Università Politecnica delle Marche

Facoltà di Ingegneria

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Magistrale in Ingegneria Informatica e dell'Automazione

Tesi di Laurea

Automatizzazione del processo di dynamic malware analysis in un Security Operation Center

Automation of the dynamic malware analysis process in a Security Operation Center

Relatore

Prof. Domenico Ursino

Candidato

Jacopo Carloni

Correlatori

Ing. Paolo Russo

Ing. Rossella Oliva

Anno Accademico 2018-2019

Indice

Elenco delle figure

Elenco delle tabelle

Elenco dei listati

Introduzione

La sicurezza informatica, o meglio la sicurezza delle informazioni, è una tematica con la quale oramai qualsiasi organizzazione o impresa si è imbattuta, indipendentemente dalla sua dimensione o settore di riferimento. Al giorno d'oggi, chiunque voglia avviare un'attività che prevede l'impiego, anche in minima parte, di apparecchiature informatiche, deve tenere bene a mente la necessità di investire nella messa in sicurezza di tale strumentazione.

> "[...] Ci sono solo due tipologie di aziende: quelle che sono state hackerate, e quelle che lo saranno. Ed entrambe convergeranno verso un'unica categoria: quelle che sono state hackerate e quelle che saranno hackerate di nuovo."

Questa affermazione [19] dell'ex direttore dell'FBI, Robert Mueller, risale al 2012 e lascia intendere quale sia la strada che otto anni fa si stava imboccando. Infatti, gli enti, le aziende o le organizzazioni che si occupano di redigere report periodici riguardanti i trend delle minacce informatiche e degli attacchi rilevati, sono tutti d'accordo nel sostenere che c'è un costante aumento del numero di eventi e una sempre più importante sofisticazione degli attacchi. Di pari passo, cresce anche il numero e cambia la tipologia di attori: se nel ventennio scorso gli attacchi informatici erano perlopiù perpetrati da singoli individui scarsamente organizzati, che agivano spinti da curiosità e passione per la tecnologia, oggi si parla quasi esclusivamente di attività di cybercrime, cyberwarfare o cyber espionage da parte di governi o grandi organizzazioni. Onde evitare danni maggiori, economici e non solo, è necessario correre ai ripari e investire per preparare una buona strategia di difesa e un buon piano di risposta agli incidenti, in caso di fallimento della prima. È per questo motivo che molte aziende o enti governativi hanno iniziato da diverso tempo ad impiegare le proprie risorse per la creazione di centri esclusivamente dedicati alla protezione dei sistemi informatici. Tali centri sono chiamati *Security Operation Center (SOC)*. In alternativa, quelle organizzazioni che non riescono a sopportare lo sforzo economico di creare, gestire e mantenere un SOC, sempre più frequentemente affidano la propria difesa ai fornitori di servizi di sicurezza (*Managed Security Service Provider (MSSP)*.

I SOC rappresentano la fusione di tre elementi: persone, tecnologie e processi. Affinché funzionino al meglio, questi tre devono essere di qualità e ben integrati tra loro. Dunque, c'è bisogno di tecnologie all'avanguardia, personale con competenze

molto approfondite e processi ben progettati e applicati. La capacità di prevenire gli attacchi e, soprattutto, la rapidità con la quale vengono rilevati e con la quale si rimedia agli incidenti sono estremamente importanti. In questo senso, due sono le strategie che si stanno adottando per rendere sempre più efficienti ed efficaci le attività dei SOC: l'utilizzo di Intelligenza Artificiale e l'automatizzazione dei processi.

L'automazione nel settore della cybersecurity sta crescendo così velocemente che Gartner, in [4], predice che entro il 2021 il 70% delle organizzazioni con un SOC dedicato utilizzerà strumenti automatici, come i Security Orchestration, Automation and Response (SOAR).

L'automazione diventa sempre più rilevante per due motivi: la carenza di personale adeguato da impiegare nei SOC e la rapidità ed efficacia di azione. Infatti, come riportato in [31], il 62% della popolazione intervistata conferma la mancanza di personale adeguato da impiegare nel SOC. L'automazione, quindi, può essere d'aiuto sia per sopperire a quanto detto, sia per analizzare una quantità di dati sempre maggiore in poco tempo, eseguendo task frequenti e ripetitivi. Gli strumenti automatici non prenderanno il posto dell'operatore umano. Piuttosto, le attività degli analisti del SOC risulterebbero snellite e indirizzate verso compiti in cui c'è maggior bisogno di ragionamenti di alto livello o dove è necessario prendere decisioni. Quindi, per esempio, lo strumento automatico rileva l'anomalia, l'operatore, con il suo senso critico, identifica la minaccia e l'attore.

Le tipologie d'attacco dai quali occorre difendersi sono numerose, di vario genere ed interessano tutti i livelli dello stack ISO/OSI (a partire dal livello fisico fino ad arrivare al quello applicativo e al fattore umano, se lo si vuole inserire come ottavo livello della pila).

In particolare, la posta elettronica è considerata il principale vettore di attacchi informatici, come testimonia anche il Clusit nel suo rapporto annuale [33]. Secondo tale report, il 70% di messaggi del campione analizzato è costituito da posta indesiderata (spam, malware, phishing e altri attacchi).

Fatte queste premesse, il presente elaborato vuole proporre una soluzione al problema degli attacchi tramite posta elettronica da adoperare all'interno di un Security Operation Center. Come spiega il National Cyber Security Center (NCSC) inglese in [27], la difesa contro gli attacchi che sfruttano le email ha una natura stratificata e non può essere raggiunta con un unico strumento. Il sistema che proponiamo nella presente tesi, infatti, rappresenta un primo livello di difesa di supporto alle operazioni degli analisti, che viene eseguito in maniera del tutto automatica.

Solitamente, senza automazione, quando un analista si trova di fronte a una minaccia da contenere e bloccare, deve eseguire una serie di azioni molto dispendiose in termini di tempo, utilizzando anche molteplici software e strumenti diversi. Abbiamo pensato di riunire alcune di queste attività all'interno di un unico flusso di lavoro completamente automatico. In particolare, abbiamo voluto fare in modo che un qualsiasi messaggio di posta elettronica in arrivo venisse analizzato, un po' come accade per i filtri antispam. Piuttosto che basare l'analisi dell'email sull'header e la struttura, abbiamo deciso di implementare un nostro parser, chiamato Ematex, il quale estrae dal messaggio gli elementi che potrebbero comportare la distribuzione di malware, ovvero i file allegati e i link presenti nel testo.

Per capire se tali elementi siano malevoli o meno, Ematex li invia a uno strumento di dynamic malware analysis. Questo tipo di analisi consiste nell'eseguire file o nell'accedere a pagine web, tramite link, in un ambiente isolato e controllato, per capire le conseguenze dell'esecuzione del file o dell'apertura del link. Il software utilizzato per effettuare l'analisi dinamica è CAPE Sandbox. Questo è una soluzione di sandboxing open source che permette di analizzare file di diversi formati su macchine virtuali o reali con Windows.

Abbiamo cercato di rendere l'analisi di allegati e URL un po' più approfondita, aggiungendo al nostro flusso di lavoro un controllo su una piattaforma di Threat Intelligence, chiamata MISP. Questa è un raccoglitore di eventi di sicurezza, che gli utenti possono consultare o condividere con la comunità. Ovviamente, possono essere stabilite delle restrizioni riguardo cosa condividere e con chi. Lo scopo principale di MISP è quello creare un bacino di raccolta di informazioni utili su minacce, attori e attacchi già avvenuti e analizzati, per evitare di eseguire di nuovo analisi già svolte. Il nostro utilizzo di questa piattaforma è molto limitato e per il momento non ne vengono sfruttate appieno le potenzialità; tuttavia ci è sembrato utile inserirla all'interno del nostro flusso cosicché, all'occorrenza, in futuro sarà possibile effettuare piccole modifiche per sfruttarla meglio.

Per concludere, una volta che si è appurata la natura malevola di allegati e/o link, Ematex notifica questi eventi su un sistema di Incident Response, chiamato TheHive. Questo è un *Case Management System* utilizzabile all'interno di un SOC per coordinare le attività degli analisti. Questi, quando tramite TheHive rilevano la presenza di un'email sospetta, possono reagire informando il destinatario del pericolo per evitarne l'apertura, oppure, se decidono di approfondire le analisi, possono lanciare gli Analyzer di Cortex. Quest'ultimo è un ulteriore software, altamente integrabile con TheHive, il quale permette di avviare su nuovi processi dei programmi di analisi di IP, URL o file (gli Analyzer) o dei programmi che costituiscono la reazione a eventi accaduti (i Responder).

La presente tesi descrive in dettaglio il flusso automatico di lavoro, qui riportato in breve. Abbiamo organizzato questo elaborato come di seguito specificato:

- Il primo capitolo è incentrato sui SOC: si discuterà della loro nascita ed evoluzione, dei modelli esistenti e del loro funzionamento.
- Il secondo capitolo introduce l'analisi di malware, differenziando tra le tecniche di analisi statica e dinamica e tra le tecniche più basilari e quelle più avanzate. Avendo adoperato l'analisi dinamica per il nostro lavoro, descriveremo più in dettaglio la soluzione scelta e utilizzata, ovvero CAPE Sandbox.
- Il terzo capitolo descrive le piattaforme di Incident Response utilizzate, ovvero TheHive e Cortex. Inoltre, verranno introdotti il concetto di Cyber Threat Intelligence e, per sommi capi, la piattaforma MISP, altamente integrabile con TheHive e Cortex.
- Il quarto capitolo descrive l'architettura dei sistemi di posta elettronica e i protocolli SMTP e IMAP/POP3 per l'invio e la ricezione di email. Verrà descritto, infine il software di web hosting utilizzato per creare un sistema di web mail.
- Nel quinto capitolo, viene illustrato come tutto quanto esposto precedentemente viene integrato per formare la nostra soluzione di analisi di email. Inoltre, verrà introdotto il nostro parser di email, nonché regista di tutto il flusso, ovvero Ematex.

- Nel sesto capitolo, inizialmente, saranno valutati i punti di forza, di debolezza, le opportunità e le minacce relative alla soluzione proposta. Successivamente, verranno descritte le lezioni apprese durante lo studio e l'implementazione del nostro sistema.
- Infine, nel settimo capitolo, verranno tratte le conclusioni in merito al lavoro realizzato e verranno esaminati alcuni possibili sviluppi futuri.

1

Security Operation Center

Questo capitolo descrive un Security Operation Center (SOC), le sue caratteristiche e le attività che lo identificano all'interno di un'organizzazione.

1.1 Introduzione

La dicitura "Security Operation Center" (SOC) non è inequivocabile ed è figlia delle evoluzioni del mercato della sicurezza delle informazioni. Ad essa non viene associata una definizione univoca e ben precisa. Piuttosto, un SOC può essere definito sulla base della sua composizione e delle funzioni che esercita.

> Un Security Operation Center (SOC) è il componente interno o esterno a una organizzazione costituito da esperti del settore IT il cui scopo è quello di difendere i sistemi e le infrastrutture informatiche dell'organizzazione stessa da attività non autorizzate perpetrate da agenti esterni (ma spesso anche interni).

In generale un SOC è costituito da analisti di sicurezza che hanno il compito di assicurare la confidenzialità, l'integrità e la disponibilità (triade CIA) di informazioni e di sistemi e infrastrutture informatiche in forza all'organizzazione. Quindi, un SOC può essere visto come un insieme di persone, tecnologie e processi, ed è risaputo come, sempre più frequentemente, la sicurezza di un'intera struttura o rete informatica non è legata al solo aspetto tecnologico, ma un ruolo importante viene svolto anche dai processi e, soprattutto, delle persone. Come si vedrà poi nel prosieguo di questo documento, un SOC può essere costituito all'interno di un'organizzazione oppure è possibile acquistare i corrispettivi servizi erogati da provider di servizi di sicurezza (Managed Security Services Provider, MSSP).

1.2 Cenni Storici

Il whitepaper redatto da Hewlett-Packard nel 2013 [39] descrive l'evoluzione dei SOC nel corso degli anni.

I primi Security Operation Center vennero creati nel contesto di entità militari e governative degli Stati Uniti d'America, in quanto furono i primi a utilizzare le reti TCP/IP; inoltre in quegli ambienti, i concetti di *Intelligence*, *Risk Management* e *Operations* erano all'ordine del giorno e ben compresi da tutti.

Naturalmente, i metodi e le tecniche, sia di attacco che di difesa, si sono costantemente evoluti e sono diventati sempre più sofisticati man mano che la tecnologia avanzava; tuttavia, per gli analisti SOC rimane costante non solo la sfida di rilevare nel più breve tempo possibile le minacce correnti ed emergenti, ma anche quella di predire i futuri metodi di attacco.

Di seguito si descriverà come i SOC siano evoluti nel tempo attraverso 5 tappe fondamentali.

Prima generazione (1975-1995)

La prima generazione appartiene al ventennio '75-'95, praticamente a partire dagli albori di Internet, il quale non era stato inventato e creato seguendo il più recente principio di *Security by design*. I primi attacchi che venivano fatti erano quasi esclusivamente figli della creatività e dell'ingegno di pochi e singoli individui. Non c'era ancora niente di organizzato e ripetibile. Nel corso di questo ventennio, il mondo "cyber" venne allo scoperto e ottenne più visibilità grazie a film hollywoodiani, libri e pubblicazioni. Cominciarono, dunque, a nascere i primi strumenti di difesa delle reti: firewall e antivirus. In questo contesto, però, il SOC rimane identificato in una singola persona o un ristretto gruppo di persone aventi un forte background di networking. Al termine del ventennio, le organizzazioni militari e governative videro nascere i primi Security Operation Center, i quali risultavano essere ancora poco strutturati e con scarse risorse.

Seconda generazione (1996-2001)

In questo periodo si cominciò a parlare più assiduamente di codice malevolo (malware). Si ebbe prova di come virus e worm potevano generare danni attraverso la rete. Si cominciò anche a parlare di individuazione di vulnerabilità (nel 1999 MITRE crea la repository/sistema CVE) e patch di sistema. Si iniziano ad avere i primi SOC all'interno di organizzazioni commerciali e si iniziò a vendere i primi servizi di monitoring della sicurezza a clienti paganti. Firewall, antivirus, proxy, Intrusion Detection System (IDS) e strumenti per la ricerca delle vulnerabilità si affermano e prendono sempre più campo nel mercato. Il focus in particolare è posto sugli IDS: è proprio in questo periodo, infatti, che nascono strumenti quali SNORT (1998) e tcpdump. Nelle battute finali di questo quinquennio si comincia a parlare di SIEM come di una tecnologia utilizzata per correlare diversi eventi sotto un unico sistema.

Terza generazione (2002-2006)

Gli anni di riferimento della terza generazione di Security Operation Center sono quelli che vedono evolvere le minacce cyber in attacchi finalizzati al guadagno economico. Iniziano a nascere i primi mercati underground. Il numero di attacchi cresce considerevolmente e iniziano ad essere create le prime organizzazioni cybercriminali.

Si passa sempre più, quindi, da attacchi con worm e virus dai forti connotati distruttivi e dannosi per l'intera infrastruttura, a malware più particolarizzati e specifici per un certo tipo di attacco. Si comincia ad avere una consapevolezza maggiore della sicurezza e della protezione dei dati. Dal punto di vista dei SOC, diventano sempre più maturi, anche quelli esterni alle organizzazioni governative e militari; nasce il US-CERT nel 2003. Si comincia a puntare sulla prevenzione piuttosto che sul solo rilevamento di attacchi; nascono le prime tecnologie di prevenzione come gli IPS (Intrusion Prevention System).

Quarta generazione (2007-2012)

Gli anni che vanno dal 2007 al 2012 vedono la nascita del fenomeno di cyber attacchi da parte di stati nazionali con lo scopo di rubare proprietà intellettuale e know-how alle aziende estere, nonché di effettuare sabotaggi o spionaggi. Il primo caso di cyber attacco "di stato" è quello della Russia ai danni dell'Estonia nel 2007. In questo periodo si diffonde particolarmente il fenomeno dell' "Hacktivism" grazie a dimostrazioni di alcuni gruppi contro organizzazioni o individui, con i social media sullo sfondo a fare da piattaforme di comunicazione e coordinamento. È in questo quinquennio che accade il celebre attacco contro i sistemi SCADA delle centrali nucleari iraniane tramite il Trojan "Stuxnet" (2010). Dal lato della difesa,invece, si diventa sempre più consci del fatto che non è più questione di rilevamento o prevenzione di intrusioni; con buona probabilità queste accadranno, perciò si deve essere in grado di rilevare le eventuali esfiltrazioni di dati e di contenere i danni. Anche il settore privato è sempre più consapevole dell'importanza di un Security Operation Center che possa prevenire, rilevare e rimediare ai danni subiti a seguito di incidenti.

Quinta generazione (2013-?)

I SOC di quinta generazione sono in continua e costante evoluzione. Ciò è dovuto al fatto che le minacce da contrastare hanno analoga caratteristica. Le minacce sono sempre di più e sempre nuove, i prodotti software che fanno fronte ad esse devono essere sempre più avanzati e all'avanguardia. I SOC di ultima generazione sono non solo reattivi, ma proattivi, sono sempre più efficienti e, in particolar modo, integrati con funzionalità avanzate di machine/deep learning e big data analytics. Rispetto agli analisti delle generazioni precedenti, quelli che si trovano a lavorare in un SOC di ultima generazione dispongono di un gran numero di sistemi SIEM, IDS/IPS, EDR che permettono di essere integrati tra loro e ottenere una visione complessiva di ciò che sta accadendo nella rete o, persino, nei singoli endpoint delle organizzazioni.

I SOC di quinta generazione sono "analysis-focused": memorizzano grandi quantità di informazioni strutturate o meno individuando pattern di attacchi o comportamenti anomali per poi predire eventuali eventi futuri. Con l'evoluzione dei SOC, evolvono anche le persone che li costituiscono: prima l'analista era un operatore con forti conoscenze informatiche e di reti, ora, tenendo in considerazione questi enormi sviluppi, il SOC deve essere formato anche da molteplici figure provenienti dal mondo della big data analytics, della business intelligence, della matematica, della statistica e, ovviamente, dell'informatica e delle reti.

Sempre più importante diventa la condivisione di conoscenze legate alle tecniche di attacco, alle vulnerabilità sfruttate o anche agli attori stessi che le sfruttano. La Threat Intelligence assume, dunque, un ruolo sempre più centrale.

I SOC più moderni e avanzati stanno anche cercando di allestire capacità di contrattacco. Si è notato come, per ottenere buone capacità difensive e attuare efficienti misure di prevenzione, sia necessario conoscere i metodi e le tecniche di attacco. È per questo che i governi e le più grandi aziende e organizzazioni mantengono gruppi di Red Team e Blue Team: da una parte si testano e si "stressano" i sistemi di sicurezza, dall'altra si cerca di migliorare ed aggiornare le tecniche di protezione.

1.3 Modelli di Security Operation Center

Come precedentemente accennato, un SOC può essere costruito internamente ad un'organizzazione, ma ciò risulterebbe molto costoso e time-consuming; in alternativa, si possono acquistare esternamente servizi offerti da terze parti per la prevenzione, la protezione e la risposta alle minacce informatiche. Una seconda distinzione va fatta tra i SOC centralizzati, che si trovano vicini geograficamente (al limite nello stesso edificio) all'organizzazione da difendere, e i SOC distribuiti, che si dispongono, invece, in zone o, addirittura, aree geografiche completamente diverse.

Fatta questa prima distinzione, occorre tenere presente che non esiste una sola tipologia di SOC. Gartner, in una sua pubblicazione, descrive ben cinque modelli di Security Operation Center [23].

Virtual SOC (VSOC)

Questo modello di SOC non risiede in una struttura ad hoc e non ha infrastrutture dedicate. Esso è costruito su tecnologie di sicurezza decentralizzate con un team virtuale che diviene attivo all'occorrenza. Questi SOC sono perlopiù reattivi e possono essere migliorati tramite SIEM e strumenti di Analytics. Non rimangono in funzione ventiquattro ore al giorno e sette giorni alla settimana, ma per un tempo più limitato. Essi risultano essere la tipologia di SOC più adatta per le piccole e medie imprese.

I vantaggi di questo modello sono la velocità e semplicità di implementazione, la scalabilità e i costi ridotti. Il VSOC è utile per quelle organizzazioni che sono ben informate e sensibili alla problematica della cybersecurity, ma che non possiedono risorse economiche o competenze di settore per poter allestire una soluzione 24/7 "in casa".

Lo svantaggio principale da tenere in considerazione riguarda il fatto che dati interni ad un'azienda transitano attraverso sistemi di terze parti.

Multifunction SOC/NOC

Security Operations Center (SOC) e Network Operations Center (NOC) hanno molti aspetti in comune. Spesso le funzioni di entrambi sono organizzate a livelli, con ruoli simili ai livelli più bassi. Essi condividono dei tool, pur mantenendo ciascuno

le proprie tecniche e i propri strumenti specifici. Entrambi richiedono operatori con vaste conoscenze tecniche.

Tuttavia, se, da una parte, il NOC è a supporto del business, il ruolo del SOC è quello di proteggerlo. Per esempio, se viene rilevato un evento, il personale NOC assocerà quest'ultimo a un malfunzionamento di un dispositivo o a un problema di un sistema e provvederà a risolvere il tutto, per esempio rimpiazzando tale dispositivo. Il personale SOC, invece, sarà più incline ad associare tale evento ad un'attività malevola, andandola a investigare e, all'occorrenza, estirpare. Le caratteristiche congiunte di NOC e SOC producono insieme una potente sinergia che potrebbe portare grossi benefici per un'organizzazione, come un tempo di risposta agli incidenti ridotto, nonché il miglioramento della pianificazione delle contromisure, grazie ad una celere individuazione delle cause.

In questo modello si hanno strutture e infrastrutture dedicate, con un team di esperti che non operano solo azioni di sicurezza continuativamente, ma anche operazioni più prettamente associate al comparto IT. In questo modo i costi vengono ridotti.

Tale modello risulta particolarmente adatto alle piccole e medie imprese o alle compagnie di medie dimensioni con una bassa esposizione al rischio.

Co-managed SOC

Il tradizionale Co-managed SOC è il servizio di sicurezza offerto dai vendor MSSP (Managed Security Service Providers) per le aziende di gradi o medie dimensioni, il cui core business non è legato all'IT o alla sicurezza.

Il servizio offerto, solitamente, copre otto ore per cinque giorni nel caso delle operazioni, e ventiquattro ore per sette giorni nel caso del monitoring dei sistemi. Se i fondi sono limitati, si preferisce mantenere interni i business a più alto valore e le funzionalità più critiche. Tra queste troviamo, per esempio, il design dell'architettura, la governance, la gestione dei rischi e l'analisi e la risposta a incidenti. Le funzioni di sicurezza di base, come la gestione dei dispositivi, possono essere, invece, delegate ad un provider, il quale sicuramente sarà in grado di soddisfare delle SLA più facilmente a un costo minore per l'organizzazione. Un MSSP può, inoltre, fornire assistenza per altri eventi inusuali.

Dedicated SOC

Per quelle organizzazioni di grandi dimensioni che sono mature dal punto di vista IT e della sicurezza, la scelta migliore ricade sulla creazione e sul mantenimento di un SOC in casa. Ciò, ovviamente, significa che l'azienda debba essere in grado di affrontare i costi necessari. Uno dei maggiori vantaggi di mantenere un SOC interno è quello di avere il controllo completo sulla rete e su tutto l'ambiente. Anche il team sarà interno e avrà la capacità di monitorare tutto.

Possibili svantaggi sono legati, per esempio, al difficile arruolamento e mantenimento di talenti, ai costi di investimento abbastanza elevati e ai tempi di avvio molto alti.

Una versione avanzata di questo modello è rappresentata dal *Fusion SOC* che incorpora le tradizionali funzioni con alcune nuove, come Data Science, Threat

Intelligence, Computer Incident Response Team (CIRT) e Operational Technology (OT).

Command SOC

In alcuni casi, per organizzazioni veramente molto grandi che offrono una grande varietà di servizi, come le organizzazioni statali, potrebbe essere necessaria la collaborazione tra più SOC. In certi casi i SOC possono operare in maniera del tutto autonoma come centralizzati o distribuiti; in altri casi, invece, i SOC potrebbero collaborare ed essere, per questo, organizzati in modo gerarchico. Nel secondo caso, è chiaro come uno di essi debba obbligatoriamente svolgere il ruolo di coordinatore di tutti gli altri in modo da eseguire in maniera coordinata le funzioni tradizionali; tale SOC che svolge il coordinamento è proprio il Command SOC.

1.4 SOC, CERT, CSIRT, CIRT: Quale dei tanti?

Prima di passare a descrivere più dettagliatamente quali sono i compiti di un SOC, è doveroso fare un po' di chiarezza sulla differenza che c'è tra le sigle SOC, CSIRT, CIRT e CERT e molte altre riportante in Tabella 1.1. Questi sono tutti termini che ricorrono quando si parla di Incident Response o Cyberthreat Intelligence o, in generale, di difesa delle reti e infrastrutture informatiche.

IRT	Incident Response Team
IRC	Incident Response Capability
IHT	Incident Handling Team
IMT	Incident Managing/Management Team
CSIRT	Computer Security Incident Response Team
CIRT	Computer Incident Response Team
CIRC	Computer Incident Response Capability/Center
SIRT	Security Incident Response Team
SERT	Security Emergency Response Team
CERT	Computer Emergency Response Team

Tabella 1.1. Significato delle sigle che si incontrano in ambito di Incident Response

Le prime quattro sigle individuano i ruoli, mentre le restanti indicano i team veri e propri. Per quanto riguarda l'acronimo CERT, esso sta per Computer Emergency Response Team (o anche Computer Emergency Readiness Team) e, di fatto, rappresenta un marchio registrato negli USA dalla Carnegie Mellon University. Esso nasce nel 1988, originariamente come un gruppo di incident response all'interno dell'Università, in collaborazione con il governo e alcune agenzie di law enforcement. Oggi rappresenta uno dei centri di ricerca in tematica di cybersecurity più importanti al livello mondiale. Esso non va confuso con lo US-CERT, che venne creato solo agli inizi degli anni duemila dal dipartimento di sicurezza interna degli Stati Uniti e dalla stessa Carnagie Mellon University; questo è un CSIRT a tutti gli effetti.

Il CERT/CC (Computer Emergency Response Team-Coordination Center), il primo descritto e nato, studia, ricerca e analizza, in generale, gli eventi e le tematiche di sicurezza informatica a livello globale; lo US-CERT interviene e gestisce casi che riguardano la sicurezza nazionale degli Stati Uniti. Una delle funzioni del primo, inoltre, è anche quella di promuovere la cooperazione e la condivisione di informazioni preziose tra i vari CERT nazionali e privati.

