UNIVERSITÀ
DEGLI STUDI
DI PALERMO

NDS
NETWORKING AND DISTRIBUTED SYSTEMS
RESEARCH GROUP

Progetto e sviluppo di un sistema di analisi dinamica per l'individuazione di malware

Tesi di Laurea Magistrale in Ingegneria Informatica

Giorgio Pitarresi

Relatore: Prof. Giuseppe Lo Re
Correlatore: Ing. Alessandra De Paola

Progetto e sviluppo di un sistema di analisi dinamica per l'individuazione di malware

Relatore: Ch.mo Prof. Giuseppe Lo Re
Correlatore: Ing. Alessandra De Paola
Controrelatore: Ing. Daniele Peri
Candidato: Giorgio Pitarresi

20 marzo 2019

Sommario

In questo lavoro di tesi è stato progettato e sviluppato un sistema che sfrutta la tecnica di analisi dinamica per l'individuazione di malware in ambiente Windows. Il sistema è composto dal software Cuckoo che funge da sandbox per l'esecuzione, l'analisi e l'estrazione delle informazioni comportamentali del malware e da un classificatore che sfrutta algoritmi di *machine learning* per determinare se il software analizzato può essere considerato un malware o meno.
Il lavoro di tesi qui descritto ha analizzato diverse rappresentazioni degli eseguibili, in termini di vettori delle caratteristiche, valutando l'impatto sulle prestazioni del classificatore. Tra le caratteristiche prese in considerazione in questo lavoro vi sono le chiamate alle *Application Programming Interface* (API) ed informazioni relative al traffico di rete generato dal file. Il terzo vettore è composto sia dalle informazioni relative alle API sia dalle informazioni relative al traffico di rete. Le informazioni di rete, in particolare, sono state analizzate sia in maniera diretta che tramite l'utilizzo di servizi esterni per la verifica all'appartenenza di note in blacklist. L'analisi condotta mira ad individuare quali informazioni consentono di discriminare maggiormente software malevolo da software benevolo.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Con il rapido sviluppo di Internet, trovandoci in una società basata sempre di più sulla informatizzazione dei dati e delle informazioni, il problema della sicurezza informatica assume un'importanza sempre maggiore. I sistemi informatici sono soggetti ad attacchi informatici che possono compromettere sia il loro corretto funzionamento che la disponibilità, integrità e confidenzialità dei dati. Una particolare categoria di minacce molto diffusa è costituita dai *malware*. Con il termine *malware*, abbreviazione per ***mal****icious soft****ware***, si indica un qualsiasi programma informatico utilizzato per fini malevoli.

Questi tipi di minacce, sfruttando vulnerabilità hardware o software, possono eseguire azioni in grado di compromettere l'integrità del sistema. Un problema fondamentale è quindi il riconoscimento tempestivo di questo tipo di minaccia e di nuove famiglie o di mutazioni di malware già conosciuti.

La maggior parte degli attuali strumenti di rilevazione dei malware in commercio si basano sul confronto del file da analizzare con un archivio in cui sono schedati tutti i malware conosciuti [3], [4]. Nella comunità scientifica, da alcuni anni vengono proposti sistemi di rilevamento dei malware che fanno uso di metodi di *machine learning* per supportare l'apprendimento automatico di un sistema di classificazione. Questi sistemi spesso si basano sull'uso di un insieme di caratteristiche che consente di rappresentare in maniera concisa il software da analizzare. Tali metodi possono essere distinti in due classi a secondo dell'approccio con cui vengono ottenute queste caratteristiche: l'analisi statica e l'analisi dinamica. Durante il processo di analisi statica non viene eseguito il malware ma viene analizzato il codice (qualora non sia offuscato) del software malevolo a basso livello. Per sistemi Windows un modo può essere quello di analizzare il contenuto del campo PE estratto [1], [8], [26], [27]. Visto che il malware non viene eseguito, il processo di analisi statica è considerato abbastanza sicuro. Inoltre il processo è generalmente più veloce rispetto alla controparte dinamica anche se è più facile

camuffare il codice malevolo di basso livello, compromettendo l'esito dell'analisi. A seconda della complessità delle tecniche usate e del livello di approfondimento l'approccio dell'analisi statica può essere suddiviso in:

- Analisi statica base;
- Analisi statica avanzata.

Uno degli approcci più comuni dell'analisi statica base consiste nell'utilizzare dei software che eseguono la scansione del malware tramite antivirus, l'hashing e l'analisi della struttura del software [2].
Nei metodi che utilizzano l'analisi tramite antivirus vengono sfruttate le firme del file per identificarne il comportamento sospetto. Le firme di malware conosciuti sono archiviate in un database [3], [4]. L'antivirus, dopo aver trovato una firma (se presente), la confronta con le firme che sono state memorizzate nel database. Se la firma è nel database, il campione viene contrassegnato come malevolo, altrimenti contrassegnato come benigno. Anche se la scansione antivirus funziona in modo rapido e accurato per rilevare malware noti, non riesce a rilevare malware sconosciuti. I malware sconosciuti sono generalmente versioni dello stesso malware che cambiano la propria firma per bypassare lo scanner dell'antivirus [3].
La tecnica chiamata *hashing* consiste nell'estrarre il valore hash come ad esempio MD5, SHA-1, SHA-2 [3] e confrontarlo con hash appartenenti a malware già identificati.
L'analisi statica avanzata invece utilizza strumenti appositi di disassemblaggio per identificare la struttura del flusso del software.
L'analisi dinamica (o *behavior analysis*) invece consiste nell'eseguire direttamente il malware per poterne analizzare il completo comportamento dinamico e registrarne le azioni [5]. Una infezione da malware può causare diversi danni al nostro sistema, quindi l'esecuzione dei file malevoli avviene in un ambiente protetto, chiamato sandbox. Una sandbox è un ambiente operativo completamente slegato dal sistema su cui gira in cui è possibile testare programmi e applicazioni senza correre il rischio di arrecare danni al sistema. Inoltre consente di monitorare il comportamento dei programmi (le chiamate alle funzioni API, i loro parametri, i file creati o cancellati, il traffico di rete, ecc.) registrando i risultati in un file.

Un ulteriore approccio consiste nell'usare entrambi i tipi di analisi [6]. Questo approccio viene chiamato generalmente "analisi ibrida" [2] e consiste nel prendere in considerazione sia le informazioni estratte dall'analisi dinamica che le informazioni relative all'analisi statica.

Per quanto l'analisi dinamica sia un approccio più efficace dell'analisi statica, presenta diversi svantaggi. In primo luogo si esegue il malware e quindi i risultati delle analisi possono avere tempi di risposta eccessivi. Inoltre alcuni malware possono accorgersi di essere eseguiti in un ambiente protetto e posso nascondere il loro comportamento malevolo.

L'obiettivo di questa tesi è sviluppare un sistema più robusto di classificazione di malware su sistema Windows basata sull'apprendimento automatico facendo uso di Cuckoo Sandbox [10], software utilizzato per effettuare l'analisi dinamica dei file. Questa sandbox fornirà le informazioni del comportamento dei malware che verranno utilizzate successivamente come input per gli algoritmi di *machine learning*. Per la classificazione si farà uso di diversi metodi di classificazione sfruttando ad esempio una rete neurale, una rete bayesiane e l'algoritmo Random Forest. L'obiettivo è determinare il metodo migliore, attraverso l'estrazione di caratteristiche, in grado di distinguere i file malevoli da quelli benigni in modo accurato. Il classificatore sfrutterà set di caratteristiche estratti dai report di analisi generati dalla sandbox.

