Alma Mater Studiorum - Università di Bologna

Campus di Cesena

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA
Corso di Laurea in Ingegneria e scienze informatiche

Analisi di malware in ambiente virtualizzato: l'esempio di Cuckoo Sandbox

Elaborato in System Integration

Relatore:
Chiar.mo Prof. Vittorio Ghini

Presentata da:
Davide Schiaroli

Seconda sessione di Laurea A.A. 2019-2020

Io penso che i virus dei computer debbano essere considerati come una vita. Io penso che ci insegni qualcosa sulla natura umana, dato che l'unica forma di vita che abbiamo creato è fino ad ora puramente distruttiva. Abbiamo creato la vita a nostra stessa immagine.

Stephen Hawking

Indice

Elenco delle figure

Capitolo 1

Premessa

Fino a prima dell'esplosione di Internet tutti i malware, o virus informatici, potevano circolare solo attraverso l'uso di supporti fisici come ad esempio i floppy disk. Il primo virus studiato fu "Elk Cloner", sviluppato negli U.S.A. nel 1982. Esso era in grado, una volta scritto su di un floppy disk, di infettare ogni sistema operativo in cui il floppy veniva inserito e fu in grado di diffondersi su diverse centinaia di computer. Come molti altri virus di quegli anni, non era in grado fare ingenti danni al sistema operativo, ma era piuttosto una sorta dimostrazione. L'esplosione del World Wide Web di fine anni 90' portò i virus non più a diffondersi attraverso dispositivi fisici, ma utilizzando la rete Internet, in particolare tramite e-mail. Nacque una seconda generazione di malware, chiamati worm[1], che fu molto pericolosa poiché erano in grado di eseguire il proprio codice direttamente nel sistema operativo. Notevole il caso di SQL Slammer, che all' inizio del terzo millenio riuscì in soli 15 minuti dopo il primo attacco a infettare circa 75000 server internet, mettendo in crisi la Bank Of America, il numero unico di emergenza americano 911 e molte linee aeree che furono cancellate. Fu stimato che i danni economici procurati furono superiori al miliardo di dollari. Insieme ai virus, nacquero gli antivirus, finalizzati a proteggere dai malware rilevandoli dapprima e rendendoli inoffensivi poi. Inizialmente gli antivirus utilizzavano il metodo delle firme per poter identificare i virus; ogni virus infatti veniva contrassegnato da una o più stringhe che venivano trovate analizzando l'eseguibile. Poco dopo però i

[1]Worm, o verme tradotto dall'inglese, rende l'idea di come facile sia la sua diffusione

creatori di virus iniziarono a fare delle leggerissime modifiche al codice dei virus, che non ne cambiavano il funzionamento ma che li rendeva irriconoscibili agli antivirus. Questo, unito alla difficoltà della diffusione delle firme tra database di antivirus diversi portò alla nascita delle tecniche di analisi euristica, in cui gli algoritmi cercavano di capire se le porzioni di codice presenti potessero essere malevole oppure no. Nel tempo ci si è accorti che questo approccio non sarebbe bastato, i creatori di virus sarebbero stati sempre più veloci di chi li avrebbe dovuti identificare e fermare. Così si è iniziato ad analizzare i malware per identificarli e sviluppare adeguate contromisure. Le tecniche più utilizzate erano e sono ancora quelle delle analisi statica e dinamica che possono e devono essere usati in modo complementare per essere più efficienti possibili. Spesso però per perfezionare queste analisi servono numerosi software, come ad esempio un programma per analizzare il traffico di rete, uno per analizzare lo storico dei processi aperti, uno per la memoria. Ma non basta, ognuno di questi programmi deve essere usato da una persona qualificata che riesca ad analizzare i risultati e possa esprimere un parere sulla natura del software controllato. Fortunatamente nell'ultimo decennio sono stati sviluppati diversi programmi che permettono di analizzare software. Cuckoo Sandbox è l'esempio più apprezzato di questo tipo di software e vanta una discreta facilità di utilizzo e di espansione e ne vedremo in questa tesi gli aspetti più importanti.

Capitolo 2

Introduzione

In questo capitolo andremo ad analizzare quali sono gli aspetti di sicurezza informatica più importanti da prendere in considerazione nella realizzazione di un sistema di analisi malware virtualizzato, in particolare cercheremo di capire perché è importante saper riconoscere le diverse tipologie di malware e quali sono i meccanismi di sicurezza utilizzati nei sistemi informatici, in particolare in quelli Unix[1].

2.1 Sandboxing

Nel campo della sicurezza informatica, per Sandboxing[5] si intende il meccanismo utilizzato per separare istanze di programmi in esecuzione dall'ambiente in cui vengono eseguite, cioè solitamente il sistema operativo. Ciò viene fatto per scongiurare gli effetti negativi di possibili fallimenti di sistema o per evitare che software malevolo possa infettare il sistema operativo. I sistemi virtualizzati consentono il Sandboxing poiché il sistema operativo ospite non può influenzare negativamente il sistema operativo ospitante, tranne per una vulnerabilità che vedremo più avanti. Una Sandbox tipicamente fornisce un insieme di risorse come memoria volatile e di archiviazione, mentre altre vengono disabilitate come la rete e la possibilità di utilizzare direttamente i dispositivi di Input/Output. Esistono diversi modi

[1]Sistema operativo sviluppato alla fine degli anni 60 nei laboratori Bell e At&T principalmente da Ken Thompson e Dennis Ritchie, pionieri dell'informatica moderna

di applicare questo meccanismo di sicurezza, come ad esempio l'uso di macchine virtuali o di container, sfruttando quindi i benefici delle virtualizzazione, sia a livello hardware che a livello di sistema operativo, oppure utilizzando alcune funzionalità del Kernel Linux.

2.2 Sandboxing nei sistemi Unix

2.2.1 Seccomp

Seccomp[3], abbreviazione di secure computing mode, è una funzionalità che permette ad un processo di passare ad uno stato "sicuro", disabilitando le System Call. Ogni tentativo di chiamata porterebbe infatti alla terminazione del processo con un a SIGKILL o con SIGSYS. Una estensione più utilizzata di Seccomp è Seccomp-bpf, che permette di filtrare quali System Call si possono utilizzare attraverso l'uso di politiche configurabili.

2.2.2 Namespaces

I Namespaces sono una caratteristica molto importante dei sistemi Linux che permette di isolare i processi in base a diversi criteri. I namespace attualmente esistenti sono:

- Cgroup
- IPC, usato per la comunicazione tra processi
- Network, permette di virtualizzare lo stack network di un sistema. Ogni namespace avrà un proprio set di indirizzi IP privati, una sua tabella di instradamento, un proprio elenco di porte e risorse di rete.
- Mount, un processo può vedere un diverso mount point diverso da quello originale. Per farlo viene creato un diverso e separato file System che viene di volta in volta associato ai diversi processi, cosicché non effettuino modifiche al file System originale.
- PID (Process Id) Isola il PID di un processo dalla gerarchia dei PID. E' possibile quindi avere in uno spazio dei nomi di un PID, uno identico ad un altro PID al esterno, così come posso avere un altro PID 0, cioè di Init.

- UTS (hostname e NIS domain name), un processo può avere un differente set di nomi di dominio e di host.
- User (UID, User ID e GID,group ID)
- Time, i processi possono vedere un diverso sistema temporale.

2.2.3 Cgroups

Cgroups o control sono un strumento presente nel Kernel Linux fin dalla versione 2.6 (2007) che permette di nascondere le caratteristiche del control group al quale appartiene il processo. Un processo che controllerà il control group di un'altro processo, vedrà un percorso relativo non al vero control group ma al control group assegnato dal namespace.

2.2.4 Chroot

Chroot è un meccanismo di Sandboxing presente in Unix fin dalla fine degli anni 70 e permette di "eseguire o comandi o una Shell interattiva usando una speciale directory di amministratore", secondo la sua documentazione ufficiale. Questo permette di evitare che processi che devono rimanere isolati dal sistema su cui vengono eseguiti possano "vedere" il filesystem principale. Per fare ciò è necessario che venga creata una nuova directory, che useremo come mount point, poi al suo interno inseriamo le librerie necessarie al processo per essere eseguito. Se invece volessimo poter eseguire un'applicazione in modalità Sandbox senza dover configurare nessuna opzione, è possibile utilizzare Firejail. Firejail è un programma scritto in C che permette di isolare programmi per ragioni di sicurezza o di debugging e lo fa utilizzando creando un filesystem temporaneo e utilizzando lo spazio dei nomi dell'utente.

2.3 Classificazione dei malware

Per malware si intendono tutti quei programmi che, una volta penetrati in un sistema informatico, cercano di disturbare le azioni svolte dagli utenti o di sottrarne informazioni importanti. A seconda della modalità in cui entrano in un sistema informatico e di come riescano quindi a diffondersi è possibile classificarli.

- Virus → Sono tipi di malware che una volta eseguiti, cercano di diffondersi modificando l'ambiente circostanze, quindi altri software e file, infettandoli col proprio codice. Una volta che il codice in un file infetto passa ad un altro sistema il virus cercherà di nuovo di diffondersi nel nuovo ambiente, così ricorsivamente finché non viene fermato. Secondo un articolo del Il sole 24 ore, a proposito di una ricerca di Cybersecurity Ventures nel 2015 i virus informatici nel mondo hanno causato danni per 3 miliardi di dollari[1].

- Worms → Sono malware che non necessitano di legarsi ad altri software o malware ma attaccano il sistema operativo per potersi replicare e diffondersi attraverso Internet. Per diffondersi fanno largo uso di Ingegneria Sociale. Il primo Worm creato nella storia, nel 1988, quando ancora Internet contava meno di 10000 mila computer connessi e riuscì ad infettarne circa il 5%. Questo worm, chiamato Morris Worm fu creato sfruttando le vulnerabilità di Finger e Sendmail, entrambi software Unix usati per lo scambio di informazioni, in particolare email.

- Trojan Horse → Sono malware che vengono nascosti in software che sono apparentemente normali. Infatti sono proprio gli utenti ad installarli, navigando su internet e cercando versioni gratuite di software a pagamento per esempio. Non contengo funzionalità per la replicazione quindi devono essere scambiati attraverso Internet o supporti di massa. Nonostante questo, sono la tipologia di malware più diffusa, circa il 70

- Backdoor → Sono dei software che consentono di scavalcare le normali procedure di autenticazione di un sistema. Alcune volte sono utilizzati lecitamente per il recupero dati, altre permettono ad attaccanti di avere una via facile e veloce per accedere da remoto ad un sistema.

- Spyware → Software che raccogliono dati, talvolta sensibili, su tutto quello che è presenta nel sistema. Vengono poi mandati ad un destinatario che li potrà usare per organizzare una truffa, usando anche l'Ingegneria sociale

- Hijacker → Questi tipi di malware infettano sopratutto i browser per dirottare la navigazione su pagine web non richieste. Non sono particolarmente dannosi per il

sistema, ma vengono utilizzati per incrementare i guadagni relativi alle pubblicità dei siti web.

- Rootkit → Sono uno o più software che permettono l'accesso non autorizzato ad un sistema. Di solito vengono utilizzati in coppia, dove uno dei due svolge la funzione di un normale software, come per esempio quello per il login ad un sistema (/bin/login per Unix) mentre l'altro viene utilizzato per effettuare azioni malevoli all'insaputa del utente.

- Scareware → Tipo di software non propriamente malware ma che vengono pubblicizzati attraverso le metodologie tipiche dell'ingegneria sociale. Sono di questo tipo i software che mirano a risolvere i problemi del computer o che promettono di aggiornare i driver . Solitamente non contengono al loro interno malware, ma piuttosto simulano il comportamento di un normale software, notificando l'utente di aver trovato dei problemi e che per risolverli il programma necessità di una licenza a pagamento

- Rabbit → Tipo di malware che si duplica molto velocemente fino ad utilizzare tutte le risorse disponibili nel sistema.

