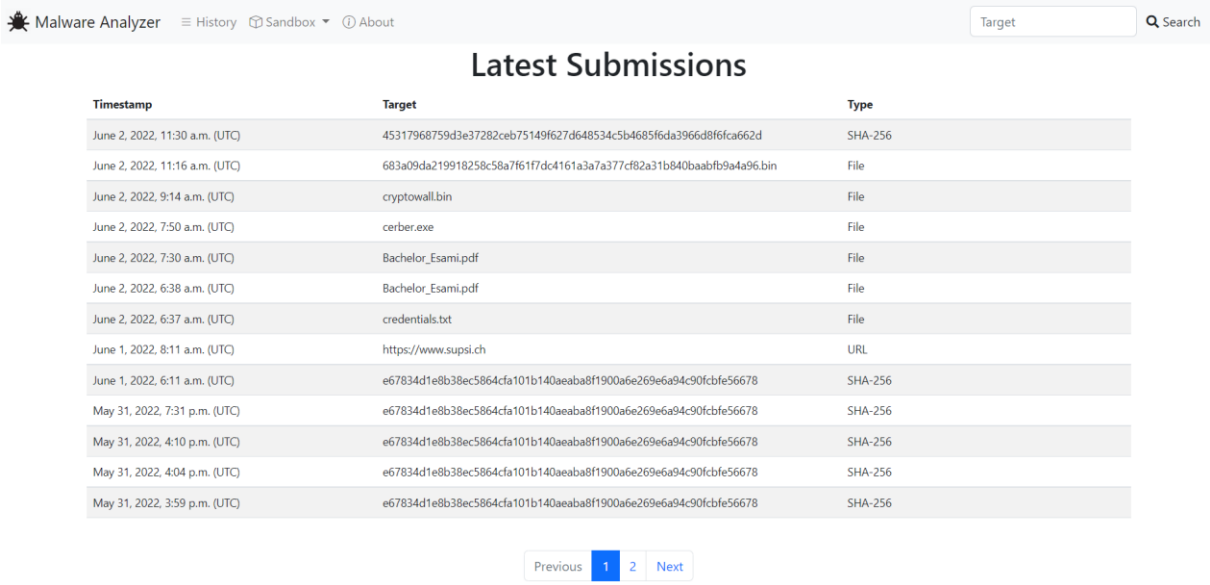


Figura 16: Storico Analisi

5.2.3 Sandbox

Il menu Sandbox non è altro che un dropdown menu che manda a delle pagine statiche contente una breve descrizione dei sistemi di Sandboxing utilizzati. Gli endpoint utilizzati sono stati i seguenti:

- *GET /info/cuckoo*: mostra una breve descrizione di Cuckoo
- *GET /info/virus-total:* mostra una breve descrizione di Virus Total.
- *GET /info/hybrid-analysis*: mostra una breve descrizione di Hybrid Analysis.
- *GET /info/file-scan-io*: mostra una breve descrizione di FileScanIO.

5.2.4 About

La seguente pagina statica è raggiungibile mediante il seguente endpoint *GET /about* e torna alcune informazioni di dettaglio dell'applicativo quali tecnologie e relative versioni.

5.3 Dettaglio sulle Analisi

Il fulcro dell'applicativo è ovviamente il report che viene generato dall'analisi di un target. L'idea di base sostanzialmente è stata quella di cercare di raccogliere più informazioni possibili da tre differenti API: FileScanIO + Hybrid Analysis, VirusTotal e Cuckoo. La decisione di combinare in un'analisi FileScanIO e Hybrid Analysis è dovuta al fatto che FileScanIO non tratta nessuna parte dell'analisi dinamica, mentre questo viene svolto da Hybrid Analysis. La schermata principale presenta 3 tab dove cliccando si può andare a vedere il dettaglio dell'analisi effettuata una per ogni servizio utilizzato. Inoltre, ogni tab possiede un menu laterale con i risultati raggruppati per contenuto informativo.
Le tab puntano alla pagina di overview. I valori possibili del campo sandbox nell'URL possono essere:

- *combined:* si intende un'analisi combinata di FileScanIO e Hybrid Analysis
- *virus-total*
- *cuckoo*