Inizialmente verrà proposto un metodo che consiste nel considerare solo le chiamate di sistema effettuate dal malware. Come si vedrà successivamente, le interazioni tra il software malevolo e il sistema operativo permettono di identificare comportamenti sospetti comuni a tutte le principali famiglie di malware conosciuti. In questo lavoro di tesi verrà proposto anche un ulteriore approccio che fa uso anche delle informazioni relative al traffico di rete generato dal software malevolo. A causa della rapida diffusione dei dispositivi connessi ad internet (IoT) [28], molti malware sfruttano la rete per compiere azioni malevole. In questo metodo proposto verranno presi in considerazione gli IP di destinazione contattati dal software malevolo. Inoltre per la costruzione del vettore di caratteristiche relativo alle informazioni del traffico di rete, verrà preso in considerazione un approccio che fa uso di un metodo di analisi statica che consiste nella ricerca degli IP, estratti con il

metodo dell'analisi dinamica, in delle blacklist online. Gli approcci di *machine learning* esposti precedentemente permetteranno di valutare il metodo migliore per la classificazione dei malware ma anche le migliori caratteristiche da considerare per la descrizione comportamentale del software malevolo. Queste considerazioni verranno trattate nel corso della tesi.

In particolare, nel Capitolo 2, verranno descritte le principali metodologie di classificazione di malware e i vari approcci utilizzati in letteratura per l'analisi di malware ed estrazione di caratteristiche.

Nel Capitolo 3 verranno illustrati invece i software utilizzati per l'analisi e la classificazione. In particolare verrà descritto il software Cuckoo che permette di eseguire l'analisi dinamica dei malware e il software Weka per la classificazione.

Nel Capitolo 4 verrà descritta più nel dettaglio la struttura del file di report generato da Cuckoo e le informazioni in esso contenute. In questo capitolo verranno discussi inoltre tutti gli approcci utilizzati per l'estrazione e rappresentazione delle caratteristiche. In particolare verranno illustrati i metodi per l'estrazione e classificazione delle API e del traffico di rete.

La descrizione degli approcci utilizzati per la classificazione delle caratteristiche verrà illustrata nel Capitolo 5. In questo capitolo verranno valutate le caratteristiche e le differenze tra i vari algoritmi di *machine learning* utilizzati. Verranno presi in considerazione una rete neurale, una rete bayesiana, l'algoritmo Random Forest e l'algoritmo J48.

I risultati sperimentali ottenuti e il confronto tra gli approcci utilizzati verranno discussi nel capitolo 6.

Capitolo 2

Stato dell'arte

L'analisi e il rilevamento di software malevolo può fornire informazioni molto importanti sulle caratteristiche dei malware. Lo studio di questo tipo di minaccia permette di apprendere, ad esempio, gli obiettivi principali dei software malevoli, le modalità di diffusione, le interazioni con i sistemi operativi e anche il traffico di rete generato. Lo studio di questo tipo di informazioni permette di identificare misure di prevenzione per limitarne la diffusione e la tempestiva individuazione.
Questo tipo di problema è stato affrontato utilizzando due metodi molto differenti tra loro: l'analisi statica e l'analisi dinamica. È stato adottato anche un ulteriore metodo chiamato analisi ibrida che sfrutta le caratteristiche dell'analisi statica e dell'analisi dinamica come in [2] e [6].
L' analisi statica consiste nell'estrarre informazioni analizzando il codice senza quindi eseguire il potenziale malware. Questo tipo di analisi permette di estrarre una *signature*, una stringa o un hash che descrive la struttura completa del programma [5]. Per i malware già noti, questo approccio è abbastanza veloce e preciso. L'analisi dinamica permette invece di estrarre informazioni eseguendo il malware in un ambiente protetto. Questo tipo di analisi sfrutta una sandbox per l'esecuzione sicura del software e per la registrazione del comportamento. L'analisi dinamica è un approccio generalmente più robusto e largamente usato in letteratura. Generalmente un metodo classico per il riconoscimento dei malware consiste nell'estrarre le chiamate ad API *(Application Programming Interface).*
In [4] viene illustrato come le chiamate ad API possano descrivere in modo dettagliato il comportamento di un malware in ambiente Windows. L'approccio che consiste nell'estrazione delle chiamate ad API è stato utilizzato nel lavoro proposto in [5] che consiste nell'utilizzare come caratteristiche per la classificazione sequenze di *n-grams* di API*,* cioè insiemi di n stringhe (in questo lavoro è stato utilizzato la formula con *3-grams*).

In [7] è stato utilizzato il software Cuckoo per l'analisi dinamica dei malware e tecniche di addestramento profondo. Sono stati confrontati i risultati dell'addestramento di una *Feedforward Network*, di una *Convolutional Network* e di un insieme composto da una *Convolutional Network* e una *LSTM Network.*
I risultati migliori sono stati ottenuti da questo ultimo insieme.
Un metodo simile è stato proposto in [13]. Il metodo consiste sempre nell'uso del software Cuckoo per l'estrazione delle API. Di questo insieme di API sono stati estratti gli *1-grams*, cioè insiemi composti da una sola stringa. Dopo aver scartato gli *1-grams* presenti in ogni file analizzato, sono stati selezionati i 20.000 *1-grams* più frequenti e convertiti in un vettore di 20.000 posizioni. Anche in questo caso è stato adottato un approccio che fa uso del *Deep Learning.* In questo caso è stata addestrata una *Deep Belief Network.*
In [15] viene utilizzato il metodo degli *n-grams* per creare diverse "API word", cioè un insieme di stringhe che contiene oltre al nome dell'API anche il nome del suo argomento e il rispettivo valore. In questo lavoro è stato utilizzato un metodo basato sull'algoritmo NMF.
In [11] invece sono stati utilizzati altri ambienti per l'individuazione del comportamento di un malware. È stato utilizzato il software Wine che permette di simulare un'applicazione Win-32 sotto ambiente Unix monitorandone il comportamento e il software di emulazione virtuale Qemu. In questo lavoro è stato utilizzato un vettore binario delle caratteristiche che sfrutta le informazioni dell'analisi dinamica dei file dopo un processo di *parsing* delle informazioni. Le tecniche usate per la classificazione consistono nell'utilizzo di una rete bayesiana, dell'algoritimo Randm Forest e dell'algoritmo J48 sfruttando il metodo della validazione incrociata (*k-cross-validation).* Si ottengono risultati migliori con Random Forest e J48.
In [12] invece il vettore delle caratteristiche consiste nello sfruttare la sequenza di API. In questo caso è stato preferito l'approccio che consiste nel costruire un vettore binario derivato da una *Action-List* globale, cioè una lista di descrizioni dell'API estratte tramite funzioni euristiche. I risultati migliori sono stati ottenuti tramite l'utilizzo di alberi decisionali e Random Forest. L'approccio utilizzando una SVM ha prodotto il risultato peggiore.

Per quanto riguarda i metodi che fanno uso dell'analisi ibrida in [2] vengono estratte caratteristiche chiamate PSI (P*rintable String Information*) per la parte statica mentre le API per la parte dinamica costruendo un vettore binario. Anche in questo caso è stata adottata la tecnica che fa uso degli *n-grams*.
I lavori descritti precedentemente basano lo studio del comportamento dinamico di un malware principalmente sull'estrazione di caratteristiche relative alle chiamate ad API. Il comportamento dinamico di un file può essere determinato considerando anche il tipo di traffico di rete che produce.
In [16] viene sfruttato un *Internet Emulator* che permette di fornire informazioni relative al traffico di rete più dettagliate circa il comportamento di un file.
In [17] invece, oltre alle API *calls*, viene analizzato il traffico di rete considerando la frequenza di *behaviour unit* contenente i due tipi di informazioni. La classificazione è stata effettuata considerando il metodo *Radial Basic Function* in un modello SVM.
Un approccio che utilizza solamente informazioni relative al traffico di rete viene proposto invece in [18]. In questo lavoro il vettore di caratteristiche viene strutturato in cinque parti:

- Transaction che rappresenta l'interazione tra un client e un server;
- Session che consiste in una tupla composta da quattro elementi relativi agli IP e alle porte sorgente e destinazione considerando i protocolli TCP e UDP;
- Flow che consiste in un gruppo di sessioni tra due coppie di indirizzi IP durante il periodo di aggregazione, cioè il periodo di tempo che va dall'inizio della prima sessione nel flusso, fino al tempo di inattività massimo tra due sessioni;
- Conversation Windows consiste in un gruppo di flussi tra un client e un server riferito ad un certo intervallo di tempo.