- Adware → Per Adware si intendono tutti quei software distribuiti gratuitamente, che si sostengono attraverso l'uso di pubblicità. Molti di questi però contengono rischi per la sicurezza poiché rallentano il sistema operativo aprendo pop-up o pagine HTML non richieste. Altre volte diffondendo informazioni personali a server remoti come le abitudini di navigazione e la cronologia.

- Malvertising → Tipo di pubblicità presenti in siti web che viene usata nella diffusione di malware, invitando per esempio l'utente a scaricare software contraffatto.

- File batch → Sono file di testo interpretabili dal prompt dei comandi Windows. Possono essere molto pericolosi se un malintenzionato ha accesso (fisico o remoto) al sistema poiché permettono di effettuare qualsiasi operazione sul sistema operativo.

- Keylogger → Sono software in grado di memorizzare tutte le stringhe che vengono digitate, copiate o incollate nel sistema operativo. Talvolta vengono installati in particolari dispositivi USB, come delle chiavette dati che vengono poi posizionate nei

computer di utenti ignari. Le informazioni vengono poi inviate all'attaccante sfruttando la connessione internet del computer dove vengono inserite Viene solitamente usato per sottrarre agli utenti le credenziali di accesso a siti web .

- Rogue antispyware → I Rogue antispyware, o Rogue o anche FraudTool sono malware che si presentano come un normale antivirus. Durante il funzionamento permettono di effettuare scansioni, che però danno risultati falsati, trovando ad esempio molti malware in realtà inesistenti, allo scopo di far acquistare all'utente la licenza completa.

- Ramsonware → Particolare virus che cifra tutti i dati personali presenti nel disco di un computer e avverte al contempo all'utente che per decifrarli dovrà versare un "riscatto" attraverso dei sistemi di pagamento non rintracciabili, come Bitcoin.

- Bomba logica o a tempo → è un tipo di malware che al verificarsi di determinate condizioni (numero di record di un database,cancellazione di un file, scadere di un timer) mette in atto determinate funzioni malevole. Il nome "bomba" deriva dal fatto che l'utente è ignaro della presenza di questa tipologia di malware, finché le azioni malevole non vengono intraprese, cancellando ad esempio dei file.

- Bomba a decompressione → E' un archivio malevolo organizzato in modo da rendere inutilizzabile il sistema, o parte di esso, sul quale viene aperto. Queste tipologie di archivi vengono infatti costruite ricorsivamente di file che estratti, contengono in realtà file di dimensioni molto maggiori. Ad esempio il file "42.zip" che come archivio ha una dimensione di soli 42 Kilobytes, una volta estratto pesa in realtà 4.5 Petabytes.

In generale i malware sfruttano bug e vulnerabilità di software di largo uso, che siano sistemi operativi o applicazioni molto usate, per compiere azioni malevole. Nella maggior parte dei casi queste azioni sono mirate ad un ritorno economico da parte dell'attaccante, ma negli ultimi anni sono sempre più frequenti attacchi "etici", cioè mirati a divulgare informazioni, come forma di protesta o per condannare governi corrotti.

2.4 Common Vulnerabilities and Exposure

Il Common Vulnerabilities and Exposure, o CVE è un registro pubblico che contiene tutte le falle di sicurezza note. Questo standard permette una forte comunicazione nel mondo della sicurezza informatica. I CVE riguardano tutti i tipi di software rilasciati pubblicamente, anche le versioni beta, e sono formati da una stringa del tipo: CVE – Anno – Numero. Ad esempio, a Meltdown[4], una delle più famose vulnerabilità degli ultimi anni, che permetteva agli attaccanti di eseguire codice in aree protette di un computer che lavorano con CPU Intel prodotte dopo il 1995, è stato assegnato il codice CVE-2017-5754.

2.5 Principio del privilegio minimo

In informatica, il principio del privilegio minimo prevede che ogni processo, thread o unità di elaborazione abbia accesso e che possa quindi "vedere" solo ed esclusivamente le risorse necessarie al suo funzionamento.I meccanismi che attuano questa politica sono le capabilities, o più frequentemente le liste a controllo di accesso. Quando parliamo di controllo di accesso, però, ci stiamo in realtà riferendo a 4 "livelli" di sicurezza:

- Impedire l'accesso, se voglio essere sicuro che nessun utente possa danneggiare un altro utente
- Consentire l'accesso, cioè selezionare cosa, in termini di risorse, possano avere in comune due utenti
- Revocare l'accesso, dopo aver permesso l'accesso, è possibile che io voglia che un utente non abbia più quel permesso

Le capabilities cercano di rispondere a queste esigenze creando dei Token da assegnare ai diversi programmi. Quindi se il programma x vuole accedere al oggetto y necessità di un Token e ogni Token specifica per un oggetto quali operazioni sono permesse (lettura, scrittura, etc..). Il sistema in cui devono essere memorizzate però è il problema principale per cui non esistono molti sistemi che utilizzano le capabilities. Oggi vengono universalmente usate le ACL, cioè Access Control List. La soluzione adottata è stata quella di fornire al filesystem il diritto di accesso a se stesso, in modo da usare l'identità dell'utente come

modalità di accesso. Ad ogni oggetto è collegata una ACL che ne permette oppure no determinate azioni.

2.6 Ingegneria sociale

L'ingegneria sociale è lo studio del comportamento di una persona, delle sue abitudini e di come potrebbero reagire a determinati eventi. Nella sicurezza informatica si approfondiscono i comportamenti umani per sottrarre dati riservati, per poi utilizzarli principalmente a scopi illegittimi . Per riuscire nell'intento un attaccante, definito "Social Engineer" deve riuscire a fingersi un'altra persona, oppure convincere un individuo nella buona fede di un messaggio. Solitamente un attacco ha bisogno di alcune fasi preliminari, come per esempio ricavare tutte le informazioni più importanti sul bersaglio, come e-mail, telefono e anche talvolta data di nascita e residenza. Poi, dopo aver validato le informazioni raccolte si passa ad indagare riguardo informazioni più personali come stile di vita, dialettica e abitudini.

2.6.1 Pretexting

Durante questa fase viene creato un pretesto, cioè una situazione che porti l'utente a fidarsi dell'interlocutore. Spesso l'attaccante si finge qualcuno di autorevole, come ad esempio una società o una persona famosa, un amico della vittima oppure il personale di forze della polizia. In questa fase l'attaccante si occupa appunto di creare questo scenario, utilizzando anche i dati raccolti nelle fasi preliminari

2.6.2 Phishing

Il Phishing è una tecnica utilizzata per ricavare informazioni personali di una vittima. Spesso vengono usate mail in cui l'attaccante si finge una società o persona famosa e dove vengono proposte offerte molto allettanti, come vincite di denaro o prodotti regalati. La vittima viene indotta quindi a scaricare file infetti, oppure a compilare form con informazioni riservate, spesso anche coordinate bancarie.

2.6.3 Contromisure

Non esistono software o azioni immediate che possono permettere di evitare del tutto il fenomeno del Phishing o più in generale di essere vittima di ingegneria sociale. Tra le azioni fondamentali da svolgere è importante controllare sempre gli indirizzi dei siti web visitati per evitare di essere stati reindirizzati in siti malevoli, controllare sempre i mittenti delle email ricevuti, installando anche filtri anti-spam all'occorrenza, cambiare spesso password e non usarne uguali per diversi siti web. All'interno di una azienda però, soprattutto se non lavora prettamente in ambito informatico, è necessario formare il personale in modo da rendere sicura la rete aziendale. Gli attaccanti che sono intenzionati ad attaccare l'azienda infatti, saranno sicuramente invogliati ad attaccare prima l'anello debole, cioè l'essere umano, prima di passare ad attaccare la rete aziendale.

2.6.4 Il caso Associated Press

Nell'Aprile del 2013, l'account Twitter dell'Associated Press[2] pubblicò un falso tweet riguardante un incidente avvenuto alla Casa Bianca. Nei pochissimi minuti in cui questo tweet fu pubblico, milioni di utenti lo videro e il Dow Jones perse 151 punti in borsa, equivalenti a circa 136 miliardi di dollari. L'attacco informatico avvenne grazie ad uno dei più semplici escamotage dell'Ingegneria sociale. Una mail, ovviamente fasulla indirizzata all'Associated Press che contenteva un form Twitter. Le informazioni di login furono poi usate dagli attaccanti per pubblicare un tweet dall'account officiale e questo permise loro di speculare in borsa, anche se non sono noti i guadagni ottenuti da questo gruppo criminale e non è più stato chiarito il coinvoglimento di gruppi terroristici[2].

[2]L'Associated Press, nota anche come APTN, è la prima agenzia di stampa internazionale, con sede negli U.S.A.

Capitolo 3

Cuckoo Sandbox

3.1 Introduzione a Cuckoo Sandbox

Cuckoo Sandbox è un software open source per l'automazione dell'analisi di malware scritto in Python, composto di diversi componenti che permettono l'esecuzione di file sospetti in un ambiente isolato. Tra le sue caratteristiche principali, Cuckoo annovera:

- 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

3.2 Casi d'uso

Cuckoo Sandbox può essere utilizzato per analizzare diverse tipologie di file, come:

- Eseguibili Windows
- Documenti office o PDF
- URL e file HTML
- Script PHP

- Archivi ZIP, RAR etc..
- Eseguibili Java (JAR)
- Script Python

3.3 Architettura

Cuckoo è costruito su di un architettura master/slave, dove il master è in esecuzione sul sistema operativo ospitante (host) e si occupa della gestione delle analisi e della generazione dei report. Gli slave invece sono in esecuzione su una o più macchine virtuali (guest) e si occupano di raccogliere dati sull'esecuzione del malware. Per comunicare questi due componenti usufruiscono di una interfaccia virtuale creata e gestita dall' hypervisor. Virtualbox per esempio fornisce fino ad 8 interfacce di rete PCI virtuali da utilizzare tramite macchina virtuale, dove per ognuna di esse è possibile selezionare il tipo di architettura hardware che verrà virtualizzato.

3.4 Estensione

Oltre ai moduli già presenti è possibile ampliare il funzionamento di Cuckoo Sandbox scrivendo altri moduli (sempre in Python).

3.4.1 Moduli Auxiliary

Cuckoo permette di ampliare le funzionalità di analisi tramite i moduli Auxiliary. Questi moduli definiscono le procedure che devono essere eseguite in parallelo ad ogni singola analisi. Ogni modulo deve ereditare la classe Auxiliary e deve implementare principalmente due funzioni, start e stop. Il metodo start verrà eseguito prima di far partire la macchina virtuale dedicata all'analisi del malware mentre end verrà eseguito alla fine dell'analisi e prima della fase di reporting Ad esempio, il modulo sniffer.py contiene le istruzioni per eseguire lo sniffing dei pacchetti tramite Tcpdump.

