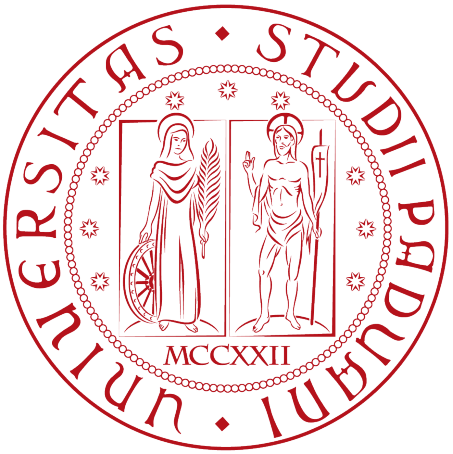
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



Deployment di strumenti di sicurezza in IaaS

Tesi di laurea triennale

Relatore

Prof. Mauro Conti

Co-relatore

Dr. Federico Turrin

Laureando

Dardan Kokollari

Anno Accademico 2020-2021

Dardan Kokollari: *Deployment di strumenti di sicurezza in IaaS,* Tesi di laurea triennale,
© Gennaio 2022.

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di circa 320 ore, dal laureando Dardan Kokollari presso l'azienda PricewaterhouseCoopers Business Services Srl.
Lo scopo di questo progetto di stage è stato effettuare il *deployment* della strumentazione utilizzata dal *team* di *Cybersecurity & Privacy*, con particolare attenzione negli *engagement* di *Cyber Incident Response* e *Cyber Threat Intelligence*.

Ringraziamenti

Ringrazio infinitamente i miei genitori per il supporto e l'aiuto datomi con cui è stato possibile tutto ciò.

Esprimo la mia gratitudine al Prof. Mauro Conti, relatore della mia tesi, a Federico Turrin, dottorando del gruppo di ricerca del Prof. Conti, a Matteo Brunati, tutor aziendale per l'aiuto e il sostegno fornitomi durante lo stage e durante la stesura della tesi.

Infine non basterebbero le righe per ringraziare i miei amici, chi più lontani e chi più vicini, per gli anni passati insieme, le storie e i momenti condivisi. Un grazie speciale a coloro che son stati una parte importante di questo periodo di crescita.

Padova, Gennaio 2022 Dardan Kokollari

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

PricewaterhouseCoopers SpA (PwC) è una società per azioni che ha per oggetto sociale la revisione, nonché ogni altra attività, anche consulenziale, inerente, collegata o correlata alle questioni di natura o di contenuto contabile. L'obiettivo che si pone durante lo svolgimento delle proprie attività di business è avere uno scopo chiaro e definito al fine di essere coerente e consentire un maggiore impatto sia come organizzazione che come professionisti. PwC interpreta un ruolo importante nel guidare i propri clienti all'interno di sistemi complessi, aiutando le imprese, l'economia, la comunità e la società in senso lato. In questo stage vengono installati e configurati alcuni strumenti per dare supporto ai clienti, in caso sia necessario fare Incident Response e Threat Intelligence.

1.1 L'azienda

Figura 1.1: Logo PwC

Fonte: https://www.pwc.com/it/it/

PwC è un *network* con oltre 284.000 professionisti in 155 Paesi del mondo, di cui oltre 6.000 in Italia. Il *network* PwC è costituito da società che sono entità legali separate.

Le *firm* che compongono il *network* PwC forniscono servizi di revisione, fiscali e di consulenza. Nel dettaglio, i servizi offerti si dividono in:

* *Audit and assurance*: l'obiettivo è offrire servizi di rendicontazione aziendale per i clienti al fine di per mantenere la fiducia dell'azienda nel sistema finanziario;
* *Consulting*: l'obiettivo è aiutare i clienti a costruire organizzazioni efficaci, innovare, crescere e ridurre i costi. Sostenere le aziende nella progettazione, nella gestione e nell'esecuzione di un cambiamento positivo e vantaggioso;

* *Deals*: l'obiettivo è realizzare il potenziale dalle fusioni, acquisizioni, cessioni e transazioni sui mercati dei capitali;
* *Tax and Legal*: l'obiettivo è supportare e assistere i clienti nazionali e internazionali in molteplici aree di specializzazione in materia legale, fiscale e HR/giuslavorista.

In tale contesto, tra i servizi professionali di *Consulting*, rientrano anche i servizi di *Cyber Security*. Le aree delle soluzioni dei servizi di *Information Security* sono:

* *Identity* i cui servizi si dividono in:
  - *Identity Governance & Administration Access Management* → il servizio operativo *Identity & Access Management* (IAM) di PwC fornisce supporto operativo quotidiano per gestire, controllare e certificare l'accesso degli utenti attraverso le funzionalità IAM. Come parte di questo servizio, vengono forniti anche il monitoraggio della stabilità / integrità del sistema, eseguiti miglioramenti del sistema, facilitate le campagne di certificazione degli accessi ed eseguiamo report IAM;
  - Gestione degli accessi privilegiati → il servizio *Privileged Access Management* (PAM) di PwC supporta le operazioni quotidiane della soluzione PAM dei clienti e delle relative funzionalità. Il tutto al fine di eseguire, mantenere, monitorare e migliorare continuamente le capacità del cliente in tutta l'azienda.
* *Vulnerability Management*: il servizio operativo *Vulnerability Management* (VM) di PwC offre la gestione *end-to-end* del programma VM di un cliente, inclusi scansione, analisi, prioritizzazione delle vulnerabilità, assistenza per la riparazione, reporting, metriche e manutenzione della tecnologia. PwC apporta valore a questo servizio aiutando un cliente a creare una VM e un programma di *governance*, oltre a eseguire integrazioni tecniche con vari sistemi client (ad esempio, ITSM, CMDB, GRC, SIEM);
* *Threat Detection & Response*: il servizio operativo *Threat Detection & Response* di PwC fornisce operazioni di monitoraggio e risposta di livello 1 (L1) - Livello 3 (L3) 24 ore su 24, 7 giorni su 7 e capacità di ingegneria tecnologica. Ciò include l'accesso alla libreria di casi d'uso di PwC e lo sviluppo di casi d'uso personalizzati;
* *High Volume Assessments*: il servizio operativo di PwC *High Volume Assessments* fornisce la valutazione come servizio completando grandi quantità di valutazioni (ad esempio TPRM – *Third Party Risk Management*) all'anno in tempo. Le capacità operative di PwC riflettono decenni di esperienza in materia di sicurezza informatica e privacy e sfruttano un modello di fornitura di servizi globale e centralizzato per fornire qualità con ogni valutazione eseguita.

L'amministrazione della Società è affidata a un Consiglio di Amministrazione, che può essere composto da un numero di membri variabile da 3 a 9, eletti dall'assemblea degli azionisti, scegliendo tra i soci. Al Consiglio di Amministrazione sono conferiti i più ampi poteri per la gestione della società, inclusa la responsabilità dello sviluppo e dell'implementazione delle direttive aziendali e della strategia. Il Consiglio di Amministrazione ha inoltre la responsabilità complessiva del sistema di controllo interno (comprensivo dei controlli relativi alla qualità), e del suo riesame periodico per valutarne

l'efficacia. Il Consiglio di Amministrazione rimane in carica per tre esercizi. Tutti i membri del Consiglio hanno la rappresentanza legale della società.

1.2 Organizzazione della tesi

Il capitolo 2 riporta il piano di lavoro concordato con il relatore e l'azienda.

Il capitolo 3 descrive l'ambiente e gli strumenti utilizzati.

Il capitolo 4 riguarda l'analisi dei requisiti del lavoro eseguito durante lo stage.

Il capitolo 5 approfondisce il processo di installazione, configurazione e distribuzione per raggiungere l'obiettivo finale dello stage.

Il capitolo 6 riporta i test effettuati per verificare il corretto funzionamento degli strumenti configurati e distribuiti.

Il capitolo 7 riporta il consuntivo finale e lo stato di raggiungimento degli obiettivi.

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

- i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*;
- gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del documento, p.es. API;
- una fonte viene indicata con un numero tra parentesi quadrate che rimanda a una descrizione più approfondita in calce all'elaborato: TheHive [14];
- sotto la descrizione di una figura viene riportata la fonte ove possibile.

Capitolo 2

Descrizione dello stage

In questo capitolo viene riportato il Piano di Lavoro concordato con il relatore e l'azienda.

2.1 Scopo dello stage

Lo scopo di questo progetto di stage è stato di effettuare il deployment della strumentazione utile a:

* l'analisi e l'investigazione degli incidenti nelle attività di Incident Response, con cui rispondere e gestire gli attacchi informatici subiti, per preparare meglio le difese dell'organizzazione per il futuro;

* lo studio e la valutazione dei gruppi criminali e delle tattiche, tecnice e procedure (TTP) possibili al fine di aiutare a mitigare i potenziali attacchi informatici (attività anche conosciuta come Threat Intelligence).

Ogni requisito è stato acquisito attraverso l'esecuzione di diverse interviste al team di *Cybersecurity & Privacy*, considerando anche eventualmente i servizi che possono assistere alle attività progettuali di Ethical Hacking. Lo studente ha avuto il compito di comprendere le necessità del team, studiando gli strumenti già individuati/utilizzati, valutandone delle alternative. Infine ha effettuato il rilascio della strumentazione proposta, ovvero il pacchetto TheHive [14] comprensivo delle soluzioni Cortex [1] e MISP [12], eseguendone l'installazione, la configurazione e la messa in funzione. Il progetto infine è stato integrato con il servizio cloud *Microsoft Azure*.

2.2 Modalità di svolgimento dello stage

La modalità di svolgimento dello stage è stata duale. Le attività sono state eseguite principalmente da remoto ma è stato comunque previsto un incontro settimanale presso la sede dell'azienda. Lo studente ha avuto la possibilità di discutere quotidianamente con il tutor aziendale per qualsiasi tipo di problema o dubbio. Sono stati previsti incontri settimanali col team di *Cybersecurity & Privacy* per verificare lo stato di avanzamento, chiarire eventualmente gli obiettivi e aggiornare il piano di lavoro stesso. Lo studente ha lavorato, secondo un contratto full-time, dal lunedì al venerdì, dalle ore 9:00 alle ore 18:00. Tale orario è stato comprensivo di un'ora di pausa pranzo.

2.3 Contenuti formativi previsti

Durante questo progetto di stage lo studente ha avuto l'occasione di acquisire competenze di DevSecOps e consulenziali nel campo della *Cybersecurity*. Lo studente ha potuto approfondire le sue conoscenze con lo studio della documentazione a supporto degli strumenti valutati e utilizzati, congiuntamente con le risorse di *e-learning* messe a disposizione dalla società.

2.4 Prodotti attesi

Tabella 2.1: Prodotti attesi

Tipo	Nome	Descrizione
Documento	Requisiti di strumentazione	Documento che dettaglia i requisiti funzionali e non funzionali raccolti attraverso il confronto con l'azienda ed emersi durante la fase di analisi
Documento	Requisiti hardware e software per il deployment	Documento che definisce i requisiti hardware e software per il deployment della strumentazione alternativa proposta
Documento	Specifiche SW	Diagramma architetturale, diagrammi di flusso
Documento	README	File *markdown* che spiega passo passo come effettuare il deployment della strumentazione

2.5 Pianificazione del lavoro

2.5.1 Pianificazione settimanale

- **28/06/21 – 30/06/21 - Introduzione e analisi (20 ore)**
  - Introduzione al team;
  - Acquisizione credenziali e strumenti di lavoro assegnati;
  - Incontro con i membri del team per discutere i requisiti di strumenti da utilizzare negli *engagement* di Ethical Hacking, *Cyber Incident Response*, *Cyber Threat Intelligence*;
  - Presa visione dell'infrastruttura esistente;
- **30/06/21 – 09/07/21 - Formazione e studio strumentazione (60 ore)**
  - Formazione sulle tecnologie adottate;
  - Valutazione dei requisiti minimi hardware e software;
- **12/07/21 – 21/07/21 - Valutazione alternative e redazione analisi (60 ore)**
  - Studio di tecnologie simili e/o architetture differenti;
  - Confronto fra le tecnologie attualmente in uso e le tecnologie da proporre;

* **21/07/21 – 27/07/21 - Definizione requisiti (30 ore)**
  - Definizione requisiti per l'implementazione delle tecnologie proposte in alternativa;
* **27/07/21 – 30/07/21 - Disegno architetturale (30 ore)**
  - Disegno dell'architettura di rete/software per il deployment della tecnologia alternativa;
* **02/08/21 – 10/08/21 - Deployment strumentazione (50 ore)**
  - Configurazione delle tecnologie proposte in alternativa;
  - Installazione delle tecnologie proposte in alternativa;
* **10/08/21 – 18/08/21 - Documentazione (50 ore)**
  - Redazione dei documenti tecnici delle soluzioni proposte;
* **18/08/21 – 20/08/21 - Conclusione (20 ore)**
  - Discussione delle tecnologie proposte;
  - Demo;

2.6 Obiettivi

2.6.1 Notazione

Si farà riferimento ai requisiti secondo le seguenti notazioni:

* *O* per i requisiti obbligatori, vincolanti in quanto obiettivo primario richiesto dal committente;
* *D* per i requisiti desiderabili, non vincolanti o strettamente necessari, ma dal riconoscibile valore aggiunto;
* *F* per i requisiti facoltativi, rappresentanti valore aggiunto non strettamente competitivo.