La classificazione è stata effettuata tramite una rete bayesiana, l'algortimo J48 e Random Forest.

In questo lavoro di tesi verrà proposto e valutato un metodo che farà uso, oltre che dalle informazioni relative alle chiamate di sistema, anche del traffico di rete.

Rispetto ai lavori illustrati precedentemente, il metodo proposto valuterà le differenze tra la classificazione basata su caratteristiche estratte dall'analisi dinamica e caratteristiche costruite grazie all'uso di blacklist esterne in cui verranno ricercati gli IP estratti.

Capitolo 3

Ambiente di sviluppo

In questo capitolo verranno presentati i principali software utilizzati per l'analisi dinamica e la classificazione dei malware. Verrà descritto, in particolare Cuckoo, il software utilizzato per l'analisi dinamica degli eseguibili in ambiente Windows, Virtualbox, software necessario per creare l'ambiente virtuale in cui far eseguire i malware in sicurezza e Weka, software utilizzato per l'apprendimento automatico del sistema di classificazione.
Il software MongoDb invece servirà per la registrazione delle informazioni generati da Cuckoo. Le estrazioni delle caratteristiche, le analisi dei malware e i processi di classificazione sono stati eseguiti in un sistema con sistema operativo Ubuntu 17.10.

3.1 Cuckoo

Cuckoo Sandbox è un sistema open source automatizzato scritto in linguaggio Python che sfrutta la tecnologia di virtualizzazione per effettuare l'analisi statica e dinamica dei malware [9], [10].
Grazie alla sua natura open source e alla sua struttura modulare, Cuckoo permette di personalizzare qualsiasi aspetto dell'ambiente di analisi, dell'elaborazione dei risultati e della fase di reporting. Sfruttando il processo di analisi dinamica, il software è in grado di monitorare il comportamento e il funzionamento del malware ed a estrarre maggiori informazioni rispetto alla sola analisi statica. Vengono monitorate le richieste HTTP, TCP, UDP e anche le chiamate alle API. Cuckoo inoltre fornisce anche i risultati dell'analisi statica sul malware [10]. L'obiettivo di questo lavoro di tesi è sfruttare le informazioni estratte per apprendere quali comportamenti sono comuni tra i malware e possono quindi essere utilizzati per identificarli.

I componenti principali dell'infrastruttura di Cuckoo sono una macchina host e un numero arbitrario di macchine g*uest*. Il sistema permette infatti di eseguire l'analisi di più malware su diverse macchine guest. L'host esegue il software che gestisce l'intero processo di analisi, mentre le macchine g*uests* sono gli ambienti isolati in cui i malware vengono effettivamente eseguiti e analizzati in modo sicuro.
Cuckoo utilizza un agente personalizzato, cioè un particolare codice scritto in Python, che viene eseguito all'interno della macchina virtuale e che gestisce la comunicazione e lo scambio di dati con l'host. La macchina host è connessa alle macchine virtuali grazie ad una rete virtuale, che permette alla macchina host di rimanere isolata dalla rete esterna. La comunicazione tra tutte le entità avviene inoltre grazie ad un web server, avviato nella macchina host, che si occupa di gestire lo scambio di informazioni tra la macchina host e le varie macchine guest. Lo scenario operativo è abbastanza semplice: non appena il nuovo file viene inviato al server, viene assegnato un ambiente virtuale, il file viene eseguito e tutte le azioni eseguite nel sistema virtualizzato vengono registrate all'interno di un file di formato JSON [9], [10].
È possibile avviare le analisi sia tramite una comoda interfaccia web sia da riga di comando. Quest'ultima opzione permette di automatizzare il processo di analisi.

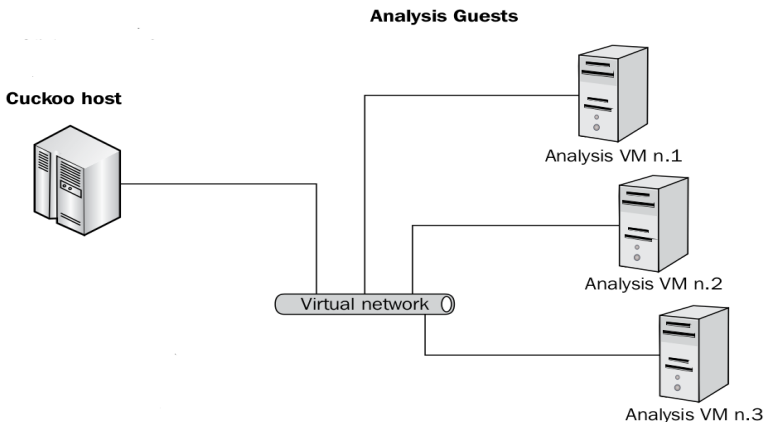

Fig. 1: Rappresentazione schematica dell'infrastruttura di Cuckoo

Per poter automatizzare il processo di analisi è stato creato inoltre un sistema

Client/Server che permette di avviare le analisi e scaricare i file di report generati da Cuckoo da remoto.
La macchina host rimane in ascolto su una porta in attesa di richieste da parte di un Client che vuole effettuare le analisi da remoto. Dopo aver controllato che non ci fossero report pendenti da inviare, il server host, che funge da server, scarica i malware da analizzare inviati da un client e avvia in automatico le analisi. Non appena un'analisi viene stata completata, l'host comunica di aver terminato l'analisi e invia il rispettivo file di report al client.

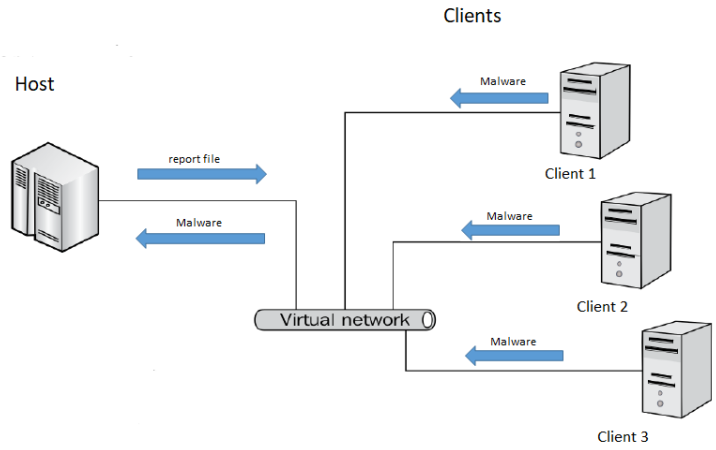

Fig. 2: Rappresentazione schematica dell'infrastruttura Client Server

3.2 VirtualBox

Virtualbox è un software gratuito open source per l'esecuzione di macchine virtuali. La virtualizzazione è un processo che permette di creare una macchina virtuale all'interno di un altro sistema. Questo software è necessario per il corretto funzionamento di Cuckoo visto che grazie ad esso è possibile creare la sandbox in cui eseguire i malware. La macchina virtuale funziona come una normale macchina fisica e quindi il comportamento del malware generalmente sarà identico a quello