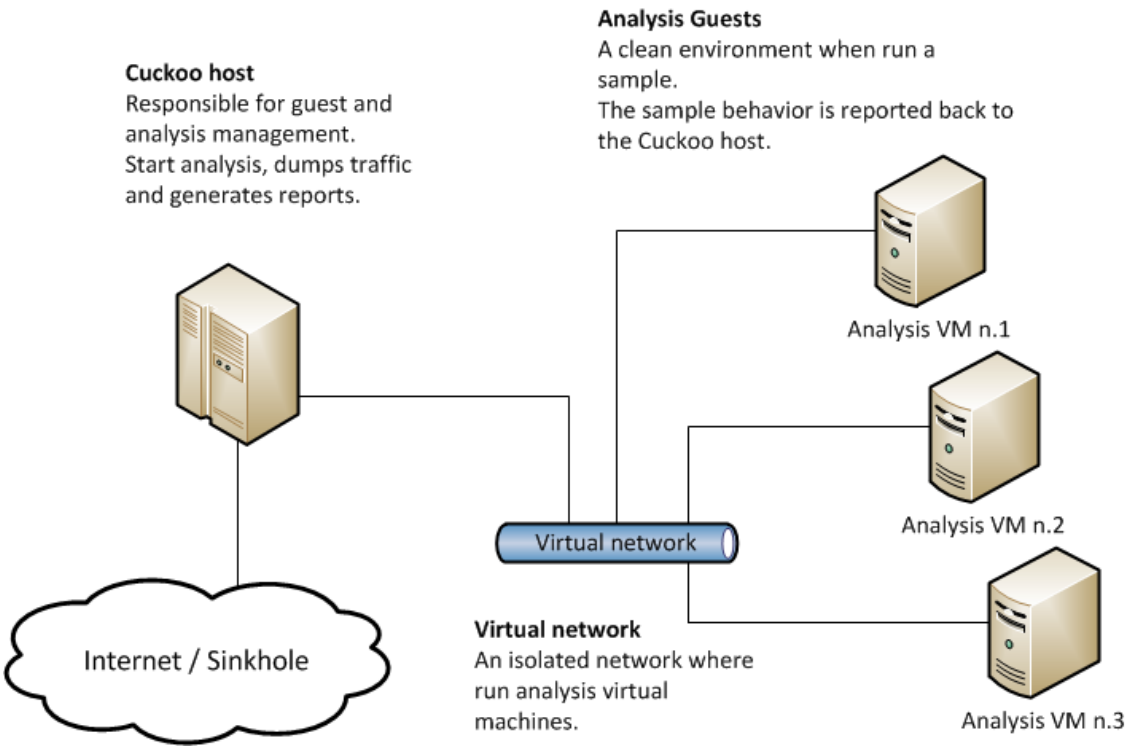

Figura 3.1: Architettura master/slave di cuckoo sandbox

```
from cuckoo.common.abstracts import Auxiliary

class MyAuxiliary(Auxiliary):

    def start(self):
        # Do something.

    def stop(self):
        # Stop the execution.
```

Figura 3.2: Esempio codice moduli Auxiliary

3.4.2 Moduli machinery

I moduli Machinery forniscono indicazioni su come Cuckoo deve interagire con il software di virtualizzazione scelto (o con la macchina fisica). Poiché la personalizzazione è uno dei punti

focali della struttura di Cuckoo, non c'è alcun software di virtualizzazione predefinito, anche se VirtualBox e VMware sono preferiti per via della numerosa documentazione presente in rete, ed è quindi possibile creare il proprio modulo per interfacciare qualsiasi Hypervisor. Per ogni modulo creato è necessario creare un file di configurazione in cui specificare il nome della macchina di analisi, la piattaforma (sistema operativo) e l'indirizzo IP. Nelle ultime versioni di Cuckoo lo sviluppo di nuovi moduli è facilitato da LibVirt.

3.4.3 Moduli Analysis

I pacchetti di analisi sono classi Python che descrivono come i componenti di analisi di Cuckoo devono condurre le analisi per ogni diverso tipo di file. Come per gli altri moduli è possibile scriverne di nuovi partendo dalla classe base, diversa per ogni tipologia di file. I metodi principali in questo caso sono start, check, execute e finish. Start contiene le operazioni di inizializzazione, come ad esempio salvare lo stato della memoria prima di far partire l'eseguibile malware Check viene eseguita ripetutamente, ogni secondo circa, finché il malware è in esecuzione. Ad esempio, posso porre come condizione di uscita dalla analisi quando un determinato file viene creato o quando una chiave di registro viene creata. Execute Gestisce l'esecuzione del malware e le DLL Injection. Finish è chiamata da Cuckoo prima del termine dell'analisi e di norma contiene un'opzione per effettuare il dumping della memoria.

3.4.4 Moduli processing

I moduli processing permettono di analizzare le informazioni ottenute indipendetemente dagli altri moduli installati, inviando i dati ad un container globale. Tra i moduli processing sono presenti:

- Snort
- String
- Suricata
- VirusTotal

Un modulo processing di esempio può essere scritto nel seguente modo:

```
from cuckoo.common.abstracts import Processing

class MyModule(Processing):
    enabled = False

    def run(self):
        self.key = "key"
        data = do_something()
        return data
```

3.5 Interfaccia web

Cuckoo presenta un'interfaccia web costruite tramite framework Django. Questa interfaccia, che permette di caricare file da analizzare e visualizzare i report, è estremamente personalizzabile, in modo da adattarsi al tipo di sistema in cui andrà ad essere installato Cuckoo. Tutte le impostazione dell'interfaccia sono nel file local_settings.py, mentre le informazione vengono recuperate dal database MongoDb, quindi è necessario che sia abilitato. Per configurare l'interfaccia web possiamo gestire la lingua le dimensione massime dei file caricati e aggiungere o togliere amministratori. Se invece volessimo costruire la parte web di Cuckoo in maniera più rubusta è possibile esporre l'interfaccia tramite applicazione WSGI ad un server Web, utilizzando uWSGI e Nginx.

3.6 Signatures

Cuckoo permette inoltre la creazione di signature personalizzate che possono essere utilizzate per identificare dei pattern specifici. Per farlo, basta creare un nuovo script come il seguente:

```
from cuckoo.common.abstracts import Signature
```

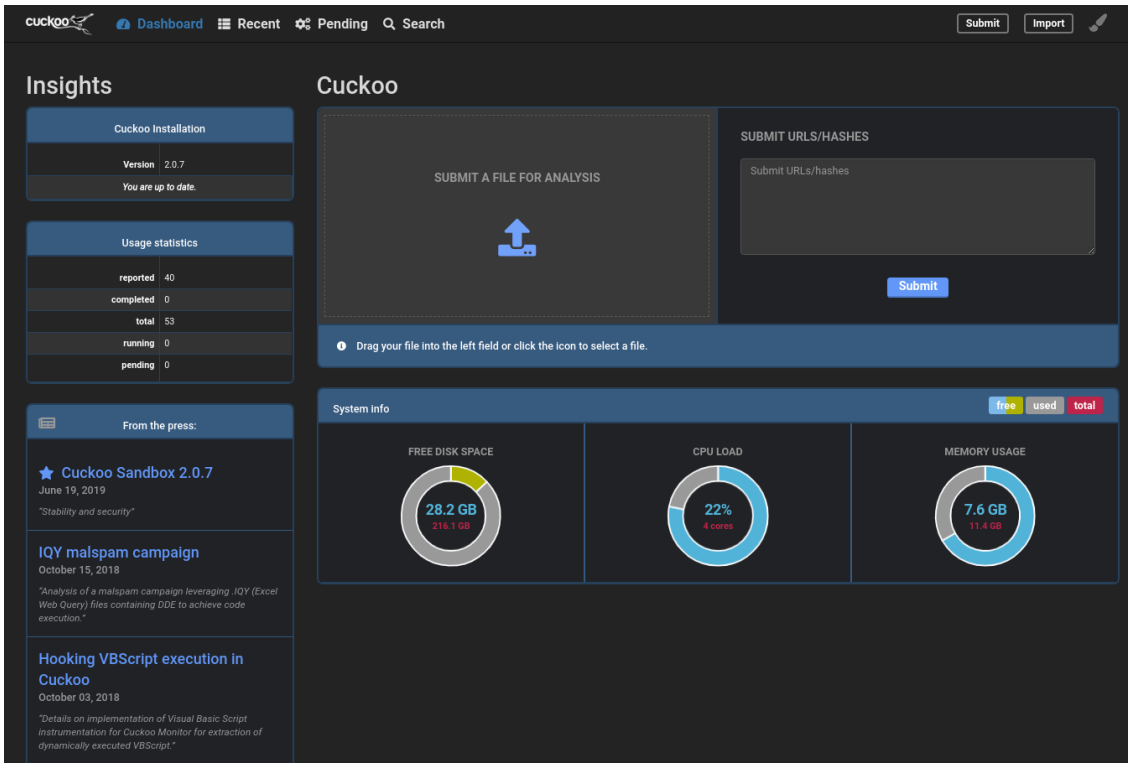

Figura 3.3: Interfaccia web default di cuckoo sandbox

```
class CreatesExe(Signature):
    name = "creates_exe"
    description = "Creates a Windows executable on the filesystem"
    severity = 2
    categories = ["generic"]
    authors = ["Cuckoo Developers"]
    minimum = "2.0"

    def on_complete(self):
        return self.check_file(pattern=".*\\.exe$", regex=True)
```

Questo script è molto semplice, in quanto restituisce un risultato booleano, in cui viene dato per vero qualsiasi file che termina con estensione ".exe". E' possibile creare signature anche molto più complesse, per esempio inserendo dei risultati basati su eventi.

Capitolo 4

Tecnologie e protocolli

4.1 Strumenti di rete

4.1.1 IDS e IPS

Per IDS, o Intrusion Detection System si intendono quei sistemi, sia solo software che embedded, utilizzati al fine di analizzare il traffico in ingresso e in uscita da una rete. Lo scopo principale è senza dubbio quello di rilevare eventuali pacchetti sospetti diretti agli host della rete osservata, tramite il salvataggio in file di log o in appositi database. Gli IPS invece, acronimo di Intrusion Prevention System, sono sistemi che si occupano di prevenire che pacchetti malevoli possano entrare in una rete, di solito LAN, attraverso il filtraggio in base a delle specifiche regole. Ad ogni pacchetto che deve entrare nella rete infatti, l'IPS deve confrontarlo con le regole e in base a quelle decidere per l'eventuale scarto (DROP) del pacchetto. Molte volte le funzioni di questi sistemi sono svolte simultaneamente dagli IDPS. Sono presenti due diverse categorie di IDS, Signature-based IDS e Anomaly-based che hanno pro e contro che sostanzialmente si equivalgono, pertanto anche i venditori hanno cominciato a realizzare sistemi che offrono entrambe le funzionalità Nei Signature-based IDS, le regole o pattern sono basati su minacce già analizzate. Una volta che un oggetto analizzato ha riscontro con le minacce nel db viene inviato un alert all'amministratore di sistema. E' così possibile scoprire attività di netowrk scanning, tentativi di attacchi o traffico di malware. Nei Anomaly-based IDS invece, vengono fortemente monitorate le