Le sigle precedentemente indicate saranno seguite da una coppia sequenziale di numeri, identificativo del requisito.

2.6.2 Obiettivi fissati

Si prevede lo svolgimento dei seguenti obiettivi:

* Obbligatori
  - *O1*: stesura del documento "Requisiti di strumentazione".
  - *O2*: stesura del documento "Requisiti hardware e software per il deployment".
  - *O3*: stesura del documento "Specifiche SW".
  - *O4*: stesura del documento "README".

  - *O5*: installazione e configurazione di una soluzione di *Incident Management* (p.es. TheHive).
  - *O6*: installazione e configurazione di una soluzione di *Communication Management* e *Document Sharing* per l'Incident Response (p.es. Nextcloud).
  - *O7*: installazione e configurazione di una soluzione di analisi collaborativa per l'*IR* (p.es. Timesketch).

* Desiderabili
  - *D1*: installazione e configurazione di una soluzione di Threat Intelligence (p.es. OpenCTI).
  - *D2*: installazione e configurazione di una soluzione di collaborazione per Ethical Hacking (p.es. Dradis Framework).
  - *D3*: esposizione del lavoro di tesi ad un meeting interno "Lunch&Learn".
* Facoltativi
  - Stabiliti durante il periodo di stage.

Capitolo 3

Strumenti

In questo capitolo si fornisce una descrizione dell'ambiente e degli strumenti utilizzati.

3.1 TheHive Project

Figura 3.1: Logo TheHive Project

Fonte: https://github.com/TheHive-Project/TheHive

TheHive [14] è una piattaforma *open source*, gratuita e *scalabile* di Incident Response 3-in-1 progettata per facilitare l'operato di SOC, CSIRT, CERT e di qualsiasi altro professionista della *Information Security* che si occupa di incidenti di *security* da indagare o agire rapidamente. È possibile sincronizzarlo con una o più istanze di MISP [12] per avviare indagini sugli eventi di MISP. Non di meno, l'esportazione dei risultati di un'indagine può avvenire come eventi di MISP per aiutare i colleghi a rilevare e reagire agli attacchi pervenuti. Inoltre, quando TheHive viene utilizzato insieme a Cortex, gli analisti e i ricercatori di *security* possono facilmente analizzare decine se non centinaia di observable.

3.1.1 Overview

TheHive si specializza nei punti di focus di seguito elencati.

* Collabora con gli analisti e gli *stakeholder*:
  - più analisti possono lavorare insieme e simultaneamente sullo stesso caso;
  - grazie al *live stream* di TheHive tutti possono sapere cosa succede osservando la piattaforma in tempo reale.

* Elabora le informazioni ottimizzando le attività:
  - ogni investigazione corrisponde ad un caso che può essere creato da zero oppure dagli eventi di MISP, alerts SIEM, reports di email o qualsiasi altra sorgente di eventi di sicurezza:
  - ogni caso può essere suddiviso in più *tasks* a cui può essere dato un tipo anche grazie al *template engine* di TheHive;
  - i *template* possono anche essere usati per associare metriche a tipologie specifiche di casi, identificando le investigazioni che richiedono più tempo e cercando di automatizzare i *tasks* che risultano più tediosi.
* Analizza gli observable, ovvero gli eventi sospetti rilevati da una rete:
  - TheHive ha l'abilità di identificare automaticamente gli observable che sono stati visti in precedenza;
  - gli observable possono anche essere associati a TLP (*Traffic Light Protocol*) o alla sorgente fornita;
  - gli analisti possono anche segnare gli observable come IoC e isolarli per poi esportarli in SIEM o altri archivi di dati;
  - gli analisti possono analizzare centinaia di observable in un paio di click usando uno o più istanze di Cortex [1]: DomainTools, VirusTotal, PassiveTotal, o qualsiasi altro pacchetto disponibile al sito `https://github.com/TheHive-Project/Cortex-Analyzers`.

3.1.2 Funzionalità

* Multi-tenant. TheHive fornisce supporto multi-tenant. Consente le seguenti strategie:
  - multi-tenant in silos: molte organizzazioni possono essere definite senza consentire loro di condividere dati;
  - multi-tenant collaborativa: è possibile consentire a un insieme di organizzazioni di collaborare su casi/attività/observable specifici, utilizzando profili utente definiti e personalizzati (RBAC).
* RBAC. TheHive fornisce una serie di autorizzazioni e diversi profili utente preconfigurati:
  - **admin**: permessi amministrativi completi sulla piattaforma; non può gestire alcun caso o altri dati relativi alle indagini;
  - **org-admin**: gestisce gli utenti e tutta la configurazione a livello di organizzazione, può creare e modificare i casi, le attività, gli observable ed eseguire gli *analyzers* e *responders*;
  - **analista**: può creare e modificare i casi, le attività, gli observable ed eseguire gli *analyzers* e *responders*;
  - **sola lettura**: può solo leggere, dettagli dei casi, le attività e gli observable.

  Gli amministratori possono creare nuovi profili dalla piattaforma stessa.
* Autenticazione. TheHive attualmente fornisce supporto ai seguenti metodi di autenticazione:

  - account locali;
  - *Active Directory*;
  - LDAP;
  - *Basic Auth*;
  - *chiavi API*;
  - *OAUTH2*;
  - *Multi Factor Authentication*.

* Statistiche e *dashboard*. TheHive è dotato di un potente modulo di statistiche che consente di creare *dashboard* significative per guidare le attività e supportare le richieste di budget.

3.1.3 Integrazioni

* MISP: TheHive può essere configurato per importare eventi da una o più istanze di MISP. È inoltre possibile utilizzare TheHive per esportare i casi come eventi di MISP su uno o più istanze di MISP.

* Cortex: pensato e sviluppato per integrarsi con TheHive. E' possibile utilizzare uno o più istanze di Cortex per analizzare gli observable su larga scala.

* E molti altri: è presente una *repository* contenente tutti i dettagli noti e i riferimenti sulle integrazioni esistenti al seguente indirizzo: `https://github.com/TheHive-Project/awesome`.

3.2 Cortex

Figura 3.2: Logo Cortex

Fonte: `https://github.com/TheHive-Project/Cortex`

Cortex [1] consente di analizzare gli observable, come indirizzi IP, e-mail, URL, nomi di dominio, file o *hash*, uno per uno o in modalità *bulk* utilizzando un'unica interfaccia web.
L'interfaccia web funge da *frontend* per numerosi *analyzer*, che sono applicazioni autonome gestite ed eseguite attraverso il Cortex *core engine*, eliminando la necessità di integrarli da soli durante l'analisi. Gli analisti possono anche utilizzare l'API REST di Cortex per automatizzare parti della loro analisi.

3.2.1 Integrazioni

* Cortex e TheHive: TheHive consente di analizzare decine o centinaia di observable in pochi click sfruttando una o più istanze di Cortex a seconda delle esigenze OPSEC e dei requisiti di performance.
* Cortex e MISP: Cortex può essere integrato con MISP nei seguenti modi:
  - Cortex può invocare i moduli di MISP;
  - MISP può invocare gli *analyzer* di Cortex.

3.3 MISP

Figura 3.3: Logo MISP

Fonte: https://github.com/MISP/MISP

MISP [12], Malware Information Sharing Platform and Threat Sharing, è una soluzione software *open source* per la raccolta, l'archiviazione, la distribuzione e la condivisione di indicatori e minacce di sicurezza informatica sull'analisi degli incidenti e del *malware*. MISP è progettato per gli analisti di incidenti, professionisti della sicurezza e ICT per supportare le loro operazioni quotidiane nella condivisione di informazioni strutturate in modo efficiente. L'obiettivo di MISP è favorire la condivisione di informazioni strutturate all'interno e all'esterno della comunità della *security*. Lo strumento fornisce funzionalità per supportare lo scambio di informazioni ma anche il consumo di tali informazioni da parte di *Network Intrusion Detection Systems* (NIDS), *Learning Intrusion Detection System* (LIDS) e strumenti di analisi dei log, SIEM.

3.3.1 Funzionalità

* Dispone di un efficiente *database* IoC e di indicatori che consentono di archiviare informazioni tecniche e non tecniche su campioni di *malware*, incidenti e aggressori.
* Correla automaticamente i dati per trovare relazioni tra attributi e indicatori da *malware*, campagne di attacco o analisi. Il motore di correlazione include la correlazione tra attributi e funzioni più avanzate come la correlazione di *hashing Fuzzy* (p.es. *ssdeep*) o la corrispondenza dei blocchi CIDR.
* Utilizza un modello di dati flessibile in cui oggetti complessi possono essere espressi e collegati tra loro per *Threat I0ntelligence*, incidenti o elementi connessi.
* Facilita la condivisione dei dati utilizzando diversi modelli di distribuzione. È possibile sincronizzare automaticamente eventi e attributi tra diverse istanze MISP. Vi sono funzionalità di filtraggio avanzate per soddisfare la politica di condivisione di ciascuna organizzazione.

* Dispone di un'interfaccia utente intuitiva che consente di creare, aggiornare e collaborare su eventi e attributi/indicatori. La navigazione è ideata per consultare gli eventi e le loro correlazioni, anche grazie al grafico degli eventi usato per creare e visualizzare le relazioni tra oggetti e attributi. Vi sono funzionalità di filtraggio avanzate ed elenchi di avvisi per aiutare gli analisti a contribuire e limitare il rischio di falsi positivi.

* Memorizza i dati in un formato strutturato (consentendo l'uso automatizzato del *database* per vari scopi) con un ampio supporto di indicatori di *Cyber Security* con indicatori di frode come nel settore finanziario.

* Possibilità di esportare in output OpenIoC, testo normale, *CSV*, *MISP XML* o *JSON* da integrare con altri sistemi (ID di rete, ID degli host, strumenti personalizzati), formato *Cache* (utilizzato per strumenti forensi), STIX *1*/STIX *2* (*XML* e *JSON*), esportazione NIDS. Molti altri formati possono essere facilmente aggiunti tramite i moduli MISP.

* Possibilità di importare in blocco, in *batch*, da OpenIoC, GFI SandBox, *ThreatConnect CSV*, con il formato standard MISP o STIX *1.1/2.0*. Molti altri formati possono essere facilmente aggiunti tramite i moduli MISP.

* Dispone di API flessibile per integrare MISP con le soluzioni personalizzate. MISP è in *bundle* con *PyMISP* che è una libreria *Python* flessibile per recuperare, aggiungere o aggiornare gli attributi degli eventi, gestire campioni di *malware* o cercare attributi. Non di meno è disponibile un'esaustiva API *restSearch* per cercare facilmente gli indicatori ed esportarli in tutti i formati supportati da MISP.

* Tassonomia regolabile per classificare e contrassegnare gli eventi seguendo schemi personalizzati di classificazione. La tassonomia può essere locale per il proprio MISP ma anche condivisibile tra diverse istanze di MISP.

* Dispone di vocabolari di *intelligence* chiamati *MISP galaxy* e messi in *bundle* con attori di minacce esistenti, *malware*, *Remote Access Trojan* (RAT), *ransomware* che possono essere facilmente collegati con eventi e attributi in MISP.

* Possibilità di espandere MISP con i propri servizi in Python.

* Possibilità di crittografare e firmare le notifiche tramite *GnuPG* e/o *S/MIME* a seconda delle preferenze dell'utente.

* Architettura *publish-subscribe* in tempo reale per ottenere automaticamente tutte le modifiche (p.es. nuovi eventi, indicatori, avvistamenti o tag).

3.4 Docker

Figura 3.4: Logo Docker

Fonte: https://www.docker.com/

Docker [3] è una piattaforma per lo sviluppo, la distribuzione e l'esecuzione di applicazioni. Docker consente di separare le applicazioni dall'infrastruttura in modo da poter distribuire rapidamente il software. Con Docker, si può gestire l'infrastruttura nello stesso modo in cui si gestiscono le applicazioni. Sfruttando le metodologie di Docker per il test e la distribuzione rapida del codice, si possono ridurre notevolmente il ritardo tra la scrittura del codice e l'esecuzione in produzione.
Docker offre la possibilità impacchettare ed eseguire un'applicazione in un ambiente isolato chiamato *container*. L'isolamento e la sicurezza consentono di eseguire molti *container* contemporaneamente su un determinato host. I *container* sono leggeri perché non richiedono il carico aggiuntivo di un *hypervisor*, ma vengono eseguiti direttamente all'interno del *kernel* della macchina host. Ciò significa che si possono eseguire più *container* su una data combinazione di hardware rispetto a utilizzare macchine virtuali. Si possono persino eseguire *container* Docker all'interno di macchine host che sono in realtà macchine virtuali.