CSIRT, CIRT e tutte le altre sigle sono sinonimi. Probabilmente queste due diciture sono le più azzeccate dal punto di vista tecnico per indicare un pool di esperti nell'individuazione e nella risposta ad un'intrusione. Una differenza che si ha tra CSIRT e CERT è che il secondo è molto più incentrato nella Threat Intelligence e nella condivisione di IOC e di informazioni con enti analoghi (esistono, infatti, svariati CERT, ciascuno con la propria particolare dicitura. In Italia, solo per citare alcuni dei 27 CERT, esiste il CERT-PA della Pubblica Amministrazione o il CERT di Poste Italiane. A livello europeo troviamo il CERT-EU). L'ente europeo per la sicurezza informatica e delle reti, nominato ENISA, raccoglie in una mappa [26] i principali CERT/CSIRT dei paesi europei.

Comunque, che venga chiamato con CERT, CSIRT o CIRT, i compiti fondamentali e le caratteristiche di questo soggetto sono descritti in [18] e vengono riassunti nella figura 1.1

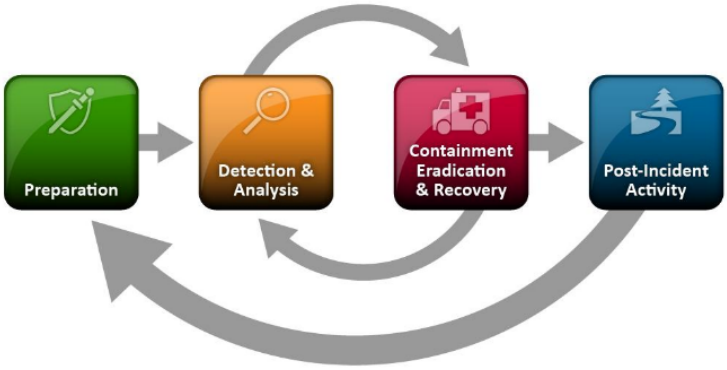

Figura 1.1. Ciclo di vita dell'Incident Response

Per quanto riguarda il Security Operation Center, invece, la prima cosa da notare è che rappresenta l'unica sigla di quelle in considerazione non presente nel glossario del NIST. Ciò potrebbe essere considerata come prova del fatto che il termine SOC sia stato sviluppato dagli addetti ai lavori nel mercato. Il Security Operation Center, tuttavia, è considerato da molti come una sorta di soggetto ancora più ampio rispetto ai due precedentemente analizzati. Infatti, alcuni ritengono che il CSIRT possa essere costituito all'interno di un SOC. I compiti di un SOC non si limitano alla IR (Incident Response), ma riguardano, in generale, tutto ciò che può essere fatto per mettere in sicurezza un'organizzazione, a partire dalla raccolta costante di tutte quelle informazioni necessarie per rilevare un incidente fino alla reazione e risposta all'incidente, a volte anche in maniera proattiva.

Dunque per riassumere e concludere, un CSIRT è quel team di esperti che risponde ad un incidente informatico cercando di attutirne il più possibile gli effetti. Il CERT ricopre funzioni simili, e in più è associato al concetto di Cyber Threat Intelligence, e quindi alla condivisione di informazioni con strutture analoghe. Il SOC svolge compiti a più ampio respiro per ciò che riguarda il mantenimento in sicurezza (informatica) di un'organizzazione, il rilevamento nel più breve tempo possibile di un attacco o un'intrusione e la reazione adeguata per evitare o limitare i danni.

Alcuni siti online ([25, 24]) riportano l'immagine mostrata in Figura 1.2 per descrivere aspetti in comune e non tra i tre soggetti analizzati.

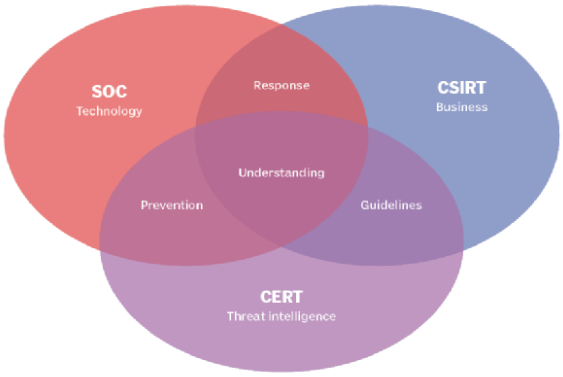

Figura 1.2. Confronto tra funzioni di SOC, CERT, CSIRT

1.5 Funzionamento di un SOC

Un Security Operation Center può essere piccolo e costituito da poche persone, oppure può essere molto grande, con centinaia di persone e più centri che cooperano, magari organizzati in modo gerarchico, come accennato nella Sezione 1.3.

Tipicamente un SOC di medie dimensioni si occupa delle seguenti attività:

- Prevenzione di incidenti di sicurezza informatica, tramite l'analisi continua delle minacce, lo scanning continuo di reti e computer alla ricerca di vulnerabilità o infezioni, lo sviluppo di contromisure da attuare all'occorrenza, la definizione di policy di sicurezza.
- Monitoraggio, rilevamento e analisi di potenziali intrusioni in real-time e attraverso pattern individuati grazie all'analisi di dati storici
- Risposta a incidenti confermati.
- Garanzia della cosiddetta *Situational Awareness* e confronto con strutture e organizzazioni appropriate per rimanere aggiornato su trend, nuove minacce, nuovi incidenti e così via.
- Ingegnerizzazione e messa in funzione di strumenti per la protezione della rete e dei sistemi, come IDS/IPS, SIEM, EDR.

Dunque, riassumendo un po' il tutto, un SOC cerca di ottenere quanti più dati disponibili di sistemi IT da proteggere e mette in atto delle analisi continue e real-time per individuare degli eventi.

Un evento viene definito come "una qualsiasi accadimento osservabile in un sistema e/o una rete" [21], e, di fatto, non è altro che un dato grezzo. Un evento potrebbe rappresentare l'indizio che qualcosa di anomalo stia avvenendo nel sistema, tuttavia è necessaria l'analisi umana, magari correlando tale evento con altri, per poter generare un allarme. Si può intuire, dunque, come un SOC costantemente memorizzi grandi moli di dati provenienti dai sistemi da proteggere; tali dati saranno successivamente analizzati.

Gli operatori di un SOC sono principalmente di due tipologie:

- *Tier 1*, che si occupa del monitoraggio in real-time dei sistemi e delle reti e del triage. Questi operatori stabiliscono se una serie di eventi supera una certa soglia, nel qual caso viene generato un allarme dal quale scaturisce l'analisi dell'incidente. In genere sono gli analisti con meno esperienza
- *Tier 2*, che esegue investigazioni più approfondite per determinare cosa sia realmente successo. L'operatore di secondo livello non opera in real-time come il precedente, in quanto spesso per ricostruire un attacco o un'intrusione possono essere necessarie numerose evidenze raccolte nel tempo. La funzione di un Tier 2 è quella di determinare se, ed eventualmente come, un incidente è accaduto e quali sono stati i danni prodotti. Solitamente per ricoprire il ruolo di analista Tier 2 è necessaria una maggiore esperienza del Tier 1.
- *Tier 3*. Questa figura viene descritta in [17]. Poiché, generalmente, i primi due livelli consumano gran parte delle risorse, umane e non, di un SOC, in media, le aziende che possiedono un SOC non investono su operatori di terzo livello. L'operatore Tier 3 è quello con la più grande esperienza in analisi e risposta a incidenti. La sua attività viene definita *Threat Hunting*: ricerca continuamente minacce complesse che i software e i sistemi di difesa non hanno rilevato come sospette.

In accordo con quanto specificato in [21], per incidente si intende "un accadimento comprovato che ha o avrebbe potuto minare la confidenzialità, l'integrità o la disponibilità di un sistema informativo o delle informazioni che il sistema processa, immagazzina o trasmette; oppure un accadimento che costituisce una violazione, o imminente minaccia, di violazione di policy o procedure di sicurezza".

È utile considerare come non sempre generare allarmi e prendere contromisure al primo sospetto di attività malevola sia la scelta più azzeccata. L'attuazione di contromisure dovrebbe essere ben ponderata, in quanto a volte si rischierebbe di bloccare attività benevole e ciò potrebbe avere enorme impatto, anche economico, sull'organizzazione. Altre volte attuare contromisure risulterebbe più costoso dei danni provocati dall'incidente accaduto. Un'altra strategia potrebbe essere, inoltre, quella di non mettere in atto contromisure per arginare intrusioni o attacchi in modo da poter monitorare e studiare il comportamento dell'attaccante.

Lo schema in Figura 1.3 mostra quali sono le attività e i ruoli all'interno di un SOC.

Un aspetto fondamentale è quello della Cyber Intelligence. Infatti non è sempre sufficiente analizzare dati provenienti dai sistemi monitorati per rilevare un attacco.

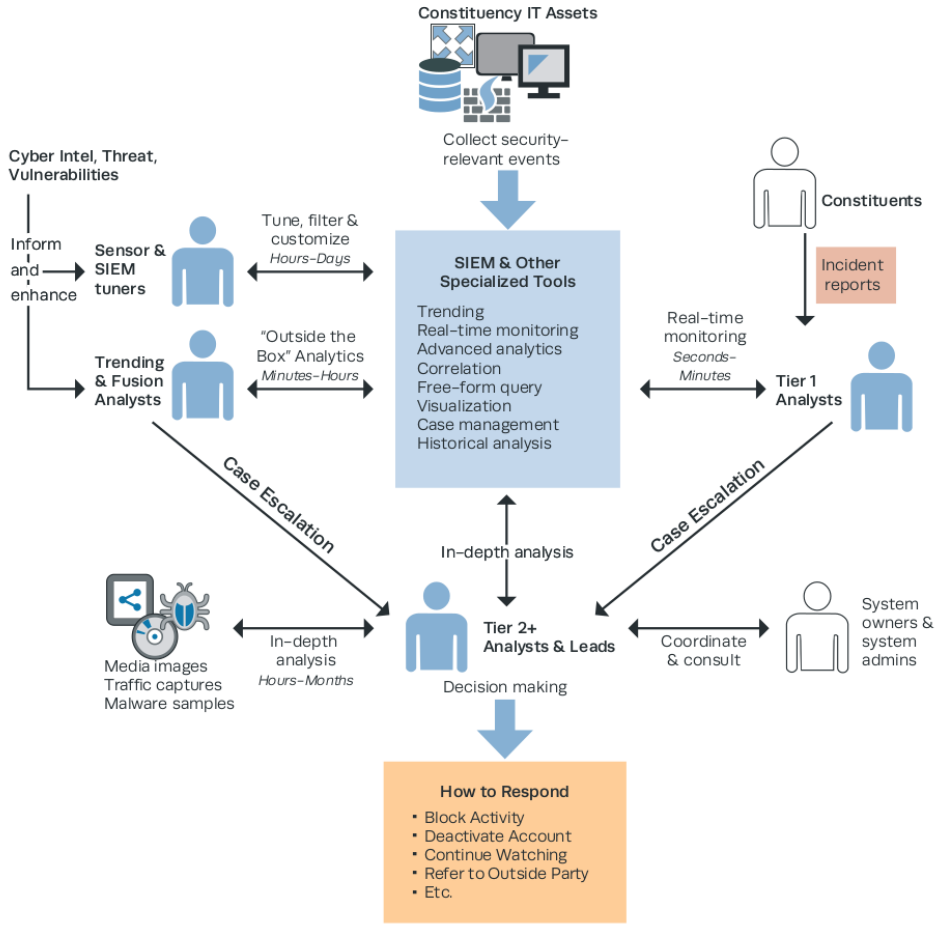

Figura 1.3. Funzionamento di un SOC

Spesso anche informazioni provenienti dall'esterno, come TTP (Tattiche, Tecniche e Procedure) di attori di minaccia, IOC (Indicator Of Comporomise), vulnerabilità scoperte e pubblicate su bollettini appositi, report di incidenti o aggiornamenti di signature, offrono un importante valore aggiunto alle analisi eseguite in locale e permettono di mantenere il passo con l'ambiente circostante in continua evoluzione e sofisticazione.

Il seguente elenco riporta e riassume tutte le macro attività che un SOC esercita:

- analisi tempo reale;
- intelligence e Trending;
- analisi e risposta agli incidenti;
- investigazioni e analisi forensi;
- scansione e Valutazione;
- produzione di Report;
- miglioramento dell'Awareness.

1.6 Conclusioni

Come si è potuto evincere dalla lettura di questo capitolo, creare e mettere in funzione una struttura come quella descritta non è triviale. Molti sono i fattori da tenere in considerazione e altrettante sono le scelte che si possono effettuare in termini di modello, di tecnologie da utilizzare, sensori da posizionare e monitorare, personale da arruolare, tenere e formare e così via. Si è visto come, essendo veramente numerose le funzioni che esegue un Security Operation Center, appare piuttosto complicato eliminare del tutto la presenza dell'operatore umano. Che gli operatori siano di tipo Tier 1 o Tier 2, sono sempre fondamentali.

Dato che un SOC può essere visto come la fusione di tecnologie, persone e processi, ciò che si è tentato di fare in questo elaborato, e che verrà descritto nei capitoli successivi, è automatizzare determinati processi in modo tale da lasciare all'analista la "semplice" supervisione degli stessi, e far si che le diverse fasi di triage o analisi di evidenze raccolte, che magari possono essere effettuate da un operatore con l'uso di strumenti software diversi in momenti diversi, siano connesse automaticamente. All'analista SOC non rimane altro che prendere decisioni contestualmente ai risultati ottenuti.

È ben noto come le tipologie di attacchi informatici siano molteplici, e ciascuna interessi un particolare livello dei sette dello stack ISO/OSI, ai quali andrebbe aggiunto anche l'elemento umano in cima a tutti. E sono proprio agli attacchi informatici che sfruttano l'uomo tramite la Social Engineering sui quali ci concentreremo nel prosieguo del documento, realizzando un'integrazione di software per l'estrazione di allegati e link da email, la loro analisi e generazione di eventuali alert. Relativamente all'analisi di email e allegati e/o link presenti, non esiste, al giorno d'oggi, nel mercato uno strumento all-in-one che sia in grado di valutare l'affidabilità della mail e delle informazioni ivi contenute e generare all'occorrenza allarmi in un SOC. Esistono strumenti per l'estrazione di allegati e link, esistono strumenti antispam, esistono software utilizzati dagli analisti per generare alert. Ma non esiste un unico prodotto in grado di prendere una mail, estrarre tutti gli "observable", analizzarli in tempo reale e creare automaticamente un allarme se necessario.

2

Dynamic Malware Analysis: Cuckoo Sandbox e CAPE Sandbox

Il presente capitolo passa in rassegna le tecniche per l'analisi di file, distinguendo tra l'analisi statica e quella dinamica. Si porrà, successivamente, l'attenzione sull'analisi dinamica e sulle due soluzioni di sandboxing utilizzate.

2.1 Introduzione

Nel capitolo precedente abbiamo visto un Security Operation Center, la storia della sua nascita, l'evoluzione, i modelli, l'organizzazione, la funzione e le attività che lo caratterizzano. Il lavoro dell'analista SOC di secondo livello (che nel Capitolo 1 veniva definito *Tier 2*) è quello di analizzare artefatti ed evidenze raccolte nel tempo a seguito di allarmi generati dagli operatori *Tier1*. In particolare, l'analisi di file rappresenta una delle principali attività che permette di capire se uno o più endpoint del sistema monitorato sia infetto ed, eventualmente, la gravità dell'infezione.

I software malevoli, o malware, giocano un ruolo importante in quasi tutte le intrusioni o gli incidenti di sicurezza. Esistono numerose categorie e famiglie di malware, che non riportiamo in questo documento in quanto ciò esula dallo scopo dello stesso. Quando le signature dei vari strumenti di difesa, quali IDS/IPS o antivirus, non sono sufficienti a rilevare un'intrusione, risulta fondamentale individuare le modalità con cui è stato infettato il sistema, il malware distribuito e il comportamento di quest'ultimo allo scopo di capire come i software di difesa sono stati aggirati.

La *malware analysis* è l'arte di sezionare un programma per capire come funziona, come identificarlo, come sconfiggerlo o come rimuoverlo. Lo scopo della malware analysis è, solitamente, quello di generare informazioni per rispondere a una intrusione in rete. Infatti, una volta individuato il (o i) file malevolo(i), e una volta studiato il suo (loro) comportamento, si passa allo sviluppo di nuove signature che possano permettere in futuro il rilevamento di tale malware. Le signature possono essere di due tipologie:

- *Host-based signature*, o indicatori, quando vengono usate per rilevare codice malevolo su un computer vittima. Servono a rilevare file modificati o creati dal malware. Si basano su cosa un malware fa in un certo sistema piuttosto che su cosa il malware è. Sono utili per la tipologia di malware polimorfico o per quelli rimossi dall'hard disk.

- *Network signature*, quando vengono usate per rilevare codice malevolo monitorando il traffico di rete. Questo tipo di firme non sono fortemente legate alla malware analysis, ma quelle generate con il supporto di essa risultano essere largamente più efficienti, diminuendo anche il numero di falsi positivi.

In [35] gli autori suddividono l'analisi di malware in due macro categorie: l'analisi statica e l'analisi dinamica. Ciascuna di esse può essere fatta in maniera basilare o avanzata, e ciascuna di esse viene svolta con particolari strumenti.

Solitamente ciò che ha a disposizione un analista è un eseguibile: si cerca di ottenere tutte le informazioni possibili a partire da questo sfruttando tool e tecniche particolari.

2.1.1 Basic Static Malware Analysis

L'analisi statica di malware ha come oggetto di studio il codice e la struttura di un programma in modo da riuscire a risalire alla sua funzione. Tipicamente in questa fase vengono messe in atto tre tecniche per avere una prima idea sull'eseguibile:

- lo scanning di antivirus;
- l'hashing del file;
- l'estrazione di stringhe nonché funzioni e analisi dell'header di file.

L'analisi preliminare che può essere fatta per capire se ci si trova di fronte a qualcosa di malevolo è quella tramite antivirus. Ciò è utile per capire in breve tempo se il file in esame sia stato analizzato in passato. Gli antivirus si basano su due aspetti: la signature del file e le euristiche. Il problema sta nel fatto che gli scrittori di malware, modificando il codice o riscrivendolo da zero, riescono a bypassare le scansioni di antivirus. Una soluzione potrebbe essere quella di eseguire scansioni su molteplici antivirus differenti, dato che ciascuno di essi lavora con un proprio database, proprie euristiche e signature. Il sito web Virustotal fornisce un servizio del genere.

Il secondo metodo utilizzato per identificare un file malevolo è quello che sfrutta l'hashing. Tramite un algoritmo di hashing (per esempio, MD5 o SHA-1) viene generato un digest, ovvero una stringa alfanumerica, che serve a identificare il file in modo univoco. La firma del file appena creata può essere, a questo punto, confrontata con quelle di malware noti e condivisa con altri analisti di sicurezza.

Un ulteriore modo per ottenere informazioni utili su un file è quello di trovare tutte le stringhe contenute in esso. Questo perché se nel codice è stato inserito un indirizzo IP, piuttosto che un URL o il nome di una funzione, si può, così facendo, ottenere degli indizi sul comportamento del malware. La tecnica di ricercare le stringhe all'interno di un eseguibile può essere, tuttavia, aggirata tramite l'offuscamento o l'impacchettamento. Altre informazioni preziose che possono essere raccolte tramite un'analisi statica sono quelle derivanti dallo studio dell'header del file. Per esempio, Windows utilizza il formato PE (Portable Executable) per i file eseguibili, DLL, e codice oggetto. L'header dei file PE sono delle strutture dati che comunicano al sistema operativo come deve essere eseguito il file. Esso, in una parte iniziale, è caratterizzato da informazioni sul codice, le funzioni di libreria richieste, i requisiti di spazio di memoria e così via.

Un'altra fonte preziosissima di informazioni su un file eseguibile è costituita dall'insieme di librerie che esso utilizza. Molto spesso, infatti, i programmi utilizzano delle funzioni molto comuni che sono definite in altri file. Le librerie devono essere importate e ciò può essere fatto staticamente, dinamicamente o runtime. Solitamente gli autori di malware offuscato utilizzano il linking a runtime: Microsoft, per esempio, mette a disposizione delle funzioni, come *LoadLibrary* e *GetProcessAddress* che permettono ai processi di accedere a qualunque funzione di libreria nel sistema. Il metodo più utilizzato, tuttavia, sembra essere quello del linking dinamico, cioè nel momento in cui il sistema operativo carica in memoria il programma. L'header del PE contiene la lista di tutte le librerie che verranno caricate, e ciò fornisce ulteriori indizi sul comportamento dell'eseguibile.

Quella statica consiste solamente nella prima fase dell'analisi di un malware, fornendo indizi e informazioni più o meno utili sul comportamento del file in esame, senza che esso venga eseguito.

2.1.2 Advanced Static Malware Analysis

L'analisi statica descritta nella sottosezione precedente rappresenta solo una fase iniziale e abbastanza superficiale che può essere effettuata su un sospetto malware. Per esempio, è solamente possibile ipotizzare in che modo una certa funzione importata venga utilizzata, ma non si ha l'evidenza certa di come essa sia realmente usata. Se si vuole approfondire l'analisi, sempre evitando di eseguire il file, si deve utilizzare la tecnica del reverse engineering.

Quando un malware, o in generale un qualsiasi file eseguibile, si trova nel disco, esso è salvato in codice binario e risulta illeggibile. Il processo di reverse engineering, o disassembly, è necessario per poter ottenere codice assembly a partire dal binario. L'assembly è il linguaggio di programmazione di più basso livello. In rete si trovano molti strumenti per il disassembly di file in formato PE, ELF (Executable and Linking Format), COFF (Common Object File Format), come IDA Pro, a pagamento, o Ghidra, sviluppato dall'NSA americana, di cui è stato pubblicato recentemente il codice sorgente.

2.1.3 Basic Dynamic Malware Analysis

Come suggerisce il nome, con questa tecnica si studia il comportamento di un file eseguendolo. Solitamente rappresenta il secondo step dell'analisi di un malware, dopo aver raccolto il maggior numero di informazioni possibili con le tecniche e i tool dell'analisi statica.

Eseguendo il malware, si può osservare realmente il suo comportamento e quali effetti produce in una macchina. Avere individuato una stringa particolare o il nome di una funzione sospetta, non implica che essa sia veramente eseguita. Si può intuire, dunque, che questo modo di analizzare i file sia molto potente; di contro, però, se non viene ben pensato e architettato, potrebbe portare a dei rischi. Inoltre, come spesso accade, se da un lato viene implementata una nuova misura di sicurezza, dall'altro viene ideato il modo per evadere tale misura: anche in questo caso, se non ben fatta, l'analisi dinamica può risultare inefficiente a causa di tecniche di evasione.

Per evitare di mettere a rischio il proprio sistema, gli analisti di malware utilizzano le cosiddette *Sandbox*. Essi non fanno altro che creare degli ambienti chiusi, limitati e controllati, nel quale eseguire un file sospetto e controllarne il comportamento. In questo modo il sistema esterno alla sandbox risulta protetto. Una sandbox può essere una macchina virtuale con o senza connessione Internet (nel caso si voglia mantenere la connessione, si dovrebbero usare delle accortezze per evitare di mettere a rischio la macchina host) o una macchina reale. Entrambi i casi presentano vantaggi e svantaggi: per esempio, è molto semplice e veloce creare sandbox virtuali, ma l'analisi potrebbe essere meno potente, mentre è più rischiosa l'analisi su macchine reali.

Le informazioni che possono essere raccolte tramite l'analisi dinamica di malware vanno dall'utilizzo della rete (dato che spesso programmi malevoli devono comunicare con dei server di comando e controllo), al monitoring dei processi in esecuzione e delle DLL da essi importate, alle modifiche dei registri del sistema.

Tuttavia anche l'analisi dinamica di malware presenta dei punti di debolezza, perciò è sempre consigliabile mettere insieme i risultati di più tecniche in modo da avere un buon quadro d'insieme.

In generale, i punti di debolezza di questa tipologia di analisi sono i seguenti:

- Gli scrittori di malware hanno pensato a delle tecniche di evasione delle sandbox; perciò spesso i programmi malevoli sono in grado di riconoscere in quale ambiente si trovano e decidere se eseguire certe parti di codice o meno (le tecniche sono anche molto ingegnose, per esempio controllare la cronologia del browser per vedere se essa è vuota o meno, attendere degli input del mouse o della tastiera e così via).
- Alcuni malware potrebbero richiedere la presenza di certe chiavi di registro, o file, per essere eseguiti, che una macchina virtuale potrebbe non avere.
- La sandbox deve avere il sistema operativo giusto: alcuni malware sono scritti per particolari sistemi operativi e, se testati su altri, potrebbero dare risultati non confortanti.
- Durante l'esecuzione del file, una sandbox raccoglie evidenze che permettono di ricostruire il comportamento dell'artefatto analizzato; tuttavia non dà informazioni su che tipo di malware sia. Ciò spetta all'analista a valle di tutte le analisi fatte.
- Infine è chiaro come l'analisi di un file tramite sandbox sia time-consuming; per particolari tipologie di analisi potrebbe essere richiesto anche molto tempo, e non è detto che il risultato sia incoraggiante. Per esempio, alcuni malware implementano dei meccanismi di attesa prima di mostrare il loro aspetto malevolo, in modo tale da risultare "puliti" se eseguiti su ambienti controllati.

2.1.4 Advanced Dynamic Malware Analysis

L'analisi avanzata dinamica di malware probabilmente è, tra quelle descritte in questo documento, la più approfondita di tutte. Questa viene fatta tramite i debugger. Essi permettono di ispezionare nel modo più dettagliato l'esecuzione di codice, in quanto consentono di definire dei *breakpoint* dove bloccare il processo e forniscono informazioni riguardo i valori delle variabili ad ogni istruzione. È possibile, dunque, seguire l'esecuzione di un programma istruzione dopo istruzione e capire in che

modo esso arrivi al risultato finale. I debugger offrono una visione interna dei programmi mentre essi vengono eseguiti. Tramite esse possono essere ispezionati i valori di qualsiasi locazione di memoria, registro o argomento passato a una funzione.

Il debugging può essere effettuato in due modi:

- A livello di codice sorgente, ed è quello che interessa soprattutto gli sviluppatori nella fase di scrittura del codice (infatti molti IDE includono, tra le proprie funzionalità, anche quella di debugging).
- A livello di codice assembly, chiamato "Low-level debugging". In questo caso si opera a livello di codice assembly, piuttosto che di codice sorgente; le modalità e peculiarità di questa tecnica sono del tutto analoghe a quelle che caratterizzano la tecnica precedente, ma interessa soprattutto gli analisti di malware in quanto spesso non hanno accesso al codice sorgente, ma sono in grado di ricavare il codice assembly con in reverse engineering.