che avrebbe in un sistema reale.
Per questa tesi è stata virtualizzata una macchina con sistema operativo Windows 7. La scelta del sistema operativo da virtualizzare è ricaduta su Windows 7 visto che è un sistema leggero, compatibile con Cuckoo e molto diffuso. Essendo un sistema generalmente molto diffuso sono presenti una grande quantità di malware appartenenti a famiglie differenti.
Il sistema virtualizzato ha un comportamento del tutto simile a quello di una macchina reale e quindi l'esecuzione non attenta di un malware all'interno della sandbox collegata ad una sottorete può causare diversi problemi e infettare anche altri sistemi. È necessario quindi non solo isolare la macchina host dalla rete esterna ma anche la macchina virtuale. A questo proposito è stato necessario riconfigurare il firewall della macchina host in modo tale da interrompere qualsiasi tipo di connessione in uscita tranne quelle strettamente necessarie per il funzionamento dell'intera infrastruttura di Cuckoo, come ad esempio le connessioni verso il server che si interfaccia con la macchina host e i vari sistemi virtualizzati.
Per effettuare le analisi e rendere la sandbox riutilizzabile, Cukoo sfrutta un'importante caratteristica di VirtualBox la quale permette di creare un'istantanea (o *snapshot*) del sistema virtualizzato. Questa *snapshot* viene creata una sola volta non appena si configura il sistema virtualizzato e utilizzata successivamente per ripristinare lo stato iniziale della macchina e preparare il sistema per un'altra analisi. Ogni volta che viene avviata un'analisi viene eseguito il malware all'interno della macchina virtuale. Non appena il sistema termina l'analisi dei file, Virtulbox carica la *snapshot* ripristinando lo stato del sistema e preparando la macchina virtuale per un'altra analisi.

3.3 Weka

Weka, acronimo di **W**aikato **E**nvironment for **K**nowledge **A**nalysis, è un software open source per l'apprendimento automatico sviluppato nell'Università di Waikato in Nuova Zelanda. Weka è un ambiente software interamente scritto in Java che permette di applicare dei metodi di apprendimento automatico ad un set di dati (*dataset*), e analizzarne il risultato. È possibile attraverso questi metodi avere quindi

una previsione dei nuovi comportamenti dei dati. Weka accetta come dataset di input una tabella in cui le istanze corrispondono alle righe e gli attributi selezionati alle colonne. Il formato utilizzato in Weka per la lettura dei dataset è il formato ARFF (*Attribute Relationship File Format*), simile al formato CSV (*Comma Separated Value*), il quale è equivalente alla tabella di un database relazionale.
Weka raccoglie diversi algoritmi di apprendimento automatico e contiene strumenti per la preparazione, la classificazione, la regressione, il clustering e il mining dei dati.

3.4 MongoDB

MongoDB è un DBMS non relazionale orientato ai documenti open source. MongoDB è stato scelto per questo progetto poiché è in grado di gestire input JSON diretti e quindi si è rivelata una soluzione naturale per la memorizzazione delle informazioni comportamentali fornite da Cuckoo. Infatti MongoDB si allontana dalla struttura tradizionale basata su tabelle dei classici database relazionale in favore di documenti in stile JSON con schema dinamico di formato BSON, rendendo l'integrazione di dati di alcuni tipi di applicazioni più facile e veloce. Il formato BSON è un tipo di un formato binario utilizzato per rappresentare strutture dati semplici e array associativi (chiamati oggetti o documenti in MongoDB). Il nome "BSON" è basato sul termine JSON e significa "JSON Binario" (*Binary JSON*).

Capitolo 4

Progettazione e implementazione del sistema

Nel capitolo precedente è stato illustrato l'ambiente di sviluppo, descrivendo in particolare il software Cuckoo che permetterà l'analisi dinamica e l'estrazione delle informazioni comportamentali dei file.
In questo capitolo verrà analizzato nel dettaglio la struttura del file di report generato da Cuckoo e i metodi e le scelte progettuali per la costruzione dei modelli da analizzare. Nel paragrafo 4.1 verrà analizzata struttura del file di report generato da Cuckoo descrivendo le informazioni relative al comportamento dinamico del file analizzato. Esso contiene tutte le informazioni comportamentali di un file, fornendo una descrizione dettagliata sia della sua interazione con il sistema operativo sia del traffico di rete generato. Nel paragrafo 4.2 verranno descritti i metodi e le scelte progettuali di estrazione delle caratteristiche utili per la classificazione dei modelli. Verranno illustrati i vettori di caratteristiche costruiti a partire dalle API estratte dal file di report e i vettori costruiti con le informazioni relative al traffico di rete. Questi modelli consentiranno di effettuare una valutazione delle varie soluzioni progettuali adottate determinando quali informazioni descrivono meglio l'aspetto comportamentale dei file.

4.1 Struttura del file di report Cuckoo

I risultati dell'analisi statica e dinamica di Cuckoo vengono registrati in un file di formato JSON. Questo file contiene diverse informazioni sia relative ad attività a livello di rete che ad attività a livello di sistema. A causa della mole di informazioni il file JSON può presentare anche più di 20000 righe e un peso che può raggiungere i 700MB.
La tipica struttura di un file di report può essere suddivisa in due grandi sezioni: la sezione *behavior* e la sezione *network.*

La sezione *behavior* contiene le informazioni relative alle attività di sistema ricavate dall'analisi dinamica circa il comportamento del malware. La sezione network invece contiene invece informazioni riguardo al traffico di rete. La sezione behavior e network insieme rappresentano il comportamento dinamico del malware.

La sezione *behavior*, come la sezione *network*, contiene al suo interno diverse sottosezioni. La sottosezione *calls* contenuta in *behavior* contiene la lista di tutte le API che vengono chiamate durante l'esecuzione del malware da uno specifico processo. Le API calls mostrano tutto ciò che accade al sistema operativo, incluse le operazioni su file, sui registri, processi e altre caratteristiche. Tutte queste informazioni descrivono il comportamento del malware. Oltre al nome dell'API stessa, all'interno della sezione *calls* sono presenti anche i suoi argomenti, il valore di ritorno e lo stato. Se lo stato è uguale 1 significa che la chiamata all'API è andata a buon fine, se è 0 il contrario.

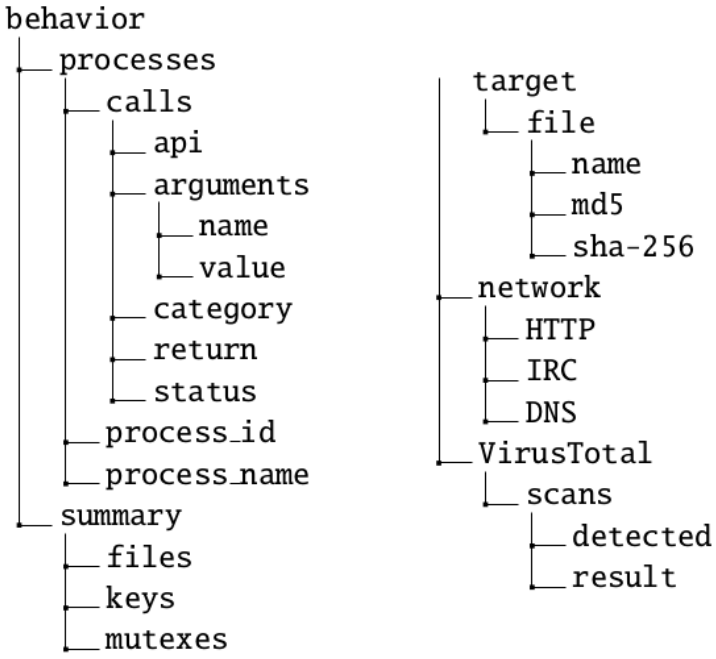

Fig. 3: Struttura file di report Cuckoo

La sezione *apistats* contiene invece la lista di tutte le API chiamate per ogni processo identificato dal suo pid e la loro frequenza, mentre *processstree* contiene l'albero dei processi da cui è possibile ricavare l'ordine di esecuzione dei processi