3.5 Traefik

Figura 3.5: Logo Traefik

Fonte: https://github.com/traefik/traefik

Traefik [16] (pronunciato *traffic*) è un *reverse proxy* HTTP e *bilanciatore del carico* che semplifica la distribuzione dei microservizi. Traefik si integra con i componenti dell'infrastruttura (Docker, Docker Swarm [8], Kubernetes, ecc.) e si configura sia automaticamente che dinamicamente. L'unico passaggio di configurazione richiesto è "puntare" Traefik verso il proprio orchestratore.

3.5.1 Overview

Distribuita una serie di microservizi con un orchestratore (come Docker Swarm o Kubernetes) c'è la necessità che gli utenti possano accedervi attraverso un *reverse proxy*.

I *reverse proxy* tradizionali richiedono che ogni *route*, che collega percorsi e sottodomini a ogni microservizio, sia configurato. In un ambiente dinamico in cui i servizi cambiano rapidamente (aggiunti, rimossi, interrotti, aggiornati o ridimensionati) più volte al giorno, il compito di mantenere i *route* diviene un'attività tediosa. Traefik ascolta l'API dell'orchestratore e genera istantaneamente le *route* in modo che i microservizi siano connessi al mondo esterno, senza ulteriori interventi.

Traefik supporta anche la configurazione manuale dei percorsi.

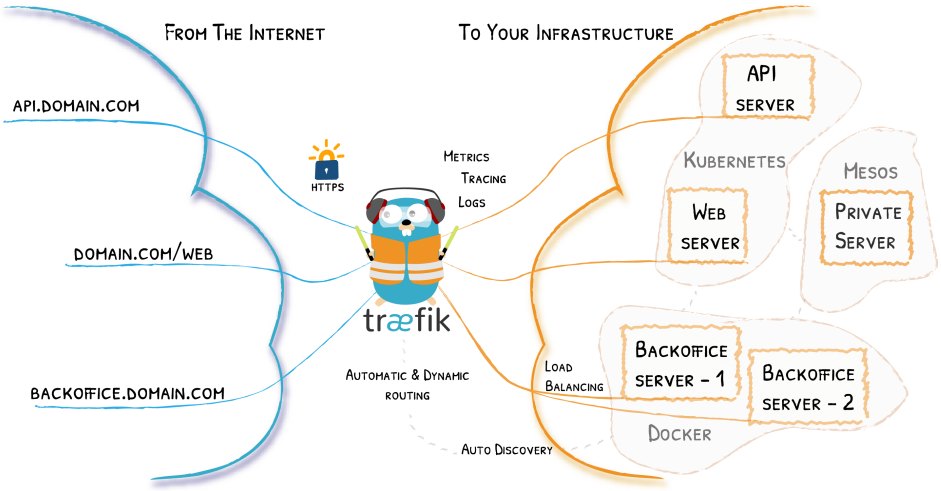


Figura 3.6: Esempio di architettura Traefik

Fonte: https://github.com/traefik/traefik

3.5.2 Funzionalità

* Aggiorna continuamente e autonomamente la configurazione (nessun riavvio è richiesto).
* Supporta più algoritmi di *bilanciamento del carico*.
* Fornisce HTTPS ai microservizi sfruttando *Let's Encrypt*.
* Predispone *Websocket*, *HTTP/2*, gRPC.
* Fornisce metriche (REST, Prometheus, Datadog, Statsd, InfluxDB).
* Mantiene i log di accesso (JSON, CLF).
* Espone un'API REST

Capitolo 4

Analisi dei requisiti

In questo capitolo si descrive l'analisi dei requisiti fatta per il progetto portato avanti durante lo stage.

4.1 Introduzione

Come menzionato nei capitoli precedenti, ciascun requisito è stato acquisito attraverso l'esecuzione di più interviste al team di *Cybersecurity & Privacy* per comprendere necessità e desideri, al fine di fornire la soluzione più adatta a tali richieste.

4.2 Attori del sistema

Sono attori primari nel sistema:

* Analista: l'utente che usufruisce dei servizi per svolgere le proprie mansioni di analisi degli incidenti informatici.
* Amministratore: l'utente che gestisce il cluster, sia in termini di macchine che di servizi, manutenendo le configurazioni e la salute dei sistemi.

4.3 Casi d'uso

In questa sezione vi sono elencati tutti i casi d'uso individuati. Ogni caso d'uso rappresenta uno scenario per uno specifico attore e, inoltre, può essere descritto tramite un diagramma. Sono così strutturati i casi d'uso:

* codice identificativo: il codice di ogni caso d'uso seguirà questo formalismo:

  UC[codice.padre].[codice.figlio]

  di cui:

  - codice.padre: numero identificativo univoco per i casi d'uso;
  - codice.figlio: numero progressivo che identifica i sotto-casi, può a sua volta includere altri livelli.

* titolo;
* diagramma UML;
* attori primari;
* attori secondari (se presenti);
* descrizione;
* inclusioni (se presenti);
* estensioni (se presenti);
* specializzazioni (se presenti);
* precondizione;
* postcondizione;

4.3.1 UC1 - Amministratore accede al cluster

* **Attori primari**: Amministratore;
* **Descrizione**: l'Amministratore effettua l'accesso al cluster;
* **Precondizione**: l'Amministratore desidera accedere al cluster;
* **Postcondizione**: l'Amministratore ottiene l'accesso al cluster;

4.3.2 UC2 - Amministratore visualizza l'elenco dei nodi nel cluster

* **Attori primari**: Amministratore;
* **Descrizione**: l'Amministratore vede l'elenco dei nodi nel cluster;
* **Precondizione**: l'Amministratore ha effettuato l'accesso al cluster;
* **Postcondizione**: il sistema stampa l'elenco dei nodi del cluster;

4.3.3 UC3 - Amministratore avvia un servizio nel cluster

* **Attori primari**: Amministratore;
* **Descrizione**: l'Amministratore avvia un servizio nel cluster;
* **Precondizione**: l'Amministratore ha effettuato l'accesso al cluster;
* **Postcondizione**: un servizio è stato avviato nel cluster;

4.3.4 UC4 - Amministratore visualizza l'elenco dei servizi nel cluster

* **Attori primari**: Amministratore;
* **Descrizione**: l'Amministratore vede l'elenco dei servizi nel cluster, siano essi in esecuzione, sospensione o terminazione;
* **Precondizione**: l'Amministratore ha effettuato l'accesso al cluster;
* **Postcondizione**: il sistema stampa l'elenco dei servizi nel cluster;

4.3.5 UC5 - Amministratore scala uno dei servizi nel cluster

* **Attori primari**: Amministratore;
* **Descrizione**: l'Amministratore scala uno dei servizi nel cluster aumentando/riducendo le sue coppie a piacere;
* **Precondizione**: l'Amministratore ha effettuato l'accesso al cluster;
* **Postcondizione**: il numero di coppie aumenta o diminuisce della quantità desiderata dall'Amministratore;

4.3.6 UC6 - Amministratore aggiorna un servizio del cluster

* **Attori primari**: Amministratore;
* **Descrizione**: l'Amministratore aggiorna un servizio per esigenze prestazionali o di configurazione;
* **Precondizione**: l'Amministratore ha effettuato l'accesso al cluster e ha aggiornato il file di configurazione del servizio;
* **Postcondizione**: il servizio viene aggiornato con la nuova configurazione;

4.3.7 UC7 - Utente accede a TheHive

* **Attori primari**: Analista, Amministratore;
* **Descrizione**: l'utente effettua l'accesso con le proprie credenziali sulla piattaforma TheHive;
* **Precondizione**: l'utente desidera accedere a TheHive;
* **Postcondizione**: l'utente ottiene l'accesso a TheHive;

4.3.8 UC8 - Utente accede a Cortex

* **Attori primari**: Analista, Amministratore;
* **Descrizione**: l'utente effettua l'accesso con le proprie credenziali sulla piattaforma Cortex;
* **Precondizione**: l'utente desidera accedere a Cortex;
* **Postcondizione**: l'utente ottiene l'accesso a Cortex;

4.3.9 UC9 - Utente accede a MISP

* **Attori primari**: Analista, Amministratore;
* **Descrizione**: l'utente effettua l'accesso con le proprie credenziali sulla piattaforma MISP;
* **Precondizione**: l'utente desidera accedere a MISP;
* **Postcondizione**: l'utente ottiene l'accesso a MISP;

4.3.10 UC10 - Amministratore aggiunge un *analyzer* a Cortex

* **Attori primari**: Amministratore;
* **Descrizione**: l'Amministratore aggiunge un *analyzer* a Cortex;
* **Precondizione**: l'Amministratore ha effettuato l'accesso a Cortex;
* **Postcondizione**: un *analyzer* è stato aggiunto alla piattaforma di Cortex;

4.3.11 UC11 - Analista crea un caso su TheHive

* **Attori primari**: Analista;
* **Descrizione**: l'Analista effettua la creazione di un caso su TheHive, che corrisponde ad un incidente;
* **Precondizione**: l'analista ha effettuato l'accesso a TheHive;
* **Postcondizione**: un caso è stato creato su TheHive;

4.3.12 UC12 - Analista aggiunge un task

* **Attori primari**: Analista;
* **Descrizione**: l'Analista aggiunge un task e la sua descrizione ad un caso su TheHive;
* **Precondizione**: l'analista ha effettuato l'accesso a TheHive ed esiste un caso;
* **Postcondizione**: un task è stato aggiunto ad un caso;

4.3.13 UC13 - Analista aggiunge un observable

* **Attori primari**: Analista;
* **Descrizione**: l'Analista aggiunge un observable ad un caso su TheHive;
* **Precondizione**: l'analista ha effettuato l'accesso a TheHive ed esiste un caso;
* **Postcondizione**: un observable è stato aggiunto ad un caso;

4.3.14 UC14 - Analista esegue uno o più *analyzer* su un observable

* **Attori primari**: Analista;
* **Descrizione**: l'Analista esegue uno o più *analyzer* su un observable identificato su TheHive;
* **Precondizione**: l'analista ha effettuato l'accesso a TheHive, esiste un observable ed è presente un analyzer su Cortex;
* **Postcondizione**: il sistema stampa i risultati prodotti dagli *analyzer* sull'observable scelto;

4.3.15 UC15 - Analista aggiunge la descrizione di un TTP

* **Attori primari**: Analista;
* **Descrizione**: l'Analista aggiunge la descrizione di una tattica, tecnica e procedura (TTP) osservata ad un caso su TheHive;
* **Precondizione**: l'analista ha effettuato l'accesso a TheHive ed esiste un caso;
* **Postcondizione**: un TTP è stato aggiunto ad un caso;

4.3.16 UC16 - Analista condivide i risultati di un caso su MISP

* **Attori primari**: Analista;
* **Descrizione**: l'Analista rende disponibile e condivide tutte le informazioni raccolte di un caso su MISP;
* **Precondizione**: l'analista ha effettuato l'accesso a TheHive ed esiste un caso;
* **Postcondizione**: i risultati di un caso sono stati condivisi su MISP;

4.4 Tracciamento dei requisiti

Ogni requisito è composto dalla seguente struttura descrittiva:

* **Codice Identificativo**: ogni codice identificativo è univoco e conforme alla seguente codifica:

  R[Tipologia][Importanza][Codice]

  Il significato delle cui voci è:

  - **Tipologia**: ogni requisito può assumere i seguenti valori:
    * *F*: funzionale;
    * *Q*: qualità;
    * *V*: vincolo;
    * *P*: prestazionale.
  - **Importanza**: ogni requisito può assumere i seguenti valori:

    - *A*: requisito obbligatorio: irrinunciabile per gli stakeholder;
    - *B*: requisito desiderabile: ha valore aggiunto riconoscibile ma non è strettamente necessario;
    - *C*: requisito opzionale: relativamente utile oppure contrattabile in seguito.
  - **Codice**: è un identificatore univoco.
- **Descrizione**: breve descrizione del requisito;
- **Fonti**: ogni requisito può essere derivato da una o più delle seguenti fonti:
  - **Piano di lavoro**: il requisito è stato individuato dalla stesura del piano di lavoro;
  - **Interno**: il requisito è stato ritenuto opportuno da aggiungere in fase di analisi;
  - **Caso d'uso**: il requisito è stato estrapolato da uno o più casi d'uso. In questo caso viene riportato il codice univoco del caso d'uso;

4.4.1 Requisiti

I requisiti individuati derivano principalmente dai casi d'uso precedentemente descritti, ma anche altre fonti (come il piano di lavoro e la descrizione del progetto stesso) risultano utili per la definizione dei vincoli. Le scelte fatte durante lo stage hanno la fonte indicata come "interno".