In aggiunta il debugging può essere fatto in user mode o kernel mode. Nel primo caso, il debugger agisce su un sistema, che è lo stesso del dell'eseguibile analizzato. In kernel mode, il debugger agisce in due sistemi perchè c'è soltanto un kernel (se il kernel si trova in un breakpoint, nessuna applicazione potrebbe essere eseguita sul sistema). Perciò un sistema esegue il debugger e l'altro il codice da analizzare. È necessario configurare appositamente il kernel per essere "debuggato" e le due macchine (il debugger e il codice analizzato) devono essere connesse.

Si può scegliere se associare un debugger ad un programma in esecuzione o se far partire un programma già all' "interno" di un debugger: nel secondo caso il programma verrà caricato in memoria e l'esecuzione si bloccherà all'entry point.

Con un debugger, si ha il pieno controllo del programma analizzato. È possibile, per esempio, modificare il flusso di esecuzione del codice, definendo un breakpoint e modificando successivamente l'instruction pointer.

In questa sezione abbiamo riportato le tecniche fondamentali per l'analisi di malware secondo quanto espresso in [35]. Le prossime sezioni saranno dedicate alle scelte che abbiamo fatto per implementare una fase di dynamic malware analysis all'interno di un flusso di analisi di email, a sua volta descritto nei capitoli successivi.

2.2 Cuckoo Sandbox

Come descritto nell'introduzione, lo scopo della presente tesi è quello di cercare di automatizzare il più possibile le attività di un analista di un Security Operation Center in modo da rendere più efficiente il flusso di lavoro dell'intera struttura.

All'interno di un SOC, è particolarmente importante l'attività di *Cyber Threat Hunting*. Quest'ultima è stata già introdotta nel definire l'operatore *Tier 3* e consiste nel ricercare in maniera proattiva le possibili minacce nella rete che si difende onde evitare attacchi.

Come già ampiamente sostenuto nella parte introduttiva, la maggior parte degli attacchi informatici, anche dei più sofisticati, inizia con il phishing, o spear phishing, utilizzando le email come vettore di malware. Si invita un utente a scaricare allegati o

a cliccare su link che infettano la macchina e danno inizio all'attacco. È importante, dunque, controllare le email e gli eventuali allegati in modo da prevenire danni.

Nell'ambito della computer security, le sandbox sono un efficace strumento di dynamic malware analysis; esse permettono di capire la natura di file o link in pochi minuti.

Nel mercato ci sono molteplici soluzioni a pagamento che, a seconda del prodotto e del pacchetto acquistato, offrono protezione per i vari sistemi operativi con analisi più o meno approfondite. Esistono anche versioni open source, la più nota delle quali è chiamata *"Cuckoo Sandbox"*[3]. Questo prodotto nasce come progetto di alcuni appassionati durante la *Google's Summer of Code* del 2010. Col passare del tempo, la community di sviluppo si è allargata e, ad oggi, Cuckoo rappresenta la principale soluzione di dynamic malware analysis open source esistente. Cuckoo è completamente scritto in Python 2.7 e il 19 giugno del 2019 è stata rilasciata l'ultima versione: la 2.0.7. Esso viene definito come un sistema di analisi di malware automatico, avanzato, completamente modulare e open source. Esso permette di eseguire analisi di file o URL in modo indipendente. In alternativa, è possibile integrarlo all'interno di un workflow più complesso che prevede una fase di analisi di observable.

Le capacità di questo strumento sono le seguenti:

- analisi di una grande varietà di tipologie di file e siti web sotto Windows, Linux, MacOS e gli ambienti virtuali di Android;
- tracciamento delle API call e del comportamento del file;
- raccolta e analisi del traffico di rete, anche cifrato, generato durante l'analisi;
- analisi avanzata dell'intera memoria (o di quella di un singolo processo) della macchina infetta, grazie al framework Volatility.

2.2.1 Architettura del sistema

L'architettura di Cuckoo è riportata nella Figura 2.1.

Come si evince dalla figura, sono presenti un unico orchestratore centrale nella macchina host e un insieme di macchine guest, virtuali o meno, connesse all'host con una rete virtuale. L'orchestratore rappresenta il core centrale dell'applicazione, essendo il responsabile della gestione dell'analisi, dell'avvio delle macchine guest ogniqualvolta avviene la submission di un job, dell'analisi dei dati ricevente al termine dell'esecuzione del file e, infine, della produzione di report. La submission di un file o di un URL può essere fatta in tre modi: *(i)* per mezzo dell'interfaccia web, sviluppata in Django; *(ii)* tramite richieste da altre applicazioni che sfruttano le API di Cuckoo; *(iii)* da riga di comando con l'apposita utility.

Le macchine guest sono un ambiente pulito e controllato, virtuale o meno, dove viene eseguito il presunto malware. La comunicazione con l'host avviene tramite un agent, installato sulla macchina guest ed eseguito all'avvio di questa con privilegi di amministrazione. L'agent funge da server in attesa delle richieste HTTP dell'orchestratore centrale. Nello specifico, quando esso riceve una richiesta dall'host, prepara l'ambiente per l'esecuzione dell'analisi, memorizzando in cartelle temporanee i file che l'host invia, manda in esecuzione l'analizzatore che l'host ha precedentemente recapitato al guest e, una volta conclusa l'esecuzione, restituisce all'orchestratore i

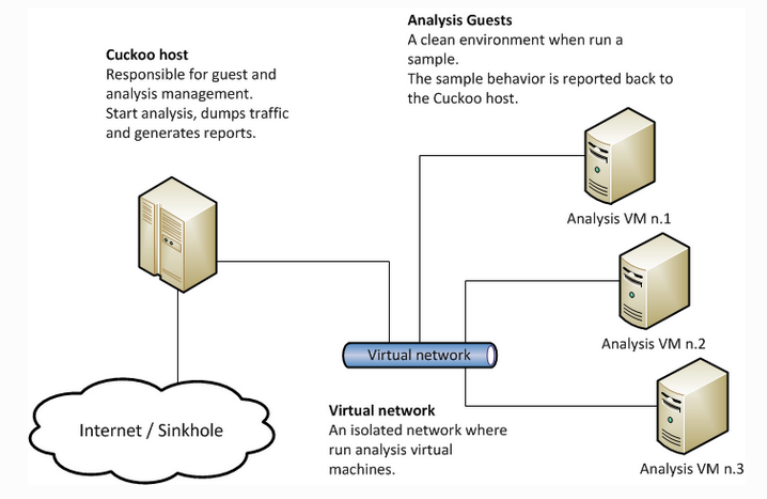

Figura 2.1. Architettura di Cuckoo Sandbox

log e i dati di comportamento prodotti. Il task termina con la fase di processing delle informazioni, con la valutazione delle signature sull'output della fase precedente e con il reporting dei risultati. Queste ultime tre azioni vengono eseguite dall'istanza di Cuckoo nella macchina host.

Da documentazione, è consigliato eseguire l'applicazione in host con sistema operativo GNU/Linux, preferibilmente Debian o Ubuntu. È raccomandato, invece, utilizzare macchine guest con Windows XP, Windows 7, MacOS X Yosemite e Debian. Tuttavia, poiché il framework è altamente modulare, estensibile e open source, chiunque può produrre moduli per lavorare anche con macchine guest aventi altri sistemi operativi.

2.2.2 Configurazione di Cuckoo

Esistono diversi modi con cui installare Cuckoo. Il più semplice e rapido tra questi è usare l'installatore ufficiale di Python, chiamato `pip`[1]. Poiché Cuckoo opera in contemporanea con altri software che potrebbero utilizzare diverse versioni di Python o di moduli, è consigliato generare un ambiente virtuale, scaricare tutte le dipendenze di Cuckoo ed eseguirlo all'interno di questo.

```
1    (venv) $ pip install -U cuckoo
```

Listato 2.1. Comando per l'installazione di Cuckoo Sandbox

Il risultato di questo comando è una cartella chiamata `cuckoo` aggiunta ai `site-packages` di Python2.7. Tale cartella contiene:

[1] Durante lo studio e il testing delle funzionalità di Cuckoo, abbiamo utilizzato un computer con Ubuntu 18.04; pertanto tutti i comandi che verranno inseriti sono riferiti a questo sistema operativo.

- il pacchetto delle utiliy;
- il pacchetto dei moduli ausiliari;
- il pacchetto del core dell'orchestratore;
- la cartella con i file che costituiranno le CWD;
- il pacchetto di machinery;
- il pacchetto di processing;
- il pacchetto di reporting;
- il pacchetto dell'interfaccia web.

Dopo aver scaricato Cuckoo, è necessario creare una Cuckoo Working Directory (CWD). Questa è la cartella dove vengono inseriti tutti i file con cui una particolare istanza di Cuckoo lavora. Al suo interno si trovano:

- i file di configurazione;
- i file di log;
- i file di monitor (di cui si discuterà in seguito);
- le signature utilizzate;
- gli analyzer;
- l'agent;
- le regole Yara;
- lo storage dove vengono memorizzate tutte le analisi e i risultati ottenuti;
- altri file di configurazione, per esempio di ElasticSearch o dell'interfaccia web.

Con il comando del Listato 2.2 si genera una nuova CWD. Se non si specifica il path con il parametro `--conf`, essa verrà creata nella home dell'utente corrente.

```
1   (venv) $ cuckoo init
```

Listato 2.2. Comando per la creazione della CWD

Il vantaggio principale di lavorare con le CWD è quello di poterne generare molteplici (ovviamente su path differenti) e configurare diversamente ciascuna di esse. In questo modo, si può passare facilmente da una configurazione all'altra senza dover ogni volta modificare i file, che, come vedremo più avanti sono molti. È sufficiente specificare,tramite l'apposito parametro, quale CWD utilizzare quando si lancia un'istanza di Cuckoo.

Un ulteriore passo preliminare all'avvio di Cuckoo è quello di scaricare tutte le signature che la comunità mette a disposizione. Ciò viene fatto con il comando mostrato nel Listato 2.3.

```
1   (venv) $ cuckoo community
```

Listato 2.3. Comando per il download delle signature dalla branch master

A questo punto, l'ultimo passaggio è quello di configurare opportunamente i file contenuti nella CWD.

FILE	DESCRIZIONE
auxiliary	In questo file si trovano le configurazioni di moduli ausiliari, cioè quei moduli che vengono eseguiti concorrentemente all'analisi del malware
avd	Contiene le configurazioni dell'emulatore Android (Android Virtual Device)
cuckoo	Questo file contiene le configurazioni dell'istanza di Cuckoo che si sta per eseguire
esx, kvm, qemu, virtualbox, vmware, vsphere, xenserver	Questi file contengono le configurazioni degli hypervisor che si possono utilizzare con Cuckoo
memory	Cuckoo permette di utilizzare il framework Volatility per l'analisi avanzata della memoria della macchina guest. In questo file è possibile configurare Volatility
physical	File usato per la configurazione delle macchine reali
processing	Con questo file è possibile abilitare o disabilitare i moduli di processing
reporting	File con il quale si possono settare le modalità in cui avviene il reporting dell'analisi
routing	In questo file è possibile configurare le impostazioni di rete della macchina guest

Tabella 2.1. File di configurazione di Cuckoo Sandbox

La cartella `conf` della CWD, contiene quindici file di configurazione (tutti con estensione `.conf`). La Tabella 2.1 riporta tali file e, per ciascuno di essi, riassume le funzioni.

Una volta aver configurato in maniera opportuna i file, è possibile eseguire un'istanza di Cuckoo con il comando mostrato nel Listato 2.4

```
1   (venv) $ cuckoo
```

Listato 2.4. Comando per l'avvio di un'istanza dell'orchestratore centrale di Cuckoo

È possibile lanciare il comando specificando diversi parametri, come quello per il debug o quello per specificare il path della CWD da utilizzare.

A questo punto l'istanza di Cuckoo rimane in attesa della submission di un file o URL con uno dei modi specificati precedentemente. Ovviamente, se la submission avviene per mezzo delle API, è necessario aver mandato in esecuzione l'apposita utility, lo stesso avviene se si utilizza l'interfaccia web. L'ultima alternativa è quella di eseguire in locale il comando nel Listato 2.5. È possibile specificare diversi flag e parametri per configurare l'analisi nella maniera desiderata.

```
1   (venv) $ cuckoo submit
```

Listato 2.5. Comando per lanciare l'analisi di un file o URL in locale.

La Tabella 2.2 riporta tutte le utility di Cuckoo e una breve descrizione della loro funzione.

UTILITY	DESCRIZIONE
api	Esegue le API di Cuckoo
clean	Ripulisce il CWD e il database associato
community	Scarica le signature della community
distributed	Comando per distribuire le utility di Cuckoo
dnsserve	Stabilisce un server DNS custom
import	Importa un vecchio setup in una nuova CWD
init	Inizializza Cuckoo creando una CWD
machine	Aggiunge o rimuove dinamicamente una macchina guest
migrate	Permette la migrazione di database
process	Processa dati grezzi e genera report (nel caso in cui si vogliano, appunto, generare i report in un momento differente rispetto alla fine dell'analisi)
rooter	Istanzia il Rooter di Cuckoo per gestire la rete delle macchine guest
submit	Esegue il submit di un observable
web	Esegue l'interfaccia web di Cuckoo

Tabella 2.2. Lista delle utility di Cuckoo

2.2.3 Workflow

All'atto della sottomissione di un file o URL, Cuckoo avvia lo snapshot di una macchina virtuale tra quelle a disposizione tramite il modulo di machinery relativo alla soluzione di virtualizzazione scelta. Successivamente, viene inviato alla macchina virtuale il pacchetto dell'analyzer, specifico per il sistema operativo della macchina virtuale, contenente il file con il codice che esegue il sospetto malware a seconda del tipo di file stesso (Python, exe, DLL e così via).

Durante l'esecuzione vengono generati file di log, con estensione ".bson", contenenti i dati grezzi raccolti sul comportamento del sistema, gli screenshot dello schermo della macchina, se questa funzione è stata attivata, il dump del traffico di rete, il dump della memoria dei processi monitorati e/o dell'intera memoria della macchina, se specificato.

Una volta terminata l'esecuzione del file, vengono inviate all'orchestratore centrale tutte queste informazioni, viene spenta la macchina guest e viene eseguito nella macchina host il processo di analisi dei dati raccolti con i moduli di processing abilitati in fase di configurazione. Questi ultimi sono file Python che eseguono dei task di analisi, come il modulo `strings` che estrae dal file tutte le stringhe presenti, il modulo `network`, che analizza il file `.pcap` generato dallo sniffing del traffico di rete della macchina host, o il modulo `memory` che sfrutta le API di Volatility per l'analisi del dump della memoria, solo per citarne alcuni.

La fase successiva prevede il controllo delle signature: vengono confrontati i dati di output della fase precedente con le signature scaricate con il comando del Listato 2.3. Ciascuna di esse è caratterizzata da un valore di *severity*, generalmente compreso tra uno e tre (ma c'è la possibilità che alcune signature utilizzino anche valori maggiori).

A conclusione di tutto il processo di analisi c'è la fase di reporting. Il report può essere prodotto in diversi modi, a seconda di come si è popolato il relativo file di configurazione. Usualmente, oltre ad essere mantenuti su un database SQLite, i risultati vengono riportati su un file in formato `.json`. Ci sono, comunque, molte altre alternative, tutte utilizzabili contemporaneamente:

- report HTML;
- report PDF;

- aggiornamento dati su ElasticSearch;
- aggiornamento delle IOC della piattaforma MISP.

Il report riassume tutte le informazioni che sono state generate dall'analisi. Di maggiore interesse rispetto a tutte le altre sono le voci relative allo *score*, che indica quanto sia malevolo il file, e le signature rilevate. Lo score finale viene calcolato come rapporto tra la somma dei valori di severity di tutte le signature rilevate e cinque. Tuttavia, a volte non è sufficiente basarsi sul valore dello score per capire la natura di un file o di un URL, ma è necessario anche valutare quali signature sono state rilevate. Per esempio, un file rilevato da poche signature "critiche" potrebbe avere un basso score ma un alto impatto malevolo se eseguito.

2.3 KVM, QEMU e Libvirt

Cuckoo Sandbox permette di utilizzare uno dei seguenti sistemi di virtualizzazione:

- VirtualBox;
- Vmware;
- QEMU;
- KVM;
- AVD (per i sistemi Android);
- ESX
- XenServer
- Vsphere

Dovendo lavorare su una macchina con sistema operativo Ubuntu 18.04, abbiamo deciso di utilizzare lo stack di virtualizzazione KVM, QEMU e libvirt, essendo tutti fortemente integrati con Linux.

La Figura 2.2 riassume come queste tre componenti sono strutturate all'interno di un computer con sistema operativo Linux.

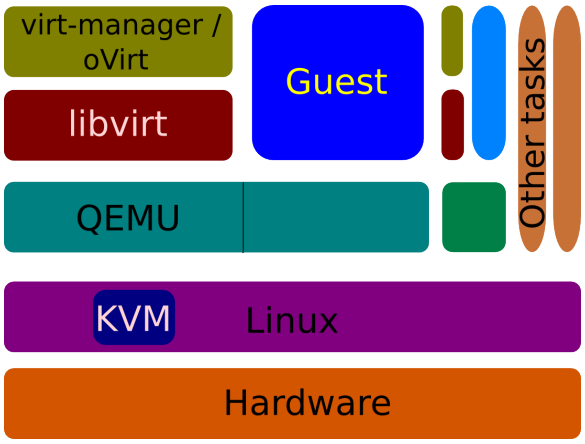

Figura 2.2. Stack di virtualizzazione

Come descritto in [22], in cima allo stack si trova l'applicativo con cui si interfaccia l'utente, per esempio *virt-manager*. Questo software fornisce un'interfaccia grafica per la gestione di macchine virtuali. Dovendo fare numerose prove per valutare le funzionalità e i risultati di Cuckoo, è stato necessario modificare molte volte l'ambiente virtuale, e *virt-manager* ci ha permesso di creare macchine rapidamente, di avviarle e di acquisirne snapshot. In alternativa, è possibile utilizzare *virsh* come interfaccia utente da riga di comando.

Sia *virt-manager* che *virsh* interagiscono con *libvirt*, che rappresenta un'interfaccia di programmazione compatibile con molteplici hypervisor. Nel caso specifico dello stack QEMU/KVM, *libvirt* si interfaccia con le API messe a disposizione da QEMU (Quick EMUlator). Sul sito ufficiale [12], quest'ultimo viene definito come un "generico emulatore e virtualizzatore di macchine open source". Se si utilizza questo stack, l'hardware virtuale che viene creato dipende dalle sue configurazioni (dalla Figura 2.2 è possibile notare come il guest si trovi proprio al di sopra dello strato di QEMU). La macchina virtuale viene lanciata come processo QEMU, per esempio con il comando `qemu-system-x86_64` per sistemi x86 a 64 bit. Esso utilizza diversi servizi del sistema operativo host Linux, come i servizi di rete, servizi di storage o le API di KVM per il controllo della macchina guest. QEMU può operare in diverse modalità:

- *User-mode emulation*, per eseguire programmi che erano stati pensati e compilati per un diverso instruction set da quello utilizzato dall'host.
- *System-mode emulation*, dove viene emulato un computer interamente, periferiche incluse. In questa modalità, QEMU può avviare differenti sistemi operativi ed emulare diversi instruction set.
- *KVM hosting*, per una virtualizzazione vera e propria con KVM a fare da hypervisor e con prestazioni "near native".

QEMU utilizza la tecnica del dynamic binary tranlsation per tradurre le istruzioni da un instruction set di partenza a quelle dell'instruction set target a tempo di esecuzione.

Per concludere la descrizione dello stack, alla base di esso, proprio sopra l'hardware, si trova KVM. L'acronimo sta per Kernel-based Virtual Machine e consiste in un modulo del kernel di Linux, `kvm.ko`, che fornisce l'infrastruttura centrale di virtualizzazione, e un modulo specifico per l'hardware, `kvm-intel.ko` o `kvm-amd.ko`. KVM permette di convertire il kernel di Linux in hypervisor. Esso espone delle API allo user space e QEMU è proprio uno degli utilizzatori di queste. Insieme KVM e QEMU ottengono buone prestazioni di virtualizzazione con tempi di esecuzione di istruzioni simili a quelli su sistemi nativi.

Affinché sia possibile utilizzare gli strumenti appena descritti, l'hardware sottostante deve supportare la virtualizzazione.

2.3.1 Virtualizzazione in Cuckoo Sandbox

Cuckoo Sandbox utilizza dei moduli di "machinery" per interagire con le macchine virtuali. Si possono utilizzare i moduli elencati nella Sezione 2.3, oppure in alternativa, come per qualsiasi altra tipologia di moduli, è possibile crearne dei nuovi: essi devono contenere la definizione di una classe, che estende la classe `Machinery`

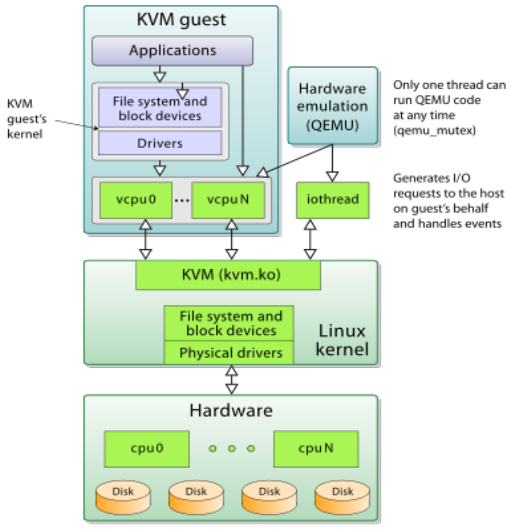

Figura 2.3. Interazioni tra KVM e QEMU

o `LibVirtMachinery`, dove viene fatto l'override dei metodi di `start` e `stop`. Una volta aggiunto il nuovo modulo nel pacchetto `Machinery`, si devono inserire le informazioni relative alla macchina virtuale nel relativo file di configurazione, tra cui il nome dello snapshot da avviare, se richiesto.

Durante i test, inizialmente abbiamo deciso di utilizzare il modulo di QEMU, che estende la classe generica `Machinery` e, quindi, non è supportato da *libvirt* e KVM. In un secondo momento, abbiamo deciso di passare a KVM poiché alcune funzionalità avanzate, come la creazione del dump di memoria e la conseguente analisi con i plugin di Volatility in fase di processing, sono attualmente sviluppate solamente per KVM (e i moduli di machinery che utilizzano *libvirt*, cioè che estendono la classe `LibVirtMachinery`) e VirtualBox.

Quando un'istanza di Cuckoo in attesa di una submission riceve un nuovo task, la prima operazione che viene effettuata è quella di avviare la macchina virtuale selezionata tramite un suo snapshot. Uno snapshot è la "fotografia" dello stato della memoria centrale della macchina in un certo istante. Usare uno snapshot per avviare una macchina virtuale è molto più rapido di avviarla normalmente; inoltre permette di avere dei processi desiderati in esecuzione direttamente all'avvio (è sufficiente catturare lo snapshot quando tali processi sono in esecuzione).

Affinché le analisi siano ottimali e forniscano una buona idea della natura di un file o URL, l'ambiente virtuale deve essere opportunamente preparato. Se, per esempio, sulla macchina virtuale gira il sistema operativo Windows, devono essere disabilitati il firewall, il sistema di Windows Defender, il sistema di protezione real-time e il sistema di controllo degli accessi (User Access Control - UAC). Per non "sporcare" lo sniffing del traffico di rete con pacchetti non direttamente associati all'esecuzione del file è necessario disabilitare tutti quei servizi, come Windows Update, che generano traffico sia in entrata che in uscita.

Configurazione di rete della macchina virtuale

Cuckoo offre numerose alternative con le quali configurare la rete delle macchine virtuali. La più elementare è quella di impedire alla macchina qualsiasi comunicazione con l'esterno. Il modo più semplice per fare ciò è quello di generare una rete virtuale isolata al momento della creazione della macchina, la quale permetta soltanto la comunicazione interna tra host e guest e impostare tale interfaccia nelle configurazioni della macchina stessa.

Questa soluzione, però, non è molto flessibile e, se la si vuole modificare, è necessario creare una nuova interfaccia di rete che utilizza, per esempio, il NAT. Per questo esiste l'utility *Rooter* che, dinamicamente, permette di effettuare il setting delle configurazioni di rete impostate nel file `routing.conf`. Le alternative sono: non permettere alla macchina guest alcun accesso ad Internet, permettere l'accesso alla rete attraverso una "dirty line", lasciando passare il traffico attraverso la macchina host, o utilizzare il servizio *InetSim* dirottando il traffico attraverso una rete che simula servizi Internet.

In aggiunta, è possibile utilizzare VPN o dirottare il traffico verso la rete TOR.

2.3.2 CAPE Sandbox

Durante i test di Cuckoo, abbiamo notato che i risultati ottenuti non erano sempre entusiasmanti, se confrontati con altre soluzioni di sandbox online. Una di queste si chiama *CAPE Sandbox* e permette di eseguire analisi di observable online in maniera completamente gratuita. CAPE, infatti, è un'estensione di Cuckoo ed è anch'essa open source. Abbiamo deciso, quindi, di migrare a CAPE Sandbox e di sfruttare le sue potenzialità, scaricandola da GitHub [13] e installandola in locale.

L'architettura rimane pressoché invariata e, per buona parte, il codice è lo stesso di quello di Cuckoo. Tuttavia CAPE estende le sue funzionalità con un maggior numero di moduli ausiliari, un maggior numero di signature, un maggior numero di moduli di processing e permette di analizzare più tipologie di file. C'è una diversa organizzazione dei file e un diverso agent, che da `SimpleHTTPServer` diviene un `SimpleXMLRPCServer`: nel primo caso, venivano usate le tradizionali richieste GET e POST del protocollo HTTP per eseguire funzioni dell'agent, nel secondo, si usa il linguaggio XML per la chiamata di procedure remote (RPC). CAPE elimina il concetto di CWD: l'analyzer (per Windows) e i file di configurazione sono unici. Come accennato, si specializza su Windows: il servizio online opera soltanto con macchine virtuali Windows 7 a 64 bit, ma in locale sembra funzionare anche con Windows 10. Mantiene più o meno la stessa possibilità di scelta di virtualizzazione, essendo stati rimossi soltanto i moduli di AVD e QEMU. Tuttavia, rispetto a Cuckoo, permette di utilizzare contemporaneamente diversi sistemi di virtualizzazione.

CAPE Sandbox è in grado di individuare un maggior numero di comportamenti, tecniche malevole e specifiche famiglie di malware. Esso è in grado di rilevare l'iniezione di processi, in uno dei possibili modi esistenti, la decompressione o l'estrazione di file eseguibili in memoria. Il payload decompresso, estratto o iniettato, viene ulteriormente analizzato da CAPE.