5.3.1 Overview

L'overview è la prima pagina che appare quando si vuole consultare un'analisi. La pagina principale mostra una scheda riassuntiva che presenta il sunto dell'analisi. Più specificatamente il nome del target, la sandbox utilizzata per l'analisi, la data dell'analisi e il valore dello SHA-256. Inoltre, se disponibili, viene anche presentato uno slideshow con degli screenshot di analisi, che mostrano il file in esecuzione all'interno di un sandbox e una tabella e tutte le tecniche di attacco basate sul framework MITRE.
A livello di database gli screenshot vengono memorizzati tutti in una cartella nel web server utilizzando il path relativo: */media/screenshots*. Ci sono alcune API, ad esempio Hybrid Analysis, che tornano gli screenshot in formato base64; per convertire la "stringa" in immagine si può utilizzare la libreria base64 di Python.
La tabella relativa al MITRE specifica quali tecniche e quali attacchi impiega il target per infettare il sistema. In più cliccando sulla voce della colonna Techniques su ogni riga, apparirà la pagina di dettaglio sul sito del MITRE.

Se il target analizzato è di tipo File apparirà in fondo al primo blocco un bottone denominato 'Download Sample' che potrà essere utilizzato per scaricare il file analizzato. L'endpoint utilizzato è il seguente: GET */analysis/{id}/download/sample.* Di fianco a questo bottone ne è presente un altro denominato 'Download Report', il quale cliccando farà apparire un dropdown menu utilizzato per scaricare il report di analisi prodotto dalla rispettiva API.

La seguente pagina è raggiungibile mediante il seguente endpoint: GET */analysis/{id}/{sandbox}/overview*

5.3.2 Details

La schermata di dettaglio include una sezione contenente le principali caratteristiche del file denominata *'FileDetails'* e una sezione che mostra l'analisi del formato PE denominata '*ExtendedDetails*'.
Nell'immagine sottostante sono raffigurate le principali informazioni del target analizzato. A livello progettuale tutti i dati vengono memorizzati all'interno del database sfruttando un modello Summary che funge da container di tutte le informazioni ricavate con opportuna chiave esterne utilizzate per referenziare altri elementi che verranno trattati successivamente. I campi che presenta il modello sono molti:

- *size*: indica la dimensione del file. A livello di template è stato implementato un tag che implementa un meccanismo per parsare la dimensione nell'unità di misura corretta.
- *MD5, SHA-1, SHA-256, SHA-512, ssdeep, import_hash, authentihash. Fuzzy_hash*: tutte le signatures possibili che si possono trovare.
- *Architecture*
- *Language*
- *Subsytem Readable*
- *Packers:* lista dei packers impiegati per la compilazione dell'eseguibile.
- *isPacked, isDotNet, isDigitallySigned:* flag di compilazione
- *Tags:* tag informativi che categorizzano il target.
- *Header*: chiave esterna al modello Header.
- *Rich_header:* chiave esterna al modello RichHeader.
- *Sections:* lista delle sezioni dell'eseguibile.
- *Processes*: lista dei processi
- *Resources:* lista delle risorse
- *Strings*: lista delle stringhe trovate.
- *Screenshots*: screenshot dell'esecuzione del malware all'interno di una sandbox.
- *Mitre_attacks*: specifica quali tecniche di attacco sfrutta il target.
- *Yara Rules*: lista di eventuali Yara Rules che possono essere utilizzate allo scopo di detettare l'eseguibile.
- *Antivirus*: analisi del target sfruttano 70 differenti Antivirus ed è fornita da Virus Total.
- *Version_info*: contiene delle informazioni medianti coppie chiave-valore del relativo target.

Tutte le signature non utilizzano TextField come campo del modello in quanto hanno una lunghezza limitata conosciuta. Perciò è stato più opportuno utilizzare il campo CharField che consente di stabilire una lunghezza massima per la memorizzazione.
L'icona allineata vicino alle signatures più lunghe permette di far comparire una finestra modale per far visualizzare il valore nella sua interezza.