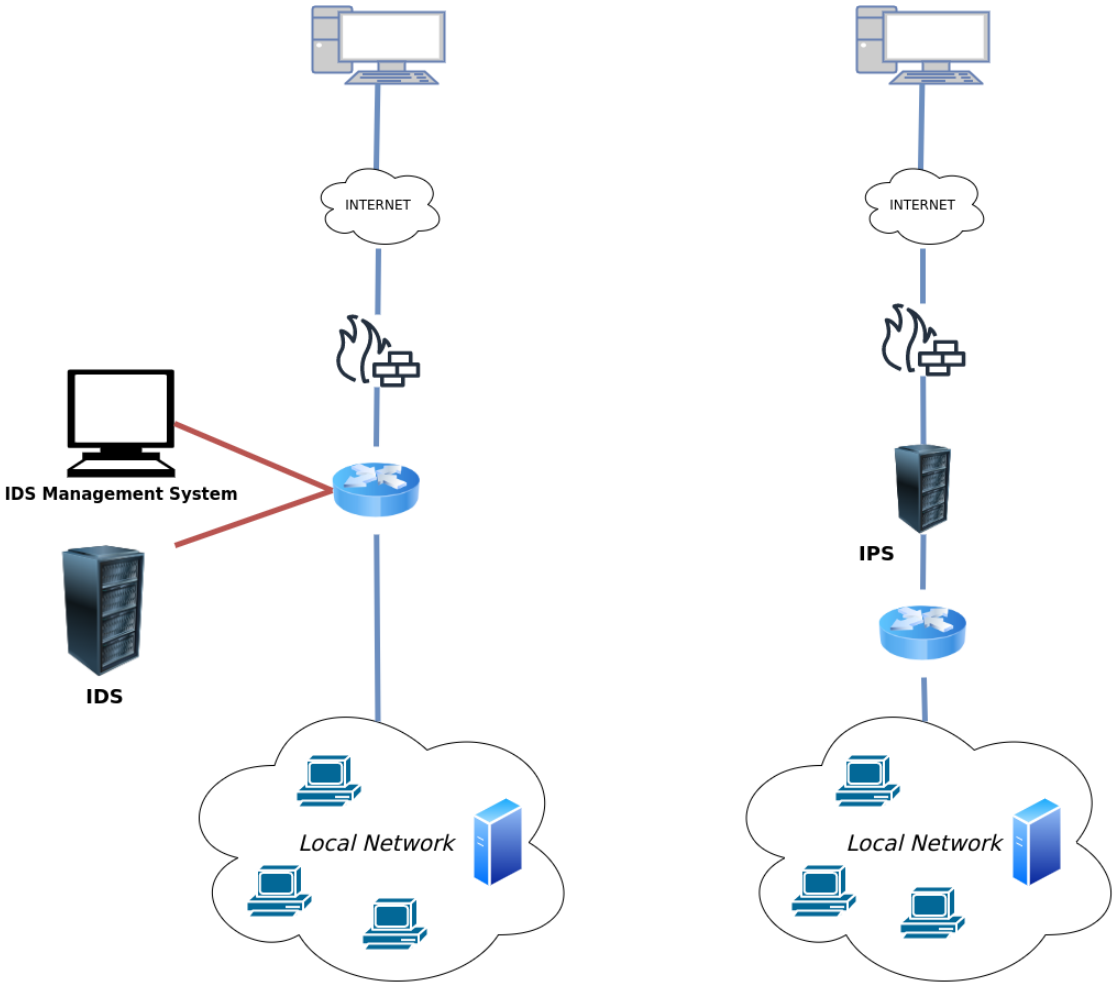

Figura 4.1: Confronto architettura di rete IDS (sinistra) e IPS (destra)

attività svolte, se queste si scostano di molto dalle attività svolte di solito, viene segnalato l'host che ha svolto l'azione ritenuta pericolosa.

4.1.2 Suricata

Suricata è un IDS sviluppato dalla Open Security Foundation, scritto in C, che supporta il real time IDS, la Inline Intrusion Prevention, il Network Security Monitoring, e il processing di file pcap. Suricata ispeziona il traffico di rete utilizzando un completo sistema di regole basato sui pattern ed ha un potente motore per la rilevazione di minacce, scritto in Lua.

Inoltre è perfettamente integrabile con sistemi come Elasticsearch e Kibana, avendo come formati standard YAML e JSON.

4.1.3 Snort

Snort è un IDPS Open Source, sempre scritto in C, ora mantenuto da CISCO. È in grado di eseguire analisi real-time del traffico Internet e Packet Logging dei pacchetti IP. Sia Suricata che snort sono Network base IDS, esistono anche Host-Based IDS. Gli Host-Based IDS, monitorano constantemente le attività svolte nei terminali di una rete, cioè negli host utilizzati dagli utenti ed in particolare si comportano da firewall o antivirus. Un esempio di HIDS è OSSEC.

4.1.4 Tcpdump

Tcpdump è un eseguibile da bash che produce delle "catture" del traffico di rete. Questo programma permette di memorizzare ogni pacchetto che passa dall'interfaccia di rete presa in considerazione e di analizzare il contenuto di esso. E' possibile inoltre specificare delle regole in base a quali pacchetti sono interessato ad osservare. Per esempio sono interessato a sniffare le credenziali che circolano in chiaro all'interno di una rete, posso specificare a Tcpdump che sono interessato ai soli pacchetti HTTP. A questo punto Tcpdump salverà solo i pacchetti interessati e lascerà passare gli altri. Inoltre sono presenti alcuni programmi, il più noto è WireShark, che permettono di analizzare attraverso una GUI i pacchetti che vengono salvati, oltre che permettere delle funzionalità di ordinamento e filtraggio più elaborate rispetto al solo Tcpdump. WireShark permette inoltre di operare in tempo reale sui pacchetti in transito e anche eventualmente di modificarli se necessario, tramite editcap.

4.1.5 Guacd

Guacd è il proxy server-side utilizzato da Apache e in questo contesto viene utilizzato per fornire il Translation Layer per i protocolli RDP, VNC, e SSH permettendo il controllo remoto dell'interfaccia web di Cuckoo Sandbox.

4.2 Strumenti di classificazione

4.2.1 Yara

Yara è uno strumento utilizzato per classificare ed identificare campioni malware. Permette di creare delle descrizioni di famiglie di malware, basate su testo o pattern binari e ognuna di queste descrizioni (o regole) consiste in un insieme di stringe ed espressioni booleane.

4.2.2 Mitre Attack

E' un database di tattica di attacco, realizzato mediante osservazione di scenari di attacco reali. Questo database permette di costruire tecniche difensive adeguate ed aiuta nella classificazione dei malware. Ogni tattica di attacco, per poter essere correttamente attuata da un attacante, viene suddivisa in alcune tecniche, per esempio se l'attaccante vuole sfruttare il movimento laterale come tattica, dovrà prima usare delle tecniche come l'exploitation dei servizi remoti, il Phishing e molti altri.

4.2.3 Virus Total

Virus Total è un sito online che permette di scovare malware all'interno di file o URL. VT aggrega i risultati di più di 70 antivirus e permette di analizzare il risultato ottenuto da ciascun antivirus riguardo ad ogni singolo oggetto controllato.

4.3 Strumenti per le basi di dati

4.3.1 MongoDb

MongoDb è un DBMS non relazionale basato su documenti in stile JSON. Le sue API unite ai moltissimi strumenti frontend per l'amministrazione fanno sì che sia uno dei DBMS non relazionali più utilizzati al mondo.

4.3.2 Elasticsearch

Elasticsearch è un sistema distribuito, RestFul per l'analisi e la ricerca di informazioni all'interno di un database. È usato da Cuckoo come sistema per navigare all'interno del database dei risultati delle scansioni. E' basato sul motore di ricerca Apache Lucene.

4.3.3 Kibana

Kibana è una piattaforma open source per la visualizzazione dei dati relativi ad Elasticsearch. Oltre la visualizzazione, Kibana consente di importare set (o flow) di dati.Permette anche di gestire le impostazione per lo stack Elastic, che vedremo tra poco in dettaglio.

4.3.4 Logstash

Logstash è uno strumento che permette di collezionare, processare e gestire eventi e log. La raccolta dei dati può avvenire tramite socket oppure da bus messaggi.

4.3.5 Stack ELK

Questi tre strumenti, Elasticsearch, Kibana e Logstash sono strettamenti collegati e permettono di creare uno strumento per la gestione dei log molto potente.

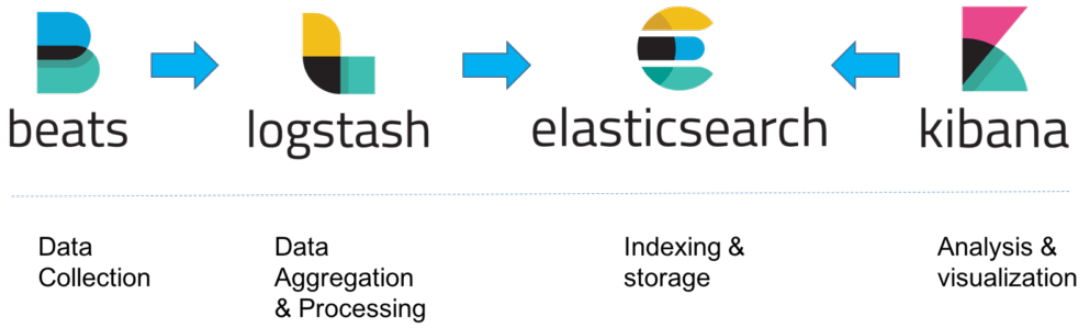

Figura 4.2: Schema processo dei log

4.4 Strumenti per la memoria

4.4.1 Volatility

Volatility è uno strumento open Source scritto in Python per l'analisi della memoria, in questo caso volatile. È uno strumento molto potente, lavora con praticamente qualsiasi operativo, da Windows Xp a 10, Windows Server,distribuzioni Linux con Kernel dalla versione 2.6 alla 5.5, Mac OsX dalla versione 10.5.x alla 10.15.x, permette inoltre di lavorare con file di ibernazione, file dd, crash dump, e snapshot di macchine virtuali (VMware, etc...), file di formato EWF,LiME, Mach-0 e moltri altri ancora.

4.4.2 M2Crypto

M2Crypto è una libreria crittografica Python, permette di utilizzare tutti i protocolli crittografici più utilizzati (OpenSSL, RSA, TLS).

4.4.3 Strings

Strings è un comando Unix che stampa le stringhe di carattere stampabili all'interno di un file. Per ogni file dato, strings ne stampa la sequenza di caratteri stampabili lunga almeno 4 caratteri (o più se viene fornita l'opzione -n).

4.5 Protocolli

4.5.1 Snmp

Snmp è un protocollo connection-less di livello 7 (Applicazione) del modello OSI che consente di gestire e supervisionare gli apparati collegati in una rete. Snmp opera attraverso il protocollo di trasporto UDP e prevede tre componenti principali, Il manager, il master agent ed un insieme di managed object. Ogni nodo della rete (router, stampante, etc..) ha in esecuzione un master agent e solitamente anche uno o più subagent. Il master agent ha il ruolo di mediare tra manager e subagent, mentre il subagent è predisposto alla attuazione delle decisioni del manager. In caso di mancanza di subagent, come nei sistemi con

meccanismi di gestione molto semplici, il master agent e il subagent possono essere inclusi nella stessa soluzione software.

4.5.2 LLDP

LLDP, acronimo di Link Layer Discovery Protocol, è un protocollo di livello Link del modello Osi usato dagli strumenti di rete per distribuire informazioni su di se agli altri dispositivi. E' un protocollo molto utilizzato nella gestione delle reti e nelle reti di monitoraggio. Tra le informazioni che vengono fornite dal dispositivo sia ha:

- System name and description
- Port name and description
- VLAN name
- IP management address
- System capabilities (switching, routing, etc.)
- MAC information
- MDI power
- Link aggregation

Capitolo 5

Virtualizzazione

La virtualizzazione è uno strumento che permette di astrarre le risorse fisiche di un sistema in risorse virtuali. L'insieme di queste componenti fisiche viene chiamato macchina virtuale. La virtualizzazione di sistemi operativi comporta alcune funzionalità molto utili, tra esse troviamo:

- Installazione software semplificata
- Esecuzione di più sistemi operativi simultanea
- Virtualizzazione di un hardware differente da quello presente
- Condivisione di risorse
- Test e ripristino di sistema

5.1 Emulazione di processore

Per emulazione si intende l'esecuzione di programmi pensati e compilati per una determinate archiettura, su di un'architettura diversa, ciò è possibile traducendo ogni singola istruzione, attraverso l'interpretazione oppure la ricompilazione dinamica.

5.1.1 Interpretazione

L'interpretazione delle istruzione è un metodo molto potente ma anche molto dispendioso poichè richiede che molte istruzioni singole siano tradotte in più istruzioni, generando quindi un sovraccarico.