e delle chiamate alle API.
Per quanto riguarda la sezione network essa è suddivisa in tante sottosezioni quanti sono i tipi di richieste effettuate dal malware verso l'esterno.
Sono presenti diverse sezioni relative alle informazioni sulle richieste UPD, TCP, HTTP, HTTPS ma anche gli indirizzi IP contattati e le porte utilizzate.
La sezione UDP, come quella TCP, contiene gli indirizzi IP sorgente e destinazione un offset, il tempo di esecuzione e le porte sorgente e destinazione.
Le sezioni HTTP_Ex e HTTPS_Ex contengono invece informazioni più complete delle sezioni HTTP e HTTPS. Esse contengono il nome di host contattato, l'IP di destinazione, le porte usate e varie informazioni relative all'URI e al tipo di richiesta effettuata.

4.2 Estrazione delle caratteristiche

*** OMISSIS ***

Capitolo 5

Metodi di classificazione

In questo capitolo verranno illustrate le tecniche utilizzate per la classificazione delle *features* estratte basate su algortimi di *machine learning.*

Con *machine learning* (o apprendimento automatico) si intende un insieme di tecniche che utilizza metodi statistici che hanno lo scopo di identificare pattern nei dati. L'apprendimento automatico permette quindi di classificare i dati in base ai tipi di pattern individuati.

È possibile individuare due tipi principali di apprendimento automatico:

- Apprendimento supervisionato;
- Apprendimento non supervisionato.

L'apprendimento supervisionato consiste nel fornire al modello dei dati già etichettati con l'obiettivo di estrarre una regola generale che associ i dati alle rispettive classi di appartenenza (in questo caso malware e file benevoli). L'apprendimento non supervisionato invece consiste nel trovare una struttura negli input forniti senza che una parte di questi dati vengano etichettati con la loro classe di appartenenza.

Per la classificazione sono state utilizzate tecniche di apprendimento supervisionato e sono stati utilizzati quattro diversi approcci: l'algoritmo Random Forest, una Rete Bayesiana, una Rete Neurale e un albero decisionale. Utilizzare diversi approcci è stato utile per testare la bontà del modello.

L'apprendimento supervisionato utilizza due sottoinsiemi dei dati forniti: l'insieme di addestramento e l'insieme di validazione. L'insieme di addestramento è l'insieme dei dati utilizzati per addestrare il sistema il modo supervisionato. L'insieme di validazione è invece un insieme di dati etichettati utilizzato per testare la correttezza dell'apprendimento. La suddivisione del primo dataset in insiemi di validazione e insiemi di addestramento è stata effettuata tramite la tecnica chiamata

k-cross-validation.

La tecnica *k-cross-validation* consiste nella suddivisione del dataset in k parti di uguale dimensione. Ad ogni passo, la k-esima parte del dataset viene utilizzata come insieme di validazione, mentre la restante parte costituisce l'insieme di addestramento. In questo modo, il modello viene addestrato per ognuno dei sottoinsiemi del dataset, evitando quindi problemi di overfitting. In questo modo si esclude iterativamente un gruppo alla volta e lo si cerca di predire con i gruppi non esclusi.

5.1 Alberi di decisione

In *machine learning* un albero di decisione è un uno strumento di apprendimento supervisionato. Un albero decisionale è una collezione di regole espresse in forma di albero binario ottenute attraverso partizionamento ricorsivo [25]. Nel caso preso in esame una semplice regola potrebbe basarsi sulla presenza o meno di gruppi di API o di indirizzi IP. In questo modo la probabilità associata alla regola definisce l'appartenenza ad una determinata classe definita nel modello.

Ogni albero binario produce in uscita, per ogni vettore della matrice binaria, un risultato che rappresenta la classe di appartenenza del vettore selezionato.

Il modello di albero decisionale standard crea quindi un solo albero che viene utilizzato per prevedere il risultato di nuove osservazioni. L'output di questa strategia è molto instabile e la struttura ad albero potrebbe essere seriamente compromessa da una piccola modifica nel set di dati di addestramento.

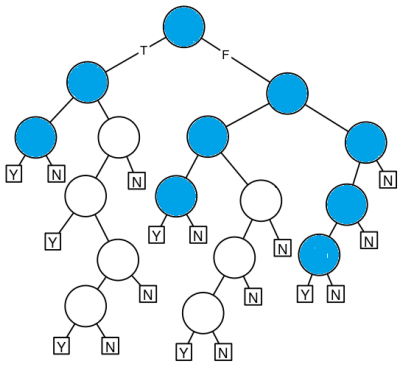

Fig.13: Esempio di albero decisionale

5.1.1 Algoritmo ID3

L'algoritmo ID3 è uno dei primi algoritmi utilizzati per la costruzione di un albero decisionale binario. L'algoritmo consiste nel costruire l'albero binario in modo top-down usando una politica nota come divide et impera (cioè dividere un problema in sotto-problemi più piccoli) [20].
L'algoritmo consiste quindi nella seguente sequenza di passi:

- Assegnare alla radice di partenza le istanze di addestramento;
- Scegliere un attributo mediante *information gain;*
- Creare tanti nodi quanti sono i possibili valori dell'attributo scelto;
- Procede ricorsivamente usando come radici i nuovi nodi.

L*'information gain* consiste nella misura della differenza di entropia calcolata prima a dopo la suddivisione del dataset su attributo. In altre parole, indica la riduzione di incertezza nel dataset dopo aver suddiviso l'insieme di dati sull'attributo:

$$H(S) = \sum_{x \in X} -p(x) \log_2 p(x)$$

Dove:

- S è il dataset corrente per il quale viene calcolata l'entropia che cambia ad ogni passo dell'algoritmo;
- X è l'insieme di classi in S;
- *p(x)* è la proporzione del numero di elementi nella classe *x* rispetto al numero di elementi nell'insieme S.

Quando $H(S) = 0$ il dataset S è perfettamente classificato..
Il valore di *information gain* IG(A), dato un attributo A, può essere calcolato in questo modo:

$$IG(S, A) = H(S) - \sum_{t \in T} p(t) H(t)$$

Dove T è il sottoinsieme creato dalla suddivisione dell'insieme S sull'attributo A tale che $S = \bigcup_{t \in T} t$.

In generale quindi l'algoritmo suddivide i dati in un insieme di dati omogeneo con l'obbiettivo di trovare quell'attributo che determina il più alto valore di *information gain* [23]. Questo procedimento consiste nel calcolare l'entropia di ogni attributo del dataset, partizionare il dataset in insiemi più piccoli utilizzando l'attributo per il quale l'entropia risultante dopo la suddivisione è minima o, equivalentemente, l'*information gain* è massimo.

Per questo lavoro di tesi, a causa dell'inefficienza della rete neurale per l'addestramento di alcuni modelli, è stata utilizzata un'implementazione Java *open-source* di albero decisionale proposto da Weka, che estende l'algoritmo C4.5.

Questa implementazione viene chiamata all'interno del software Weka J48.

J48 (C4.5) aggiunge sostanziali modifiche all'algoritmo descritto precedentemente:

- Creazione una soglia e quindi divisione degli attributi in base al loro valore rispetto alla soglia;
- I valori degli attributi mancanti non vengono utilizzati nei calcoli dell'*information gain* e dell'entropia;

  Gestione degli attributi con costi diversi;

  Potatura degli alberi dopo la creazione: J48 tenterà di rimuovere i rami che non portano alcuna informazione utile per l'addestramento sostituendoli con i nodi foglia.