Il modo con cui viene calcolato lo score finale associato al file è differente. Precedentemente, per ogni signature rilevata, veniva indicato soltanto un valore di

severity; in CAPE si hanno i valori di *severity*, *weight* e *confidence*. La *severity* è un intero compreso tra 1 e 3 caratteristici della signature, i valori di *weight* e di *confidence* sono anch'essi interi, ma cambiano in base alle evidenze trovate. Il calcolo che porta al valore finale è il seguente:

$$malscore = \begin{cases} weight \cdot (severity - 1) \cdot (\frac{confidence}{100}), & severity = 2, 3 \\ weight \cdot 0.5 \cdot (\frac{confidence}{100}), & severity = 1 \end{cases}$$

Se il valore calcolato è maggiore di dieci, il malscore è posto a dieci, se è minore di zero, diventa zero.

Un ulteriore aspetto, non marginale, che ci ha spinto a migrare su CAPE è che esso sembra molto più mantenuto di Cuckoo dalla comunità di sviluppo: i commit su Github sono molto frequenti e gli ultimi molto recenti.

2.3.3 Vantaggi e svantaggi

Molteplici sono i vantaggi di utilizzare una soluzione di dynamic malware analysis come Cuckoo o CAPE. Il principale, ma che è in comune a tutte le soluzioni di sandboxing, è quello di riuscire a farsi un'idea sulla natura del file o dell'URL analizzato in una manciata di minuti. Altri vantaggi, propri di Cuckoo e CAPE, sono la loro natura open source e modulare. Questi due aspetti insieme permettono all'occorrenza di estendere il sistema con moduli personalizzati in base alle particolari esigenze. Possono essere prodotti moduli di qualsiasi genere, da quelli relativi alle signature ai moduli di reporting o processing. Possono essere prodotti nuovi analyzer per diversi sistemi operativi, o moduli per eseguire l'analisi di qualsiasi estensione di file. Uno degli aspetti negativi, tuttavia, è che il codice di entrambi è completamente scritto nella Versione 2.7 di Python, la quale dal primo gennaio del 2020 non è più supportata.

Un ulteriore vantaggio è quello di possedere moduli di processing che eseguono anche analisi tipiche della static malware analysis, come l'estrazione di stringhe dal file, o il controllo dello stesso su virustotal. In aggiunta, è garantita un'ampia scelta di hypervisor per le macchine guest e, all'occorrenza, è possibile utilizzare anche macchine reali. Un aspetto negativo è la necessità di eseguire molte configurazioni modificando gli appositi file. Inoltre, CAPE risulta essere uno strumento abbastanza complesso e difficilmente maneggiabile dai non addetti ai lavori o con poca esperienza nel settore.

Un ultimo, ma non per importanza, svantaggio che abbiamo notato nell'eseguire dei test è che entrambi permettono analisi soltanto su macchine guest con Windows, perché su Cuckoo, per Linux il pacchetto analyzer non è completo e non sono stati scritti i package associati alle varie tipologie dei file (doc, pdf, xls e così via); su CAPE non esiste proprio il pacchetto di altri sistemi operativi.

Per quanto riguarda le prestazioni, non ci siamo concentrati molto nel valutare i tempi di esecuzione delle analisi, in primo luogo perché essi variano molto tra loro, anche sottomettendo lo stesso file; in secondo luogo, perché esse dipendono, anche, dal numero di moduli di processing o reporting che si eseguono. Per esempio, se venisse utilizzato Volatility, sicuramente l'analisi risulterà rallentata.

Sotto questo punto di vista, l'utilizzo del linguaggio Python non è di aiuto poiché il GIL impedisce l'esecuzione di più thread parallelamente. Se, per esempio, si

sottomettono due o più file nello stesso momento, verranno generati altrettanti task in maniera sequenziale; se l'architettura del calcolatore lo permette, e se sono state configurate molteplici macchine virtuali, l'esecuzione dei file sulle singole macchine guest può avvenire parallelamente, ma le fasi di processing e reporting tornano a essere sequenziali in quanto eseguite dall'unico orchestratore centrale. Per ovviare a questa che potrebbe diventare una problematica su architetture con grandi carichi di lavoro, è possibile avviare e distribuire più istanze.

2.4 Conclusione

In questo capitolo sono riportate le tecniche di analisi di malware statiche e dinamiche. L'analisi del malware rappresenta un passo fondamentale nel ciclo di vita di risposta (e prevenzione) agli incidenti; per questo più informazioni produce, maggiori saranno i benefici.

Non esiste ad oggi una tecnica all-in-one, o modalità di analisi automatica, che permetta di ottenere tutte le informazioni desiderate di un file. Le analisi statiche e dinamiche andrebbero utilizzate insieme in quanto i loro risultati favoriscono e apportano vantaggi le une alle altre. Per esempio, utilizzare solo l'analisi dinamica potrebbe non bastare perché non è detto che in quella particolare analisi il programma mostri tutti i suoi comportamenti. Il comportamento malevolo potrebbe non essere "triggerato" e rimanere nascosto. Al contrario, analizzare un eseguibile senza vederlo realmente in azione potrebbe non essere sufficiente.

In questo lavoro, lo scopo è quello di cercare di prevenire tentativi di phishing analizzando in maniera automatica le email in arrivo agli utenti di un sistema di web mail. È chiaro, dunque, che, in questo contesto, non è possibile analizzare approfonditamente tutte le email in arrivo e i loro allegati prima che l'utente possa aprirle: sarebbe molto dispendioso in termini di tempo e potrebbe causare il degrado dell'efficienza dei processi di lavoro dell'utente stesso. Per questo, è sembrato più opportuno eseguire soltanto un'analisi dinamica preliminare con sandbox, che permetta di capire in pochi minuti se le nuove email siano genuine o se sia meglio evitarne l'apertura ed eventualmente disporre ulteriori analisi più approfondite.

Quella descritta in questo capitolo, dunque, rappresenta una delle fasi che comporrà il flusso di analisi di email da implementare all'interno di un Security Operation Center.

3

Security Incident Response: TheHive, Cortex, MISP

Questo capitolo descrive tre strumenti utilizzati e altamente integrabili tra loro che permettono di automatizzare e velocizzare la generazione di alert e la risposta ad eventuali incidenti: TheHive, Cortex e MISP. Inizialmente si offre una descrizione più teorica del loro contesto di riferimento mentre in una seconda fase, si passa ad analizzare la loro architettura e le loro funzionalità.

3.1 Introduzione

Tornando a un concetto espresso nel Capitolo 1, un Security Operation Center è una combinazione di persone, tecnologie e processi. Affinché svolga al meglio il proprio compito, è necessario che tutte e tre queste componenti siano ben strutturate e di qualità.

Per quanto riguarda le tecnologie, ci sono molti software open source o proprietari da utilizzare in un SOC, a partire dagli IPS/IDS, passando per i SIEM, gli EDR, gli applicativi per attività di forensics, di reverse engineering e di malware analysis, fino ad arrivare ai SOAR (Security Orchestration, Automation and Response); sicuramente, molti altri ne verranno prodotti in futuro. In tempi recenti, inoltre, l'Intelligenza Artificiale (con Machine/Deep Learning) e la Big Data Analytics stanno irrompendo anche nel settore della cybersecurity, rendendo sempre più efficienti le attività di cyber defence (oltre a quelle offensive).

Le persone sono, indubbiamente, centrali in un SOC: sebbene si stia cercando di automatizzare il più possibile i vari processi (e questo elaborato è proprio un tentativo di rendere automatiche alcune attività di un SOC che finora erano effettuate sequenzialmente e in maniera separata), la figura dell'analista risulta sempre centrale, per esempio nell'indagare sui falsi positivi, per individuare e ridurre i falsi negativi o nell'analisi di un malware. Non a caso, i SOC sono costituiti da risorse umane dotate di grande esperienza e di conoscenze molto approfondite nei vari ambiti dell'ICT.

Per finire, un SOC ben funzionante necessita di un'ottima progettazione e implementazione di processi, uno dei quali è proprio quello di orchestrazione della risposta agli incidenti, cosicché si possa reagire celermente e in maniera ottimale.

L'americana IBM ha riassunto in una tabella i vari stadi di maturità di risposta agli incidenti [36]. Questa tabella viene riportata in Figura 3.1.

Maturity level		Ad hoc		Maturing		Strategic
Existing capabilities		As needed	Dedicated part-time	Full-time	SOC/IR+	Fusion
	People	• 0–1	• 1–3 • specialization	• 2–5 • Formal roles	• ~10 • Shifts (possible 24x7)	• 15+ • Intel, SOC, and IR teams
	Process	• Chaotic and relying on individual heroics reactive • General purpose run book • Tribal knowledge	• Situational run books; some consistency • Email-based processes	• Requirements and workflows documented as standard business process • Some improvement over time	• Process is measured via metrics • Minimal threat sharing • Shift turnover • SLAs	• Processes are constantly improved and optimized • Broad threat sharing • Hunt teams
	Technology		• SIEM • Sandboxing	• Continuous monitoring • Endpoint forensics • Tactical intelligence	• Malware analysis • Additional intelligence • IT operations	• Intel+IR drives security program • Strategic intelligence • Coordination with physical security
CMM equivalent		Initial	Repeatable	Defined	Managed	Optimized

Figura 3.1. Modello di maturità della risposta agli incidenti

Il modello descritto mappa le varie fasi da seguire per passare da una capacità di difesa e risposta ad hoc, in cui la risposta risulta essere poco strutturata e coordinata, ad una capacità di risposta completamente coordinata, integrata e continuamente ottimizzata, testata e migliorata.

Un importante strumento a supporto della attività quotidiane degli operatori di un SOC sono le cosiddette Security Incident Response Platform (SIRP).

3.1.1 Cos'è un SIRP

Una piattaforma di Incident Response ha lo scopo di guidare le contromisure contro un breach e mettere in atto una serie di risposte automatiche pianificate in anticipo. Tramite questo tipo di piattaforme è possibile pianificare, orchestrare e generare report di incidenti di sicurezza in accordo con le best practice. I task di risposta ad un incidente possono includere attività di threat hunting, rilevamento di anomalie, risposta alle minacce il tempo reale.

Le piattaforme di IR forniscono un runbook, ovvero una serie di tattiche e metodi usuali, per contenere attacchi e rimediare a incidenti. Queste tattiche di risposta possono anche essere attivate in maniera del tutto automatica a seguito del rilevamento di una minaccia nota o dell'infezione di un dispositivo.

L'obiettivo delle SIRP è quello di minimizzare il tempo e le risorse necessarie per far fronte a incidenti.

A differenza delle piattaforme di Endpoint Protection, le quali rappresentano una prima linea di difesa con la quale bloccare minacce note, le SIRP sono utilizzate nella fase successiva, quando le prime linee hanno fallito il loro compito ed è

avvenuto un incidente: a quel punto è necessario estirparne le cause il più velocemente possibile, minimizzandone gli effetti. Questi due tipi di piattaforme stanno cominciando a fondersi insieme andando a costituire i più moderni EDR.

La piattaforma SIRP che abbiamo utilizzato per questo lavoro si chiama TheHive e fa parte di TheHive Project. Questa rappresenta un *Case Management System*, cioè un sistema che facilita le investigazioni di incidenti di sicurezza, permettendo di gestire, organizzare, coordinare le attività degli analisti di un SOC ed eventualmente di rispondere. TheHive Project è sviluppato da alcuni ricercatori del CERT BDF (CERT Banque De France), i quali hanno prodotto due componenti a se stanti: TheHive e Cortex, entrambi completamente open source, scaricabili, utilizzabili e ulteriormente sviluppabili. Ovviamente le due soluzioni sono altamente integrabili e possono essere utilizzate singolarmente, o insieme, in un flusso di attività di IR. I loghi delle due piattaforme sono mostrati in Figura 3.2.

Figura 3.2. Loghi di TheHive e Cortex

3.2 TheHive

Nel sito ufficiale [14] questo strumento viene definito come una piattaforma di risposta agli incidenti di sicurezza scalabile, open source e completamente gratuita, che riesce a semplificare e supportare il lavoro all'interno dei SOC, di CSIRT o di CERT.

Il backend del software è scritto in Scala; il frontend, fruibile tramite browser, è scritto in JavaScript con AngularJS e Bootstrap. L'ultima versione rilasciata è la Versione 3.4.0.

3.2.1 Funzionamento della piattaforma

La Figura 3.3 riporta l'architettura della piattaforma.

A sinistra si trova TheHive, diviso tra il frontend, il supporto di storage costituito da ElasticSearch e il backend, il quale comunica tramite richieste HTTP con l'eventuale istanza di Cortex. Non è indispensabile eseguire in contemporanea TheHive e Cortex, in quanto ciascuno dei due svolge il proprio specifico ruolo nell'IR; il primo permette di gestire il processo di risposta all'incidente, il secondo esegue task più pratici con Analyzer o Responder. Ciò nondimeno, la loro collaborazione permette di automatizzare molti task definiti in TheHive.

La Figura 3.4, invece, descrive il workflow della piattaforma.

Una volta scaricato, installato (con una delle modalità descritte nella documentazione [8]) e avviato TheHive, si deve creare un nuovo database ElasticSearch. Il

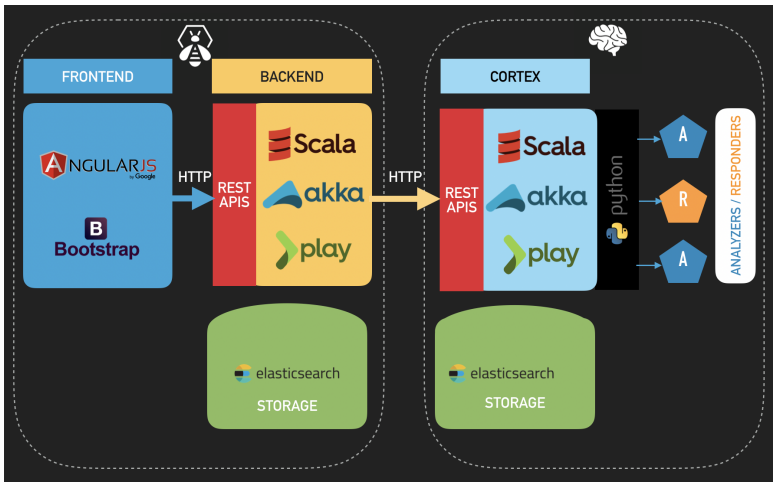

Figura 3.3. Architettura di TheHive

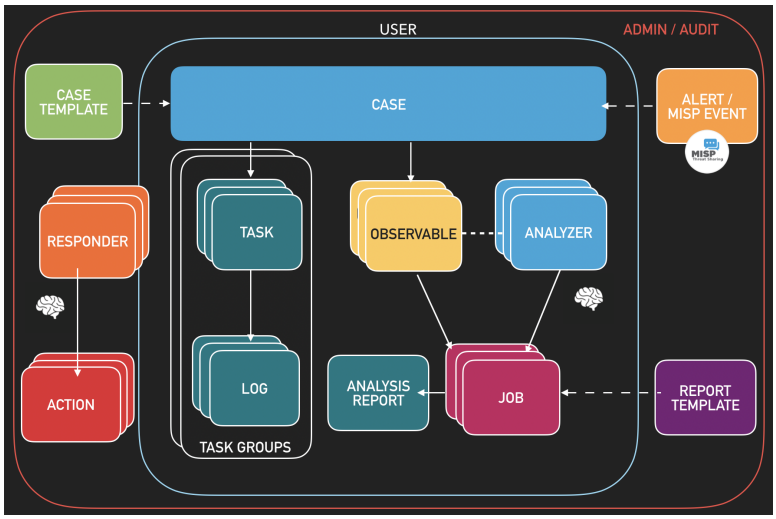

Figura 3.4. Workflow di TheHive

tutto è fatto in maniera automatica semplicemente cliccando il pulsante che compare nell'interfaccia web appena avviato il programma. Successivamente è necessario creare un utente amministratore fornendo username, nome e password. Una volta che si è acceduti, navigando sull'apposito pannello di amministrazione, si ha la facoltà di creare altri utenti. L'amministratore stabilisce la password con cui il nuovo utente può autenticarsi la prima volta e, se necessario, può generare la relativa APIkey.

A ciascun utente possono essere attribuiti permessi di lettura, scrittura, generazione di alert o privilegi di amministrazione. Per quanto riguarda la lettura, un utente con questo permesso può leggere tutti i dati non sensibili, ma non può apportare nessuna modifica. Per fare ciò, è necessario avere il permesso di scrittura, con

il quale si eredita anche quello di lettura. Se si hanno privilegi di amministrazione si eredita il privilegio di scrittura (e in cascata quello di lettura) e, in aggiunta, possono essere gestiti gli altri utenti. Infine, può creare alert solamente chi ha questo specifico permesso.

TheHive fornisce molteplici metodi di autenticazione:

- Active Directory;
- LDAP;
- API keys;
- X.509 SSO;
- OAuth 2;
- Autenticazione locale.

Il core principale di TheHive è l'investigazione dei *Case*. Questi possono essere generati direttamente dagli utenti o, in maniera automatica, da *Alert* o eventi di MISP, e riassumono informazioni su un possibile incidente avvenuto. Per informazioni si intende il nome del case, chi lo ha generato, la descrizione della causa che ha portato alla sua generazione, l'eventuale alert da cui è stato creato (se gli utenti decidono di non ignorare tale allarme), eventuali observable che meritano di essere analizzati. Ad un case si attribuisce un valore di *severity* (Low, Medium o High), si possono associare dei tag e il valore di TLP.[1] La Figura 3.5 mostra la sezione di TheHive in cui sono elencati tutti i case presenti nel database.

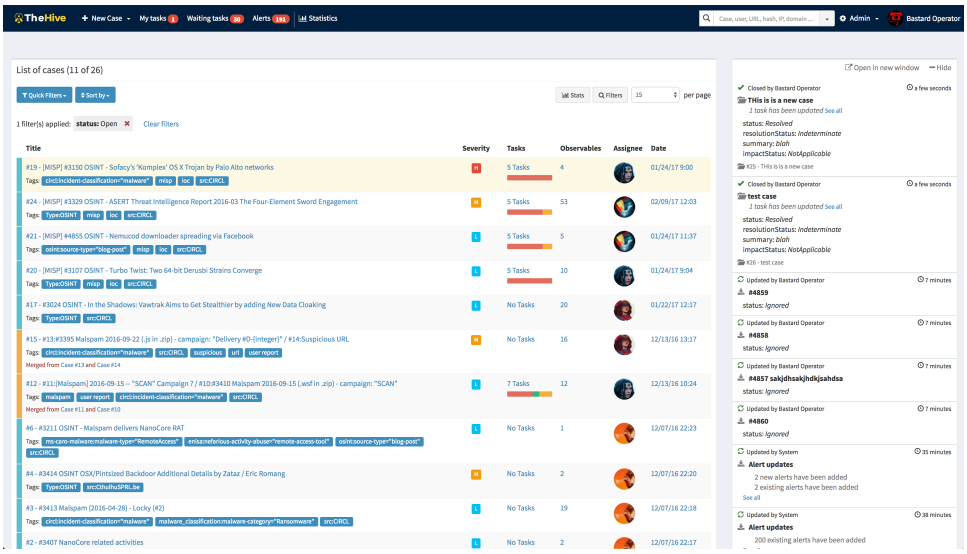

Figura 3.5. Pannello dei Case

[1] Il Traffic Light Protocol (TLP) è un protocollo sviluppato dall'NSCC (National Infrastructure Security Coordination Centre) per la condivisione di informazioni sensibili con l'audience appropriata. Esso definisce quattro colori (bianco, verde, giallo e rosso) ognuno dei quali stabilisce criteri di condivisione più o meno restrittivi.

L'analisi che porta alla chiusura di un case è scandita in task. Ciascuno di essi può essere preso in carico da un operatore differente.

Molto spesso, i case che vengono creati nella piattaforma hanno caratteristiche simili in quanto le cause che li generano sono simili (per esempio email di phishing o una serie di tentativi di login falliti). TheHive offre la possibilità di definire dei template in modo da associare ad un nuovo case una serie di informazioni prestabilite, tra cui l'insieme di task da completare per chiudere il case.

All'interno di un case possono essere inseriti gli observable. Questi sono indirizzi IP, file, HTTP URI o nomi di dominio e altre tipologie di artefatti, a cui possono essere associati dei tag, un valore di TLP, un messaggio di descrizione e altre informazioni. Ci sono molti benefici nel tracciare gli observable. Essi possono essere importati in blacklist o whitelist a seguito delle analisi. Inoltre, se è attiva anche un'istanza di Cortex, è possibile sottomettere gli observable agli *Analyzer* di quest'ultimo, in modo da ottenere dei report, in base ai quali eseguire eventuali *Responder*.

In Figura 3.6 è mostrato un case in dettaglio insieme a tutte le informazioni che lo riguardano.

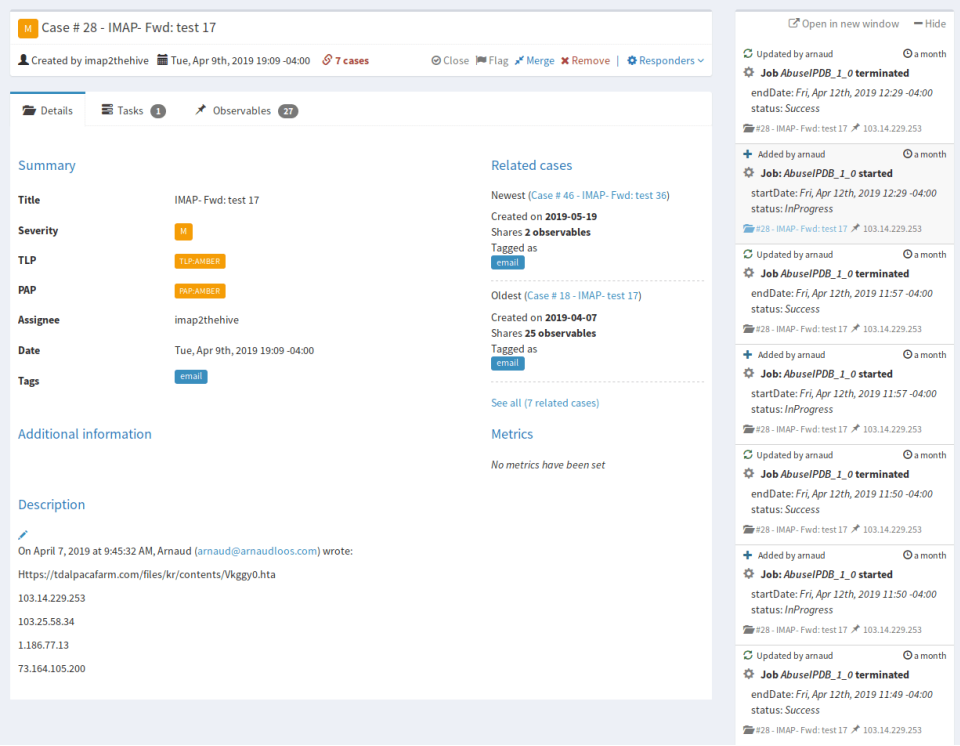

Figura 3.6. Esempio di un case

Si può notare come sono stati aggiunti 27 observable e un task. Inoltre, in maniera automatica, TheHive è in grado di identificare case simili e aggiungerli nella

colonna dei *Related cases*.

I task sono unità di lavoro da effettuare all'interno di un case. Un task può essere preso in consegna da qualsiasi analista che ne ha il permesso e, nel caso di non propedeuticità tra di essi, più task possono essere eseguiti in contemporanea. Una volta che tutti i task di un case vengono chiusi, il case è risolto. Man mano che gli operatori portano a termine dei task, producono dei log dove viene presentato un resoconto sulle attività svolte, con testo e allegati.

Un'ulteriore funzionalità di TheHive è quella che, nella barra di navigazione in alto, va sotto il nome di *Statistics*. In questa sezione si trova una dashboard con grafici generati automaticamente, che descrivono i case presenti nel database di TheHive.

3.2.2 TheHive4py

TheHive4py è un client API in Python per la piattaforma TheHive. Esso, integrato in un applicativo Python, permette di interfacciare il programma stesso con la piattaforma e, all'occorrenza, di creare o aggiornare alert, case, template di case, di aggiungere observable o di interrogare TheHive per ottenere questi elementi.

3.2.3 Collaborazione

Uno degli aspetti fondamentali di TheHive, e di un qualsiasi case management system, è la possibilità di collaborazione tra più analisti SOC. La piattaforma possiede una sezione di live stream (a destra in Figura 3.5) dove vengono inserite e mostrate in tempo reale a tutti gli utenti le notifiche sulle attività in corso, come la creazione o la modifica di nuovi casi, task, observable e IOC.

Viene implementato un sistema di notifiche che permette di gestire o assegnare nuovi task e di ricevere degli alert da più sorgenti differenti, come SIEM, CTI (Cyber Threat Intelligence) provider o report di email. Dunque, gli analisti SOC possono schedulare e dedicarsi a diversi task, magari riguardanti lo stesso caso nello stesso momento; se, per esempio, uno di essi si dedica all'analisi di un malware, un altro può andare alla ricerca di un eventuale server C2 analizzando i log del traffico di rete.

3.3 Cortex

Cortex è un motore di analisi standalone che ben si integra con TheHive e la piattaforma MISP. Esso risolve due problematiche incontrate di frequente nei SOC, CSIRT e dai ricercatori di sicurezza nel corso di attività di threat intelligence, digital forensics o risposta agli incidenti. Tali problematiche riguardano:

- Come analizzare gli observable raccolti in modo da coinvolgere un unico strumento.
- Come rispondere attivamente alle minacce e come interagire con altri team o gli utenti del sistema difeso.

Cortex mira a risolvere queste problematiche con *Analyzer*, *Responder* e API RESTful.

Anche Cortex utilizza ElasticSearch come supporto di storage. La sua fase di avvio è analoga a quella di TheHive: è necessario creare il database su ElasticSearch, un utente superadmin, fornendo nome e credenziali d'accesso, ed effettuare l'accesso con tale utente. In Cortex è necessario creare delle organizzazioni. In quella di default, chiamata *Cortex*, possono far parte soltanto gli utenti superadmin, i quali possono esclusivamente gestire organizzazioni e utenti. Per questo è necessario creare una nuova organizzazione e gli utenti che ne fanno parte. Questi ultimi possono attivare ed utilizzare gli Analyzer e i Responder.