Tabella 4.1: Tabella dei requisiti

Codice	Descrizione	Fonte
RFA1	L'Amministratore può accedere alla macchina su cui è configurato il cluster.	UC1
RFA2	L'Amministratore tramite terminale può visualizzare l'elenco dei nodi nel cluster.	UC2
RFA3	L'Amministratore tramite terminale può avviare un servizio nel cluster definendone la configurazione.	UC3
RFA4	L'Amministratore tramite terminale può visualizzare l'elenco dei servizi, che siano essi in esecuzione, sospensione o terminazione, nel cluster.	UC4
RFA5	L'Amministratore può scalare uno dei servizi nel cluster a seconda delle esigenze.	UC5
RFA6	L'Amministratore può aggiornare un servizio nel cluster dato un nuovo file di configurazione.	UC6
RFA7	L'utente, che sia esso l'Analista o l'Amministratore, può accedere alla piattaforma TheHive con le proprie credenziali.	UC7
RFA8	L'utente, che sia esso l'Analista o l'Amministratore, può accedere alla piattaforma Cortex con le proprie credenziali.	UC8

Tabella 4.1 – *continuazione da pagina precedente*

Codice	Descrizione	Fonte
RFA9	L'utente, che sia esso l'Analista o l'Amministratore, può accedere alla piattaforma MISP con le proprie credenziali.	UC9
RFA10	L'Amministratore può aggiungere un *analyzer* a Cortex, eventualmente configurandolo con le necessarie impostazioni quali API Key, path server, ecc.	UC10
RFA11	L'Analista può creare un caso su TheHive, che identifica un incidente da analizzare.	UC11
RFA12	L'Analista può aggiungere un task ad un caso su TheHive, che corrisponde ad un'attività pianificata da svolgere.	UC12
RFA13	L'Analista può aggiungere un observable su TheHive per poter effettuare le analisi del caso.	UC13
RFA14	L'Analista può eseguire l'insieme degli *analyzer*, o un suo sottoinsieme, su un observable da TheHive. Gli *analyzer* sono definiti e configurati da Cortex.	UC14
RFA15	L'Analista può aggiungere la descrizione di tattiche, tecniche e procedure (TTP) osservati ad un caso studiato su TheHive.	UC15
RFA16	L'Analista può condividere su MISP i risultati e le informazioni raccolte dei casi analizzati.	UC16
RFA17	I servizi "deployati" devono essere integrati tra di loro.	Interno
RVA1	Deve essere fornito un documento che dettaglia i requisiti funzionali e non funzionali raccolti attraverso il confronto con l'azienda ed emersi durante la fase di analisi.	Piano di lavoro
RVA2	Deve essere fornito un documento che definisce i requisiti hardware e software per il deployment della strumentazione alternativa proposta.	Piano di lavoro
RVA3	Deve essere fornito un diagramma architetturale ed eventuali diagrammi di flusso.	Piano di lavoro
RVA4	Deve essere fornito un file markdown che spiega passo passo come effettuare il deployment della strumentazione.	Piano di lavoro
RVA5	Il cluster dovrà integrarsi con i servizi cloud come Microsoft Azure e Google Cloud Platform.	Piano di lavoro
RVA6	Il deployment dei servizi deve avvenire attraverso Docker.	Interno
RVA7	Il cluster deve essere gestito e contenuto con Docker Swarm.	Interno

Capitolo 5

Installazione e configurazione

In questo capitolo si approfondisce il processo di installazione, configurazione e distribuzione dei servizi per raggiungere l'obiettivo finale dello stage.

5.1 Configurazione della macchina

Per comprendere meglio la natura della macchina su cui installare i servizi bisogna consultare i requisiti minimi di ciascun servizio:

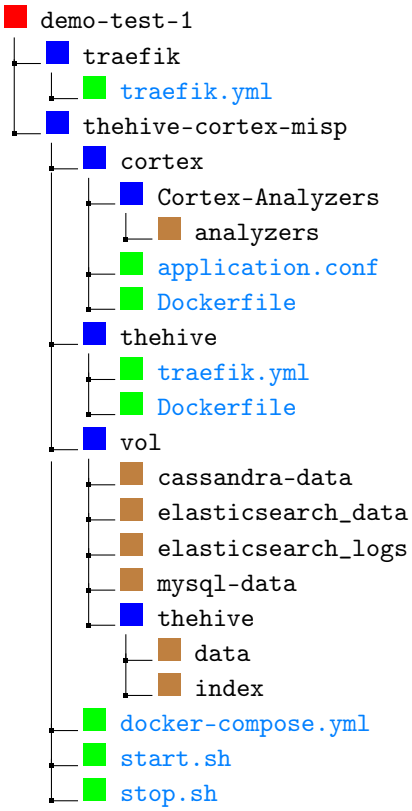
* TheHive [15]:
  - se meno di 3 utenti: 2vCPU, 4-8GB RAM
  - se meno di 10 utenti: 4vCPU, 8-16GB RAM
  - se meno di 20 utenti: 8vCPU, 16-32GB RAM
* Cortex [2]:
  - utilizza JavaVM
  - 8vCPU, 8GB RAM, 10GB storage
* MISP [11]:
  - se su macchina singola: 2GB RAM, 30GB storage
  - se su cluster: 8GB RAM, 30GB storage

Per questo elaborato si sono ipotizzati meno di 10 utenti che utilizzano TheHive, quindi sono state selezionate due macchine denominate "demo-bushi-1" e "demo-bushi-2" con 8vCPU, 8GB RAM e 30GB storage ciascuno. Il sistema operativo scelto è Ubuntu 20.04 LTS.

Poiché ogni macchina dispone di un *file system* diverso, ovvero di un spazio di archiviazione proprio, è stato preferito mantenere tutti i file su "demo-bushi-1". Una possibile soluzione a tale problema potrebbe essere l'implementazione di Gluster [10].

Di seguito una breve illustrazione della disposizione dei file all'interno di "demo-bushi-1".

I riquadri in colore blu sono cartelle, mentre in colore marrone le cartelle il cui contenuto è omesso perché generato dinamicamente. In colore verde sono identificati i file. Per facilitare la lettura il contenuto dei file è disponibile nelle appendici in calce all'elaborato.



5.1.1 Installazione Docker Engine

L'installazione di Docker Engine su Ubuntu avviene secondo i passi descritti dalla guida ufficiale disponibile online [5]. La versione utilizzata in questo elaborato è la 20.10.12 con build e91ed57, tuttavia per i file di configurazione presenti è sufficiente una versione 19.03.0 o superiore [4].

Eliminare preventivamente le vecchie installazioni di Docker:

```
sudo apt-get remove docker docker-engine docker.io containerd
    runc
```

Aggiornare l'indice dei pacchetti e permettere l'installazione dei pacchetti su HTTPS:

```
sudo apt-get update

sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

Aggiungere la chiave GnuPG ufficiale di Docker:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo
    gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.
    gpg
```

Impostare la repository *stable* (stabile) di Docker:

```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/
      keyrings/docker-archive-keyring.gpg] https://download.docker
      .com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/
      docker.list > /dev/null
```

Aggiornare l'indice dei pacchetti e installare l'ultima versione di Docker Engine e containerd:

```
sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io
```

Verificare il corretto funzionamento di Docker eseguendo un'immagine di test denominato "hello-world". Il comando scarica l'immagine e lo esegue in un container, quando quest'ultimo è in funzione stampa un messaggio e termina:

```
sudo docker run hello-world
```

Per comodità di utilizzo si è preferito gestire Docker anche da un utente *non-root* (non privilegiato). Prestare comunque attenzione perché il gruppo "docker" garantisce privilegi equiparabili all'utente *root* (privilegiato) [7].

Creare il gruppo "docker":

```
sudo groupadd docker
```

Aggiungere l'utente attuale al gruppo:

```
sudo usermod -aG docker $USER
```

Rendere effettive le modifiche con il seguente comando:

```
newgrp docker
```

Verificare di potere eseguire un comando docker senza *root* (utente con privilegi):

```
docker run hello-world
```

5.1.2 Installazione docker-compose

Durante il progetto si è optato per modificare alcune immagini scaricate. Per condividere tali personalizzazioni sul cluster è stato usato docker-compose. La versione utilizzata in questo elaborato è la 1.27.4-1.

Scaricare e installare il pacchetto docker-compose:

```
sudo apt-get install docker-compose
```

5.1.3 Configurazione Docker Swarm

Un cluster orchestrato da Docker Swarm è composto di macchine **manager** se esse gestiscono il cluster stesso (e.g. modifica dei nodi, aggiornamento dei servizi, ecc.), oppure di macchine **worker** (lavoratori) se si limitano ad eseguire i servizi. Per definizione il cluster ha almeno un **manager** (tipicamente il fondatore) e non è possibile fare operazioni sui nodi (e.g. promuovere, rimuovere, ecc.) se almeno la metà dei **manager** non sono online. I **manager**, oltre a ricoprire il ruolo organizzativo all'interno del cluster, eseguono i servizi come i colleghi **worker**.
Si fa notare che in tale contesto ciascun nodo corrisponde ad una ed una sola macchina.

Per questo elaborato sono state utilizzate due macchine **manager** denominate "demo-bushi-1" e "demo-bushi-2" su una rete locale così configurata:

* demo-bushi-1:

  IP: 192.168.255.100

  Subnet mask: 255.255.255.0

  Gateway: 192.168.255.1

* demo-bushi-2:

  IP: 192.168.255.200

  Subnet mask: 255.255.255.0

  Gateway: 192.168.255.1

Inserire sulla macchina il comando di inizializzazione del cluster specificando l'indirizzo IP su cui gli altri nodi faranno adesione:

```
docker swarm init --advertise-addr 192.168.255.100
```

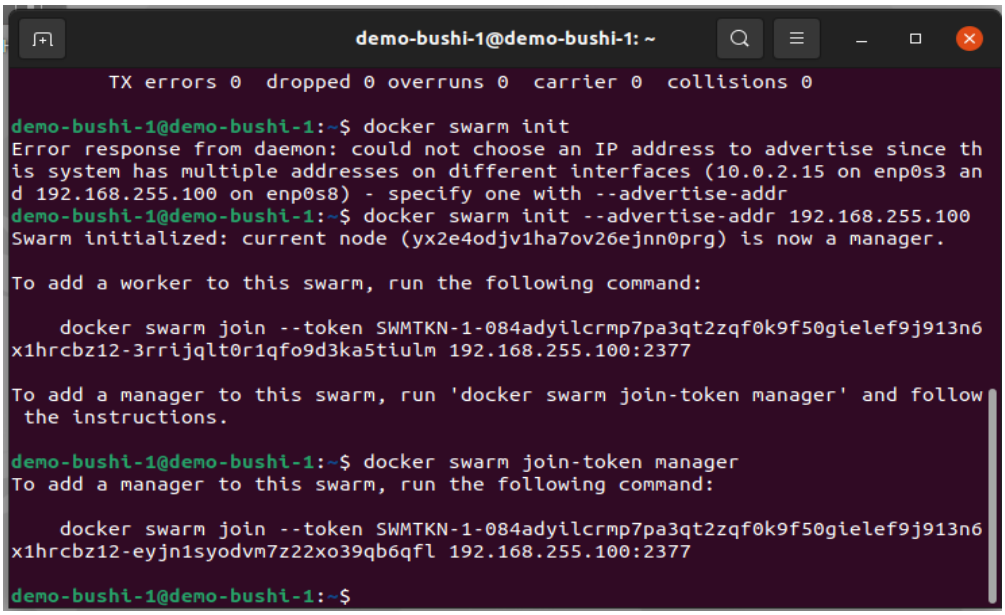


Figura 5.1: Inizializzazione del cluster su demo-bushi-1 con Docker Swarm

Conseguentemente sono stampati i *token*, cioè le chiavi, necessari per aggiungere un nodo al cluster. A seconda del *token* utilizzato nel momento dell'adesione sarà deciso anche il ruolo (i.e. *manager* o *worker*).
Inserire sulla macchina da aggiungere al cluster il comando risultato precedentemente:

```
docker swarm join --token SWMTKN-1-084adyilcrmp7pa3qt2zqf0k9f50
    gielef9j913n6x1hrcbz12-3rrijqlt0r1qfo9d3ka5tiulm
    192.168.255.100:2377
```

Viene stampato a video una conferma dell'operazione.