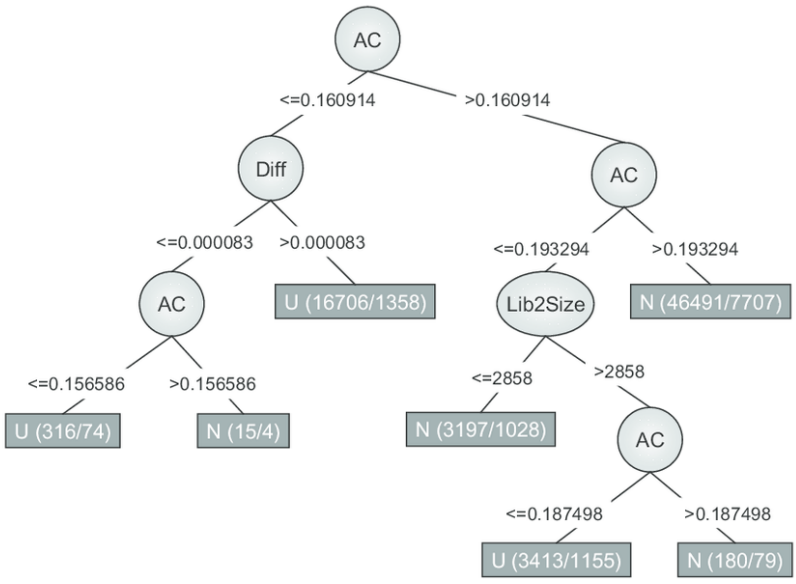

Fig.14: Esempio di albero decisionale generato da J48 [24]

5.2 Random Forest

Random Forest (o *Random Decision Forest*) è un metodo di apprendimento automatico appartenente all'insieme di algoritmi chiamato *Bootstrap Aggregation* o più semplicemente *Bagging* utilizzato per la classificazione e regressione dei dati che fa uso di diversi alberi decisionali. In questo tipo di metodo la classe prodotta al termine della classificazione dipende dall'uscita degli alberi decisionali presi individualmente.

La tecnica chiamata *Bootstrap Aggregation* (o *Bagging*), che sta alla base dell'algoritmo Random Forest, risolve questo problema e permette invece di ridurre la varianza della tecnica basata su un solo albero decisionale. *Bootstrap Aggregation* consiste nel costruire più modelli di alberi decisionali diversi a partire da un singolo set di dati di addestramento ripetutamente suddiviso in sottoinsiemi diversi. Ogni sottoinsieme di dati viene utilizzato per addestrare gli alberi e quindi ogni albero viene costruito indipendentemente dagli altri producendo un risultato generalmente diverso.

Il risultato della previsione viene calcolato utilizzando la media di tutte le previsioni di ogni albero generato rendendo la tecnica più robusta di quella basata su un singolo albero decisionale. Anche se la tecnica risulta più robusta gli alberi decisionali possono presentare però molte somiglianze strutturali compromettendo la bontà delle loro previsioni.

Random Forest rappresenta un miglioramento della tecnica *Bootstrap Aggregation. Random Forest* riduce la correlazione tra i sottoalberi generati sfruttando non solo la selezione casuale dei dati ma anche la selezione casuale delle caratteristiche rispetto alle tecniche illustrate precedentemente.

Supponendo che ci siano N osservazioni ed M caratteristiche nell'insieme di addestramento:

- Viene selezionato casualmente un sottoinsieme dell'insieme di addestramento;
- Viene selezionato casualmente un sottoinsieme delle M caratteristiche e le caratteristiche che forniscono i migliori risultati vengono usate per dividere il nodo iterativamente;

- Si costruisce l'albero più profondo possibile;
- Si ripetono i passi precedenti e la predizione finale si basa sull'aggregazione di predizione di n alberi.

Ogni *decision tree* propone quindi un risultato generalmente diverso. I risultati finali della classificazione vengono decisi tramite il sistema di *majority voting:* per ogni campione si considera il risultato proposto da ogni sotto-albero decisionale e viene decisa l'etichetta in base alla maggioranza di voti. Se ad esempio la foresta è composta da quattro alberi e tre alberi producono in output lo stesso risultato per il campione preso in considerazione, grazie al sistema di *majority voting* viene scelto come risultato definitivo quel risultato che ha avuto la maggioranza di voti.
Questo tipo di approccio permette di gestire dati di grandi dimensioni mantenendo una buona accuratezza dei dati e anche una buona efficienza.

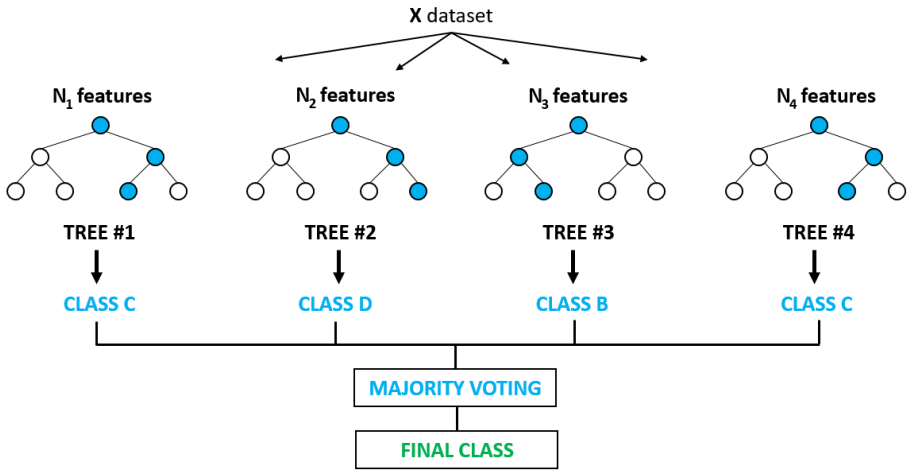

Fig.15: Esempio schema Random Forest

5.3 Rete Neurale

Una rete neurale artificiale è un modello computazionale il cui funzionamento trae ispirazione dai sistemi nervosi biologici. Una rete neurale è composta da molteplici unità, chiamate percettroni, interconnesse tra loro che si attivano quando la quantità di segnale ricevuto è maggiore di una certa soglia di attivazione. Un percettrone riceve in input diversi valori e produce un singolo valore di output [21]. Generalmente ogni vettore è costituito da:

- uno stato $a_j(t)$;
- una soglia θ_j;
- una funzione di attivazione f che calcola una nuova attivazione all'istante t+1 a partire da $a_j(t)$, θ_j e l'input: $a_j(t+1) = f\left(a_j(t), \rho_j(t), \theta_j\right)$;
- una funziona di uscita $o_j = f_{out}\left(a_j(t)\right)$.