Gli Analyzer e i Responder sono applicazioni autonome gestite ed eseguite tramite Cortex. I primi permettono agli analisti e ai ricercatori di sicurezza di analizzare observable e IOC, come nomi di dominio, file, indirizzi IP, hash di file o URL. La maggior parte degli Analyzer sono gratuiti, mentre alcuni richiedono una licenza a pagamento. I secondi sono programmi che effettuano diverse azioni; essi possono essere eseguiti su alert, case, task, log di task o observable.

La Figura 3.7 mostra l'elenco dei job effettuati da Cortex.

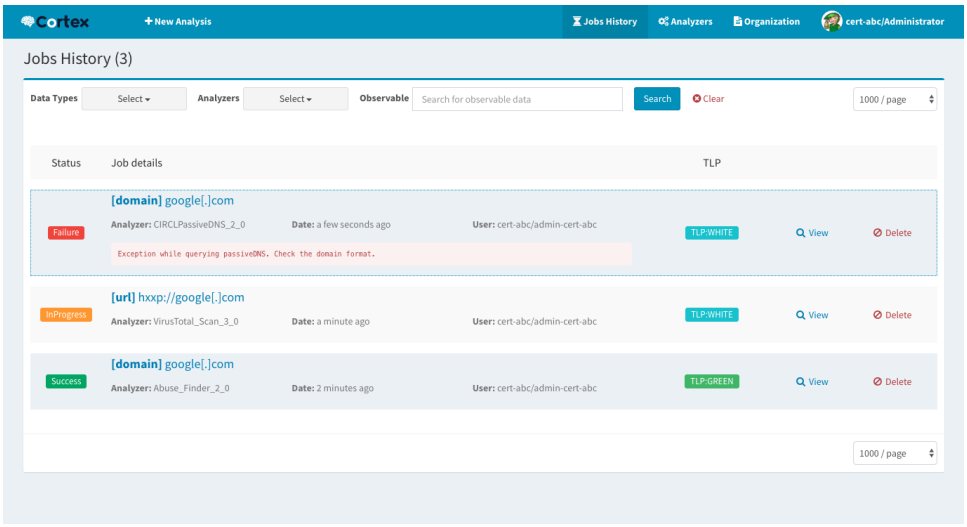

Figura 3.7. Pannello di controllo dei job di Cortex

Chiunque può scrivere un nuovo Analyzer o Responder e utilizzarlo in Cortex. Questo è proprio quello che abbiamo fatto nell'ambito della presente tesi.

3.3.1 Analyzer di CAPE Sandbox

Sebbene esista l'analizzatore per Cuckoo Sandbox tra quelli di default, scaricabili dal sito [2], non esiste l'analizzatore per CAPE Sandbox. Perciò abbiamo deciso di scriverlo noi per i nostri scopi. Poiché CAPE deriva da Cuckoo, i rispettivi Analyzer non saranno molto diversi tra loro.

Il nuovo Analyzer definisce una classe, che estende la classe `Analyzer` di Cortex, ed esegue il metodo `run` di tale classe. Inizialmente il metodo `run` sottomette un nuovo job all'istanza, precedentemente attivata, di CAPE tramite un richiesta POST alle API in ascolto su un certo IP e una certa porta (specificati in fase di configurazione). Successivamente, con il metodo GET, richiede l'eventuale report a intervalli di sessanta secondi. Se dopo un certo intervallo di tempo il report non è ancora disponibile, l'analisi è considerata fallita. Se, al contrario, l'analisi non fallisce, viene prodotto un riassunto del report, il quale viene restituito all'istanza di Cortex, che lo renderà consultabile da interfaccia web.

La produzione di questo nuovo Analyzer ci è sembrata necessaria anche perché quello già presente in Cuckoo non era molto flessibile. In particolare, ci è sembrato utile avere a disposizione un Analyzer che permettesse di impostare i parametri con cui eseguire un task su CAPE. Infatti, come descritto nel Capitolo 2, al momento della submission di un nuovo task, è possibile specificare diversi parametri e opzioni, che permettono di personalizzare l'analisi del file/URL. A tale scopo, abbiamo creato questo nuovo Analyzer che, al momento dell'attivazione su Cortex, permette di specificare i valori dei campi in Figura 3.8.

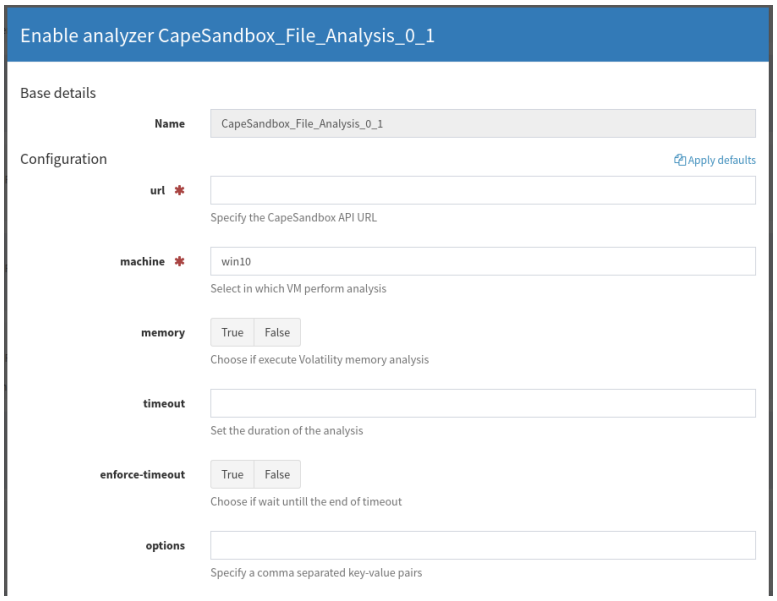

Figura 3.8. Configurazione dell'Analyzer prodotto da interfaccia web

La Figura 3.8 rappresenta la finestra di configurazione del nuovo Analyzer, quando lo si attiva da interfaccia web. Rispetto a quello di Cuckoo sono stati aggiunti nuovi campi per inserire le informazioni con le quali effettuare la submission del job all'istanza di CAPE.

Essendo l'Analyzer un programma che prende in input un observable e delle informazioni di configurazione, è necessario creare almeno due tipologie di file per definirne uno nuovo, ovvero:

- il programma stesso;
- uno o più file di configurazione;
- il file con i requisiti Python (se l'Analyzer è scritto in Python).

capesandbox_analyzer.py

Il Listato 3.1 mostra una parte del codice del nostro Analyzer per eseguire la submission di un task a CAPE Sandobox.

```
from cortexutils.analyzer import Analyzer

...

Class CapeSandboxAnalyzer(Analyzer){
   def __init__(self):
       Analyzer.__init__(self)
       self.url = self.get_param('config.url', None, 'CapeSandbox API URL is missing')
       self.url = self.url + "/" if not self.url.endswith("/") else self.url
       self.machine = self.get_param('config.machine', 'win10', '')
       self.memory = self.get_param('config.memory', False, '')
       self.timeout = self.get_param('config.timeout', '45', '')
       self.enforce_timeout = self.get_param('config.enforce-timeout', False, '')
       self.options = self.get_param('config.options', '', '')

   def run(self):
       Analyzer.run(self)

       try:

           # file analysis
           if self.data_type == 'file':
               filepath = self.get_param('file', None, 'File is missing')
               filename = self.get_param('filename', basename(filepath))
               data = {'machine': self.machine, 'memory': self.memory, 'timeout': self.timeout,
                   'enforce_timeout': self.enforce_timeout, 'options': self.options}

               with open(filepath, "rb") as sample:
                   files = {"file": (filename, sample)}
                   response = requests.post(self.url + 'tasks/create/file', files=files, data=data )
               if 'task_ids' in response.json().keys():
                   task_id = response.json()['task_ids'][0]
               elif 'task_id' in response.json().keys():
                   task_id = response.json()['task_id']
               elif response.status_code == 401:
                   self.error("API token is required by this CAPE instance.")
               else:
                   self.error(response.json()['message'])

           ...

         # Wait for the end of the analysis
           finished = False
           tries = 0
           while not finished and tries <= 15: # wait max 15 mins
               time.sleep(60)
               response = requests.get(self.url + 'tasks/view/' + str(task_id), headers=headers)
               content = response.json()['task']['status']
               if content == 'reported':
                   finished = True
               tries += 1
           if not finished:
               self.error('CapeSandbox analysis timed out')

           # Download the report
           response = requests.get(self.url + 'tasks/report/' + str(task_id) + '/json', headers=headers)
           resp_json = response.json()

           ...

           # Report building
           if self.data_type == 'url':
               self.report({
                   'signatures': list_description, 'suricata_alerts': suri_alerts, 'snort_alerts': snort_alerts, '
                           domains': domains, 'uri': uri,
                   'malscore': resp_json['malscore'] if 'malscore' in resp_json.keys()
                           else resp_json['info'].get('score', None),
                   'malfamily': resp_json.get('malfamily', None),
                   'file_type': 'url',
                   'yara': resp_json['target']['url'] if 'target' in resp_json.keys() and 'url'
                           in resp_json['target'].keys() else '-'
               })
           else:
               self.report({
```

```
74                  'signatures': list_description, 'suricata_alerts': suri_alerts, 'snort_alerts', '
                        domains': domains, 'uri': uri,
75                  'malscore': resp_json['malscore'] if 'malscore' in resp_json.keys() else resp_json['info'].get(
                        'score', None),
76                  'malfamily': resp_json.get('malfamily', None),
77                  'file_type': "".join([x for x in resp_json['target']['file']['type']]),
78                  'yara': [x['name'] + " - " + x['meta']['description'] if 'description' in x['meta'].keys() else
                        x['name'] for x in resp_json['target']['file']['yara']]
79              })
80
81              ...
82
83  if __name__ == '__main__':
84      CapeSandboxAnalyzer().run()
85  }
```

Listato 3.1. Classe `CapeSandboxAnalyzer`

Essendo CAPE Sandbox un'estensione di Cuckoo Sandbox, le API delle due soluzioni differiscono per pochi dettagli, perciò è stato sufficiente modificare il codice dell'Analyzer di Cuckoo in minima parte per produrre un Analyzer per CAPE funzionante.

Nel Listato 3.1 si pone l'attenzione a tre aspetti particolari: *(i)* la submission del file (non è riportato il codice della submission di un URL, ma, di fatto, è del tutto analogo); *(ii)* il ciclo di attesa della fine dell'analisi di CAPE; *(iii)* la costruzione del report finale.

CapeSandbox_File_Analysis.json

In questo file, riportato nel Listato 3.2, è possibile definire le impostazioni con cui attivare l'Analyzer. Tutti gli Analyzer presentano delle impostazioni di default (non riportate in Figura 3.8) e, in aggiunta, le proprie impostazioni specifiche. Queste ultime sono aggiunte inserendo nuovi elementi nella lista associata alla chiave `configurationItems`.

```
1
2   {
3     "name": "CapeSandbox_Analysis",
4     "version": "0.1",
5     "author": "Jacopo Carloni",
6     "url": "",
7     "license": "AGPL-V3",
8     "description": "CAPE Sandbox file/URL analysis.",
9     "dataTypeList": ["file", "url"],
10    "command": "CapeSandbox/capesandbox_analyzer.py",
11    "baseConfig": "CapeSandbox",
12    "configurationItems": [
13      {
14        "name": "url",
15        "description": "Specify the CapeSandbox API URL",
16        "type": "string",
17        "multi": false,
18        "required": true
19      },
20      {
21        "name": "machine",
22        "description": "Select in which VM perform analysis",
23        "type": "string",
24        "multi": false,
25        "required": true,
26        "defaultValue": "win10"
27      },
28      {
29        "name": "memory",
30        "description": "Choose if execute Volatility memory analysis",
31        "type": "boolean",
32        "multi": false,
33        "required": false,
34        "defaultValue": false
35      },
36      {
37        "name": "timeout",
```

```
38        "description": "Set the duration of the analysis",
39        "type": "string",
40        "multi": false,
41        "required": false
42      },
43      {
44        "name": "enforce-timeout",
45        "description": "Choose if wait untill the end of timeout",
46        "type": "boolean",
47        "multi": false,
48        "required": false,
49        "defaultValue": false
50      },
51      {
52        "name": "options",
53        "description": "Specify a comma separated key-value pairs",
54        "type": "string",
55        "multi": false,
56        "required": false
57      }
58    ]
59  }
```

Listato 3.2. File di configurazione dell'Analyzer prodotto

Gli item che abbiamo aggiunto sono `url`, `machine`, `memory`, `tiemout`, `enforce-timeout` e `options`, in linea con quanto mostrato in Figura 3.8.

3.4 Threat Intelligence e MISP

La difesa dei sistemi informativi in forza a un'organizzazione è data da una somma di diversi elementi. Poiché i metodi di attacco sono molti, le difese da mettere in campo sono altrettanto numerose. Quindi, per fare degli esempi, la difesa di una rete di computer può essere effettuata con la raccolta e l'analisi del traffico, sia per capire in tempo reale cosa sta accadendo, sia per conoscere le modalità con cui un attaccante è riuscito ad accedere, a incidente avvenuto; gli endpoint possono essere monitorati con HIDS (Host-based Intrusion Detection System) per rilevare intrusioni; l'attuazione di una corretta policy di scelta e di salvataggio delle password può rendere difficile il lavoro di un attaccante.

Gli attacchi sono sempre più numerosi e sofisticati e, dal punto di vista tecnologico, le modalità d'attacco sono tanto più numerose quanto più è ampia la superficie d'attacco del bersaglio. Chi difende deve essere obbligatoriamente attento a chiudere tutte le strade agli avversari. Tuttavia, al di là di qualsiasi soluzione tecnologica, hardware o software, adoperata, o policy di sicurezza stabilita e adottata, gioca un ruolo sempre più rilevante la raccolta di informazioni. La conoscenza dell'ambiente in cui si opera, dei rischi a cui si è soggetti e delle possibili minacce possono fare la differenza nella difesa dei propri asset. Per questo, negli ultimi anni, si sente sempre più frequentemente parlare di *Threat Intelligence*, o *Cyber Threat Intelligence*.

Secondo [32], una *cyber threat* è

> "una qualsiasi circostanza o evento con il potenziale di impattare negativamente su operazioni organizzative (come la mission, le funzioni, l'immagine o la reputazione), sugli asset organizzativi, sugli individui, su altre organizzazioni o la nazione attraverso accessi non autorizzati a sistemi informatici, distruzione, divulgazione o modifica di informazioni, e/o interruzioni di servizi".

Gli individui e i gruppi che costituiscono tali minacce sono detti *threat actors*, o semplicemente *actors*.

Con *Threat Information* si indica qualsiasi informazione relativa a una minaccia che può aiutare un'organizzazione a proteggersi da una minaccia o a rilevare le attività di un attore. Le principali tipologie di *threat information* sono:

- *Indicator*, observable o artefatti che suggeriscono che un attacco è imminente, in corso d'opera o già terminato. Esempi di indicatori sono: indirizzi IP o nomi di domino DNS sospetti, URL, file hash.
- *Tactics Technique and Procedure (TTP)*, che descrivono il comportamento di un attaccante. Per *tactic*, si intende la descrizione ad alto livello del comportamento dell'attaccante, le *technique* sono descrizioni più dettagliate nel contesto di una tattica, infine, a loro volta, le *procedure* sono descrizioni ancora più dettagliate e di più basso livello nel contesto di una tecnica.
- *Security alert*, i quali sono prodotti da enti come lo US-CERT e sono delle notifiche tecniche riguardanti le vulnerabilità più recenti, gli exploit e altri problemi di sicurezza.
- *Threat intelligence report*, i quali sono documenti che descrivono le TTP, gli attori, i sistemi bersagliati e altre informazioni che forniscono un'ottima *situational awareness* a un'organizzazione.
- *Tool configuration*, che sono raccomandazioni per impostare e utilizzare strumenti che supportano la collezione, lo scambio, l'elaborazione, l'analisi e l'uso automatici di informazioni sulle minacce.

Appare evidente, dunque, come lo scambio di questo tipo di informazioni possa produrre importanti benefici nella difesa delle organizzazioni. Gli attori sono sempre più competenti e sempre meglio organizzati, e la cyber defence sarebbe più efficace qualora le organizzazioni collaborassero.

3.4.1 Cos'è la Threat Intelligence

In accordo con [29], la *Threat Intelligence (TI)* è il processo di acquisizione, attraverso molteplici sorgenti, di informazioni riguardanti le minacce in un ambiente. La definizione più citata, e apparentemente la più autorevole in letteratura, è quella fornita da Rob McMillan in [20] dove descrive la TI come

> "conoscenza basata sui fatti, comprensiva di contesto, meccanismi, indicatori, implicazioni e suggerimenti riguardanti una minaccia emergente o i rischi per l'asset che può essere utilizzata per informare le decisioni in merito alla risposta del soggetto alla minaccia o rischio".

Un'organizzazione che vuole mantenere un team e vuole incrementare il proprio livello di sicurezza deve tenere in considerazione l'utilizzo della TI. Lo scopo di quest'ultima è di rilevare incidenti il più presto possibile e, all'evenienza, evitarli. La Figura 3.9, estrapolata da [37], mostra come per il 53% dei casi, il personale che si occupa di CTI lavora all'interno di un SOC. Poiché molte organizzazioni spesso faticano a trovare personale qualificato nell'ambito delle operazioni di sicurezza o di risposta agli incidenti, sembra avere senso la scelta di integrare personale dedicato alla CTI con gli analisti SOC: in questo modo si uniscono le informazioni provenienti dal lavoro dei primi con le pratiche attuate dai secondi.

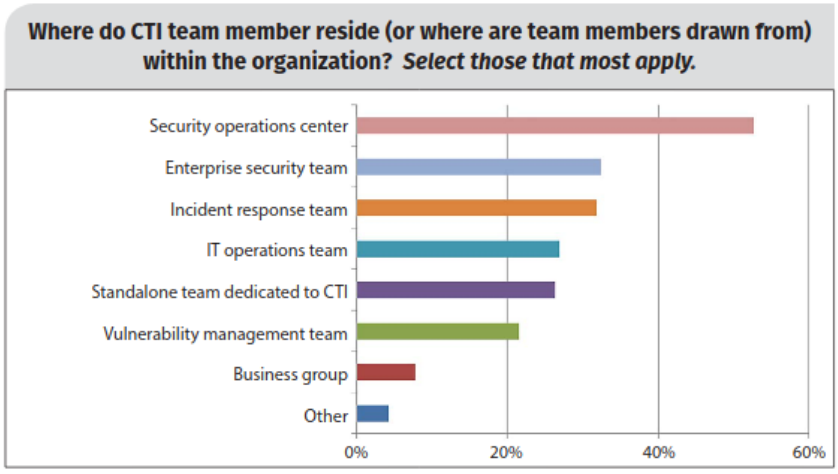

Figura 3.9. Soggetti che utilizzano maggiormente la CTI

Integrare attività di CTI all'interno di un'organizzazione non è un processo immediato. Per ottenere un impatto positivo, è necessario progettarle a dovere. Per esempio, una delle questioni più importanti è quella riguardante le sorgenti: esse possono essere esterne o interne, e nel caso di sorgenti esterne è necessario stabilirne la fiducia e la qualità delle informazioni condivise.

Come mostrato in Figura 3.10, estratta da [16], esistono quattro tipi di TI, ovvero:

- *Strategica*, che, usando analisi dettagliate sui trend e i rischi emergenti, produce dei possibili scenari sulle conseguenze di attacchi cyber. É un'intelligence di alto livello e a lungo termine.
- *Tattica*, che offre dettagli più specifici sulle TTP degli attori. Questa è rivolta maggiormente agli addetti alla protezione dei sistemi e delle reti.
- *Tecnica*, con la quale si scende ancora di livello andando a considerare dettagli e indicatori più tecnici, come le caratteristiche specifiche di malware.
- *Operativa*, che aiuta i team di sicurezza a capire la natura di specifici attacchi come gli scopi, i tempi e il livello di sofisticazione dei responsabili.

3.4.2 MISP

Una *Threat Intelligence Platform (TIP)* è un strumento di supporto alle organizzazioni nei programmi di CTI. Essa permette di collezionare, normalizzare, arricchire, correlare, analizzare e condividere informazioni su minacce e attori [30].

Sebbene esistano molte piattaforme di TI, molte delle quali open source, (come [1, 5, 11]), per questo progetto abbiamo deciso di utilizzare la piattaforma MISP (Malware Information Sharing Platform) [38]. Essa è open source ed è una delle piattaforme più conosciute e utilizzate a livello mondiale. Il motivo principale che ci ha spinto ad optare per MISP è la sua facile integrabilità con TheHive e Cortex, come mostrato in Figura 3.11.

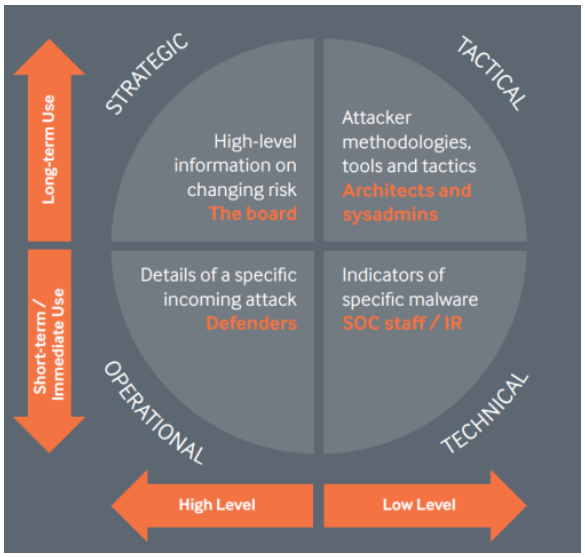

Figura 3.10. Le quattro tipologie di CTI

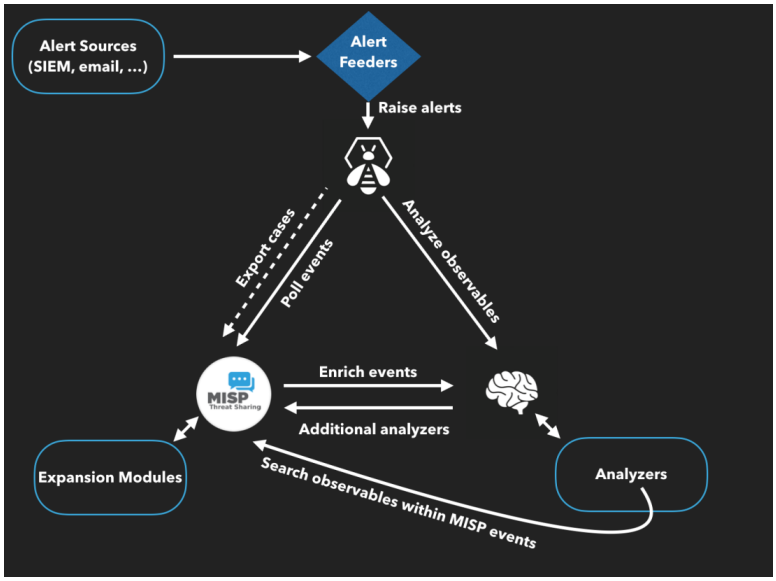

Figura 3.11. Integrazione di TheHive-MISP-Cortex

Alcune informazioni su MISP

Molte comunità, anche afferenti a diversi settori, utilizzano MISP. Chiunque in MISP può fungere da produttore e/o consumatore di informazioni. Non è obbligatorio condividere.

Il formato del core di MISP è JSON, e tutti gli *Object* sono rappresentati da oggetti JSON. MISP differenzia tra modello di dati, espressi con l'iniziale maiuscola (per esempio, Org o Event) e attributi, al contrario caratterizzati dall'iniziale minuscola; questi ultimi sono campi che descrivono alcune caratteristiche degli oggetti.

Un insieme di indicatori tra loro collegati riguardanti un certo contesto costituisce un evento. Un evento può essere creato a seguito di un incidente, dopo aver prodotto il report di un'analisi o dallo studio di un attore specifico. Quest'ultimo è costituito da una serie di attributi, come, solo per citarne alcuni, l'UUID (l'identificatore unico dell'evento), il *threat level*, il suo stato di pubblicazione, lo stato dell'analisi cui si riferisce, la data e il timestamp dell'ultima modifica, l'id dell'organizzazione che lo ha generato, la modalità di distribuzione.

Tra gli *Object* di MISP troviamo le organizzazioni (*Org*), caratterizzate da UUID, nome e ID, come mostrato nell'esempio del Listato 3.3.

```
1   "Org": {
2           "id": "2",
3           "name": "CIRCL",
4           "uuid": "55f6ea5e-2c60-40e5-964f-47a8950d210f"
5   }
```

Listato 3.3. Esempio di Org

I *MISP Attribute*, da non confondere con gli attributi di un oggetto, sono oggetti utilizzati per descrivere gli indicatori e i dati di un evento. Le informazioni obbligatoriamente presenti sono *category*, *type* e *value*. In aggiunta a questi possono essercene molti altri, come mostra il Listato 3.4.

```
1
2   "Attribute": {
3                   "id": "346056",
4                   "type": "comment",
5                   "category": "Other",
6                   "to_ids": false,
7                   "uuid": "57f4f6d9-cd20-458b-84fd-109ec0a83869",
8                   "event_id": "3357",
9                   "distribution": "5",
10                  "timestamp": "1475679332",
11                  "comment": "",
12                  "sharing_group_id": "0",
13                  "deleted": false,
14                  "value": "Hello world",
15                  "SharingGroup": [],
16                  "ShadowAttribute": [],
17                  "RelatedAttribute": []
18                 }
```

Listato 3.4. Esempio di Attribute

In MISP è possibile utilizzare i tag per classificare un evento e riassumere in una parola il suo contenuto. Il nome del tag può essere scelto liberamente, oppure preso da vocabolari predefiniti, in MISP chiamati *Taxonomy*. Anche il tag è un array JSON, costituito da quattro voci, mostrate nell'esempio del Listato 3.5.

```
"Tag": [{
            "exportable": true,
            "colour": "#ffffff",
            "name": "tlp:white",
            "id": "2"
        }]
```

Listato 3.5. Esempio di Tag

I *MISP Object* circoscrivono il contesto di una lista di attributi di un certo evento. Il loro scopo principale è quello di descrivere strutture più complesse di quelle che possono essere descritte da un singolo attributo. Ciascun oggetto è creato con un certo template. Nel Listato 3.6 è mostrato un Object.

```

"Object":{
            "name": "domain|ip",
            "meta-category": "network",
            "description": "A domain and IP address seen as a tuple in a specific time frame.",
            "version": 1,
            "uuid": "f47559d7-6c16-40e8-a6b0-eda4a008376f",

            "attributes" :
            {
                    "ip": {
                            "misp-attribute": "ip-dst",
                            "ui-priority": 1,
                            "categories": ["Network activity","External analysis"]
                    },
                    "domain": {
                            "misp-attribute": "domain",
                            "ui-priority": 1,
                            "categories": ["Network activity","External analysis"]
                    },
                    "first-seen": {
                            "misp-attribute": "datetime",
                            "disable\_correlation": true,
                            "ui-priority": 0
                    },
                    "last-seen": {
                            "misp-attribute": "datetime",
                            "disable\_correlation": true,
                            "ui-priority": 0
                    }

            },
            "required": ["ip","domain"]
}
```

Listato 3.6. Esempio di Object

Oggetti tra loro, o oggetti e attributi possono essere collegati logicamente tramite le *Object Reference*.