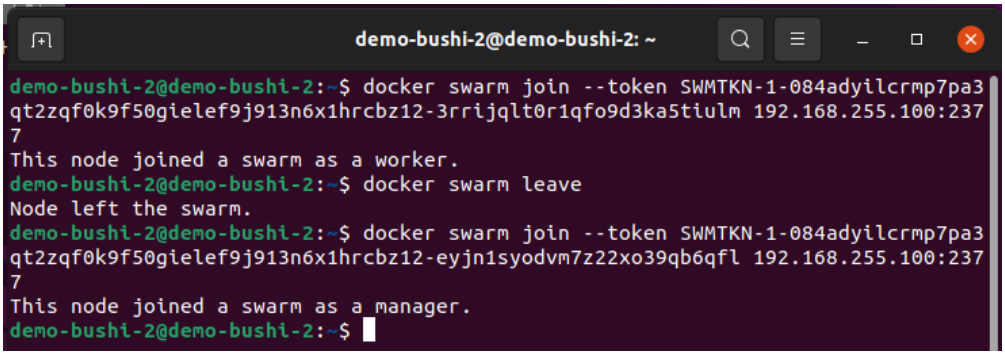


Figura 5.2: Adesione di demo-bushi-2 al cluster con Docker Swarm

Verificare la lista di nodi che compongono il cluster:

```
docker node ls
```

Sono stampati a video le informazioni dei nodi (i.e. ID, nome, status, ecc.).

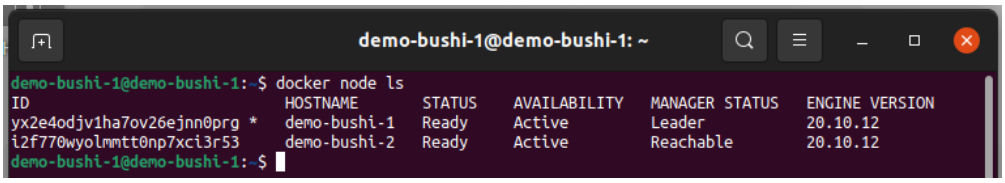


Figura 5.3: Visualizzazione della lista dei nodi che compongono il cluster con Docker Swarm

5.2 Configurazione Traefik

Il file di configurazione di Traefik necessita della definizione preliminare di alcune variabili quali email, dominio (i.e. l'indirizzo URL da cui è raggiungibile), username e password (per effettuare l'accesso), ecc.:

```
export EMAIL=dardan.kokollari@studenti.unipd.it
export DOMAIN=traefik.localhost
export USERNAME=admin
export PASSWORD=test
export HASHED_PASSWORD=$(openssl passwd -apr1 $PASSWORD)

# convalida dei certificati sul nodo attuale
export NODE_ID=$(docker info -f '{{.Swarm.NodeID}}')
docker node update --label-add traefik-public.traefik-public-
    certificates=true $NODE_ID

# creazione della rete su cui si appoggia il container
docker network create --driver=overlay traefik-public
```

Eseguire il deploy di Traefik nel cluster:

```
docker stack deploy traefik -c traefik/traefik.yml
```

Verificare che Traefik sia eseguito:

```
docker stack ps traefik
```

5.3 Configurazione e avvio servizi

Per specificare lo spazio di archiviazione in cui memorizzare i dati generati dai servizi, c'è bisogno di specificare un *label* (un'etichetta) per la macchina demo-bushi-1, qui definito come "bushi1":

```
# creazione del label bushi1
export NODE_ID=$(docker info -f '{{.Swarm.NodeID}}')
docker node update --label-add server.name=bushi1 $NODE_ID
```

Avviare lo script di deploy dei servizi:

```
sh start.sh
```

Lo script si comporta come segue:

* crea il servizio "registry";

* scarica e costruisce l'immagine di Cortex (vedi cortex/Dockerfile);
* scarica e costruisce l'immagine di TheHive (vedi thehive/Dockerfile);
* esegue il *push* delle immagini create su "registry";
* scarica le immagini ed esegue il deploy dei servizi di TheHive, Cortex, MISP e tutte le loro dipendenze necessarie (e.g. ElasticSearch, Cassandra, ecc.).

Il *push* delle immagini su un servizio ad-hoc del cluster è necessario perché così ogni nodo reperisce la stessa immagine costruita. Per tutte le altre immagini che non hanno bisogno di personalizzazioni non è necessario fare alcuna operazione aggiuntiva.

Verificare lo stato dei servizi sul cluster:

```
docker service ls
```

Verificare lo stato dei servizi di TheHive, Cortex e MISP, con più dettagli come p.es. il nome del nodo che sta eseguendo il servizio:

```
docker stack ps demo
```

Conseguentemente sono stampati a video molte informazioni utili all'amministratore del sistema.

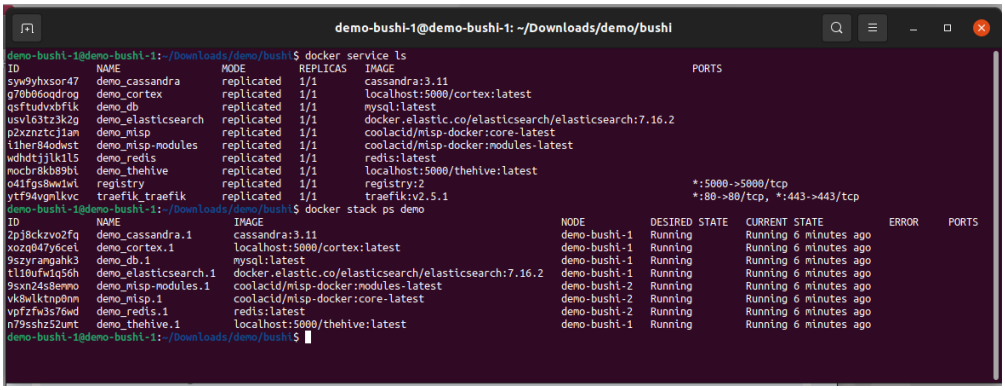


Figura 5.4: Visualizzazione della lista dei servizi eseguiti sul cluster e loro stato

Accedere all'interfaccia grafica di Traefik all'indirizzo `https://traefik.localhost` e verificare sulla *dashboard* che i servizi siano stati rilevati:

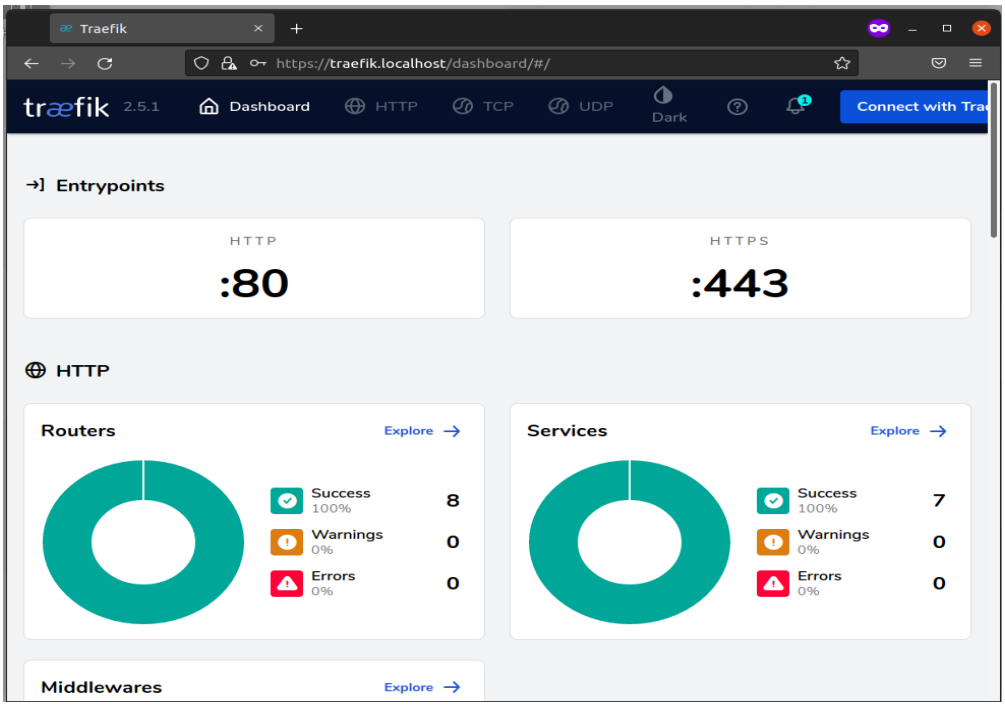


Figura 5.5: Dashboard di Traefik

Selezionare la scheda "routers" per verificare che i *path*, cioè gli indirizzi URL, siano corretti:

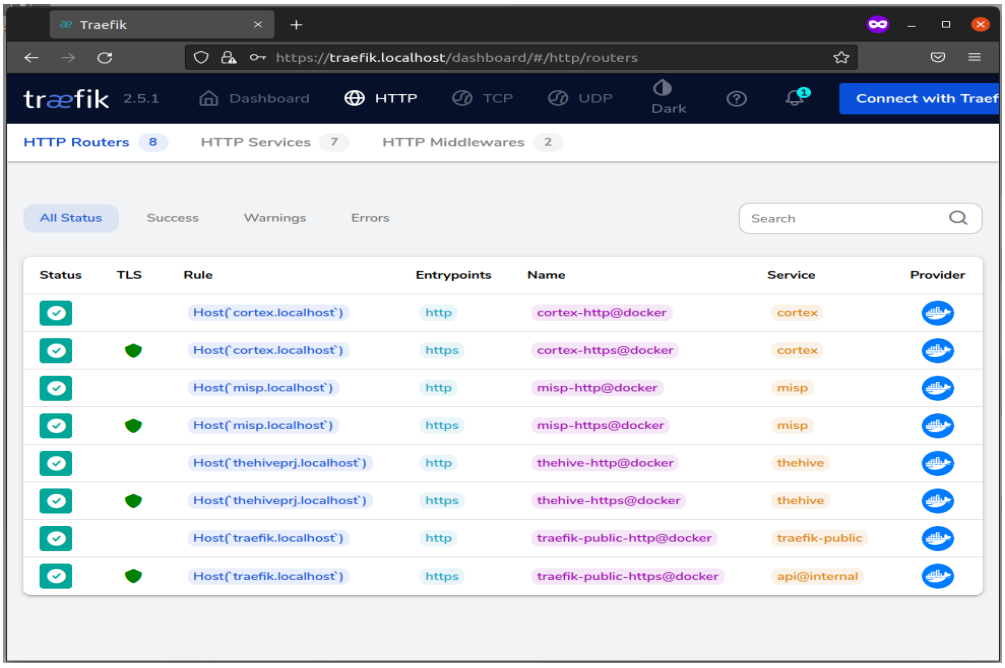


Figura 5.6: Scheda "routers" di Traefik per i servizi esposti

5.4 Configurazione Cortex

Accedere all'interfaccia grafica di Cortex all'indirizzo `https://cortex.localhost`, quindi selezionare il pulsante "Update Database" per il primo avvio.

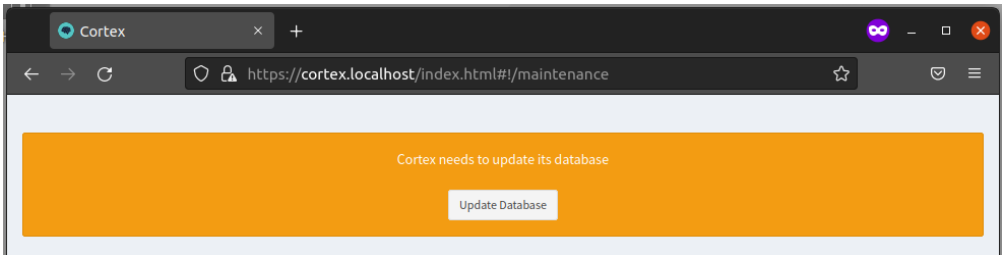


Figura 5.7: Schermata di primo avvio di Cortex

Creare un account amministratore per gestire il servizio:

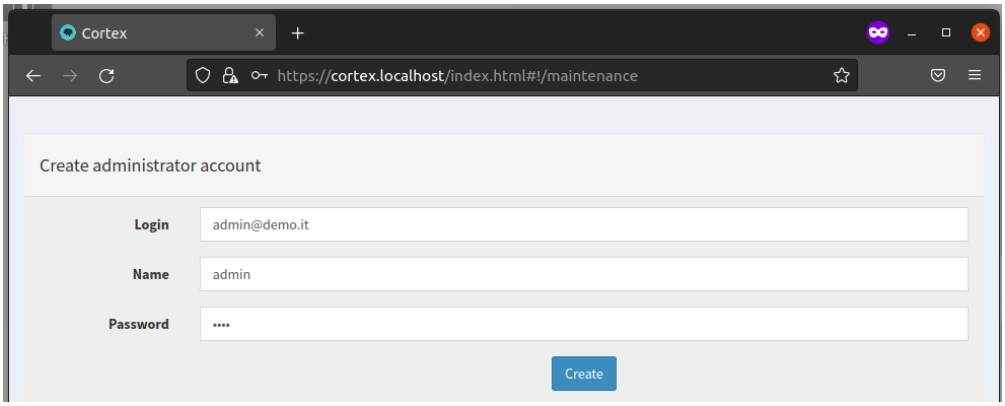


Figura 5.8: Form di inserimento dati per la creazione di un superadmin per Cortex