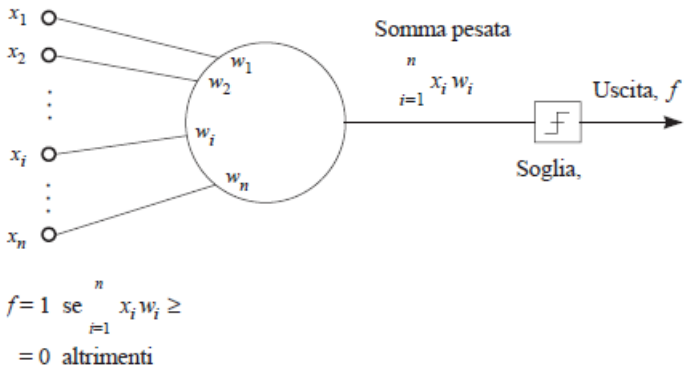

Fig.16: Percettrone [21]

Un percettrone viene quindi definito dalla sua soglia e dai pesi che possono essere rappresentati da un vettore $\boldsymbol{W}$. Dato quindi un vettore degli ingressi denominato $\boldsymbol{X}$ il singolo percettrone avrà in output il valore 1 se il prodotto scalare $s = \boldsymbol{X} \cdot \boldsymbol{W}$ è maggiore della sua soglia, 0 altrimenti.
La funzione di attivazione non lineare generalmente usata è la funzione sigmoide:

$$f(t) = \frac{1}{1 + e^{-t}}$$

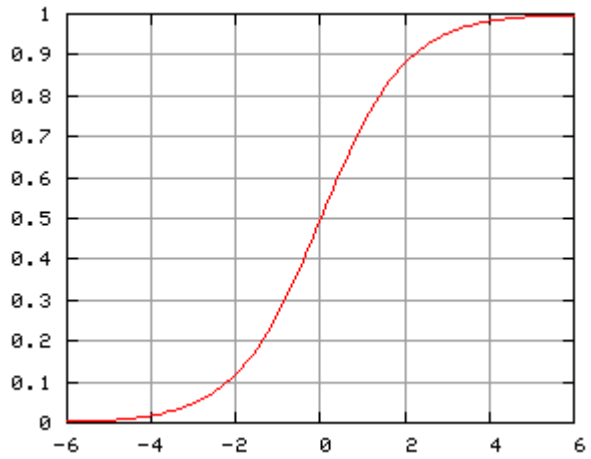

Fig. 17: Grafico funzione sigmoide

Tale modello viene rappresentato come un grafo orientato con nodi rappresentati dai percettroni ed archi pesati. Ogni connessione trasferisce l'uscita di un percettrone *i* all'ingresso di un percettrone *j*. Alcune di queste unità ricevono informazioni dall'esterno formando lo strato di input, altre comunicano solo con percettroni della stessa rete formando lo strato chiamato nascosto mentre altre forniscono informazioni all'esterno della rete formando lo strato di output.

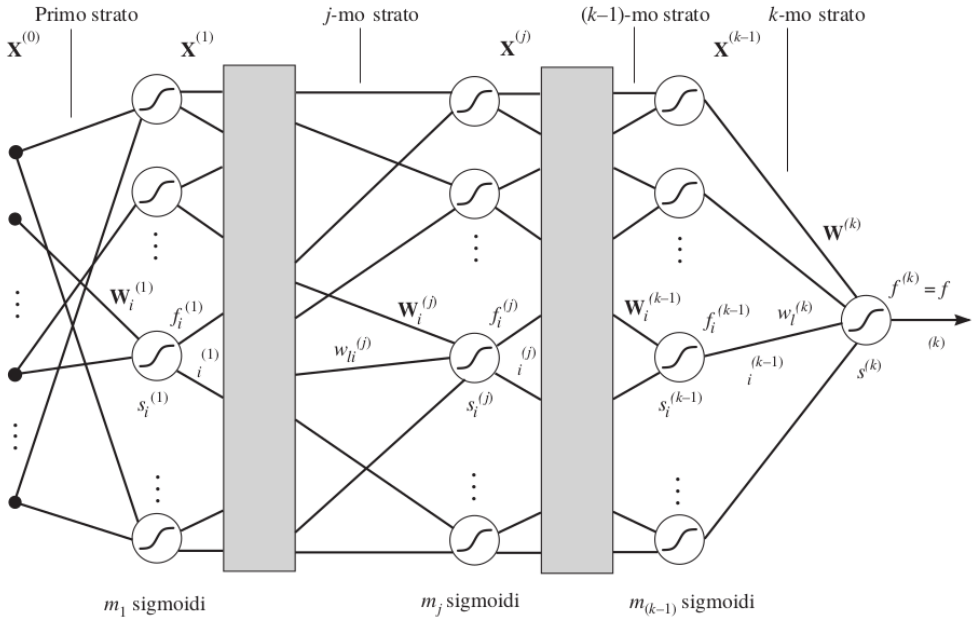

Fig. 18: Esempio schema rete neurale [21]

5.3.1 Metodo della retropropagazione

Il metodo della retropropagazione è un particolare metodo di addestramento supervisionato che sfrutta l'errore tra le uscite desiderate e le uscite effettive del modello usato in combinazione con il metodo di ottimizzazione della discesa lungo il gradiente. L'addestramento consiste quindi nell'aggiornamento ciclico dei vettori dei pesi finché l'errore non scende al di sotto di una soglia prefissata.

Come anticipato per verificare che un singolo percettrone risponda in modo adeguato ai vettori di addestramento si definisce una funzione di errore che generalmente coincide con la funzione di errore quadratico:

$$\varepsilon = \sum_{X_j \in \Xi} (d_i - f_i)^2$$

dovefi è l'uscita effettiva del percettrone per l'ingresso X_i mentre d_i è l'uscita desiderata.

Il metodo di discesa lungo il gradiente consiste quindi nel trovare il minimo della funzione di errore quadratico. Calcolando il gradiente otteniamo:

$$\frac{\partial \varepsilon}{\partial W_i^{(j)}} \stackrel{\text{def}}{=} \left[\frac{\partial \varepsilon}{\partial w_{1i}^{(j)}}, \cdots, \frac{\partial \varepsilon}{\partial w_{li}^{(j)}}, \cdots, \frac{\partial \varepsilon}{\partial w_{m_{j-1}+1i}^{(j)}} \right]$$

dove $w_{li}^{(j)}$è la componente l-esima di $W_i^{(j)}$. Grazie alla regola del concatenamento possiamo scrivere $\frac{\partial \varepsilon}{\partial W_i^{(j)}} = \frac{\partial \varepsilon}{\partial s_i^{(j)}} \frac{\partial s_i^{(j)}}{\partial W_i^{(j)}}$, ponendo$\delta_i^{(j)} = (d - f) \frac{\partial f}{\partial s_i^{(j)}}$. Questo valore indica quanto l'errore quadratico dell'uscita della rete sia suscettibile ai cambiamenti nell'ingresso.

Il gradiente infine può essere scritto così : $\frac{\partial \varepsilon}{\partial W_i^{(j)}} = -2\delta_i^{(j)} \left(X^{j-1} \right)$

I vettori dei pesi vengono modificati secondo la direzione del gradiente negativo, quindi:

$W_i^{(j)} \leftarrow W_i^{(j)} + c_i^{(j)} \delta_i^{(j)} X^{(j-1)}$dove $c_i^{(j)}$ è la costante che determina la velocità di apprendimento dei vettori dei pesi.

5.4 Rete Bayesiana

Una rete bayesiana è un modello grafico probabilistico rappresentato attraverso un grafo orientato aciclico i cui nodi sono etichettati con variabili casuali [21].
È necessario che ogni nodo V_i del grafo sia condizionatamente indipendente da qualsiasi sottoinsieme di nodi non discendenti di V_i, dati i genitori di V_i.
Una rete bayesiana può essere rappresentata attraverso la distribuzione congiunta di probabilità di un insieme di variabili aleatorie:

$$p(V_1, V_2, \dots, V_K | V) = \prod_{i=1}^{k} p(V_i | V_{i-1}, \dots, V_1, V)$$

Addestrare una rete bayesiana significa trovare una rete che corrisponda al meglio a un insieme di addestramento dei dati, cioè determinare la struttura del grafo aciclico sia le tabelle di probabilità condizionata associata ad ogni nodo del grafo.
Per fare ciò è necessaria una metrica per la valutazione delle reti e un metodo per effettuarne la ricerca. Per la scelta della metrica è possibile considerare le proprietà statistiche dei codici efficienti, come ad esempio Huffman. Se si considera l'insieme di addestramento come un vettore di bit da trasmettere, possiamo codificare il vettore con una codifica efficiente.
Dato quindi un insieme di addestramento Ξe una rete bayesiana B, secondo la teoria dell'informazione la migliore codifica di Ξ distribuito secondo le probabilità di B richiede $L(\Xi, B)$ bit:

$$L(\Xi, B) = -log(p[\Xi])$$

con $p[\Xi]$ probabilità secondo la distrubuzione congiunta determiata da B dei particolari dati di Ξ.
È possibile inoltre trovare quella rete bayesiana B_0 che minimizza $L(\Xi, B)$.
Da sola $L(\Xi, B)$ non è una metrica visto che il suo uso favorisce reti sovra-adattate

per l'insieme di addestramento. Conviene quindi aggiungere ad $L(\Xi, B)$ la lunghezza necessaria per inviare B.