Infine ci sono le *Galaxy*. Esse sono un semplice metodo per esprimere un grande oggetto che può essere allegato a eventi MISP. Una galassia, o cluster, può essere composto da più elementi del tipo chiave-valore.

```
"Galaxy": [ {
            "id": "18",
            "uuid": "698774c7-8022-42c4-917f-8d6e4f06ada3",
            "name": "Threat Actor",
            "type": "threat-actor",
            "description": "Threat actors are characteristics of malicious actors
                            (or adversaries) representing a cyber attack threat
                            including presumed intent and historically observed behaviour.",
                "version": "1",
             "GalaxyCluster": [
             {
                "id": "1699",
                "uuid": "7cdff317-a673-4474-84ec-4f1754947823",
```

```
14                "type": "threat-actor",
15                "value": "Anunak",
16                "tag_name": "misp-galaxy:threat-actor=\"Anunak\"",
17                "description": "Groups targeting financial organizations
18                   or people with significant financial assets.",
19                "galaxy_id": "18",
20                "source": "MISP Project",
21                "authors": [
22                   "Alexandre Dulaunoy",
23                   "Florian Roth",
24                   "Thomas Schreck",
25                   "Timo Steffens",
26                   "Various"
27                ],
28                "tag_id": "111",
29                      "meta": {
30                         "synonyms": [
31                            "Carbanak",
32                            "Carbon Spider"
33                         ],
34                         "country": [
35                            "RU"
36                         ],
37                         "motive": [
38                            "Cybercrime"
39                         ]
40                      }
41                   }
42             ]
43          }
44       ]
```

Listato 3.7. Esempio di Galaxy

La Figura 3.12 riassume il modello dei dati di MISP, mentre la Figura 3.13 mostra una serie di eventi.

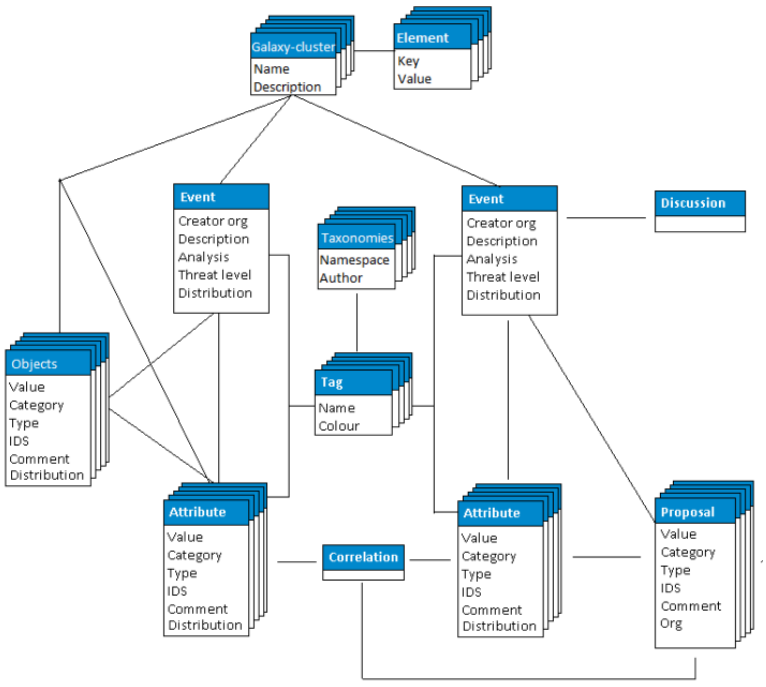

Figura 3.12. Modello dei dati di MISP

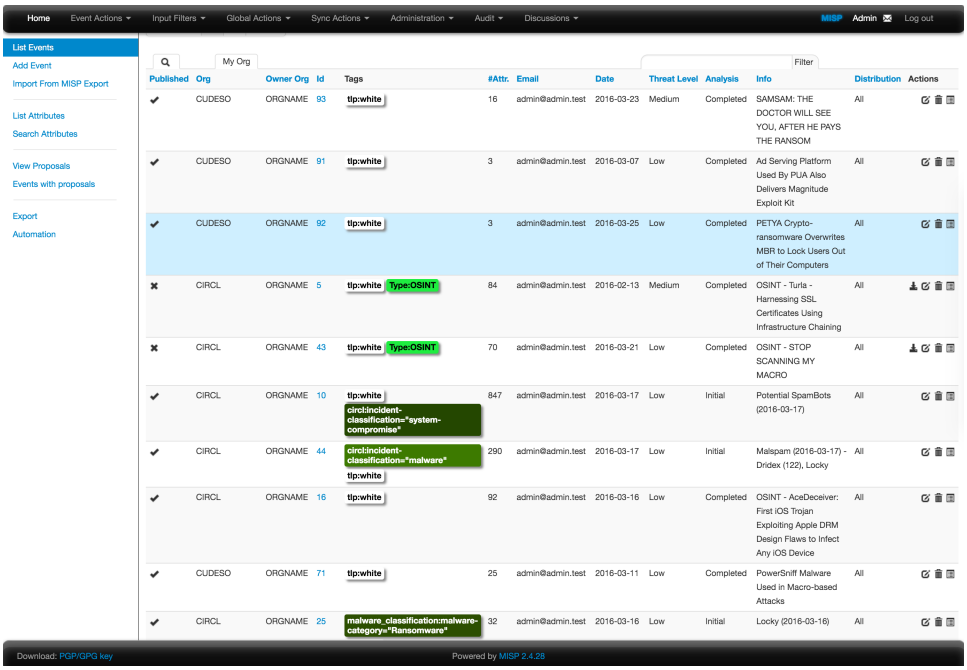

Figura 3.13. Lista di eventi in MISP

Dalla Figura 3.13, si nota come, navigando sui link della sezione di sinistra, tra le altre cose, è possibile elencare gli eventi (schermata corrente) o aggiungerne dei nuovi, elencare gli attributi o cercare particolari attributi con dei filtri di ricerca.

In questo progetto, poiché le nostre attività non sono propriamente di Cyber Threat Intelligence, ma piuttosto rientrano in quelle di Cyber Threat Hunting, o eventualmente di Incident Response, la piattaforma MISP è stata utilizzata in maniera limitata, a puro scopo consultivo. Come si vedrà nei capitoli successivi, il sistema che abbiamo cercato di costruire interroga MISP per capire se un link o un file sia già stato analizzato in precedenza e magari valutato come malevolo. Dunque, la nostra è semplicemente una consultazione degli eventi e degli attributi pubblici. L'interrogazione di MISP avviene tramite la libreria Pyhton PyMISP, utilizzando la APIkey di default come metodo di autenticazione. Come sarà descritto nei prossimi capitoli, nel nostro flusso di lavoro, ricercheremo su MISP observable e artefatti, quali IP, URL o file, associati ad attori ostili noti oppure a incidenti già avvenuti e riportati, in modo da prendere le giuste contromisure.

3.5 Conclusioni

In questo capitolo abbiamo parlato di tre strumenti open source molto conosciuti nel settore della cybersecurity, che permettono di gestire e mettere in atto la risposta agli incidenti. In primo luogo, abbiamo descritto una piattaforma di Incident Response, ovvero TheHive. Questa risulta essere uno strumento molto utile per gli

analisti SOC, in quanto permette di gestire e coordinare le attività del team per abbattere i tempi di risposta agli incidenti. Successivamente, abbiamo introdotto Cortex: le due piattaforme possono lavorare in modalità standalone, ma, facendo parte dello stesso progetto, sono altamente integrabili ed è possibile utilizzare Cortex per automatizzare i task di analisi di TheHive. Infine, pur non facendo parte di TheHiveProject, abbiamo introdotto la Cyber Threat Intelligence e la piattaforma MISP, la quale ben si integra nel flusso di lavoro di TheHive e Cortex. Questa permette la condivisione di informazioni preziose riguardanti rischi, minacce, attori ostili e attacchi già perpetrati; in questo modo, all'interno di un SOC o CSIRT, si cerca di evitare incidenti o eseguire nuovamente analisi già svolte in passato.

Unendo le caratteristiche di questi tre strumenti è possibile creare un flusso di lavoro di risposta agli incidenti efficiente ed efficace.

4

Posta elettronica e protocolli SMTP, POP3 e IMAP. Plesk Mail Server

Questo capitolo tratterà del funzionamento della posta elettronica e descriverà i protocolli attualmente utilizzati per l'invio e la ricezione di email. Successivamente introdurremo Plesk, una piattaforma di web hosting, e la sua soluzione di Mail Server, che abbiamo utilizzato per implementare un sistema di web mail.

4.1 Posta elettronica

La posta elettronica, abbreviato email, è un servizio che utilizza Internet per lo scambio di messaggi tra computer, e rappresenta uno dei primi servizi sviluppati dalla nascita di Internet. Inizialmente, la posta elettronica veniva utilizzata soltanto nel mondo accademico; tuttavia, dagli anni novanta in poi, iniziò ad diffondersi anche in altri ambienti su larga scala, e ad oggi, nonostante la diffusione di applicazioni di *instant messaging*, rimane sempre uno dei principali metodi di comunicazione su Internet, soprattutto per fini lavorativi e commerciali.

4.1.1 Architettura e Servizi

L'architettura di un sistema di posta elettronica è costituita dai seguenti agenti software: il *Mail User Agent (MUA)*, il *Mail Submission Agent (MSA)*, il *Mail Transfer Agent (MTA)*, il *Mail Delivery Agent (MDA)* e il *Mail Retrieval Agent (MRA)*. Per essere quanto più dettagliati possibile, questi appena elencati sono tutti gli agenti che esistono; tuttavia, non c'è sempre una netta distinzione tra di essi. Eccezion fatta per il MUA e l'MTA, che hanno funzioni ben precise, spesso può accadere che l'MSA e l' MRA vengano compresi rispettivamente nell'MTA e nel MUA più grandi. Il MUA è il classico client di posta elettronica che viene utilizzato dagli utenti per comporre o visualizzare i messaggi (per esempio, Mozilla Thunderbird o Microsoft Outlook); l'MSA è un agente software che riceve un'email da un MUA e coopera con un MTA per la consegna. Storicamente sia l'MSA che l'MTA utilizzano la porta 25, ma ufficialmente quella del primo è la 587. L'MSA controlla che la mail sia sintatticamente corretta e conforme a determinati standard.

Il Mail Transfer Agent riceve i messaggi dal MUA (o dall'MSA, qualora MUA ed MSA siano separati) o da un altro MTA e li inoltra al suo corrispettivo di destinazione; l'MDA riceve il messaggio dall'MTA di destinazione e si incarica di salvarlo sul disco del server nella mailbox locale del destinatario. Esso è anche chiamato *Local Delivery Agent (LDA)*. Infine, l'MRA è il programma che recupera le email da un server remoto e opera in collaborazione con un MDA per recapitare la mail ad una mailbox locale o remota. Come già accennato, esso può essere standalone o integrato all'interno di un MUA più grande. La Figura 4.1 mostra la catena di agenti software appena descritti e il contesto in cui operano.

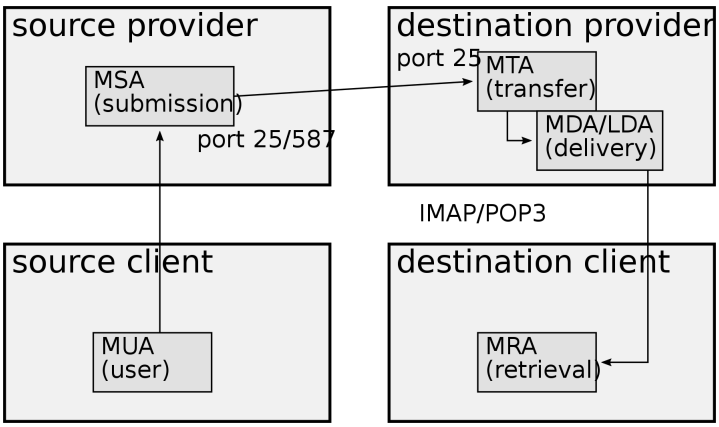

Figura 4.1. Architettura di un sistema di email

L'MTA è un processo eseguito solitamente in background nei mail server; e rimane sempre in ascolto sulla porta 25. Essi utilizzano il protocollo SMTP (Simple Mail Transfer Protocol) per spostare un messaggio da un server all'altro, fino a raggiungere quello di destinazione, ritornando lo stato di ricezione o eventuali errori.

Le *mailbox* rappresentano un sistema di memorizzazione delle email. L'utente che invia una mail deposita il messaggio nella mailbox del destinatario passando per gli agenti precedentemente descritti, che comunicano tramite protocollo SMTP. L'utente che visualizza le email recupera il contenuto della propria mailbox tramite protocollo IMAP o POP3.

4.1.2 Simple Mail Transfer Protocol (SMTP)

Il trasferimento delle email dal mittente al destinatario avviene tramite protocollo SMTP.

Inizialmente il MUA invia l'email al proprio MTA e, quindi, inserisce il messaggio nel sistema di ricezione. Ciò è rappresentato dalla fase uno, chiamata *Mail submission* in Figura 4.2. Originariamente, il MUA mittente e l'MTA erano eseguiti sulla stessa macchina. Oggi non è più così: mentre lo user agent è eseguito su laptop,

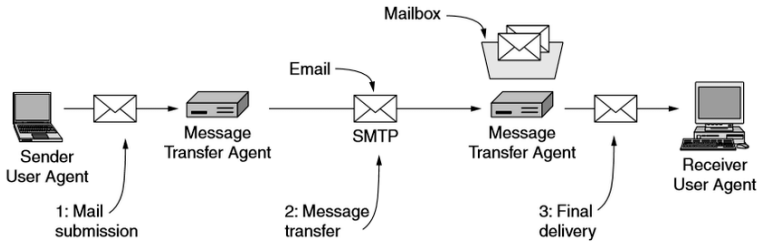

Figura 4.2. Architettura semplificata di un sistema di email

personal computer o smartphone, l'MTA è eseguito dai server dell'ISP o dei provider del servizio, i quali sono sempre connessi a Internet. Dunque, un qualsiasi client potrebbe connettersi al server SMTP, per questo, prima di connettersi al server, è necessaria l'autenticazione (tramite l'estensione AUTH del protocollo SMTP).

Il computer mittente stabilisce con la porta 25 dell'MTA una connessione TCP. Dopo aver accettato la connessione in ingresso e alcuni controlli di sicurezza, il destinatario accetta i messaggi in arrivo in codifica ASCII. Prima di inviare la mail, il mittente specifica al MTA il destinatario (o i destinatari, se più di uno). Se questo è conosciuto dal MTA allora si prosegue nell'invio del messaggio, altrimenti l'invio abortisce.

Una volta che l'MTA del mittente ha ricevuto la mail, questo deve recapitarla all'MTA del destinatario. Per capire quale server contattare, viene effettuata una richiesta DNS (Domain Name System) in base all'indirizzo del destinatario per conoscere l'identità (indirizzo IP) del server a cui inviare il messaggio. A questo punto l'MTA mittente si connette con protocollo SMTP alla porta 25 del server destinatario e invia l'email; questa, una volta ricevuta, verrà inserita nella mailbox dell'utente interessato. Perciò, in questo modo, la mail può viaggiare dall'MTA mittente all'MTA destinatario con un singolo salto, o passando attraverso più MTA, in base alla topologia della rete.

4.1.3 Internet Message Access Protocol (IMAP). Post Office Protocol, v3 (POP3)

A questo punto, rimane soltanto il passaggio della mail dalla mailbox al MUA del destinatario per essere mostrata. Per questo scopo, non si utilizza il protocollo SMTP, che è di tipo *push-based*, ma ne serve uno differente, che, tra le altre cose, permetta di gestire in maniera completa la propria mailbox. Uno dei protocolli più utilizzati per il passaggio finale è l'IMAP (Internet Message Access Protocol). Il mail server esegue un processo in ascolto sulla porta 143. Il MUA, invece, esegue un IMAP client, il quale si connette al server e inizia a inviare comandi. Dunque, il protocollo consiste in una connessione di rete iniziale tra client e server, un "saluto" iniziale da parte del server e una serie di interazioni client/server; per interazioni si intendono comandi del client, invio di dati insieme con lo stato finale dell'operazione da parte del server.

IMAP è un protocollo molto esteso e completo e, grazie al gran numero di comandi a disposizione del client, consente di eseguire molte azioni sulla mailbox. Un

aspetto importante di questo protocollo è il fatto che l'email, pur essendo visualizzata dal client di posta, e magari salvata in cache, non viene scaricata e non viene mantenuto il relativo file in locale.

Un protocollo alternativo che può essere usato per il passaggio finale dell'email dall'MTA al MUA è il Post Office Protocol, Versone 3 (POP3). Questo è meno esteso e sicuro del protocollo IMAP. Inoltre, tramite il POP3, le email sono scaricate per essere mantenute in locale e rimosse dal server. Ciò porta a dei pro e dei contro: un vantaggio è che il server risulta più semplice; tuttavia, non è più possibile leggere tutte le email da più computer differenti, e un guasto del MUA comporta la perdita delle email.

Oltre ai protocolli appena descritti, esistono anche soluzioni proprietarie, come il protocollo utilizzato da Microsoft Exchange.

4.1.4 Il formato dei messaggi

Finora abbiamo descritto il modo in cui viene inviata un'email da un mittente e come essa arriva al destinatario. Ora descriviamo come è fatta un'email; infatti, una email, per essere inviata e immessa nell'MTS (Message Transport System), deve essere posta in un formato standard.

Una email è costituita dalle seguenti parti:

- un*envelope*;
- un header;
- una riga bianca;
- il corpo del messaggio.

Lo user agent costruisce il corpo del messaggio e inserisce i capi dell'header in base alle informazioni inserite dall'utente; l'MTA costruisce l'envelope intorno al messaggio inviato ad esso dallo user agent utilizzando alcuni dei campi dell'header.

Header

I principali campi dell'header di un'email sono riportati nelle Tabelle 4.1 4.2.

CAMPO	DESCRIZIONE
To:	Indirizzo email dei destinatari principali
Cc:	Indirizzo email dei destinatari secondari
Bcc:	Come Cc, ma questa riga viene eliminata dalle copie inviate ai destinatari pricipali e secondari
From:	Persona che ha creato il messaggio
Sender:	Indirizzo email del mittente reale
Received:	Identità di ogni MTA lungo il cammino dal mittente al destinatario
Return-Path:	Aggiunto dall'ultimo MTA per esprimere il percorso all'indietro verso il primo MTA

Tabella 4.1. Campi dell'header legati al trasporto di un messaggio secondo RFC 5322

CAMPO	DESCRIZIONE
Date:	Data e ora in cui l'email è stata spedita
Reply-To:	Indirizzo email a cui dovrebbe essere spedita l'eventuale risposta
Message-Id:	Id unico del messaggio
In-Reply-To:	Se il corrente è un messaggio di risposta, questo campo indica l'id del messaggio a cui risponde
References:	Altri Message-Id rilevanti
Keywords:	Keyword schelte dall'utente
Subject:	Oggetto dell'email

Tabella 4.2. Altri campi dell'header secondo RFC 5322

Lo standard permette la definizione di nuovi campi opzionali dell'header da parte di utenti. Per convenzione, il nome di questi ultimi inizia per *X-* per distinguere tra i campi ufficiali e quelli personalizzati.

Body

Inizialmente il messaggio di una email era costituito da puro testo espresso in ASCII. Con il passare del tempo, si è sentita la necessità di arricchire il contenuto delle email anche con informazioni in altri formati. Si è passati, quindi, al formato MIME (Multipurpose Internet Mail Extension). Questo permette di creare e inviare email più "ricche" del semplice testo, consentendo l'inserimento di nuove parti e definendo regole di codifica per le parti non-ASCII.

Con il MIME vengono definiti nuovi header, riportati in Tabella 4.3

CAMPO	DESCRIZIONE
MIME-version:	Versione MIME
Content-Description:	Descrizione a parole di cosa c'è nel messaggio
Content-Id:	Id unico
Content-Transfer-Encoding:	Modalità con cui il corpo è stato codificato per la trasmissione
Content-Type:	Tipo e formato del contenuto

Tabella 4.3. Header del messaggio aggiunti dal formato MIME

Per le parti testuali del corpo in formato ASCII, la codifica è quella a 7 bit. Se il corpo contiene, ad esempio, allegati binari, la codifica a 7 bit non va bene, per questo file binari vengono trasformati in file testuali con la codifica base64.

Il campo dell'header *Content-Type* contiene il tipo e il sottotipo della parte di messaggio cui si riferisce. La Tabella 4.4 riassume i possibili tipi, e, per ciascuno di essi, alcuni esempi di sottotipi.

TIPO	SOTTOTIPO	DESCRIZIONE
text	plain, html, xml, css	Testi
image	gif, jpeg, tiff	Immagini
audio	basic, mpeg, mp4	Suoni
video	mpeg, mp4, quicktime	Video
model	vrml	Modelli 3D
application	octet-stream, pdf, javascript, zip	Dati prodotti da applicazioni
message	http, rfc822	Messaggi incapsulati
multipart	mixed, alternative, parallel, digest	Combinazione di più tipi

Tabella 4.4. Header del messaggio aggiunti dal formato MIME

4.2 Plesk

La soluzione che abbiamo scelto per implementare un mail server è Plesk. Questo prodotto è una piattaforma di web hosting, fornita di un pannello di controllo, accessibile da browser, utile per gli amministratori di sistema per gestire il proprio portale web. Tra le tante funzionalità, che non passeremo in rassegna perché non rilevanti per lo scopo di questo elaborato, Plesk permette di creare un servizio di mail per lo scambio di posta elettronica tra utenti. Plesk rappresenta il proprio mail server e permette, di nuovo grazie all'interfaccia web, di creare nuovi account e di gestirne le impostazioni, come cambiare le password, abilitare il servizio di risposta automatica, e così via.

4.2.1 Installazione

Innanzitutto, abbiamo creato una macchina virtuale con sistema operativo Ubuntu 18.04 con *virt-manager* connessa a Internet tramite NAT. Per installare Plesk ci sono diverse strade percorribili, come descritto in [7]. Noi abbiamo deciso di installarlo manualmente, sfruttando il comando nel Listato 4.1 per l'installazione "One-click".

```
1
2    $ sh <(curl https://autoinstall.plesk.com/one-click-installer || wget -O - https://autoinstall.plesk.com/one-click
         -installer)
```

Listato 4.1. Comando per la One-click Installation di Plesk

Dopo qualche minuto, l'esecuzione dello script termina ed è possibile accedere alla piattaforma che, come primo passaggio, richiede la creazione di un utente amministratore. Successivamente va creato un nuovo dominio, come mostrato in figura 4.3.

Una volta fatto ciò, è possibile attivare il servizio di mail generando gli utenti nell'apposito pannello, accessibile tramite l'icona della busta della lettera sulla barra a sinistra. Per inserire un utente è sufficiente scegliere un username univoco e la corrispettiva password. In aggiunta, si ha la possibilità di scegliere altre impostazioni, come la dimensione massima della mailbox, un indirizzo alias o messaggi di risposta automatica.

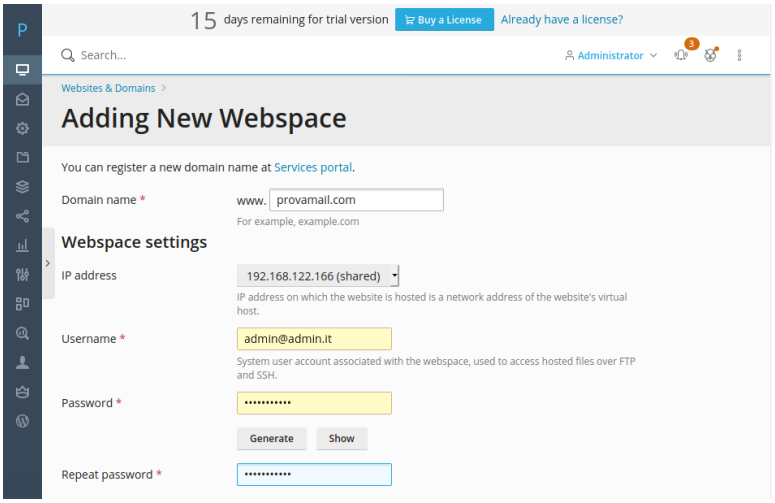

Figura 4.3. Creazione di un nuovo dominio

In Figura 4.4 viene riportato il pannello di controllo del sistema di mail, il quale mostra gli utenti presenti con alcune informazioni aggiuntive.

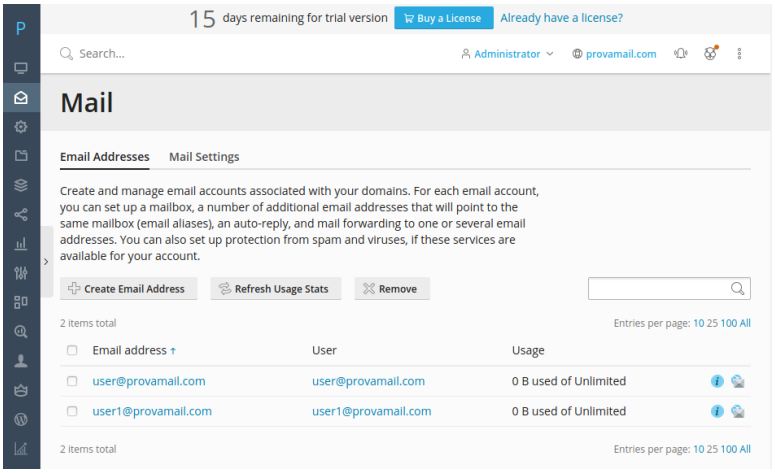

Figura 4.4. Pannello di controllo del Mail Server

A questo punto il servizio di posta elettronica è pronto per essere utilizzato.

Per le nostre prove, abbiamo utilizzato il client di posta elettronica Mozilla Thunderbird, il quale è preinstallato in Ubuntu e permette di avere più utenti attivi contemporaneamente.

4.2.2 Server IMAP e server SMTP

Per Linux, Plesk offre la possibilità di usare due prodotti di mail server: *Postfix* e *Qmail*. Per quanto riguarda i protocolli IMAP/POP3, si può utilizzare *Dovecot* o *Courier IMAP*. Tra questi ultimi due, è consigliato usare il primo in quanto più efficiente, sicuro e "leggero" del secondo.