Accedere con le credenziali precedentemente inserite per visualizzare la homepage:

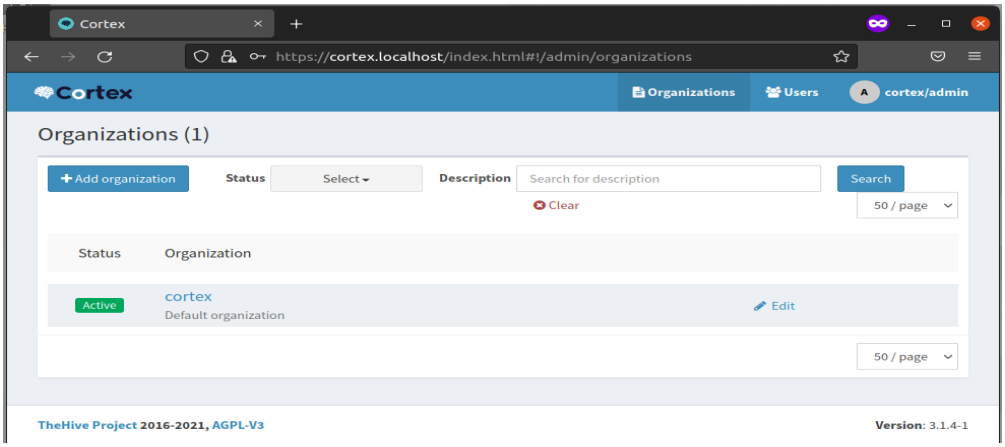


Figura 5.9: Lista delle organizzazioni, homepage di Cortex

Creare una nuova organizzazione:

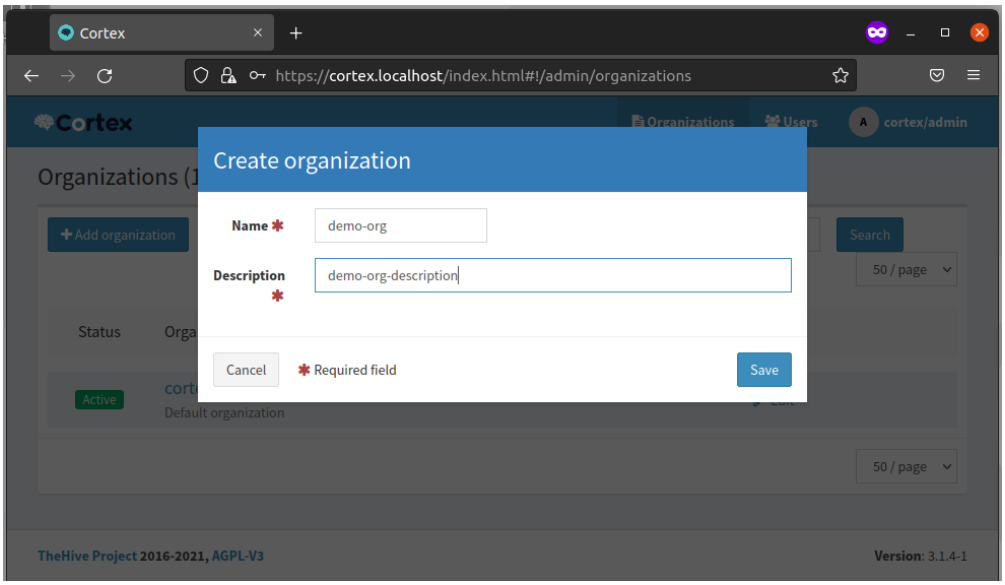


Figura 5.10: Form di inserimento dati per la creazione di un'organizzazione su Cortex

Selezionare la nuova organizzazione dalla lista delle organizzazioni e creare un nuovo utente premendo il tasto "Add user". Il ruolo "orgadmin" è necessario se si vogliono gestire e configurare gli *analyzer*:

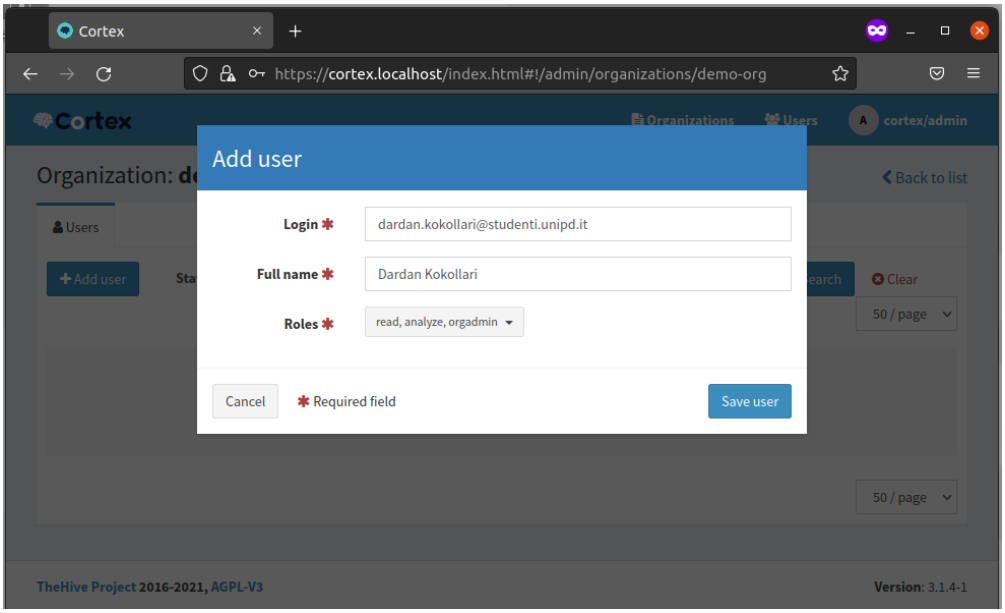


Figura 5.11: Form di inserimento dati per la creazione di un utente su Cortex

Importante: creare una nuova password per l'utenza appena creata premendo il tasto "New password", quindi inserire una password.

Per far comunicare Cortex con TheHive è necessario disporre di una *API Key*. Siccome le utenze sono nominative, è consigliabile creare un account ad-hoc. Quindi premere sul pulsante "Create API Key", poi "Reveal" e copiare la chiave. Non c'è bisogno di inserire una password per tale utenza particolare:

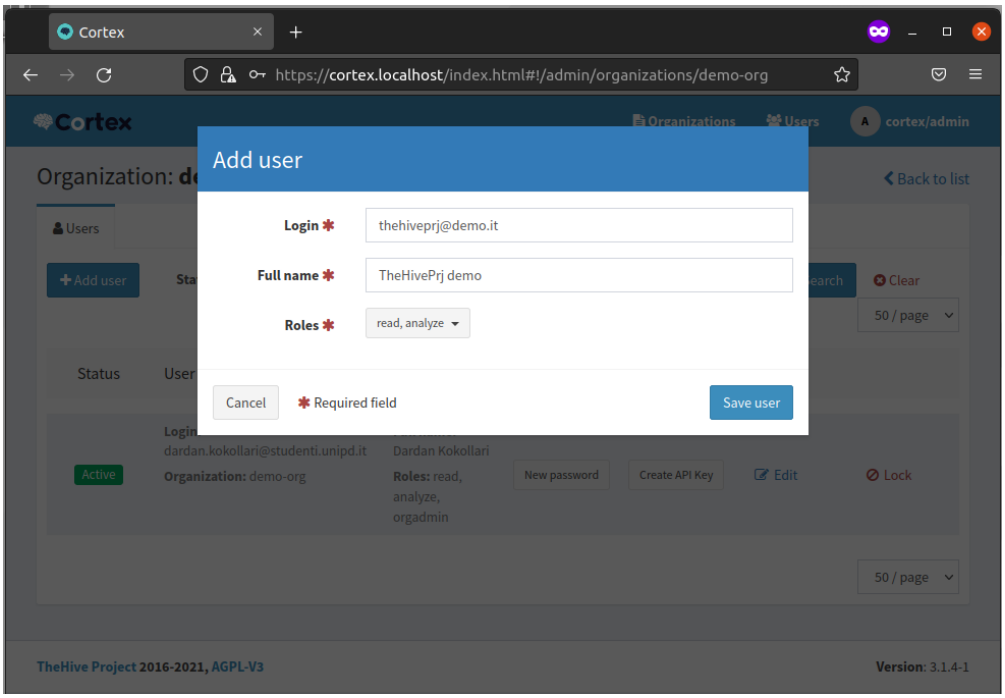


Figura 5.12: Form di inserimento dati per la creazione dell'utenza thehiveprj su Cortex

Incollare l'*API Key* su thehive/application.conf nella sezione dedicata a Cortex:

```
play.modules.enabled += org.thp.thehive.connector.cortex.CortexModule
cortex {
  servers = [
    {
      name = local
      url = "http://cortex:9001"
      auth {
        type = "bearer"
        key = "KB9tEVnphcvaf5Gep7k3DyE/Qw66zYlz"
      }
      # HTTP client configuration (SSL and proxy)
      #  wsConfig {}
    # List TheHive organisation which can use this Cortex server. All ("*") by default
    # includedTheHiveOrganisations = ["*"]
    # List TheHive organisation which cannot use this Cortex server. None by default
    # excludedTheHiveOrganisations = []
    }
  ]
  # Check job update time intervalcortex
  refreshDelay = 5 seconds
  # Maximum number of successive errors before give up
  maxRetryOnError = 3
  # Check remote Cortex status time interval
```


Figura 5.13: Posizionamento dell'*API Key* di Cortex sul file di configurazione di TheHive

Effettuare il logout e accedere con un utente con ruolo "orgadmin", quindi selezionare la scheda "Organization", poi la sotto-scheda "Analyzers" per visualizzare gli *analyzer* installati:

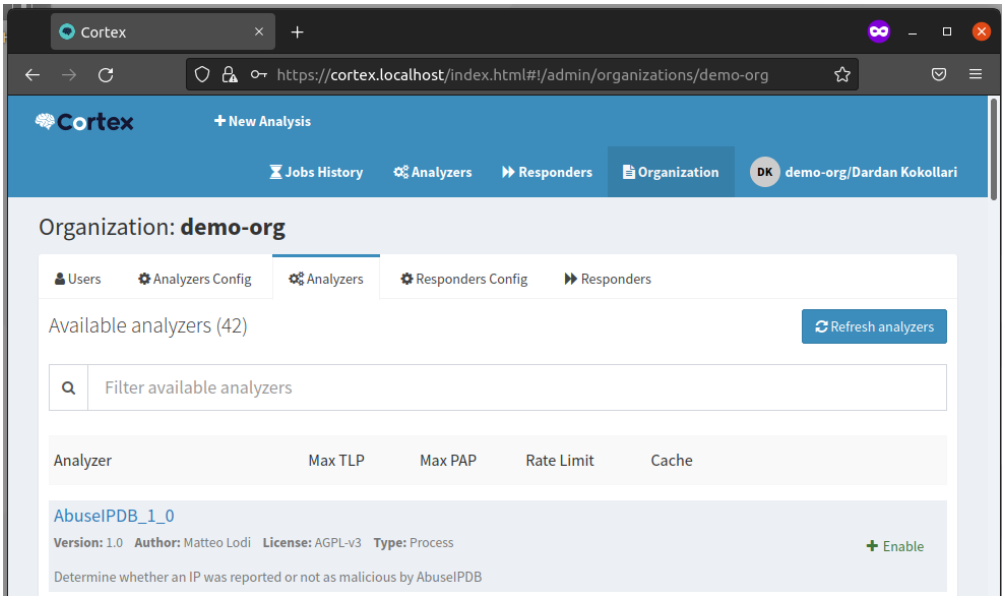


Figura 5.14: Lista degli *analyzer* installati su Cortex

Abilitare gli *analyzer* desiderati, quindi selezionare la scheda "Analyzers" per visualizzare gli *analyzer* attivi:

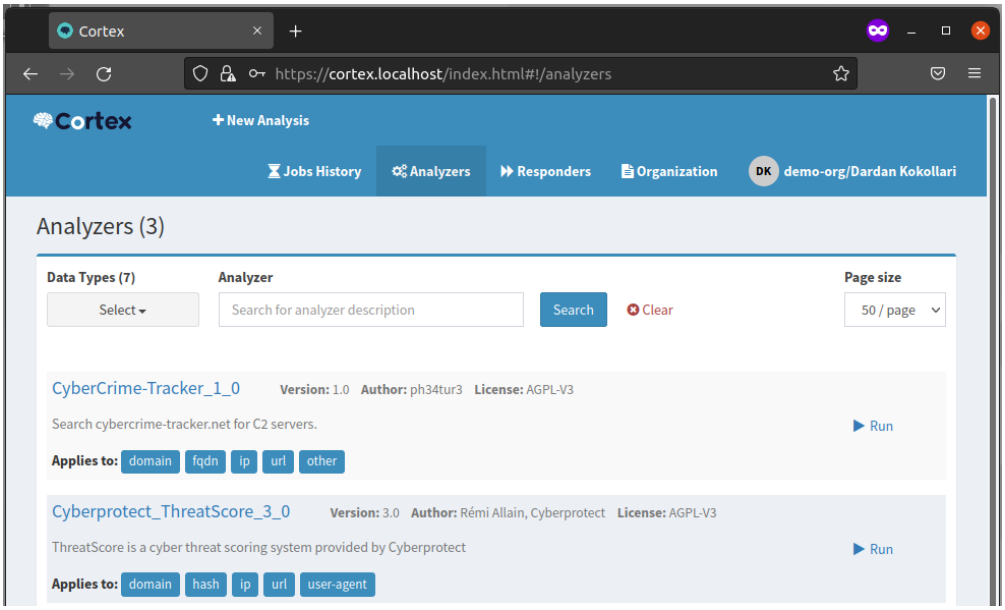


Figura 5.15: Lista degli *analyzer* attivi su Cortex

Premere "Run" sull'*analyzer* per poter eseguire analisi. I risultati sono disponibili in formato JSON.