Per quanto riguarda invece la ricerca della rete bayesiana è possibile adottare un tipo di ricerca chiamata *hill-descending* per trovare quella che minimizza $L'(\Xi, B)$. Si parte da una configurazione vuota (che rappresenta l'indipendenza tra tutte le variabili) e si applicano piccole modifiche cercando di minimizzare $L'(\Xi, B)$. Queste piccole modifiche possono consistere nella eliminazione o aggiunzione di archi nel grafo o in un cambio di direzione

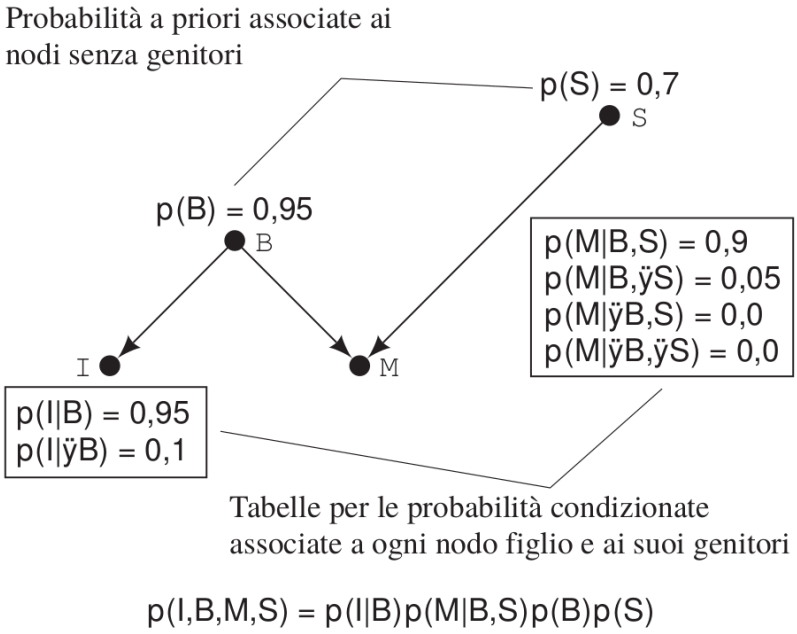

Fig. 19: Esempio schema rete bayesiana [21]

Capitolo 6

*** OMISSIS ***

Capitolo 7

Conclusioni

In questo lavoro è stato presentato un sistema per il riconoscimento di malware che sfrutta strumenti per l'analisi dinamica per analizzare il comportamento dei file e utilizza tecniche di *machine learning* per addestrare dei classificatori automatici in grado di individuare la natura malevola o benevola degli eseguibili analizzati. Il lavoro di tesi svolti ha riguardato inoltre la progettazione di diversi vettori delle caratteristiche per rappresentare in maniera sintetica il comportamento degli eseguibili analizzati.

*** OMISSIS ***

Bibliografia

[1] Andrii Shalaginov, Sergii Banin, Ali Dehghantanha, Katrin Franke. “Machine Learning Aided Static Malware Analysis: A Survey and Tutorial” (2018).

[2] Yudi Prayudi , Imam Riadi. “Implementation of Malware Analysis using Static and Dynamic Analysis Method” *in* International Journal of Computer Applications, April 2015.

[3] Ömer Aslan, Refik Samet. "Investigation of Possibilities to Detect Malware Using Existing Tools".

[4] Sikorski, Michael, Honig. “Practical malware analysis: the hands-on guide to dissecting malicious software”.

[5] Ömer Aslan. “Performance Comparison of Static Malware Analysis Tools Versus Antivirus Scanners To Detect Malware” (2017).

[6] P. V. Shijoa, A.Salimb. “Integrated static and dynamic analysis for malware detection” (2008).

[7] Bojan Kolosnjaji, Apostolis Zarras, George Webster, and Claudia Eckert. “Deep Learning for Classication of Malware System Call Sequences”.

[8] Microsoft. *Microsoft Portable Executable and Common Object File Format Specication*. Rapp. tecn. Gen. 2017.

[9] Cuckoo Sandbox Book.

[10] https://cuckoosandbox.org/.

[11] Karthik Raman. “Selecting Features to Classify Malware”.

[12] Hisham Shehata, Galal1Yousef, Bassyouni Mahdy1, Mohammed Ali Atiea. “Behavior-based features model for malware detection” (2015).

[13] Omid E. Davidt, Nathan S. Netanyahu. “DeepSign: Deep Learning for Automatic MalwareSignature Generation and Classification”.

[14] Egele, M., Scholte, T., Kirda, E., and Kruegel, C. 2012. “A survey on automated dynamic malware-analysis techniques and tools. ACM Comput. Surv”.

[15] Akinori Fujino, Junichi Murakami, Tatsuya Mori. “ Discovering Similar Malware Samples Using API Call Topics”.

[16] Daisuke Inoue, Katsunari Yoshioka, Masashi Eto, Yuji Hoshizawa, Koji Nakao. “Malware Behavior Analysis in Isolated Miniature Network for Revealing Malware’s Network Activity “.

[17] Zhao, Ming Xu, Ning Zheng Jingjing Yao, Qiang Hou. “Malicious executables classification based on behavioral factor analysis Hengli “.

[18] Dmitri Bekerman, Bracha Shapira, Lior Rokach, Ariel Bar. “Unknown Malware Detection Using Network Traffic Classification”.

[19] https://github.com/jgamblin/isthisipbad/blob/master/isthisipbad.py.

[20] https://en.wikipedia.org.

[21] Nils J. Nilsson and S. Gaglio. “Intelligenza artificiale”, 1 Feb 2002.

[22] Tina R. Patil, Mrs. S. S. Sherekar. “Performance Analysis of Naive Bayes and

J48 Classification Algorithm for Data Classification".

[23] Gaganjot Kaur Amit Chhabra. "Improved J48 Classification Algorithm for the Prediction of Diabetes".

[24] Angelica Lindlöf, Marcus Bräutigam, Aakash Chawade, Björn Olsson. "REVIEW Evaluation of Combining Several Statistical Methods with a Flexible Cutoff for Identifying Differentially Expressed Genes in Pairwise Comparison of EST Sets".

[25] Leo Breimann "BaggingPredictors".

[26] De Paola, A., Gaglio, S., Lo Re, G., Morana, M., "A hybrid system for malware detection on big data", INFOCOM 2018 IEEE Conference on Computer Communications Workshops, pp. 45-50, 2018;

[27] De Paola, A., Favaloro, S., Gaglio, S., Lo Re, G., Morana, M., Malware detection through low-level features and stacked denoising autoencoders, CEUR Workshop Proceedings, vol. 2058, 2018;

[28] S. Gaglio and G. Lo Re, Advances onto the Internet of Things: How ontologies make the Internet of Things meaningful. Springer, 2014