5.1.2 Compilazione dinamica

La compilazione dinamica, invece legge interi blocchi di codice e li traduce per la nuova architettura hardware. In termini di prestazioni la ricompilazione è molto più efficace della interpretazione, considerando che blocchi di codice ripetuti possono essere bufferizzati, evitando così di doverli tradurre di nuovo. Le tecnologie più note che usano questa tecnica sono Qemu e Virtual PC.

5.1.3 Qemu

Qemu è un software in grado di emulare l'hardware del sistema operativo host, traducendo il codice eseguito nella macchina virtuale in codice comprensibile al processore fisico. Qemu è in grado di virtualizzare le architetture di processori più utilizzate come x86, Arm, Sparc e Mips. L'unica limitazione in termini architetturali di Qemu consiste nel dover scegliere come sistema ospitante un sistema con architettura x86, x86_64 o PowerPc.

5.2 Virtualizzazione hardware

Quando si parla di virtualizzazione hardware è necessario fare alcune distinzioni. Innanzitutto si parla di virtualizzazione bare-metal (tipo 1) quando il sistema operativo è assente e al suo posto è presente solamente l'hypervisor che ne fa le veci(XenServer, Esxi). E ancora possiamo distinguere tra hypervisor monolitici, che svolgono cioè tutte le funzioni tipiche dell'hypervisor e microkernel, dove alcune funzioni sono delegate ad una macchina virtuale apposita, chiamata Service VM. Se invece il sistema operativo è presente parliamo di virtualizzazione hosted(tipo 2), dove l'hypervisor è un normale processo (Virtualbox, VMware workstation).

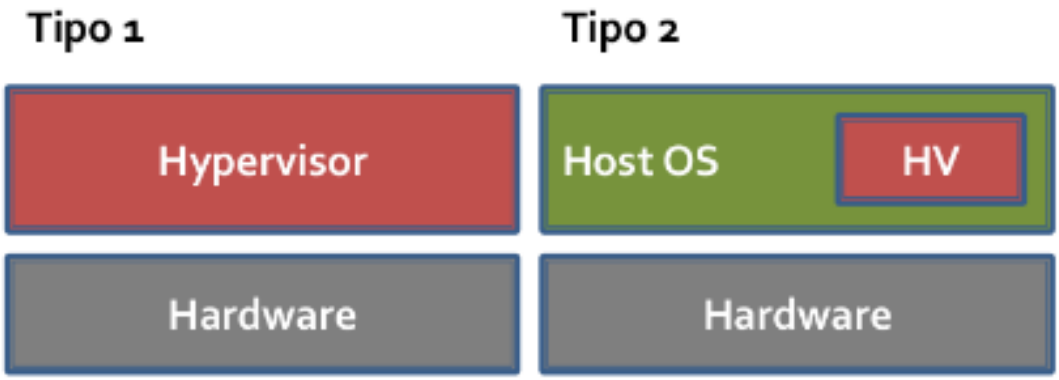

Figura 5.1: Differenze tra virtualizzazione tipo 1 e tipo 2

5.2.1 Full virtualization vs Para Virtualization

Inoltre dobbiamo distinguere tra full-virtualization e para-virtualization. Nella para-virtualization la macchina virtuale presenta una interfaccia diversa da quella della macchina fisica. Questo comporta la modifica del sistema operativo guest per poterlo adattare alla macchina virtuale e per consentirne quindi l'esecuzione. L'hypervisor fornisce però delle API che consentono la comunicazione con il sistema operativo guest e vengono chiamate Hypercall. Nella full-virtualization invece non è necessario nessuna modifica del sistema operativo e le chiamate privilegiate vengono gestite tramite oppurtuni metodi, come la binary translation o la virtualizzazione hardware-assisted.

5.2.2 Rischi della virtualizzazione hardware

Nonostante fosse ritenuto impossibile, negli ultimi anni sono state scoperte vulnerabilità che permettono ad un attaccante di eseguire codice arbitrario e di interagire con il sistema operativo ospitante, avendo accesso diretto solo alla macchina virtuale. Questo tipo di vulnerabilità prende il nome di Virtual Machine Escape. Un esempio di queste vulnerabilità è la CVE-2017-4934, che descrive come in alcune versione specifiche di VMware Fusion e Workspace sia possibile un buffer overflow di una heap nel codice della vmnat, l'interfaccia di rete utilizzata. Queste tipologie di vulnerabilità sono molto serie poiché nella maggior parte dei casi i software per la virtualizzazione, come VirtualBox e VmWare, vengono

eseguite con I privilegi di root. Un'altra vulnerabilità degna di nota è Venom (Virtualized Environment Neglected Operations Manipulation), un bug nel Floppy Driver Controller (FDC) in Qemu. Il codice del Fdc vulnerabilità è presente anche in altre piattaforme di virtualizzazione come Xen, KVM e VirtualBox. Appurata questa categoria di rischi, è bene notare quali siano invece i lati positivi della virtualizzazione, i quali hanno portato la virtualizzazione ad essere il pilastro delle più recenti tecnologie (Container, Kubernetes, Cloud, etc..).

5.2.3 Installazione software più semplice

Quando l'installazione di un determinato software diventa molto complessa e prevede l'installazione di numerose librerie, è possibile, se previsto dal venditore, prendere un'immagine di un sistema operativo in cui quel software è già installato e configurato e usarla come macchina virtuale, in modo da avere quel software pronto all'uso.

5.2.4 Test e ripristino

Uno dei punti forti delle macchine virtuali è la possibilità di considerarle come una scatola chiusa e gestirle senza interessarsi di cosa c'è dentro. E' possibile fermarne l'esecuzione e riprenderla senza che le applicazioni che sono attualmente in uso se ne accorgano. E' possibile anche realizzare dei backup, chiamati snapshot, che non solo salvano tutti i dati in un determinato momento, ma congelano lo stato di una macchina nel instante in cui effetto lo snapshot. Se qualcosa va storto durante l'uso della macchina virtuale, è sempre possibile ripristinarla attraverso l'uso di snapshot, senza dover eseguire ogni volta backup o ripristini di sistema.

5.2.5 Condivisione di risorse

La condivisione di risorse viene sfruttata dai fornitori di servizi come i data center per risparmiare i costi per l'hardware ed i consumi. E' noto infatti che i computer usano per la maggior parte del tempo solo una parte delle proprie risorse, quindi invece di eseguire su ogni hardware un solo sistema operativo, si è pensato di virtualizzare su ogni hardware

diversi sistemi operativi, cosicché sia possibile assegnare ad ogni sistema operativo solo le risorse di cui esso necessità e in caso bilanciarle.

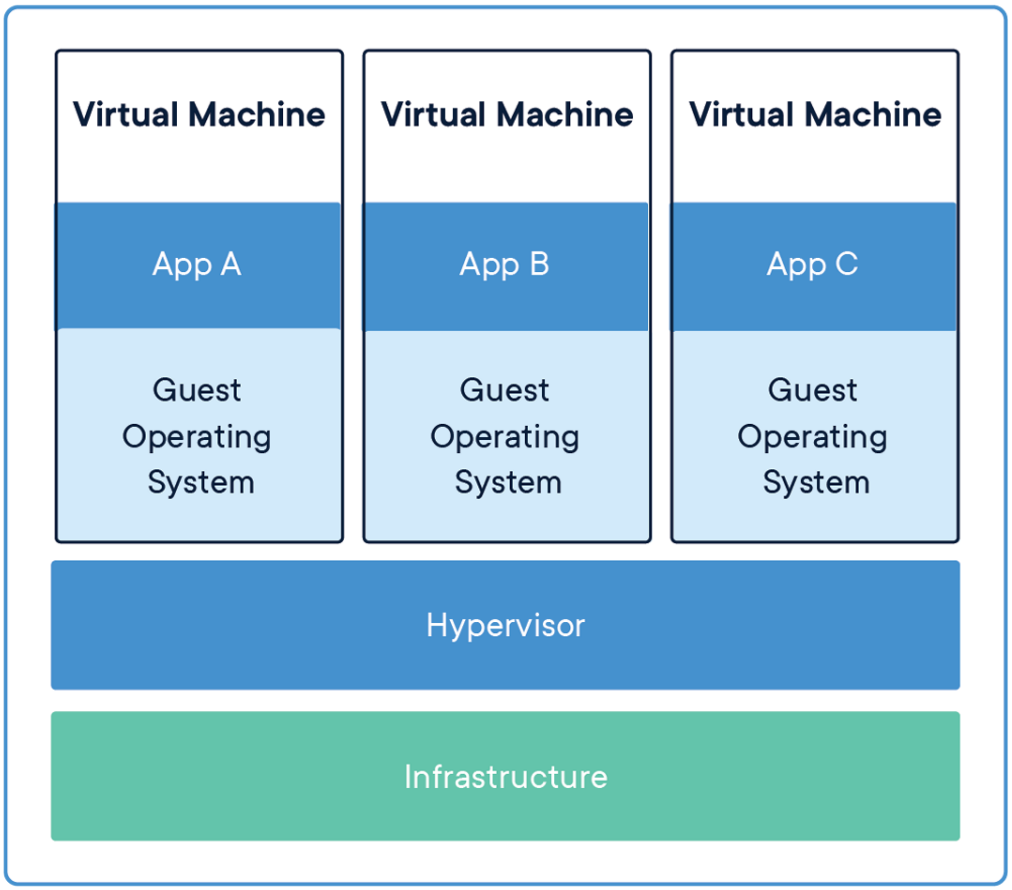

Figura 5.2: Virtualizzazione livello hardware

5.3 Virtualizzazione a livello Os (Containers)

Quando si parla di containers si parla di virtualizzazione a livello di sistema operativo, viene infatti presentata all'utente una partizione del sistema operativo sul quale eseguire le applicazioni, che rimarranno isolate in quella partizione. Ciò è possibile utilizzando un unico kernel, quello del sistema operativo host e multiple istanze di partizioni, o user space, gestibili indipendentemente l'una dall'altra. Ognuna di queste partizioni viene chiamata container. Ad ogni container viene garantito l'isolamento di rete e di storage, inoltre

viene fornito un meccanismo di gestione per la Cpu, la memoria e la rete. Questo tipo di virtualizzazione porta alcuni vantaggi rispetto a quella di livello hardware, per esempio un basso overhead per il context-switch e un basso overhead di memoria. Oltre che per isolare diverse partizioni, i container vengono usati per il deployment di applicazioni. Avendo uno spazio utente isolato, ognuno con librerie e files, è possibile infatti testare un applicazione senza dover configurare una macchina virtuale, cosi da rendere estramente veloce il processo di pacchettizzazione del software. I principali sofware di virtualizzazione sono Docker, Hyper-V (Microsoft) e Container di Windows Server.