Per sistemi Windows, si possono utilizzare anche altre soluzioni di mail server, che, tuttavia, vanno installate separatamente; tra questi citiamo *IceWarp* e *SmarterMail*.

La soluzione di default di Plesk per implementare il server IMAP/POP3 è Dovecot; quella per gli MTA, cioè i server SMTP, è Postfix. Quest'ultimo rappresenta un MTA open source e gratuito, che può essere eseguito su qualsiasi sistema operativo "unix-like", con un compilatore C e un ambiente di sviluppo con standard POSIX. Poiché i nostri esperimenti non sono incentrati sul recapito delle email attraverso gli MTA, non verrà riportata, di seguito, una dettagliata descrizione di Postfix. Al contrario, ci è sembrato doveroso descrivere più in dettaglio il funzionamento di Dovecot, in quanto sarà utile per capire le scelte fatte nell'implementazione di un nostro parser (che verrà illustrata nel capitolo successivo).

Dovecot e Maildir

Dovecot rappresenta l'IMAP o il POP3 server, che permette al MUA di un utente di accedere alle proprie email, previa autenticazione. Esso consente la realizzazione del passaggio numero tre in Figura 4.4. Dovecot non è, quindi, responsabile della ricezione delle email dagli altri utenti, ma permette: *(i)* l'inoltro di email già memorizzate nella mailbox di un utente verso i client IMAP/POP3; *(ii)* la memorizzazione nell'apposito store locale di email in arrivo dall'MTA.

Dovecot supporta numerosi formati di mailbox, come *mbox*, *maildir*, *dbox*, solo per citarne alcuni. Dovecot lavora con *maildir*.

In *maildir* [9, 10], ogni utente ha la sua directory e ogni email è inserita in un file con un nome univoco (esistono altre soluzioni in cui più email sono inserite all'interno dello stesso file). Questo permette di trattare ogni email singolarmente senza affliggere altri messaggi. Un appunto da fare riguarda il fatto che lo standard di maildir non supporta pienamente il protocollo IMAP; perciò, Dovecot ha aggiunto delle estensioni, utilizzabili dai MUA che le supportano.

I nomi dei file vengono creati seguendo una determinata procedura. Il formato è del tipo `<base>:2,<flags>`. La base è composta da tre parti separate da un punto. La prima è il risultato della funzione `time()` all'arrivo, dunque, un numero intero che indica il numero di secondi trascorsi dalla mezzanotte del primo gennaio del 1970. La terza parte è composta dal risultato della funzione `gethostname()`. La parte intermedia è data dal *delivery identifier*, il quale deve essere unico. L'unicità è garantita dalla concatenazione di un numero sufficiente dei seguenti campi:

- `#n`, dove `n` è l'output in esadecimale della system call `unix_sequencenumber()`;
- `Xn`, dove `n` è l'output in esadecimale della system call `unix_bootnumber()`
- `Rn`, dove `n` è l'output in esadecimale della system call `unix_cryptorandomnumber()` (non supportata da tutti i sistemi operativi);
- `In`, dove `n` è, in esadecimale, il numero dell'inode del file corrente;

- `Vn`, dove `n` è il numero del device del file corrente;
- `Mn`, dove `n` è l'output in decimale della funzione `gettimeofday()`;
- `Pn`, dove `n` è il process ID in decimale;
- `Qn`, dove `n` è il numero delle consegne fatte dal processo corrente in decimale.

Le prime due voci dell'elenco precedente sarebbero sufficienti per garantire l'unicità, tuttavia molti sistemi operativi non supportano queste due system call.

Un esempio di base del nome di file, può essere:

```
1580984867.M636022P11704.localhost
```

Inoltre, alla base possono essere associati altri due campi:

- `S=<size>`, dove `size` indica la dimensione del file;
- `W=<vsize>`, dove `vsize` indica la dimensione virtuale del file come descritto in RFC822.

Se sono presenti questi due campi nel nome del file, Dovecot non ha bisogno di impiegare del tempo per calcolare la dimensione del file.

Dunque, un esempio di base completa è:

```
1580984867.M636022P11704.localhost,S=957,W=981
```

Il carattere "`:`" nel nome del file separa la parte univoca dalla restante. Il carattere "`2`" specifica la versione del campo *flag* dopo la virgola. Ad oggi esiste solo la versione relativa al valore 2. Se si trova il valore 1, significa che si sta usando una versione sperimentale. I flag principali sono:

- P (passed), indica che l'email è stata inoltrata;
- R (replied), indica una email a cui è stata inviata una risposta;
- S (seen), indica che l'email è stata aperta;
- T (trashed), l'email è stata spostata nel cestino;
- D (draft), indica una bozza;
- F (flagged), indica che l'email è stata etichettata da flag definiti dall'utente.

Oltre a questi flag, ne esistono altri non standard. Dovecot, nel file `dovecot-keywords`, mantiene il mapping tra questi, espressi da lettere dalla "`a`" alla "`z`", e il loro significato.

Un nome completo di file è, per esempio:

```
1580984867.M636022P11704.localhost,S=957,W=981:2,S
```

La mailbox di ogni utente registrato al servizio di posta elettronica è costituita dall'albero di directory mostrato in Figura 4.5.

Le directory `tmp`, `new` e `cur` sotto la cartella `Maildir` sono le più importanti. La prima viene utilizzata durante l'arrivo del messaggio; infatti, il file corrispondente a questo messaggio, con nome unico, viene creato dentro questa cartella. Una volta terminato l'arrivo di tutti i dati e composto il file completo, quest'ultimo viene spostato nella cartella `new`, dove rimane finché l'utente non apre un client di posta e viene scaricato il messaggio. Lo spostamento dell'email dalla cartella `tmp` alla cartella `new` può avvenire con l'operazione atomica di `rename` o sfruttando la sequenza

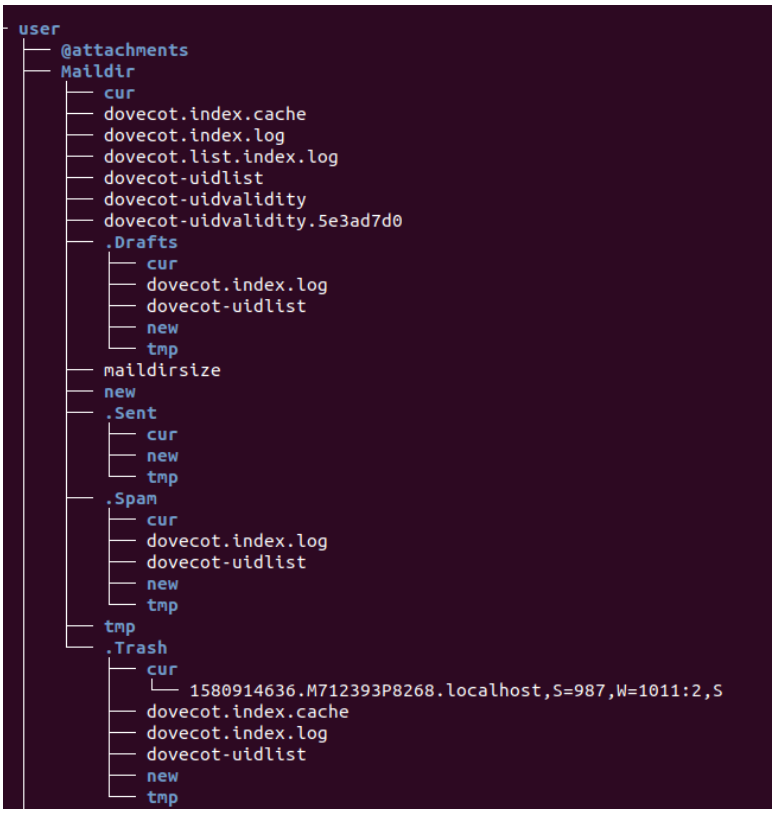

Figura 4.5. Organizzazione della mailbox di un utente

creazione-cancellazione di hard link. Questo passaggio è utile per far si che non vengano aperte email incomplete. Come anticipato, quando un client, con protocollo IMAP o POP3, richiede la mail, questa passa alla directory `cur`. Mantenere i messaggi nella directory `new` prima di trasferirli in `cur` è un modo utile semplicemente per notificare all'utente che ci sono un certo numero di messaggi non letti.

Oltre a queste cartelle, ci sono anche quelle nascoste (il cui nome inizia con ".") dove vengono mantenute le email inviate (`.Sent`), le email etichettate come spam (`.Spam`) e quelle spostate nel cestino (`.Trash`). Ciascuna di queste, a loro volta, è organizzata nelle sottocartelle `tmp`, `new` e `cur`.

Il formato Maildir presenta diversi vantaggi: per prima cosa, è un formato sicuro e le chance di corrompere i file sono ridotte al minimo; conoscendo la struttura delle cartelle, la localizzazione e il recupero dei file è semplice; non c'è bisogno di gestire il locking dei file perché questo è a carico del file system locale durante lo spostamento, l'aggiunta o la cancellazione di file; infine, esso può essere usato per NFS (Network File System). Gli svantaggi riguardano il fatto che ogni email è mantenuta in un file differente, quindi potrebbe essere oneroso gestire tanti piccoli file, inoltre non risulta efficiente la ricerca per testo nelle email perché ciò comporta l'apertura di tutti i file.

5

Integrazione finale: Ematex

Il presente capitolo descrive come tutto ciò che è stato precedentemente introdotto sia stato integrato per automatizzare il flusso di analisi delle email.

5.1 Obiettivo

Nell'Introduzione abbiamo evidenziato il concetto secondo il quale la resistenza di una catena è data dalla resistenza dell'anello più debole. Nell'ambito dell'Information Security, spesso l'uomo è ritenuto l'anello più debole di tale catena. Egli, interagendo con la tecnologia, può essere considerato una vulnerabilità da sfruttare al pari di quelle presenti nei software codificati in malo modo, nei protocolli mal progettati o nei sistemi mal configurati.

Dunque, un attore ostile potrebbe decidere di attaccare un'organizzazione sfruttando la sua vulnerabilità principale: il fattore umano. Perpetrare attacchi tecnologicamente sofisticati che sfruttano vulnerabilità o mal configurazioni dei sistemi informatici necessita, oltre che di attori profondamente preparati ed esperti, di risorse economiche e di tempo. Perciò, la vulnerabilità più conveniente da sfruttare è il fattore umano. Attacchi di questo genere non necessitano di conoscenze approfondite, non richiedono un eccessivo costo di inizializzazione, il rischio è relativamente basso e, se vanno a buon fine, il guadagno è alto. Il vettore d'attacco più utilizzato in questo contesto è rappresentato dalle email di phishing.

Dati questi presupposti, appare evidente come gli analisti di un SOC, tra le tante attività che svolgono, debbano anche controllare le email in arrivo nei sistemi di posta elettronica supervisionati.

Ad oggi, in molte organizzazioni le email vengono analizzate solo se ritenute sospette da chi le riceve, che provvede a contattare chi si dedica alla difesa dell'infrastruttura, o da sistemi SIEM, che generano alert in maniera automatica.

In commercio si trovano molti sistemi antispam, che filtrano le email in arrivo inserendo, in apposite directory, i messaggi considerati indesiderati. Tali sistemi, tuttavia, non sono sempre efficaci e possono generare molti falsi positivi, o ancora peggio (dal punto di vista della cybersecurity), falsi negativi. Gli attacchi più completi e sofisticati, come le *Advanced Persistent Threat (APT)* o le *Business Email*

Compromise (*BEC*, detti anche *CEO Fraud*), sono difficili da rilevare perché l'attaccante, attraverso una fase iniziale, in cui utilizza tecniche di social engineering, riesce a generare email che sfuggono ai sistemi antispam e che potrebbero ingannare gli utenti meno attenti.

Il nostro obiettivo è quello di implementare un sistema del tutto automatico, che permetta l'analisi delle email in arrivo a tutti gli utenti del sistema difeso. Qualora vengano rilevate email sospette o pericolose, si innesca una catena di azioni che previene l'apertura di tali messaggi.

5.2 Progettazione e implementazione

La macchina con la quale abbiamo lavorato presenta le seguenti caratteristiche:

- processore Intel i3;
- 16 GB di memoria RAM;
- sistema operativo Ubuntu 18.04;
- stack di virtualizzazione KVM/QEMU, libvirt, virt-manager;
- interprete Python, Versione 3.6, con installatore di pacchetti *pip*.

I software e le piattaforme, di cui abbiamo ampiamente discusso nei capitoli precedenti, svolgono funzioni diverse all'interno di un SOC. Una sandbox viene utilizzata dagli analisti per comprendere come un file o URL interagisce con un sistema quando viene aperto/eseguito. La piattaforma MISP è utile per mantenere e scambiare informazioni su eventi di sicurezza avvenuti e studiati in passato per evitare di ripetere analisi già effettuate. TheHive e Cortex sono degli applicativi più "gestionali", in quanto permettono di coordinare e, in parte, automatizzare il lavoro degli operatori del SOC. In Figura 5.1 è mostrata l'architettura del sistema di analisi delle email.

In alto, è riportata l'organizzazione (semplificata) delle mailbox degli utenti all'interno della macchina server. Per implementare un sistema di posta elettronica abbiamo utilizzato Plesk. In un ambiente virtuale con sistema operativo Ubuntu 18.04, abbiamo installato Plesk, creato un dominio e alcuni utenti del servizio di email, come descritto nel Capitolo 4; abbiamo lasciato immutate le impostazioni di default, che prevedono l'utilizzo di Postfix (SMTP) e Dovecot (IMAP o POP3). A questo punto, la directory `/var/qmail/mailnames/<nome_dominio>/` contiene una cartella per ogni utente.

Più in basso è riportato lo schema delle componenti del programma che gestisce il flusso di analisi dei messaggi. Tale flusso è scandito in quattro passaggi:

1. ottenimento delle email;
2. parsing del testo delle email e confronto di URL con feed presenti su MISP.
3. estrazione allegati, controllo su MISP ed eventuale creazione di un task su CAPE Sandbox.
4. ottenimento dei risultati e, se necessaria, creazione di un case su TheHive.

Si noti come il nostro programma, in Python 3.6, sia costituito da quattro file principali. Partendo dall'ultimo, il file `Config.py` definisce una classe che legge il

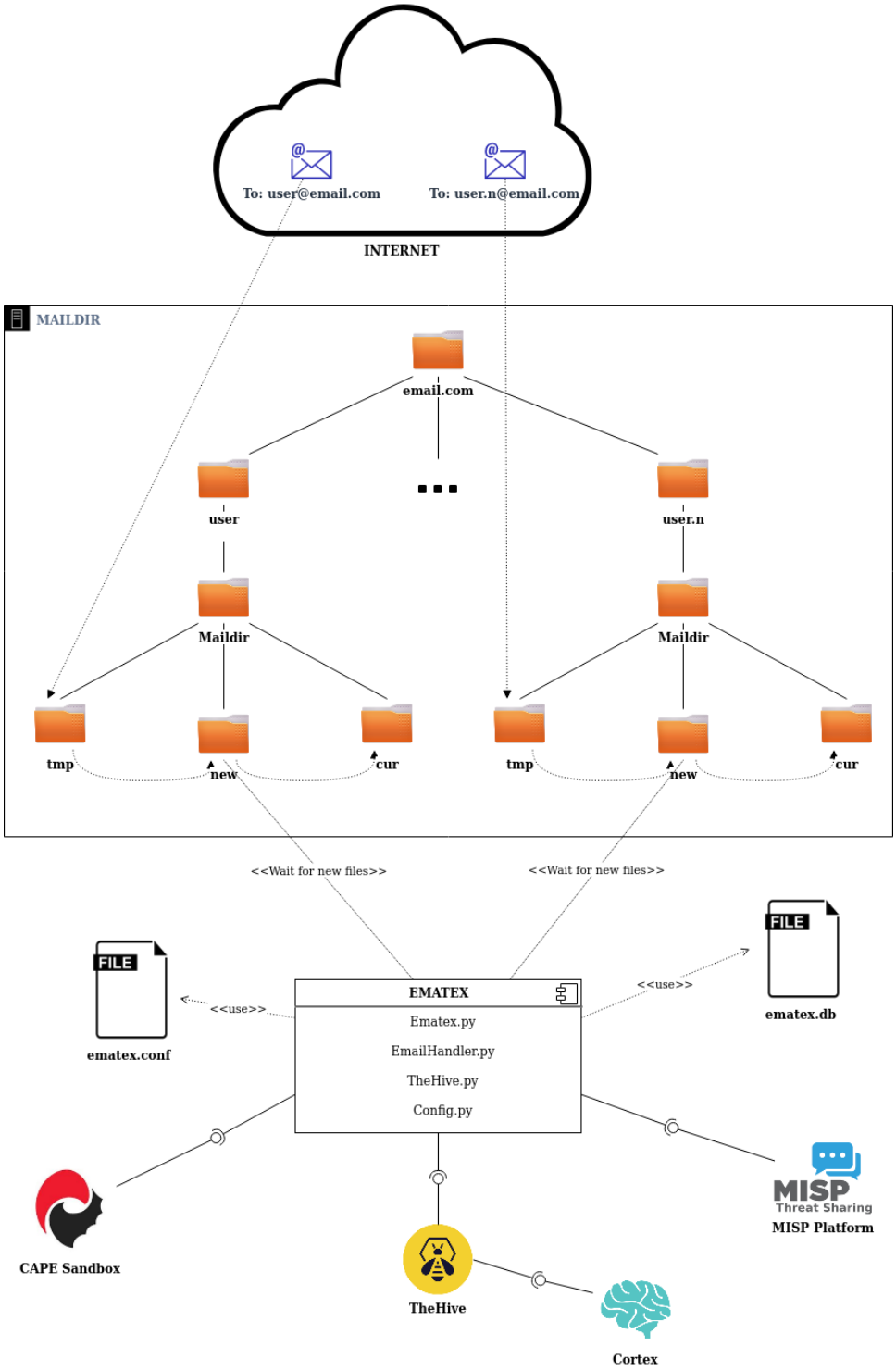

Figura 5.1. Architettura del sistema di monitoring delle email.

file di configurazione e importa tutte le coppie chiave-valore di ogni sezione in un dizionario. Il Listato 5.1 mostra lo schema di un file di configurazione.

```
1   [<SECTION1>]
2
3   <chiave1> = <valore1>
4   <chiave2> = <valore2>
5   ...
6
7   [<SECTION2>]
8
9   <chiave3> = <valore3>
10  ...
```

Listato 5.1. Esempio di file di configurazione

Il file `Ematex.py` è l'entry point dell'applicazione e si occupa di effettuarne il setup, importando le configurazioni, controllando la connessione con i software con cui collabora, fino ad avviare il ciclo di attesa e l'inserimento delle nuove email in una coda.

Il file `EmailHandler.py` contiene due classi, entrambe estensioni della classe `Thread`: `QueueExtractor` e `EmailHandler`. La prima estrae elementi dalla coda e li passa all'`EmailHandler`, avviandolo. La seconda rappresenta il thread che gestisce l'estrazione e l'analisi di link e di allegati in un'email.

5.2.1 Fase 1: preparazione e ottenimento dell'email

Per prima cosa, affinché l'applicativo possa svolgere il proprio compito, è necessario che siano attivi MISP, CAPE Sandbox e TheHive. Nel caso in cui, almeno uno tra i primi due non sia funzionante, il programma abortisce; se TheHive risulta spento o irraggiungibile, il programma può comunque proseguire ed elaborare le email in arrivo, tuttavia, qualora fosse necessario, non sarà possibile interagire con la piattaforma. Nella fase di inizializzazione, vengono importati i dati di un file di configurazione. Il vantaggio di mantenere alcune informazioni in un file di configurazione è quello di poterle modificare in qualsiasi momento, senza intaccare il codice sorgente. Tale file, chiamato `ematex.conf`, contiene:

- il dominio da monitorare;
- l'URL su cui sono in ascolto le API di CAPE;
- l'URL e l'APIkey di MISP;
- l'URL e l'APIkey di TheHive;
- il minimo valore di malscore per cui generare un case su TheHive.

Successivamente, è necessario escogitare un modo per rilevare l'arrivo di nuovi messaggi. Ne abbiamo trovati principalmente due: *(i)* codificare un client IMAP per ciascun utente, *(ii)* eseguire un listener di eventi nella mailbox di ogni utente.

Apparentemente, la prima alternativa sembra migliore perché, con gli ultimi aggiornamenti del protocollo IMAP, è stata introdotta la funzionalità *IDLE*, che permette al server di inviare notifiche in tempo reale a un client in ascolto. Per implementare questa scelta, dunque, è sufficiente eseguire un client per ogni utente, il quale rimarrà in attesa di una notifica da parte del server. In questo modo si evita il polling del client e la soluzione potrebbe risultare più efficiente. Ciò nondimeno,

poiché il client deve essere autenticato dal server, questa soluzione implica la conoscenza delle password di tutti gli utenti da monitorare. In aggiunta, perché client e server comunicano in rete, la soluzione non sembra scalabile in quanto, all'aumentare del numero di utenti, aumenterebbe il traffico, fino a rendere la rete un collo di bottiglia.

La seconda alternativa, invece, prevede l'implementazione di un applicativo che opera sul server stesso, senza utilizzo della rete. Questo, come mostrato in Figura 5.1, rimane in ascolto degli eventi registrati nel file system del server. In Python, la libreria `inotify` [6] implementa l'omonima funzionalità del kernel di Linux, che permette di inviare ad un processo notifiche sugli eventi registrati nelle directory scelte. Anche in questo caso, poiché è implementato un sistema di block-and-wait, si evita il polling. Quindi, è sufficiente un ciclo che registra un evento alla volta e che, nel mentre, si blocca, permettendo l'esecuzione di altri task.

Per i nostri scopi, abbiamo scelto di rimanere in attesa degli eventi sulla directory `new` di ciascun utente. Perciò, ogni volta che si registra la creazione di un nuovo file in queste directory, utilizzando la libreria `email` di Python, creiamo una tupla, costituita da destinatario, testo del messaggio e id dell'email, che inseriamo in una coda.

L'utilizzo della coda con thread produttore e thread consumatore permette al primo di non rimanere bloccato fino al termine dell'analisi di una mail, quanto piuttosto di tornare ad ascoltare eventi sulle cartelle, dopo aver inserito l'elemento in coda.

Una volta estratto l'elemento dalla coda, il `QueueExtractor` (consumatore) genera un ulteriore thread, chiamato `EmailHandler`.

Uno dei limiti di Python è il GIL (Global Interpreter Lock), il quale impedisce l'esecuzione parallela di più thread. Tuttavia, l'organizzazione in thread e la concorrenza che ne deriva risultano essere vantaggiosi, in quanto i thread sono spesso bloccati in attesa di operazioni di I/O.

5.2.2 Fase 2: estrazione e controllo di URL

Il Listato 5.2 mostra il contenuto di un file che rappresenta un'email inviata tramite Mozilla Thunderbird (ricordando che, utilizzando Maildir, si ha un file per ogni email).

```
1  Return-Path: <sender@loremipsum.com>
2  X-Original-To: receiver@loremipsum.com
3  Delivered-To: receiver@loremipsum.com
4  Received: from [192.168.122.102] (plesk [192.168.122.102])
5          by localhost.localdomain (Postfix) with ESMTPSA id 54B331419E0
6          for <receiver@loremipsum.com>; Mon, 10 Feb 2020 22:33:41 +0100 (CET)
7  Authentication-Results: plesk;
8          dmarc=none (p=NONE sp=NONE) smtp.from=loremipsum.com header.from=loremipsum.com;
9          spf=pass (sender IP is 192.168.122.102) smtp.mailfrom=sender@loremipsum.com smtp.helo=[192.168.122.102]
10 Received-SPF: pass (plesk: connection is authenticated)
11 To: receiver@loremipsum.com
12 From: "sender@loremipsum.com" <sender@loremipsum.com>
13 Subject: lorem ipsum
14 Message-ID: <90ade82a-28b8-8992-656a-bb63bfc19ed1@loremipsum.com>
15 Date: Mon, 10 Feb 2020 22:33:41 +0100
16 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
17  Thunderbird/68.4.1
18 MIME-Version: 1.0
19 Content-Type: multipart/mixed;
20  boundary="------------12E3B4844A31F44A3AFB26B1"
21 Content-Language: en-US
22
```

```
23  This is a multi-part message in MIME format.
24  --------------12E3B4844A31F44A3AFB26B1
25  Content-Type: text/plain; charset=utf-8; format=flowed
26  Content-Transfer-Encoding: 7bit
27
28  "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
29  tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim
30  veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea
31  commodo consequat. Duis aute irure dolor in reprehenderit in voluptate
32  velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint
33  occaecat cupidatat non proident, sunt in culpa qui officia deserunt
34  mollit anim id est laborum."
35
36
37  --------------12E3B4844A31F44A3AFB26B1
38  Content-Type: application/pdf;
39   name="Lorem-Ipsum.pdf"
40  Content-Transfer-Encoding: base64
41  Content-Disposition: attachment;
42   filename="Lorem-Ipsum.pdf"
43
44  JVBERi0xLjMKJeLjz9MKMiAwIG9iago8PAovQ3JlYXRpb25EYXRlIChEOjIwMTQwNjIOMDgz
45  OTEOLTA3JzAwJykKL01vZERhdGUgKEQ6MjAxNDA2MjQwODM5MTQtMDcnMDAnKQovUHJvZHVj
46
47  ...
48
49  Zm8gMiAwIFIKL0lEWzw5ZDM3MGIwYTA5M2RkOTFlZjgOODkyNDlkMTFhZmNmMz48OWQzNzBi
50  MGEwOTNkZDkxZWY4NDg5MjQ5ZDExYWZjZjM+XQo+PgpzdGFydHhyZWYKNTMONQolJUVPRgo=
51  --------------12E3B4844A31F44A3AFB26B1--
```

Listato 5.2. Esempio di email

Per i nostri scopi, terremo conto soltanto di alcuni campi dell'header. In particolare, questi sono:

1. `Message-ID`, per non analizzare più volte lo stesso messaggio con più destinatari;
2. `Content-Type`, per individuare il testo del messaggio;
3. `Content-Disposition`, per individuare gli allegati.

L'`EmailHandler`, prima di analizzare il contenuto dell'email in input, calcola l'hash MD5 del file e lo confronta con quelli presenti su un database SQLite per capire se la mail sia già stata analizzata. Per questo scopo, abbiamo utilizzato la liberia Pyhton `sqlitedict` [15], la quale rappresenta un wrapper del database `sqlite3` e che permette di interfacciarsi con i dati e le tabelle utilizzando dizionari.

Se viene appurato che quella mail non è stata precedentemente analizzata, utilizzando una regex, vengono cercati, nella sezione con `Content-Type` uguale a `plain/text`, tutti gli URL. Gli URL così determinati vengono confrontati con gli attributi degli eventi presenti su MISP[1].