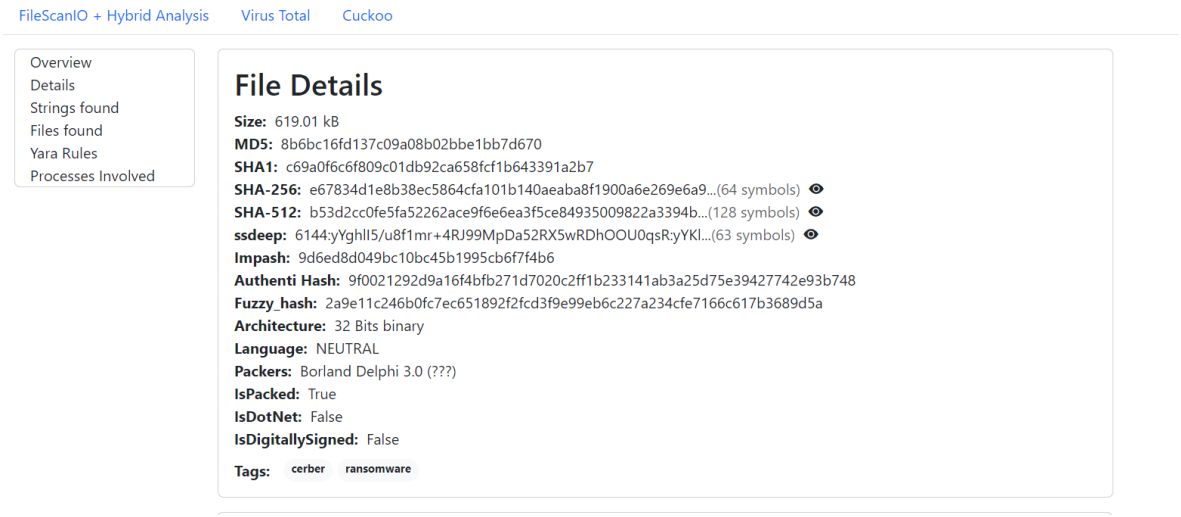


Figura 17: Dettaglio Analisi - Parte 1

La sezione Extended Details è stata suddivisa per tab dove vengono riportate tutte le informazioni sull' analisi del formato PE. In particolare:

- Nella tab Header sono presenti tutte le informazioni dell'Header descritte nel secondo capitolo.
- Nella tab Header sono presenti tutte le informazioni del Rich Header descritte nel secondo capitolo.
- Nella tab VerInfo è presente una tabella contenente alcune proprietà del target mediante coppie chiave-valore.
- Nella tab Sections è presente l'elenco di tutte le sezioni e relativi attributi del target
- Nella tab Resources è presente l'elenco di tutte le risorse e relativi attributi del target.
- Nella tab Imports si riportano tutte le funzioni che il target importa o esporta venendo raggruppate per DLL.

La pagina è raggiungibile mediante il seguente endpoint
GET/analysis/{id}/{sandbox}/details