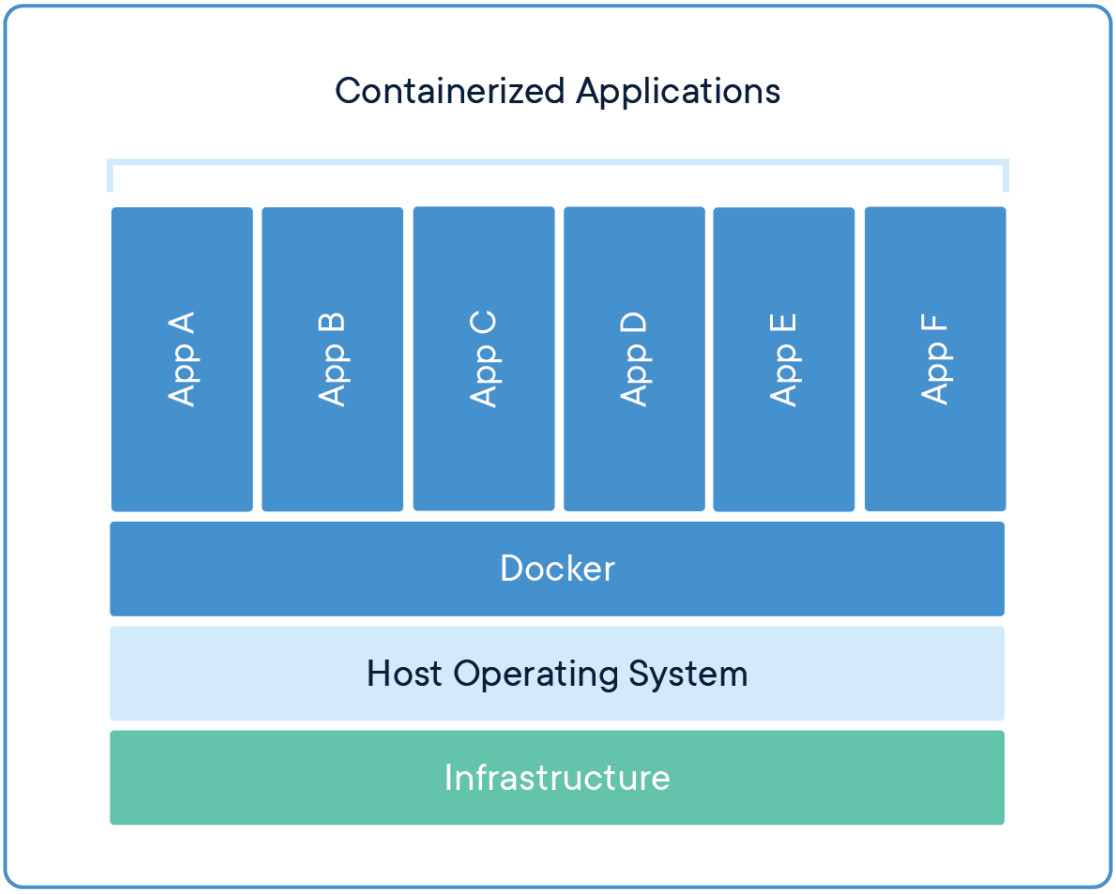

Figura 5.3: Virtualizzazione livello OS

5.4 Docker

Docker è una piattaforma per la containerizzazione creata nel 2013 per permettere ai programmatori eseguire e distribuire applicazioni attraverso appunto l'uso di container. Docker basa il suo funzionamento sulle immagini, che non sono altro che le istruzione che Docker usa per costruire uno specifico container. Nello specifico la architettura client/server prevede che il daemon si occupi di compilare, eseguire e distribuire i container, mentre il client si interfaccia con l'utente. Client e server comunicano attraverso una REST API, usando le sockets Unix o l'interfaccia di rete. Dockerd, che è il daemon, resta in ascolto di richieste dal client per gestire gli oggetti Docker(immagini,container,reti, volumi), oppure può comunicare con altri daemon per gestire i servizi Docker. Per esempio questo codice permette di avviare un container che stampa il classico "Hello, world!"

```
docker run hello-world
# 1 - Il docker client contatta il daemon
# 2 - Il daemon scarica (pull) l'immagine da Docker-Hub
# 3 - Il daemon crea un nuovo container
# 4 - Il daemon ridirige l'output nel terminale che esegue Docker
```

5.4.1 Immagini Docker

Un immagine Docker è una sequenza di istruzione che permette a Docker di creare un container. Solitamente si parte da una distribuzione conosciuta, come Ubuntu, Debian o Alpine e si aggiungono tutte le librerie necessarie al funzionamento dell'applicazione. Le istruzioni sono contenute in un file chiamato Dockerfile, per esempio analizziamo questo esempio:

```
FROM node:current-slim

WORKDIR /usr/src/app

COPY package.json .
```

```
RUN npm install

EXPOSE 8080

CMD [ "npm", "start" ]

COPY . .
```

Innanzitutto, con FROM definiamo qual'è l'immagine di partenza, in questo caso vogliamo che il nostro sistema abbia solo l'interpete Node.Js e le sue dipendenze. Con il comando WORKDIR, stabiliamo il percorso base per tutte le operazioni da qui in avanti. Il comando COPY invece specifica di copiare il file package.json dall'host alla immagine.Il comando RUN invece serve per eseguire istruzioni nella nostra immagine. In questo caso eseguiamo npm[1] install, che installerà il nostro pacchetto package.json. Expose notifica che il container ascolerà sulla porta 8080. CMD invece specifica come dovrà rispondere il container una volta eseguito. Per eseguire il container però, è necessario compilare prima il Dockerfile, per fare ciò basta digitare:

```
docker build --tag nodetest:1.0 .
```

Oltre alla opzione tag che permette di specificare un nome al container, è presente un punto, che è di fondamentale importanza in quanto specifica a Docker di prendere come Dockerfile di partenza quello presente in questa directory. A questo punto è possibile eseguire il nostro container, digitando questa riga:

```
docker run --publish 8000:8080 --detach --name node nodetest:1.0
```

In questa riga specifico di eseguire il container nodetest, alla versione 1.0 (specificata in fase di build), in background (detach) e redirigendo il traffico dalla porta 8000 alla 8080.

[1]Node Package Manager, gestore di pacchetti di default per ambienti Node

5.4.2 Docker Compose

Docker Compose è uno strumento che permette l'esecuzione di applicazione che necessitano di più container. Per utilizzare Docker compose è sufficiente descrivere in un file YAML[2] le istruzione per ogni container, chiamato servizio e alcune opzioni per configurare l'ambiente di esecuzione, come i volumi e le porte da esporre. Un esempio di file compose è il seguente:

```
version: '2.0'
services:
  web:
    build: .
    ports:
    - "5000:5000"
    volumes:
    - .:/code
    - logvolume01:/var/log
    links:
    - redis
  redis:
    image: redis
volumes:
  logvolume01: {}
```

In questo esempio abbiamo due container, web e redis, il primo non dispone di un immagine già presente ma chiede di compilare il Dockerfile, specifcando l'opzione build con il punto.Inoltre espone la porta 5000 e specifica che necessita di Redis per funzionare. Redis invece ha già un'immagine presente e quindi non deve essere compilato.

[2]YAML, acronimo ricorsivo di YAML Ain't a Markup Language, è un formato per la serializzazione dei dati, fortemente ispirato al linguaggio XML

5.4.3 Docker Swarn

Docker Swarn è una tecnologia che permette di gestire cluster di nodi Docker. Questo cluster, chiamato Swarn è completamente gestibile attraverso la CLI[3] di Docker. I cluster consistono in diversi Docker host, che vengono eseguiti in modalità swarn e possono essere manager e/o worker.Docker swarn ha l'obiettivo, una volta stabiliito uno stato ottimale(in termini di risorse disponibili), di lavorare per mantenerlo. Se per esempio un nodo worker non è più disponibile, i task di quel nodo passano ad un altro nodo worker. Per task si intende un container in esecuzione che fa parte di uno swarn. Le caratteristiche principali di Swarn sono:

- Scalabilità
- Bilanciamento di carico
- Design decentralizzato
- Alto livello di sicurezza
- Instradamento multi-host

[3]CLI, acronimo di command line interface, è l'interfaccia a riga di comando, o console, sul quale vengono impartiti i comandi da eseguire ad una determinata applicazione o al sistema operativo

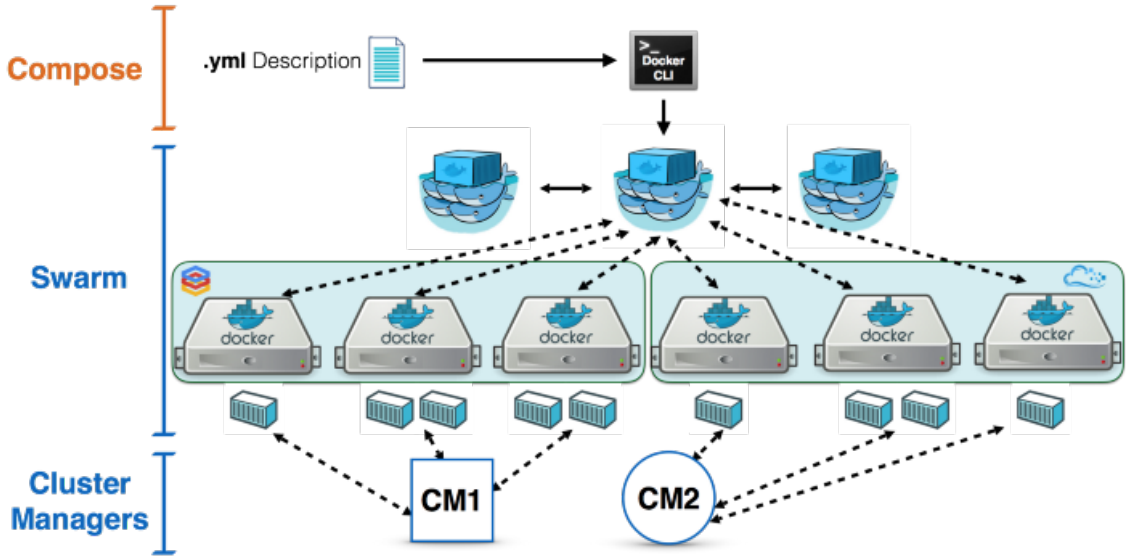

Figura 5.4: Schema architettura Docker Swarn

Capitolo 6

Realizzazione

Il primo passo per l'installazione di questo sistema è stato lo studio della documentazione di Cuckoo e la predisposizione di un sistema "pulito" sul quale installarlo. Come immagine di partenza ho scelta la distribuzione Gnu/Linux Debian 10/Buster. Come macchina d'analisi ho invece scelto di utilizzare Windows 7, poiché è la piattaforma per il quale sono stati pensati e realizzati più malware, oltre che essere più semplice da configurare rispetto a Windows 10 oppure Windows Xp. Come metodo di installazione di Cuckoo, è possibile scegliere tra la compilazione del codice sorgente oppure l'installazione tramite PIP.

6.1 Configurazione Guest

Per la configurazione il primo passo è stato, dopo aver scelto la piattaforma di virtualizzazione, quello di creare una macchina virtuale con Windows 7 come sistema operativo. Ho scelto di utilizzare la virtualizzazione hosted, con VirtualBox come HyperVisor perché mi permetteva di configurare la macchina d'analisi in maniera più semplice. Il passo successivo è stato rimuovere tutte le protezioni installate su Windows 7 di default, come l' UAC (User Account Control) e Windows Firewall. Ho installato poi OpenOffice e Acrobat Reader per aprire i file PDF. In seguito, ho installato Python, nella versione 2.7 e la libreria Pillow. Dopo aver configurato il sistema operativo sono passato alla configurazione dell'interfaccia di rete e ho impostato un indirizzo IP e DNS statico e disabilitato il DHCP, in questo modo ad ogni avvio la macchina virtuale avrebbe avuto lo stesso indirizzo IP, facilitandone la

configurazione a lato host. A questo punto la macchina virtuale era pronta e ho utilizzato i seguenti comandi iptables per far comunicare host e macchina virtuale. Dopo aver uti-

```
$ sudo iptables -t nat -A POSTROUTING -o eth0 -s 192.168.56.0/24 -j MASQUERADE

# Default drop.
$ sudo iptables -P FORWARD DROP

# Existing connections.
$ sudo iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT

# Accept connections from vboxnet to the whole internet.
$ sudo iptables -A FORWARD -s 192.168.56.0/24 -j ACCEPT

# Internal traffic.
$ sudo iptables -A FORWARD -s 192.168.56.0/24 -d 192.168.56.0/24 -j ACCEPT

# Log stuff that reaches this point (could be noisy).
$ sudo iptables -A FORWARD -j LOG
```

Figura 6.1: Comandi iptables

lizzato nella macchina host questi comandi, ho clonato la macchina virtuale con lo script agent.py in esecuzione e ho concluso così la configurazione della parte guest. Per clonare la macchina virtuale, ho utilizzato il comando:

```
"VBoxManage snapshot "<Name of VM>" take "<Name of snapshot>" --pause"
```

Un'altra possibile configurazione di Cuckoo, sarebbe stata quella di utilizzare una macchina fisica come macchina d'analisi. In questo caso le impostazioni lato host avrebbero coinvolto l'uso di Fog per la gestione delle immagini. Nell'ambito di questo progetto non andremo a studiare questa configurazione.