Prima di passare all'analisi degli allegati, che nel file dell'email sono posti dopo il testo, se MISP ha fornito dei risultati, viene impostato un flag che indica la presenza di almeno un link malevolo.

5.2.3 Fase 3: estrazione e controllo degli allegati

In un'email, il `Content-Type` di un allegato può essere di varia natura, a seconda del tipo di file (nel Listato 5.2 si ha `application/pdf`). Perciò, discriminiamo un allegato dalle altre parti dell'email tramite il `Content-Disposition`, che contiene, proprio, la parola *"attachment"* (alla Riga 41 del Listato 5.2).

[1] Il rilevamento su MISP di un observable non è sempre indice di elemento malevolo. Ciò che si dovrebbe fare, di cui si discuterà nelle conclusioni, è analizzare approfonditamente i dati ottenuti da MISP per capire il ruolo dell'observable in quel particolare evento.

Gli allegati di una mail vengono scaricati e inseriti in apposite cartelle temporanee sul server. Anch'essi vengono sottomessi a MISP. Qualora non ci siano feed su questi file, essi sono sottomessi a CAPE Sandbox, tramite le relative API, generando ciascuno un task diverso.

A questo punto, viene avviato un ultimo thread, denominato `RetreiveFromCAPE`. Esso si occupa di reperire il report, interrogando CAPE ogni 30 secondi sullo stato di avanzamento del task. Se il task non fallisce entro un intervallo di timeout, si ottiene il report. Di quest'ultimo, l'unica informazione utile è il `malscore`, che indica se e quanto, in una scala da 0 a 10, è malevolo il file. Se tale valore supera una soglia prestabilita (e modificabile da file di configurazione) viene definito un flag che evidenzia la presenza di allegati malevoli.

5.2.4 Fase 4: creazione del case su TheHive

Quando tutti gli URL e gli allegati sono stati analizzati, se almeno uno dei due flag è stato impostato perché rilevato qualcosa di malevolo, si procede alla creazione di un case su TheHive.

La libreria `Thehive4py` permette a un'applicazione di interagire con la piattaforma TheHive. Un prerequisito obbligatorio è aver creato un utente di TheHive con permessi di lettura e scrittura e con le sue APIkey, inserite nel file di configurazione di Ematex. In prima battuta è necessario stabilire una connessione autenticata tra la piattaforma e l'applicazione stessa utilizzando l'APIkey. A questo punto è possibile generare un case, corredato di observable e di tutte le informazioni utili agli operatori del SOC, per effettuare analisi e arrivare alla chiusura del caso.

Per riassumere un po' il tutto, in Figura 5.2, viene riportato un diagramma di sequenze che ha l'obiettivo di descrivere a grandi linee il flusso di lavoro, facendo ulteriore chiarezza su quali siano i thread e le istanze in gioco e mostrando come questi interagiscano tra loro.

Finora non abbiamo mai discusso di come Cortex entri a far parte di questo flusso. In realtà, come mostrato in Figura 5.1, esso si interfaccia con TheHive e non viene in nessun modo controllato dal nostro programma. Questo perché, come ampiamente descritto nel relativo capitolo, Cortex mette a disposizione meccanismi automatici per analizzare o per attuare contromisure a eventi di sicurezza. Una prima analisi degli URL o dei file sospetti è già stata effettuata senza passare per Cortex e, in base all'esito di questa, si è aperto o meno un case. Cortex entra in gioco una volta aperto il case, permettendo agli analisti di eseguire analisi automatiche più approfondite o di reagire in maniera rapida ed efficace, direttamente dall'interfaccia di TheHive. Il connettore di cui si è parlato precedentemente è utile qualora si volessero eseguire ulteriori analisi con CAPE Sandbox modificando i parametri con cui sottomettere un task oppure qualora si volessero scansionare observable non ancora analizzati.

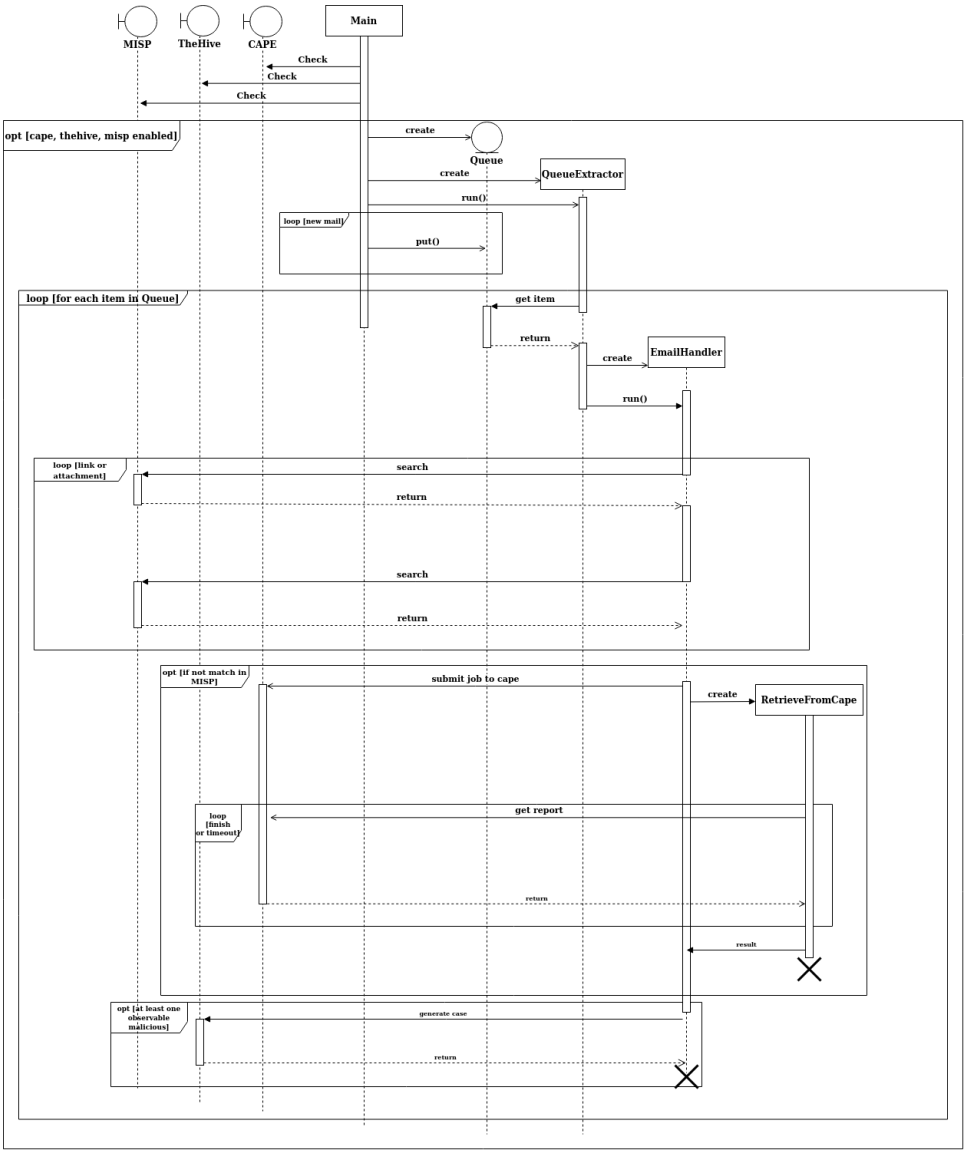

Figura 5.2. Diagramma delle sequenze del flusso di lavoro.

6

Discussione in merito al lavoro svolto

Nel presente capitolo riportiamo una discussione generale sul lavoro svolto e sui risultati conseguiti, evidenziandone i punti di forza e di debolezza.

6.1 SWOT Analysis

La soluzione proposta in questo elaborato non rappresenta il rimedio definitivo al phishing e alla diffusione di malware tramite email. Essa può essere considerata una prima linea di difesa, con vantaggi e svantaggi.

6.1.1 Punti di forza

La soluzione presenta diversi vantaggi e punti di forza. Abbiamo prodotto un sistema automatico di analisi delle email, con lo scopo di prevenire attacchi informatici che distribuiscono malware con messaggi di posta elettronica. Spesso, all'interno delle organizzazioni, le email vengono analizzate soltanto se il destinatario si insospettisce e le sottopone a chi ha il compito di difendere i sistemi informatici, o se vengono generati alert automatici da sistemi SIEM. Con Ematex, è possibile intercettare tutte le email che arrivano agli utenti di un certo dominio e sottomettere gli observable estratti a un sistema di analisi dinamica.

L'architettura di questo strumento risulta estremamente semplice. I software esterni integrati sono pochi, oltre ad essere molto utili ed efficaci per il proprio scopo. Il nostro applicativo Python non è complesso, essendo costituito da poche classi, e sfrutta la concorrenza per effettuare più analisi contemporaneamente. Inoltre, scegliendo di implementare un listener locale con *inotify* per il rilevamento di nuove email, si evita l'utilizzo della rete per questo scopo, sebbene le connessioni di rete siano, comunque, indispensabili per l'utilizzo delle API degli altri software (MISP, TheHive e CAPE Sandbox).

Gli operatori di un SOC che lavorano con TheHive possono facilmente integrare il programma con questa piattaforma; in alternativa, è ancora possibile utilizzare Ematex, ma è necessario controllare i log di output per conoscere il risultato di

un'analisi. Ovviamente, MISP e CAPE Sandbox sono indispensabili per il funzionamento di Ematex, dato che, senza di essi, non si avrebbe nessuna informazione sulla natura degli observable.

Un ultimo, ma non meno importante, punto di forza di questa soluzione è il suo costo di avvio praticamente nullo. Tutti gli strumenti utilizzati sono open source, scaricabili, installabili e utilizzabili gratuitamente, nonché, all'occorrenza, ulteriormente sviluppabili con nuove funzionalità.

6.1.2 Punti di debolezza

A dispetto dei vantaggi appena esposti, è doveroso sottolineare che non mancano punti di debolezza. Il più evidente è che Ematex non impedisce l'arrivo delle email ai destinatari. Esso, infatti, allo stato attuale, analizza ogni email in arrivo ma non le filtra in base ai risultati; queste arrivano intatte all'utente. In questo sistema, è onere degli operatori del SOC, una volta appurata la minaccia, informare del pericolo il destinatario interessato.

Alcuni svantaggi sono legati all'utilizzo della sandbox. Trascurando le debolezze intrinseche della dynamic malware analysis (ad esempio, sandbox evasion e alte problematiche analoghe), per prima cosa, questa necessita di una buona configurazione, altrimenti le analisi risultano inefficaci, con la possibilità di generare falsi positivi o falsi negativi. La sandbox provoca un inevitabile rallentamento del flusso, e più approfondite si desiderano le analisi, maggiore è il tempo necessario. In alcuni casi potrebbero essere impiegati anche diversi minuti per un singolo file. Inoltre, se la soluzione di sandbox non è ben progettata e dimensionata (in termini di numero di macchine disponibili e task concorrenti eseguibili), si rischia di rallentare ulteriormente il tutto in attesa di task in coda.

Un ulteriore svantaggio riguarda il controllo della presenza degli observable tra gli attributi nella piattaforma MISP. Anche in questo caso, trascurando le problematiche proprie della Threat Intelligence, Ematex utilizza MISP soltanto per capire se ci sono corrispondenze tra gli artefatti estratti dalle email e gli attributi degli eventi presenti sulla piattaforma. Tuttavia, in generale, la presenza di una corrispondenza non implica necessariamente la natura malevola del link o dell'allegato (per esempio, durante i test abbiamo notato come tra tutti gli attributi pubblici di MISP ci sia anche il link `www.google.com` che, ovviamente, di per sé non è malevolo); per questo l'utilizzo di MISP dovrebbe essere raffinato.

6.1.3 Opportunità

Ciò che rende appetibile questa soluzione è la costante crescita di attacchi informatici che sfruttano le email ed il fattore umano. I report prodotti da importanti associazioni e organizzazioni nazionali o internazionali sono tutti d'accordo nel constatare una crescita costante degli attacchi informatici. Sempre più spesso, in particolare a seguito di campagne di ransomware, si mette in evidenza come gli attaccanti sfruttino email malevole nella fase di *delivery* della loro *Cyber Kill Chain*.

Dunque, in questo contesto, la soluzione proposta rappresenta un prima utile linea di difesa per arginare gli attacchi di questo genere.

È da sottolineare il fatto che tutto avvenga in maniera automatica, senza l'interazione di utenti. All'interno di un SOC, infatti, è sempre più importante cercare di automatizzare il più possibile le operazioni frequenti e ripetitive, in modo da lasciare agli operatori la possibilità di dedicarsi alle attività in cui l'intervento umano è indispensabile.

6.1.4 Minacce

Uno degli ostacoli ai quali questa soluzione potrebbe andare incontro è legato al supporto dei software utilizzati. Essendo tutti open source e supportati da comunità di sviluppo più o meno ampie, c'è il rischio che con il tempo questi prodotti non vengano più aggiornati, migliorati e controllati fino a diventare inutilizzabili.

Un'altra minaccia, che potrebbe rendere inutile questo strumento, è lo sviluppo di soluzioni che sfruttano l'Intelligenza Artificiale. La nostra soluzione non usa tecniche di Intelligenza Artificiale e non filtra le email; questa si limita ad analizzare i messaggi in arrivo e a mostrare i risultati, in base ai quali attuare contromisure. In futuro potrebbero essere prodotti dei filtri di email, resi ancora più efficienti dal Machine/Deep Learning, che, in pochi istanti, riescono a discernere le email legittime, facendole pervenire al destinatario, rispetto a quelle malevole, che, al contrario, non saranno recapitate.

6.2 Lezioni Apprese

Durante lo studio, la progettazione e l'implementazione del sistema discusso nella presente tesi abbiamo appreso dei concetti importanti che vogliamo qui riportare.

Per quanto concerne la soluzione di sandbox, abbiamo appurato quali siano le difficoltà per implementare in maniera efficace una soluzione del genere. Poiché le macchine guest vanno preparate, se la soluzione scelta permette di utilizzare più sistemi operativi (come Cuckoo Sandbox), è necessario scegliere in anticipo quale adottare. Inoltre, si è constatato come versioni diverse di uno stesso sistema operativo producano risultati differenti, anche sugli stessi file; dunque, è necessario scegliere anche una versione adeguata del sistema operativo.

Se il software lo permette, si deve scegliere quale virtualizzatore utilizzare in base alle sue caratteristiche. Per esempio, sia Cuckoo che CAPE hanno moduli che consentono di scegliere tra più hypervisor, ma alcune funzionalità, come l'utilizzo di Volatility, sono possibili soltanto con KVM o Virtualbox.

Infine, è consigliabile fornire alle macchine virtuali un'interfaccia di rete NAT, rimandando la scelta se concedere o meno l'accesso ad Internet all'atto della sottomissione di un task. Sfruttando l'utility *rooter* di CAPE/Cuckoo, infatti, è possibile inserire in maniera automatica apposite regole di *iptables* nell'host, con le quali filtrare o meno il traffico da e verso la macchina guest.

Qualora si decidesse di fornire l'accesso ad Internet alla macchina guest, per evitare che siano rilevate signature erroneamente, è indispensabile eliminare (e Cuckoo/CAPE lo permette sia impostando delle regole del Berkeley Packet Filter, sia

aggiungendo dei domini in una whitelist) il traffico legittimo (o comunque non malevolo) dal dump dei pacchetti; in alternativa, si dovrebbero disattivare tutti i processi o i servizi che generano traffico di rete.

Per quanto riguarda la piattaforma TheHive, poiché nel nostro flusso vengono creati dei case tramite le API (*thehive4py*), non viene segnalata nessuna notifica sulla barra di navigazione in alto. Infatti, essa non contiene l'icona che segnala il numero di case; sono segnalati, invece, il numero di alert, il numero di nuovi task e il numero dei task presi in carico dell'utente autenticato.

Poiché il nostro flusso non genera alert, la creazione di un case potrebbe passare inosservata agli utenti della piattaforma. È consigliabile, dunque, che, durante la creazione di un case, venga attribuito a quest'ultimo un template, precedentemente definito, contenente i task da effettuare per chiudere tale case (nel nostro caso, un task potrebbe essere quello di informare il destinatario che la mail appena arrivata contiene un malware). In questo modo, una volta generato il case, TheHive segnalerà, sulla barra in alto, la presenza di nuovi task che è possibile prendere in carico.

7

Conclusioni e sviluppi futuri

L'azienda Negg s.r.l., presso la quale si sono svolte le attività finalizzate alla scrittura di questa tesi, sta progettando la costruzione di un Security Operation Center interno. Una delle attività dalle quali è sembrato essenziale partire è l'analisi delle email. Questo è ciò che ci ha spinto a implementare il sistema automatico di analisi di file e URL, presenti nei messaggi in arrivo agli utenti di un servizio di posta elettronica, decritto in questo elaborato. In particolare, dalle email si estraggono gli artefatti, che potrebbero essere vettori di malware; tali artefatti sono successivamente analizzati con la tecnica dell'analisi di malware dinamica, previo controllo sulla piattaforma MISP di Threat Intelligence, per appurare l'esistenza di analisi già effettuate in passato.

Il lavoro che ha portato alla soluzione decritta in questo elaborato è iniziato con lo studio del sistema di dynamic malware analysis, chiamato Cuckoo Sandbox. Dopo una fase iniziale di studio teorico del software, siamo passati all'installazione e configurazione, fino ad arrivare ai test pratici, utilizzando sia malware noti, sia file "puliti". Non essendo soddisfatti dei risultati che Cuckoo ci forniva, se confrontati con altre soluzioni di sandboxing online, abbiamo deciso di utilizzare CAPE Sandbox, il quale, pur essendo un progetto derivato da Cuckoo, ereditandone in larga parte l'architettura, sembrava essere più completo e affidabile.

Una volta compreso approfonditamente il funzionamento della sandbox e iniziati ad avere buoni risultati nei test, siamo passati allo studio delle piattaforme di TheHive Project. La stessa procedura, seguita per Cuckoo prima e CAPE dopo, si è ripetuta con Cortex e TheHive. Abbiamo valutato, dal punto di vista teorico, le caratteristiche di ciascuna delle due piattaforme e, successivamente, siamo passati a valutare il loro comportamento.

Per quanto concerne MISP, il nostro lavoro è stato più superficiale, in quanto per i nostri scopi era sufficiente installare la piattaforma, scaricare gli eventi pubblici e renderla disponibile a essere interrogata.

Infine, terminato lo studio dei singoli applicativi, abbiamo progettato e realizzato il nostro sistema, chiamato Ematex. Esso si pone come obiettivo di fungere da regista completamente autonomo del nostro processo di analisi delle email. I suoi compiti sono: il rilevamento dell'arrivo di nuove email, l'estrazione di file/URL, l'analisi di questi ultimi e, se una mail è ritenuta malevola, l'opportuna notifica.

L'integrazione di software per l'analisi di email presentata in questa tesi rappresenta una piccola parte di un processo ben più ampio e complesso che ha come obiettivo la prevenzione degli attacchi informatici condotti tramite email. Il nostro scopo era quello di implementare un sistema di analisi e rendere quest'ultimo il più automatico possibile. In parte ci siamo riusciti, ma sicuramente sono possibili a delle migliorie che si possono apportare alla nostra soluzione per renderla più efficiente ed efficace. In prima battuta, poiché il nostro sistema è costruito per proteggere gli utenti che utilizzano Windows, sarebbe interessante estendere le sue funzionalità su altri sistemi operativi, come Linux o Android. Ciò dipende dalla sandbox utilizzata, ma essendo CAPE open source e modulare, è sufficiente scrivere e integrare il codice dell'Analyzer per i nuovi sistemi operativi.

In seconda istanza, avendo evidenziato come l'analisi degli allegati tramite sandbox rallenti il nostro flusso, è possibile ovviare a questo problema implementando dei controlli preliminari più snelli, magari utilizzando il Machine Learning, che evitino la creazione di un task della sandbox per ogni nuovo allegato.

Un ultimo aspetto da tenere a mente è quello legato al recapito delle email. Nel nostro sistema, una email in arrivo, sia che venga giudicata legittima sia che sia considerata malevola, viene recapitata al destinatario. Sarebbe opportuno individuare un modo tramite il quale evitare che email etichettate come malevole vengano scaricate dal client di posta elettronica degli utenti.

Riferimenti bibliografici

1. Collaborative Research Into Threats. https://crits.github.io/.
2. Cortex Analyzers. https://github.com/TheHive-Project/Cortex-Analyzers/tree/master/analyzers.
3. Cuckoo Sandbox - Automated Malware Analysis. https://cuckoosandbox.org/.
4. Emerging Technology Analysis: SOAR Solutions. https://www.gartner.com/en/documents/3895089.
5. GOSINT - Open Source Threat Intelligence Gathering and Processing Framework. https://github.com/ciscocsirt/GOSINT.
6. Inotify 0.2.10. https://pypi.org/project/inotify/.
7. Install Plesk. https://get.plesk.com.
8. Installation Guide. https://github.com/TheHive-Project/TheHiveDocs/blob/master/installation/install-guide.md.
9. Maildir. https://wiki2.dovecot.org/MailboxFormat/Maildir.
10. Maildir. http://cr.yp.to/proto/maildir.html.
11. MineMeld Threat Intelligence Sharing. https://www.paloaltonetworks.com/products/secure-the-network/subscriptions/minemeld.
12. Qemu the fast! processor emulator. https://www.qemu.org/.
13. Repository github di cape sandbox. https://github.com/ctxis/CAPE.
14. Security Incident Response For The Masses. https://thehive-project.org/.
15. Sqlitedict 1.6.0. https://pypi.org/project/sqlitedict/.
16. Threat Intelligence Infographic. https://www.cpni.gov.uk/advice/cyber/Threat-Intelligence.
17. What is a SOC? https://www.paloaltonetworks.com/cyberpedia/what-is-a-soc.
18. Computer security incident handling guide. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-61r2.pdf, 2012.
19. Robert Mueller's Speach. https://archives.fbi.gov/archives/news/speeches/combating-threats-in-the-cyber-world-outsmarting-terrorists-hackers-and-spies, 2012.
20. Definition: Threat Intelligence. https://www.gartner.com/en/documents/2487216, 2013.
21. CNSS Instruction No. 4009. https://rmf.org/wp-content/uploads/2017/10/CNSSI-4009.pdf, 2015.
22. Live Migrating QEMU-KVM Virtual Machines. https://developers.redhat.com/blog/2015/03/24/live-migrating-qemu-kvm-virtual-machines/, 2015.
23. The five models of Security Operation Centers. https://www.gartner.com/en/documents/3155618/the-five-models-of-security-operation-centers, 2015.

24. https://securityboulevard.com/2018/03/certs-csirts-and-socs-after-10-years-from-definitions/, 2018.
25. https://searchsecurity.techtarget.com/tip/CERT-vs-CSIRT-vs-SOC-Whats-the-difference, 2019.
26. Csirts by country - interactive map. https://www.enisa.europa.eu/topics/csirts-in-europe/csirt-inventory/certs-by-country-interactive-map, 2019.
27. Phishing attacks: defending your organisation. https://www.ncsc.gov.uk/guidance/phishing, 2019.
28. A. S. Tanenbaum, D. J. Wetherall. *Computer Networks*. Pearson College Div, 2010.
29. M. Bromiley. *Threat Intelligence: What It Is, and How to Use It Effectively*, 2016.
30. C. Beard, S. Brown, A. Dulaunoy, J. Ginn, P. Stirparo. *Exploring the opportunities and limitations of current Threat Intelligence Platforms*, 2017.
31. C. Crowley, J. Pescatore. *The Definition of SOC-cess? SANS 2018 Security Operations Center Survey*, 2018.
32. C. Johnson, L. Badger, D. Waltermire, J. Snyder, C. Skorupka. *Guide to Cyber Threat Information Sharing*, 2016.
33. Clusit. *Rapporto Clusit sulla sicurezza ICT in Italia*, 2019.
34. E. Eilam. *Reversing. Secret of Reverse Engineering*. Wiley Publishing, Inc., 2005.
35. M Sikorski, A Honig. *Practical Malware Analysis. The Hands-On Guide to Dissecting Malicious Software*. No Starch Press, 2012.
36. IBM Security. *Six steps for building a robust incident response function*, 2017.
37. D. Shackleford. *CTI in Security Operations: SANS 2018 Cyber Threat Intelligence Survey*, 2018.
38. Cynthia Wagner, Alexandre Dulaunoy, Gérard Wagener, and Andras Iklody. Misp: The design and implementation of a collaborative threat intelligence sharing platform. In *Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security*, pages 49–56. ACM, 2016.
39. C. Zimmerman. *Ten Strategies of a World-Class Cybersecurity Operations Center*. MITRE, 2014.

Ringraziamenti

Ringrazio, e non sarà mai abbastanza, il Professor Domenico Ursino per la sua infinita disponibilità, per l'enorme passione che profonde nella sua professione e per avermi permesso di vivere un'esperienza speciale, impensabile senza il suo aiuto, per lo svolgimento della tesi.

Ringrazio infinitamente la mia famiglia. Non credo sia necessario elencarne tutti i motivi. È sufficiente scrivere che è semplicemente la mia più grande fortuna.

Ringrazio il mio Amico e compagno di avventure a Reggio Calabria, Paolo. Non fosse stato per lui e per la squisita persona qual è, sarebbe stato molto difficile per me affrontare il tirocinio che ha portato all'elaborazione di questa tesi.

Ringrazio la mia ragazza Lalli che, pur non riuscendo ad essermi vicina fisicamente per il periodo del tirocinio, è sempre stata molto presente. È incredibile come sia riuscita e riesca sempre a supportarmi in qualsiasi cosa io faccia, facendomi sempre sentire apprezzato. Grazie per aver sopportato la mia assenza e per aver aspettato il mio ritorno.

Ringrazio la famiglia Negg per l'accoglienza, la fiducia e la possibilità concessami di entrare in contatto con la loro realtà. Avendo toccato con mano la loro professionalità e cordialità non credo sia una semplice frase di rito affermare che sono "leader" in entrambi i settori. Ringrazio Paolo e Rossella per aver supervisionato il mio lavoro, nonostante i numerosissimi impegni in agenda ogni giorno, e per avermi insegnato molto. Ringrazio Francesco e Rodolfo per avermi concesso questa opportunità per me importantissima e per aver sempre speso belle parole nei miei confronti.

Ringrazio i miei amici (in ordine alfabetico per non fare torto a nessuno) Filippo, Mattia e Simone perché mi hanno dimostrato e continuano a dimostrarmi quotidianamente il senso dell'amicizia vera e quanto sia importante avere al proprio fianco persone su cui poter contare in ogni situazione.