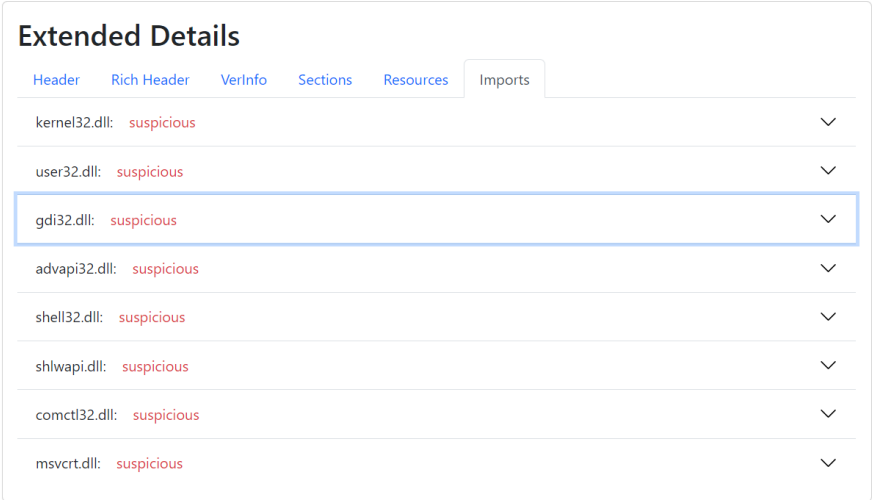


Figura 18: Dettaglio Analisi - Parte 2

5.3.4 Strings

Questa pagina riporta la lista esaustiva della raccolta delle stringhe effettuata all'interno dell'eseguibile. Sia chiaro che non tutte le stringhe sono utili al fine di comprendere il funzionamento del malware, in quanto alcune sono soltanto delle mere sequenze di bytes interpretate malamente.
La seguente pagina è raggiungibile mediante il seguente endpoint:
GET/analysis/{id}/{sandbox}/strings

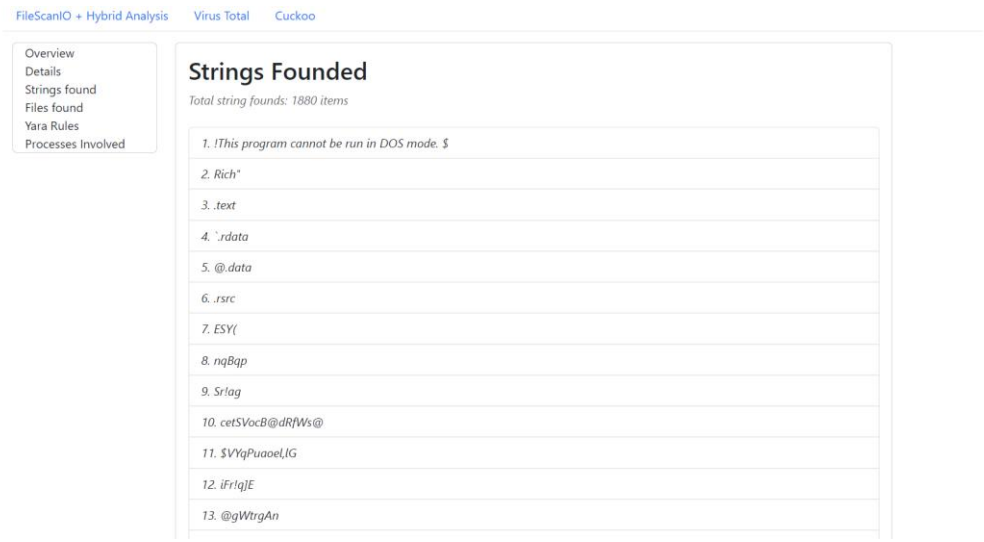


Figura 19: Dettaglio Stringhe

5.3.5 Files

La pagina presenta. A livello di database ogni 'file' estratto viene rappresentato attraverso il modello Extracted Files che presenta i seguenti parametri:

- Nome del file
- Dimensione del file
- Descrizione aggiuntiva
- Mime-Type
- Hashes tra cui: MD5, SHA-1, SHA-256, SHA-512, ssdeep.
- Meta dati tra cui: Resource ID ed Entropy

Cliccando sull'icona vicino al valore dello SHA512 e del ssdeep si apre una finestra modale che permette di visualizzarne il relativo valore nella sua interezza.
La seguente pagina è raggiungibile mediante il seguente endpoint:
GET/analysis/{id}/sandbox/files