6.2 Configurazione Host

La configurazione dell'host ha previsto l'installazione di alcune librerie Python, di MongoDb e se necessario di PostgreSql. Per permettere il Dump delle attività di rete era necessario uno sniffer di rete, cioè uno strumento che rimane in ascolto di tutti i pacchetti che passano per l'interfaccia di rete. Per comodità è stato scelto Tcpdump, in quanto è il network

sniffer, più utilizzato e documentato, oltre che presente di default in strumenti per l'analisi di rete come Wireshark. Per la sua configurazione Cuckoo prevede nella sua directory di configurazione alcuni file, ognuno contenente le impostazione riguardanti un aspetto del funzionamento di Cuckoo. Cuckoo.conf In questo file, è possibile indicare quale software di virtualizzazione è stato scelto e le opzioni per interagire con la macchina d'analisi creata. Inoltre è possibile indicare le credenziali e la configurazione del database interno. Auxiliary.conf I moduli Auxiliary sono script che vengono eseguiti insieme all'analisi come ad esempio l'utilizzo di uno sniffer o del proxying MITM (Man In The Middle). Memory.conf Specifica il profilo del sistema operativo installato per permettere un Dump della memoria più veloce e permette anche di nascondere alcuni processi dal Dump. Inoltre contiene le impostazione per la configurazione di Volatility e di altri Plug-in eventualmente installati. Processing.conf Permette di abilitare o disabilitare alcuni moduli, tra cui Suricata, Virustotal, Snort, String e altri. Virtualbox/vmware.conf Contiene impostazioni specifiche sulla macchina virtuale come il nome dello snapshot e l'indirizzo della macchina virtuale.

6.3 Utilizzo

Per utilizzare Cuckoo, basta digitare da bash il comando Cuckoo. Dopo aver cercato eventuali aggiornamenti, Cuckoo si metterà in attesa di file da analizzare. Digitando "Cuckoo –help" è possibile ricevere una sorta di breve documentazione contenente i comandi principali. Prima di iniziare con le prime analisi è consigliato però utilizzare il comando Cuckoo community, che permette di scaricare tutte le signatures presenti nel server di Cuckoo. A questo punto possiamo iniziare a caricare alcuni file tramite il comando submit. Per permettere a Cuckoo di eseguire l'analisi è necessario far partire il vero motore di Cuckoo, cioè il suo daemon, attraverso il comando cuckoo -d. Se invece volessimo utilizzare un'interfaccia web per caricare i file, far partire le analisi e visualizzarne i risultati, possiamo utilizzare il comando cuckoo web.

```
$ cuckoo --help
Usage: cuckoo [OPTIONS] COMMAND [ARGS]...

Invokes the Cuckoo daemon or one of its subcommands.

To be able to use different Cuckoo configurations on the same
machine with the same Cuckoo installation, we use the so-called
Cuckoo Working Directory (aka "CWD"). A default CWD is
available, but may be overridden through the following options -
listed in order of precedence.

* Command-line option (--cwd)
* Environment option ("CUCKOO")
* Environment option ("CUCKOO_CWD")
* Current directory (if the ".cwd" file exists)
* Default value ("~/.cuckoo")

Options:
  -d, --debug             Enable verbose logging
  -q, --quiet             Only log warnings and critical messages
  -m, --maxcount INTEGER  Maximum number of analyses to process
  --user TEXT             Drop privileges to this user
  --cwd TEXT              Cuckoo Working Directory
  --help                  Show this message and exit.

Commands:
  api          Operate the Cuckoo REST API.
  clean        Clean the CWD and associated databases.
  community    Fetch supplies from the Cuckoo Community.
  distributed  Distributed Cuckoo helper utilities.
  dnsserve     Custom DNS server.
  import       Imports an older Cuckoo setup into a new CWD.
  init         Initializes Cuckoo and its configuration.
  machine      Dynamically add/remove machines.
  migrate      Perform database migrations.
  process      Process raw task data into reports.
  rooter       Instantiates the Cuckoo Rooter.
  submit       Submit one or more files or URLs to Cuckoo.
  web          Operate the Cuckoo Web Interface.
```

Figura 6.2: Output comando Cuckoo -help

6.4 Rodaggio

Dopo aver confrontato i risultati ottenuti sottoponendo alcuni malware e alcuni file genuini ho effettuate alcune modifiche per regolare lo score nel report. Lo score è uno strumento che Cuckoo ha introdotto nel report per dare una valutazione della pericolosità di un file. Le prime analisi effettuate infatti erano viziate da alcuni processi che durante lo snapshot della macchina virtuale erano in esecuzione, mentre cuckoo li riteneva aperti dal file che veniva controllato, per cui venivano ritenuti processi malevoli. Ho quindi proceduto a segnarmi i PID dei processi e a scriverli in una apposita whitelist, cosicché che non intralciassero la valutazione dello score.

6.5 Dockerizzazione Cuckoo

Dopo aver valutato molti campioni di malware, ho deciso di creare una versione dockerizzata di Cuckoo Sandbox. In seguito ad aver usato per molto tempo questo programma ho infatti capito le sue potenzialità, ma sono rimasto molto deluso dalle difficoltà che ho riscontrato nell'installazione e nella configurazione. La sfida della containerizzazione infatti mi avrebbe aiutato a installarlo in altri sistemi, sia Linux che Windows, in molto meno tempo, rendendendo più facile anche la configurazione di elemententi in container separati come MongoDb e Yara. La creazione di una versione containerizzata di Cuckoo è stata abbastanza semplice, ho dapprima recuperato tutte le informazioni riguardanti i pacchetti che erano necessari e sono partito creando un Dockerfile apposito. Come immagine di partenza ho scelto alpine perché volevo partire da un sistema su cui potevo contare in termini di velocità e leggerezza, anche se era possibile ricreare la stessa immagine partendo da Debian, in modo che il container risultante fosse più stabile. Confrontandomi anche con altre versioni già presenti sul web, ho deciso di utilizzare container già presenti su docker hub di MongoDb, Yara e Volatilty. Il problema più grande nell'utilizzo di questa versione di Cuckoo è stato il collegamento con il software di virtualizzazione. Infatti, per utilizzare la versione dockerizzata di Cuckoo con virtualbox avrei dovuto avere la possibilità di comunicare con l'hypervisor, da dentro l'hypervisor. Questo avrebbe portato ad una Virtual Machine Escape, una vulnerabilità molto grave, come già visto in precedenza nella sezione

5.2.2. La soluzione più semplice avrebbe previsto l'installazione di VirtualBox all'interno del container, ma questo avrebbe portato ad una crescita a dismisura della dimensione dell'immagine, oltre ad aver reso impossibile la portabilità, perché avrebbe anche richiesto di montare il device vbox nel container in esecuzione. La soluzione adottata è stata quella di utilizzare il servizio web di VirtualBox e comunicare con Docker attraverso la SOAP Api. In particolare, è stato usato il wrapper Python di questa Api, cioè remotevbox. Questo è il Dockerfile dell'immagine di Cuckoo su Docker di partenza:

```
FROM blacktop/volatility:2.6

ENV CUCKOO_VERSION 2.0.5.3
ENV CUCKOO_CWD /cuckoo

COPY requirements.txt /tmp/requirements.txt
RUN apk add --no-cache tcpdump py-lxml py-chardet py-libvirt py-crypto
RUN apk add --no-cache postgresql-dev gcc g++ python-dev libpq
RUN pip install --upgrade pip wheel
RUN apk del --purge postgresql-dev gcc g++

RUN apk add --no-cache -t .build-deps linux-headers openssl-dev
RUN apk add --no-cache libxml2-dev python-dev  libffi-dev build-base
RUN echo "===> Install Cuckoo Sandbox..." && mkdir /cuckoo
RUN adduser -D -h /cuckoo cuckoo \
RUN export PIP_DISABLE_PIP_VERSION_CHECK=on \
RUN pip install --upgrade pip wheel setuptools
RUN LDFLAGS=-L/lib pip install cuckoo==$CUCKOO_VERSION
RUN cuckoo && cuckoo community
RUN echo "===> Install additional dependencies..."
RUN pip install -r /tmp/requirements.txt \

COPY conf /cuckoo/conf
```

```
COPY update_conf.py /update_conf.py
COPY docker-entrypoint.sh /entrypoint.sh

WORKDIR /cuckoo

VOLUME ["/cuckoo/conf"]

EXPOSE 1337 31337

ENTRYPOINT ["/entrypoint.sh"]
CMD ["--help"]
```

Il file invece docker-compose.yaml per creare una versione con compose, è il seguente:

```
    version: "2"

services:
  cuckoo:
    image: cuckoo:2.0
    command: daemon
    ports:
      - "2042:2042"
    volumes:
      - ./cuckoo-tmp/:/tmp/cuckoo-tmp/
      - ./storage/:/cuckoo/storage/
    networks:
      - cuckoo

  web:
    image: cuckoo:2.0
    ports:
      - "80:31337"
```

```
  links:
    - mongo
    - elasticsearch
    - postgres
  command: web
  volumes:
    - ./cuckoo-tmp/:/tmp/cuckoo-tmp/
    - ./storage/:/cuckoo/storage/
  networks:
    - cuckoo

api:
  depends_on:
    - postgres
  image: blacktop/cuckoo:2.0
  ports:
    - "8000:1337"
  links:
    - postgres
  command: api
  volumes:
    - ./cuckoo-tmp/:/tmp/cuckoo-tmp/
    - ./storage/:/cuckoo/storage/
  networks:
    - cuckoo

mongo:
  image: mongo
  ports:
    - 27017
  volumes:
```

```
      - mongo-data:/data/db
    networks:
      - cuckoo

  elasticsearch:
    image: elasticsearch:5.6
    ports:
      - 9200
    volumes:
      - es-data:/usr/share/elasticsearch/data
    networks:
      - cuckoo

  postgres:
    image: postgres
    ports:
      - 5432
    environment:
      POSTGRES_USER:
      POSTGRES_PASSWORD:
    volumes:
      - postgres-data:/var/lib/postgresql/data/pgdata
    networks:
      - cuckoo

networks:
  cuckoo:
    driver: bridge
```

In questo file vengono specificate alcune immagini, cuckoo, web, api, postegre ed elasticsearch, che devono già essere presenti come container. Cuckoo, web e api si riferiscono alla

stessa immagine, cioè cuckoo, ma con un comando cmd diverso. Cuckoo serve per eseguire il daemon, web per l'interfaccia web, api per la comunicazione REST.