5.5 Configurazione MISP

Accedere all'interfaccia grafica di MISP all'indirizzo `https://misp.localhost` con le credenziali di default:

* username: admin@admin.test
* password: admin

Bisogna cambiare la password al primo avvio del servizio. La nuova password richiesta deve avere una lunghezza minima di 12 caratteri e deve essere composta da almeno una lettera maiuscola, minuscola e un numero.
Successivamente a tale operazione, è consigliabile creare un nuovo utente amministratore (*superadmin*) in modo da eliminare completamente l'utenza di default fornita. Quindi selezionare dal menù "Administration" la voce "Add User" per creare il nuovo utente amministratore:

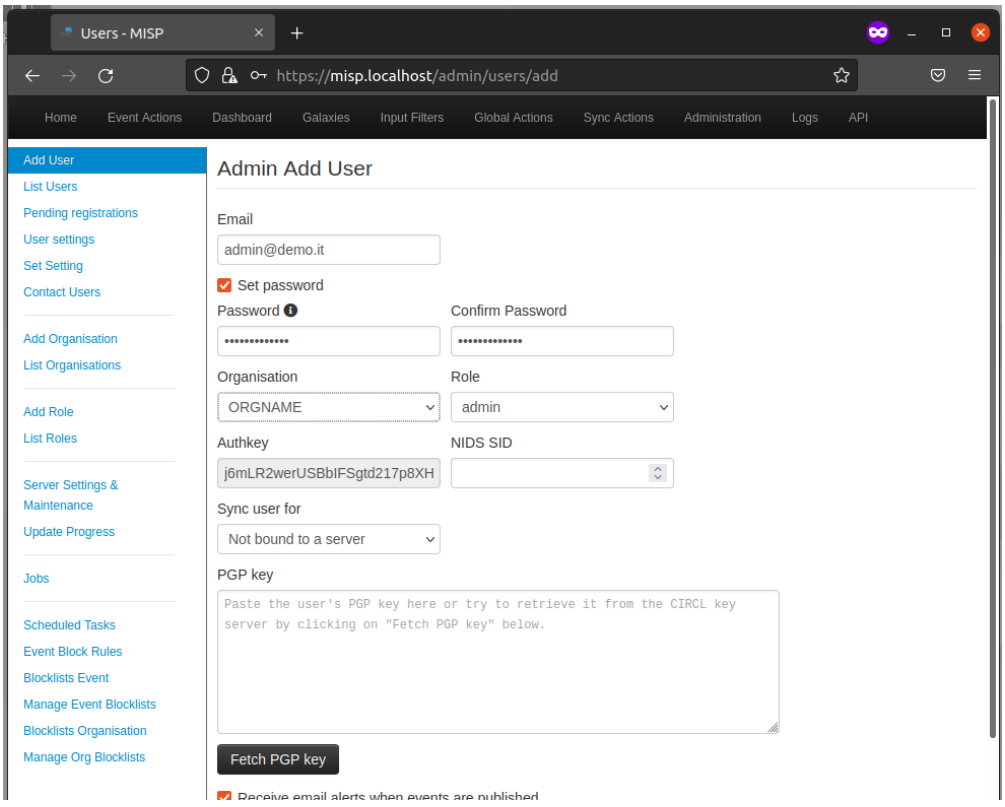


Figura 5.16: Form di inserimento dati per la creazione di un utente su MISP

Effettuare il logout e accedere col nuovo account superadmin, quindi selezionare dal menù "Administration" la voce "List Users". Eliminare permanentemente l'account admin@admin.test.

Creare una nuova organizzazione:

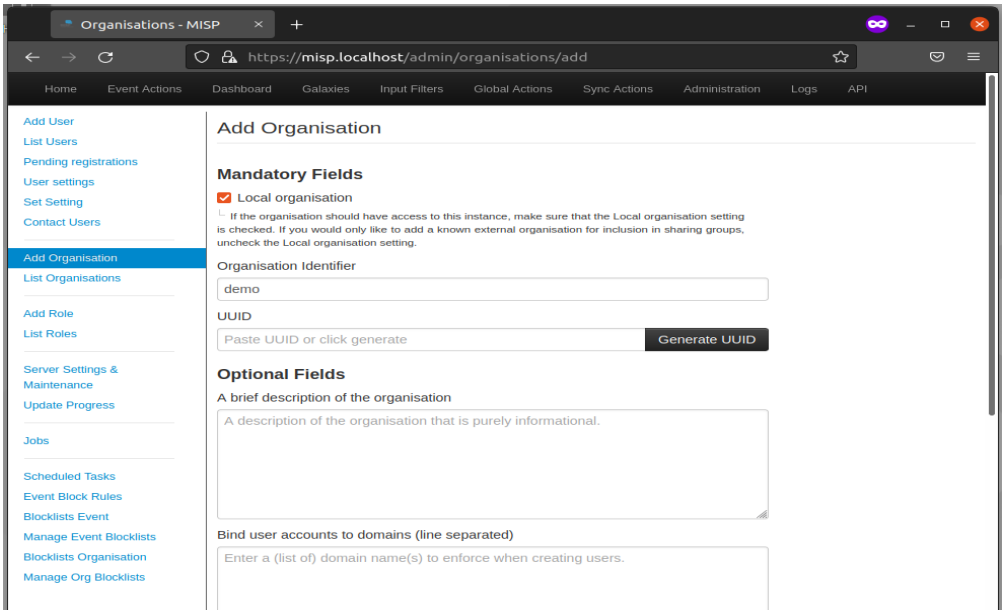


Figura 5.17: Form di inserimento dati per la creazione di un'organizzazione su MISP

Per far comunicare MISP con TheHive è necessario fare un procedimento analogo a quanto precedentemente attuato per Cortex. Quindi creare un account ad-hoc, con ruolo "Org Admin", e copiare la chiave presente nel campo "Authkey". Non c'è bisogno di inserire una password per tale utenza particolare:

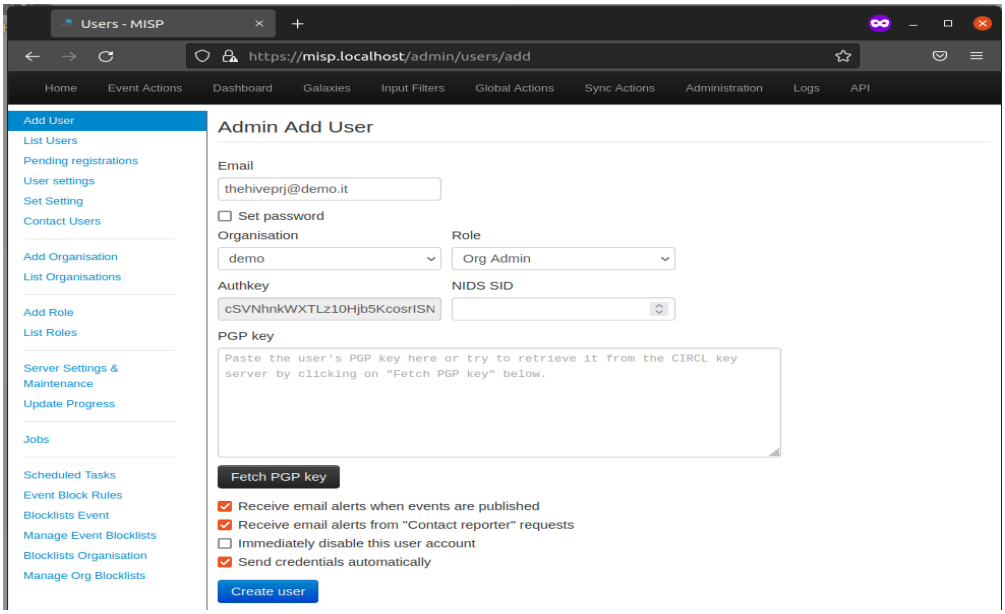


Figura 5.18: Form di inserimento dati per la creazione dell'utenza thehiveprj su MISP

Incollare l'*API Key* su thehive/application.conf nella sezione dedicata a MISP:

```
60 # MISP configuration
61 play.modules.enabled += org.thp.thehive.connector.misp.MispModule
62 misp {
63   interval: 5 min
64   servers: [
65     {
66       name = "MISP THP"            # MISP name
67       url = "https://misp/" # URL or MISP
68       auth {
69         type = key
70         key = "cSVNhnkWXTLz10Hjb5KcosrISNJgqS0yxXjq8Vxo"            # MISP API key
71       }
72       wsConfig { ssl { loose { acceptAnyCertificate: true } } }
73     }
74   ]
75 }
76
```

Figura 5.19: Posizionamento dell'*API Key* di MISP sul file di configurazione di TheHive

5.6 Configurazione TheHive

Per far sì che TheHive stia comunicando con Cortex e MISP bisogna aver prima configurato i singoli servizi. Una volta ottenute le *API Key* e averle posizionate nel file di configurazione, riavviare TheHive per rendere effettive le modifiche. Per comodità usare lo script stop.sh:

```
sh stop.sh
```

Lo script si comporta come segue:

* elimina il servizio "registry";
* arresta i servizi di TheHive, Cortex, MISP e tutte le loro dipendenze, ad eccezione di Traefik.

Eseguire lo script start.sh e attendere il corretto avvio dei servizi:

```
sh start.sh
```

Accedere all'interfaccia grafica di TheHive all'indirizzo `https://thehive.local` con le credenziali di default:

* username: admin@thehive.local
* password: secret

È consigliabile creare un nuovo utente amministratore (*superadmin*) in modo da eliminare completamente l'utenza di default fornita. Quindi selezionare l'organizzazione "admin" dalla lista delle organizzazioni e premere "Create new user":

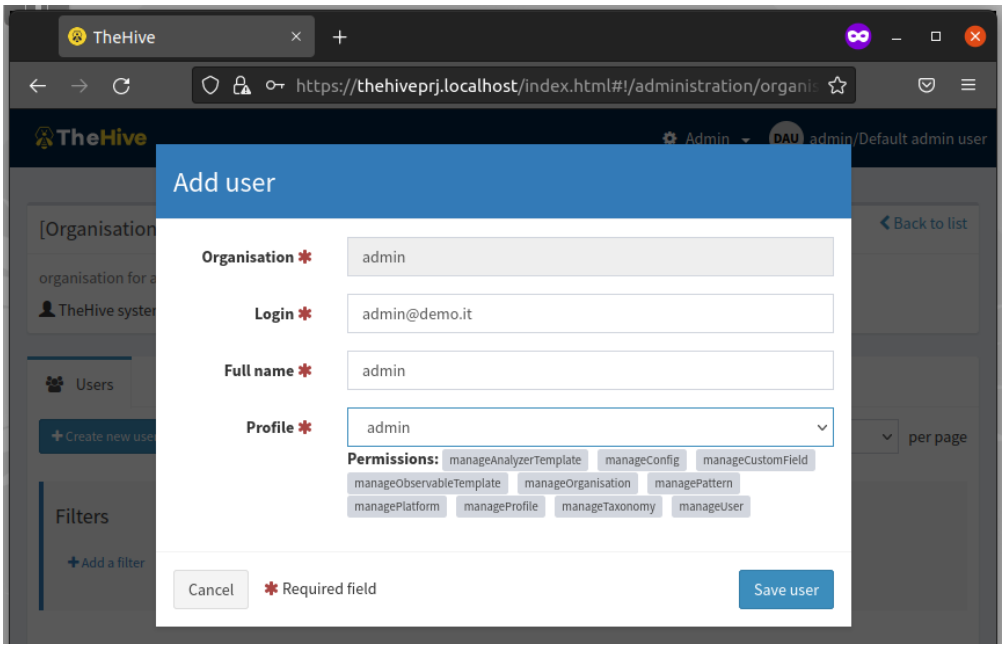


Figura 5.20: Form di inserimento dati per la creazione di un utente su TheHive

Importante: creare una nuova password per l'utenza appena creata premendo il tasto "New password", quindi inserire una password.