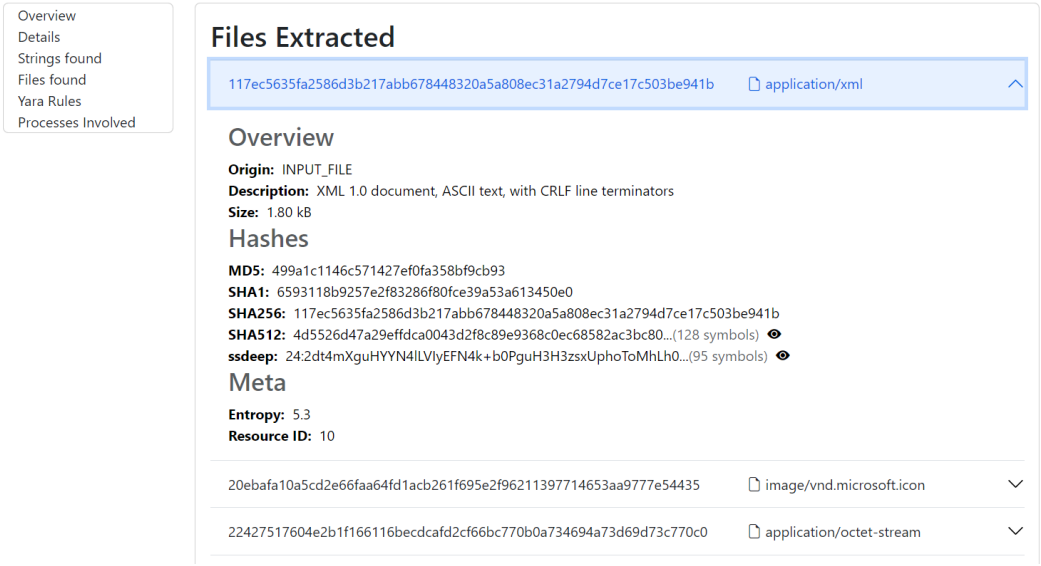


Figura 20: Dettaglio Files

5.3.6 Yara Rules

Tale pagina indica la presenza di Yara Rules che, se presenti, possono essere utilizzate per identificare un file malevolo. A livello di database ogni "blocco" rappresenta un'istanza del modello Yara Rules contenente i seguenti campi:

- Nome della regola
- Descrizione della regola
- Autore
- Data (opzionale)
- Referenza alla regola (opzionale)

La seguente pagina è raggiungibile mediante il seguente endpoint: *GET/analysis/{id}/{sandbox}/yara*

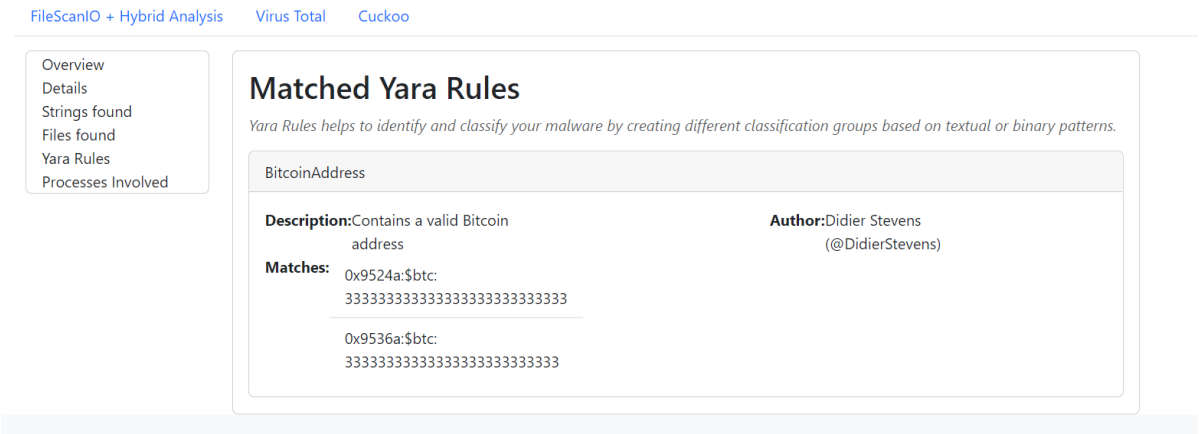


Figura 21: Dettaglio Yara Rules

5.3.7 Processi

La seguente pagina è raggiungibile mediante il seguente endpoint: *GET /analysis/{id}/sandbox/processes* e indica una lista dei processi che il target genera una volta eseguito. A livello di database ogni "accordion" rappresenta una riga del modello Process e possiede i seguenti attributi:

- UID del processo
- UID del parent process
- Command Line
- Normalized Path
- SHA-256