6.6 Cuckoo come sistema distribuito

Senza dubbio, utilizzare un singolo nodo per la raccolta di malware e l'analisi di Cuckoo è possibile solo in reti di dimensione molto modeste ed è per questo che è possibile distribuire Cuckoo in più nodi utilizzando la REST Api. In questa configurazione, una sola macchina esegue Cuckoo in modalità distribuita (master), mentre diversi nodi eseguono il daemon Cuckoo (worker). Per eseguire Cuckoo distribuito, basta lanciare il seguente comando:

```
cuckoo distributed server --help
Usage: cuckoo distributed server [OPTIONS]


Options:
  -H, --host TEXT     Host to bind the Distributed Cuckoo server on
  -p, --port INTEGER  Port to bind the Distributed Cuckoo server on
  --uwsgi             Dump uWSGI configuration
  --nginx             Dump nginx configuration
  --help              Show this message and exit.
```

A differenza della normale configurazione di Cuckoo, in questa è obbligatorio specificare il formato di output dei report, poichè una volta recuperati dal nodo master essi vengono scartati dal nodo worker. I formati principali dei report sono Json o html. I comandi principali dell'API REST[1]sono i seguenti:

```
GET /api/node  Ottieni tutti i nodi Cuckoo.
POST /api/node  Registra un nuovo nodo Cuckoo.
GET /api/node/<name>  Ottieni informazioni su un nodo.
PUT /api/node/<name>  Registra informazioni su un nodo già esistente.
```

[1]REST o Representational State Transfer, è un'architettura per sistemi distribuiti e rappresenta un sistema di trasmissione dati HTTP stateless

```
POST /api/node/<name>/refresh  Aggiorna informazioni su un nodo.
DELETE /api/node/<name>  Disabilita un nodo.
GET /api/task  Ottieni lista dei task nel database.
POST /api/task  Crea un nuovo task.
GET /api/task/<id>  Ottieni informazioni su un task.
DELETE /api/task/<id>  Elimina informazioni su un task.
GET /api/report/<id>/<format>  Ottieni report analisi.
GET /api/pcap/<id>  Ottieni informazioni sui dati pcap di una analisi.
```

Per esempio, il comando per ottenere le informazione su i nodi sarà:

```
curl http://localhost:9003/api/node
```

E l'output sarà:

```
{
    "success": true,
    "nodes": {
        "localhost": {
            "machines": [
                {
                    "name": "cuckoo1",
                    "platform": "windows",
                    "tags": []
                }
            ],
            "name": "localhost",
            "url": "http://localhost:8090/"
        }
    }
}
```

Capitolo 7

Sperimentazione

Per testare in modo completo tutte le funzionalità non rimaneva altro che analizzare dei malware veri e propri. Ho scaricato quindi da un repository online alcuni malware, diversi per tipologia, per verificare che Cuckoo innanzitutto che li rilevasse senza problemi e poi per analizzarli all'opera.

7.1 Cerber

Il primo malware che ho testato è stato Cerber, un malware di tipo Ramsonware che utilizza la cifratura AES per criptare il disco. Dopo aver cifrato tutti i dati personali, questo malware rilascia un file readme.txt, oltre che cambiare lo sfondo del pc, in quale elenca passaggi per ottenere la chiave di decriptazione. Per ottenere questa chiave è necessario pagare una cifra abbastanza alta, circa 500$, attraverso il pagamento in cryptovaluta, così da non essere rintracciabili. Spesso queste tipologie di malware vengono diffuse attraverso mail di spam oppure attraverso adware. La prima verifica fatta è stata analizzare il malware su VirusTotal, nel quale ha totalizzato 61 punti su 68, che corrisponde ad essere rilevati e contrassegnati come malware da 61 antivirus su 68. VirusTotal non esegue il malware in ambiente sandbox, ma piuttosto confronta le signatures dei diversi file caricati e li confronta con i database dei diversi antivirus. Nella modalità dinamica invece, VirusTotal utilizza proprio il software Cuckoo Sandbox per analizzare il file caricato. Dopo di VirusTotal, ho analizzato Cerber su hybrid-analysis.com, dove il livello di minaccia era al 100%. Dopo

queste analisi preventive ho caricato Cerber su Cuckoo Sanbox. In meno di due minuti ho potuto osservare come il Ramsonware si fosse impossesato della mia macchina virtuale. Nessun file era leggibile e nello sfondo comparivano le istruzioni per il riscatto.

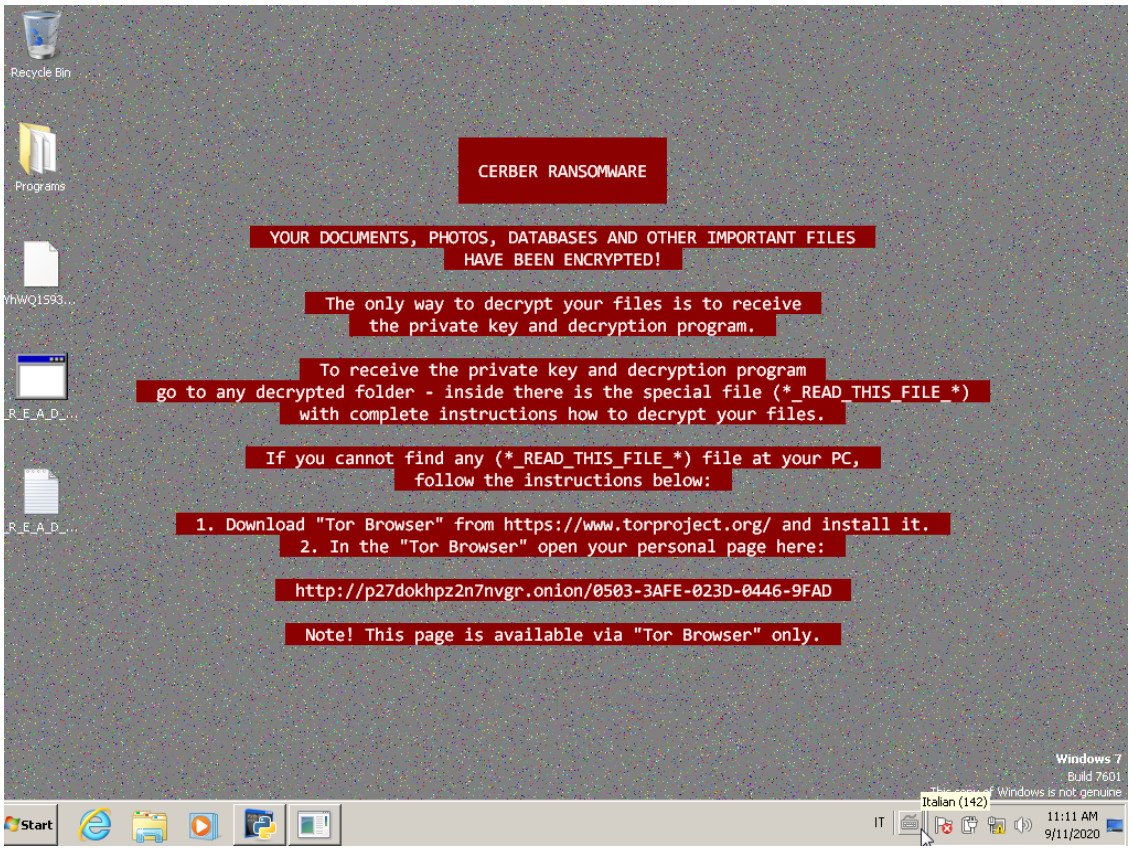

Figura 7.1: Esito esecuzione ramsonware su macchina virtuale

7.2 AgentTesla

Il secondo malware che ho testato è stato AgentTesla, un Trojan Horse che usa come veicolo una versione contraffatta di Sky Go, piattaforma per lo streaming di serie tv, eventi sportivi e altro appartenente a Sky. Questa versione prometteva di poter vedere tutti gli streaming senza dover pagare un abbonamento, cercando di convincere così le potenziali vittime a scaricare l'applicazione . Questo malware non si impossessava come Cerber dell'intero sistema ma cercava piuttosto di rimanere quanto più nascosto possibile,

cercando solo di recuperare informazioni. Alla prima esecuzione è riuscito a riavviare la macchina virtuale, eseguendo alcuni comandi particolari che hanno subito allertato Cuckoo, tra questi, è riuscito a posizionarsi tra i programmi eseguiti durante l'avvio. Tra le altre azioni pericolose, possiamo notare:

- Modifica registri
- Installazione certificati di Root
- Dumping delle credenziali di login
- Dll Injection
- NTFS Hiding (utilizza la Master File Table del file system per nascondersi)

7.3 Altri malware

Ho esaminato molti altri malware, che però avevano meno effetti visibili e quindi più difficili da dimostrare, inoltre, i malware più recenti si "disattivano" automaticamente quando riscontrano nel sistema operativo qualche elemento che gli faccia pensare che sia in esecuzione su macchina virtuale, così da rendere più difficile l'analisi.

Capitolo 8

Conclusioni

Durante questo lavoro ho avuto modo di approfondire gli aspetti cruciali della sicurezza informatica, dallo studio delle reti fino all'analisi dei malware. Ho potuto constatare quanto la sicurezza informatica non sia una mera questione di tecnologie e protocolli ma sia in realtà una disciplina che comprende moltissimi aspetti, anche psicologici, della nostra vita quotidiana. Il sistema Cuckoo Sandbox nel suo complesso si è rilevato molto efficace nella rilevazione dei malware, oltre ad essere estramemente completo e configurabile. I numerosi strumenti presenti, oltre a quelli che si possono aggiungere inoltre, lo potrebbero rendere uno degli strumenti più utilizzati al mondo per questo scopo. Purtroppo però, Cuckoo Sandbox paga il fatto di essere molto difficile da installare in ambienti desktop. Tuttavia la suo versione Dockerizzata, se sviluppata e mantenuta adeguatamente in futuro potrà essere una valida alternativa. In ottica futura il sistema potrà essere utilizzato dal campus di Cesena, in accordo con gli attuali sistemi di sicurezza, per l'analisi in tempo reale dei campioni catturati dall'IDS del campus. Per esempio è possibile creare una o più macchine virtuali nei server che ospitano la rete del campus di Cesena che lavori a stretto contatto con l'IDS, che una volta rilevato il passaggio di qualche pacchetto malevolo, lo invii a Cuckoo per l'elaborazione di un report. Il report una volta analizzato dai tecnici di laboratorio, sarà d'aiuto per intraprendere azioni su chi ha generato quel determinato pacchetto.

Bibliografia

[1] Mauro Del Corno. «Effetto attacchi informatici sulle semestrali in Borsa». In: *Il Sole 24 Ore* (2017). URL: https://www.ilsole24ore.com/art/effetto-attacchi-informatici-semestrali-borsa-AED0VSCC.

[2] Max Fisher. «Syrian hackers claim AP hack that tipped stock market by $136 billion. Is it terrorism?» In: *The Washington Post* (2013). URL: https://www.washingtonpost.com/news/worldviews/wp/2013/04/23/syrian-hackers-claim-ap-hack-that-tipped-stock-market-by-136-billion-is-it-terrorism/.

[3] Valentin Rothberg. «Improving Linux container security with seccomp». In: *Red Hat, Inc.* (2020). URL: https://www.redhat.com/sysadmin/container-security-seccomp.

[4] Graz University of Technology. «Meltdown and Spectre, Vulnerabilities in modern computers leak passwords and sensitive data.» In: *Graz University of Technology* (). URL: https://meltdownattack.com/.

[5] «What is sandboxing? How does cloud sandbox software work?» In: *Avast Inc.* (). URL: https://www.avast.com/business/resources/what-is-sandboxing.

Ringraziamenti

Vorrei ringraziare particolarmente il Prof. Ciro Barbone, che mi ha guidato in questa esperienza, come il Prof. Vittorio Ghini, che mi ha profonodamente ispirato in questi tre anni. Ringrazio la mia famiglia che mi ha sempre supportato senza la quale non avrei mai raggiunto questo obiettivo.
Infine gli amici di una vita, come quelli che ho conosciuto in questa avventura. Qualcuno diceva che ognuno ha gli amici che si merita, a me è andata bene.