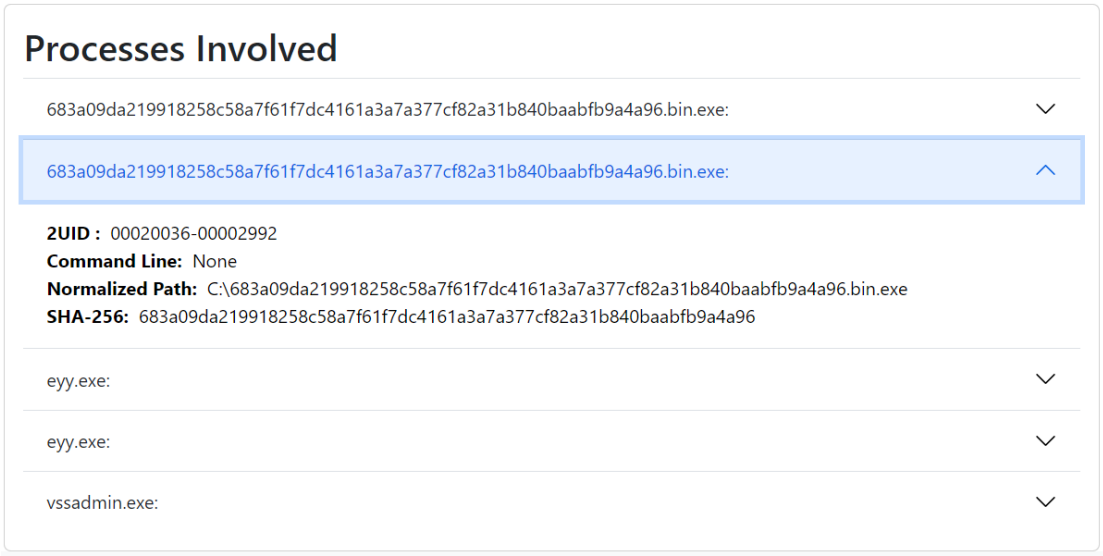
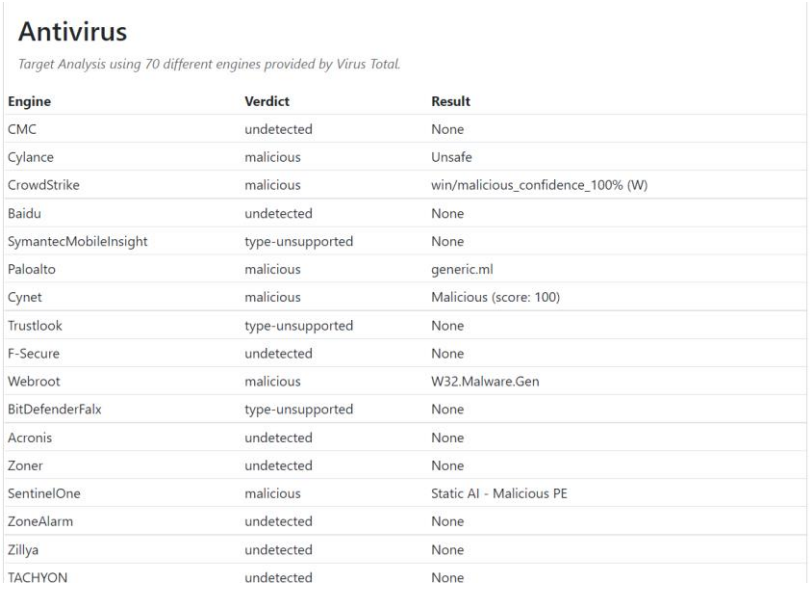


Figura 22: Dettaglio Processi

5.3.8 Antivirus

In questa pagina viene raccolto un resoconto di come il malware sia segnalato o meno in base a differenti antivirus. Tale analisi viene fornita mediante le API di Virus Total.
La seguente pagine è raggiungibile mediante il seguente endpoint: *GET analysis/{id}/{sandbox}/antivirus*



Antivirus

Target Analysis using 70 different engines provided by Virus Total.

Engine	Verdict	Result
CMC	undetected	None
Cylance	malicious	Unsafe
CrowdStrike	malicious	win/malicious_confidence_100% (W)
Baidu	undetected	None
SymantecMobileInsight	type-unsupported	None
Paloalto	malicious	generic.ml
Cynet	malicious	Malicious (score: 100)
Trustlook	type-unsupported	None
F-Secure	undetected	None
Webroot	malicious	W32.Malware.Gen
BitDefenderFalx	type-unsupported	None
Acronis	undetected	None
Zoner	undetected	None
SentinelOne	malicious	Static AI - Malicious PE
ZoneAlarm	undetected	None
Zillya	undetected	None
TACHYON	undetected	None

Figura 23: Dettaglio Antivirus

5.3.9 Networking (Sperimentale)

In questa pagina viene visualizzata una tabella contenenti i risultati di un'analisi di rete. A dipendenza dell'API utilizzata i dati relativi all'analisi di rete possono essere ottenuti attraverso un estraendo informazioni da file di estensione. pcap (Hybrid Analysis). Questa è l'estensione tipica dei log prodotti da un noto packet-filter: Wireshark. Per effettuare il parsing dei pacchetti di rete esiste una libreria in Python chiamata *scapy* impiegata per la manipolazione di pacchetti di rete. Inoltre, è anche in grado di effettuare il parsing di un log di Wireshark. La pagina è raggiungibile attraverso il seguente endpoint: *GET /analysis/{id}/{sandbox}/network*. Inoltre, se disponibile, è anche possibile scaricare il file pcap in questione attraverso il seguente endpoint: *GET /analysis/{id}/sandbox/network/download.*

5.4 Considerazioni

Il sistema in linea di massima funziona come richiesto. Sono da segnalare degli errori durante la fase di analisi che però non si ripetono spesso. Non è stata rilevata l'esatta natura, ma si pensa siano delle limitazioni delle API stesse. Inoltre, l'analisi dei file non è immediata in quanto bisogna aspettare che le API abbiano tutti i dati che devono essere parsati. Per mancanza di tempo finale e difficoltà varie per ogni servizio utilizzato non sono stati riportati tutte i possibili step di analisi come spiegato a capitolo 2, ma comunque per Cuckoo e FileScanIO + Hybrid Analysis è possibile scaricare un report in differenti formati che fornirà anche i dettagli delle analisi mancanze. In futuro si vorranno integrare le analisi mancanze con tanto di download di file particolari quali dump di memoria e i log di Wireshark. L'unico nodo da risolvere è cercare di comprendere in modo preciso dei metodi per garantire che le API ritornino tutti i dati necessari per eseguire il parsing.

Conclusioni

Il progetto ha permesso di acquisire più competenze professionali in questo ambito e una maggior sensibilità a queste tematiche. Tutto sommato il lavoro soddisfa i requisiti stipulati all'inizio anche se con alcune piccole criticità. Purtroppo, a causa delle limitazioni imposte dalle versioni premium delle API, non è stato possibile raccogliere tutte le informazioni possibili e in modo molto dettagliato, ma comunque in grado di determinare la natura esatta del target in analisi. Si spera in futuro di riuscire ad integrare gli aspetti mancanti e di allargare l'utilizzo dell'applicativo su un'utenza più larga.

Bibliografia

- [1] https://it.wikipedia.org/wiki/Malware
- [2] Analisi del malware, cos'è e come funziona - sicurezza.net
- [3] Static malware analysis - Infosec Resources (infosecinstitute.com)
- [4] Malware researcher's handbook (demystifying PE file) - Infosec Resources (infosecinstitute.com)
- [5] PE Format - Win32 apps | Microsoft Docs
- [6] FileVersionInfo Class (System.Diagnostics) | Microsoft Docs
- [7] Theta432 - Static Analysis
- [8] Module 16 - How to use Yara rules to detect malware - Blue Teams Academy - Free Training 2022
- [9] What is Cuckoo? — Cuckoo Sandbox v2.0.7 Book
- [10] Sandboxing — Cuckoo Sandbox v2.0.7 Book
- [11] Sandbox (computer security) - Wikipedia
- [12] Requirements — Cuckoo Sandbox v2.0.7 Book
- [13] How to install Python 2.7 on Ubuntu 20.04 LTS - Linux Shout (how2shout.com)
- [14] Installing Cuckoo — Cuckoo Sandbox v2.0.7 Book
- [15] Web interface — Cuckoo Sandbox v2.0.7 Book
- [16] REST API — Cuckoo Sandbox v2.0.7 Book