Effettuare il logout e accedere col nuovo account *superadmin*, quindi selezionare l'organizzazione "admin" dalla lista delle organizzazioni ed eliminare permanentemente l'account admin@thehive.local.

Creare una nuova organizzazione:

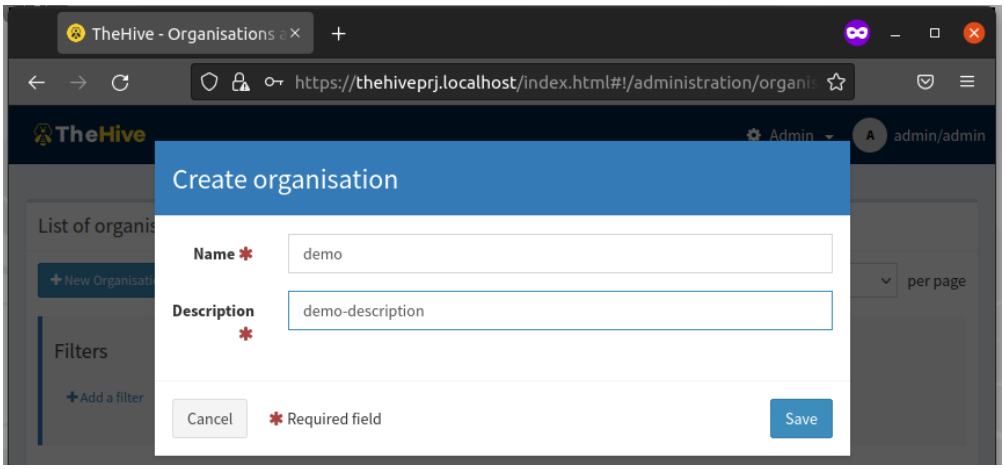


Figura 5.21: Form di inserimento dati per la creazione di un'organizzazione su TheHive

Selezionare la nuova organizzazione dalla lista delle organizzazioni e creare un nuovo

utente premendo il tasto "Add user". Il ruolo "analyst" è sufficiente per lo studio e l'analisi degli incidenti.

Importante: creare una nuova password per l'utenza appena creata premendo il tasto "New password", quindi inserire una password.

Effettuare il logout e accedere con un utente con ruolo "analyst" per iniziare a gestire i *case* (casi), che corrispondono ciascuno ad un incidente oppure in alternativa a più parti di un incidente se esso risulta complesso (in ogni caso è consigliabile fare riferimento al proprio responsabile di progetto):

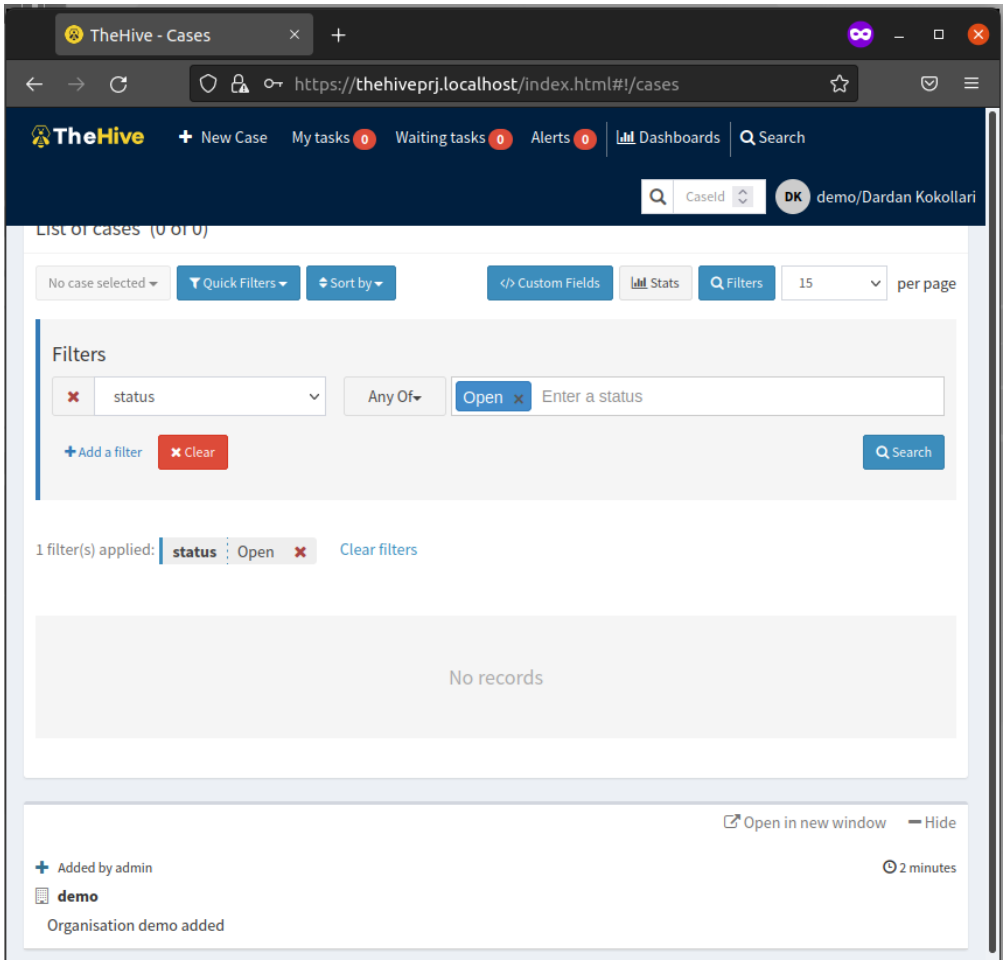


Figura 5.22: Lista dei case, pagina iniziale di TheHive

Capitolo 6

Verifica e validazione

In questo capitolo sono classificati e descritti i test effettuati per la verifica del corretto funzionamento delle configurazioni e delle installazioni effettuate.

6.1 Test di Sistema

Per il lavoro di stage non è stato richiesto lo sviluppo di codice sorgente tuttavia sono stati usati prodotti già verificati come Docker. Inoltre, i servizi Cortex, TheHive, MISP e Traefik sono testati e validati prima del rilascio al pubblico. Per questa ragione non sono stati creati test di unità e d'integrazione, ma solo test di sistema per la verifica del corretto funzionamento delle installazioni effettuate. Ogni test possiede un codice identificativo nel seguente formato:

TS[ID]

Il test può avere uno dei due differenti risultati possibili:

* **S**: il test è stato superato;
* **N**: il test non è stato superato.

Tabella 6.1: Test di sistema

Test	Descrizione	Risultato
TS1	Verifica che l'Amministratore può accedere alla macchina su cui è configurato il cluster.	S
TS2	Verifica che l'Amministratore tramite terminale può visualizzare l'elenco dei nodi nel cluster.	S
TS3	Verifica che l'Amministratore tramite terminale può avviare un servizio nel cluster definendone la configurazione.	S
TS4	Verifica che l'Amministratore tramite terminale può visualizzare l'elenco dei servizi, che siano essi in esecuzione, sospensione o terminazione, nel cluster.	S

Tabella 6.1 – *continuazione da pagina precedente*

Test	Descrizione	Risultato
TS5	Verifica che l'Amministratore può scalare uno dei servizi nel cluster a seconda delle esigenze.	S
TS6	Verifica che l'Amministratore può aggiornare un servizio nel cluster dato un nuovo file di configurazione.	S
TS7	Verifica che l'utente, che sia esso l'Analista o l'Amministratore, può accedere alla piattaforma TheHive con le proprie credenziali.	S
TS8	Verifica che l'utente, che sia esso l'Analista o l'Amministratore, può accedere alla piattaforma Cortex con le proprie credenziali.	S
TS9	Verifica che l'utente, che sia esso l'Analista o l'Amministratore, può accedere alla piattaforma MISP con le proprie credenziali.	S
TS10	Verifica che l'Amministratore può aggiungere un *analyzer* a Cortex, eventualmente configurandolo con le necessarie impostazioni quali *API Key*, *path server*, ecc.	S
TS11	Verifica che l'Analista può creare un caso su TheHive, che identifica un incidente da analizzare.	S
TS12	Verifica che l'Analista può aggiungere un task ad un caso su TheHive, che corrisponde ad un'attività pianificata da svolgere.	S
TS13	Verifica che l'Analista può aggiungere un observable su TheHive per poter effettuare le analisi del caso.	S
TS14	Verifica che l'Analista può eseguire l'insieme degli *analyzer*, o un suo sottoinsieme, su un observable da TheHive. Gli *analyzer* sono definiti e configurati da Cortex.	S
TS15	Verifica che l'Analista può aggiungere la descrizione di tattiche, tecniche e procedure (TTP) osservati ad un caso studiato su TheHive.	S
TS16	Verifica che l'Analista può condividere su MISP i risultati e le informazioni raccolte dei casi analizzati.	S
TS17	Verifica che i servizi "deployati" sono integrati tra di loro.	S
TS18	Verifica che è stato fornito un documento che dettaglia i requisiti funzionali e non funzionali raccolti attraverso il confronto con l'azienda ed emersi durante la fase di analisi.	S
TS19	Verifica che è stato fornito un documento che definisce i requisiti hardware e software per il deployment della strumentazione alternativa proposta.	S
TS20	Verifica che è stato fornito un diagramma architetturale ed eventuali diagrammi di flusso.	S
TS21	Verifica che è stato fornito un file markdown che spiega passo passo come effettuare il deployment della strumentazione.	S
TS22	Verifica che il cluster è stato integrato con i servizi cloud come Microsoft Azure e Google Cloud Platform.	S

Tabella 6.1 – *continuazione da pagina precedente*

Test	Descrizione	Risultato
TS23	Verifica che il deployment dei servizi è avvenuto attraverso Docker.	S
TS24	Verifica che il cluster è stato gestito e contenuto con Docker Swarm.	S

6.2 Tracciamento

La seguente tabella indica la corrispondenza tra i test di sistema identificati e i requisiti elencati nella sezione 4.4.1.

Tabella 6.2: Tracciamento dei test di sistema con i requisiti

Test	Requisito
TS1	RFA1
TS2	RFA2
TS3	RFA3
TS4	RFA4
TS5	RFA5
TS6	RFA6
TS7	RFA7
TS8	RFA8
TS9	RFA9
TS10	RFA10
TS11	RFA11
TS12	RFA12
TS13	RFA13
TS14	RFA14
TS15	RFA15
TS16	RFA16
TS17	RFA17
TS18	RVA1
TS19	RVA2
TS20	RVA3
TS21	RVA4
TS22	RVA5
TS23	RVA6
TS24	RVA7

Capitolo 7

Conclusioni

In questo capitolo si presentano il consuntivo finale e il raggiungimento degli obiettivi.

7.1 Consuntivo finale

L'attività di stage è iniziata in data 28 giugno 2021 e si è conclusa in data 20 agosto 2021 per un totale di 320 ore rendicontate e una valenza di 11 crediti formativi. Il lavoro è stato svolto in modalità duale, ovvero prevalentemente da remoto con un giorno alla settimana in presenza presso la sede dell'azienda. La struttura preesistente dei server, il funzionamento di Docker/Docker Swarm e dei singoli servizi ha richiesto più ore di studio rispetto a quelle preventivate, tuttavia sono stati attuati dei compromessi con il team sulla definizione dei requisiti funzionali e non funzionali, attività che ha permesso di rientrare nei tempi previsti e portare a termine il lavoro soddisfando quanti più requisiti richiesti.

Nel corso del progetto si sono verificate diverse criticità nello svolgimento delle task. In una fase iniziale è stato necessario consultare alcuni materiali forniti dal corso di "Tecnologie Open Source" dell'Università degli Studi di Padova, per istruirsi tramite esempi sull'orchestrazione dei servizi con Docker Swarm. Prima di apprendere completamente i comandi necessari per gestire il cluster, la soluzione di TheHive, Cortex e MISP è stata fatta girare su una macchina locale con docker-compose per verificarne il funzionamento, i cui servizi, se applicata la configurazione come da istruzioni disponibili online al sito `https://github.com/TheHive-Project/Docker-Templates/tree/main/docker/thehive4-cortex3-misp-shuffle`, sembrano operabili a primo impatto (ndr. più avanti si descrive il perché di questa precisazione).
Effettuati i dovuti test si è passati al deployment dei servizi sul cluster, trasponendo i file su demo-bushi-1. A causa del passaggio su cluster Elasticsearch (un servizio dipendente per Cortex) ha smesso di rispondere. Solo dopo varie consultazioni della documentazione online si è compreso che questi servizi non hanno esposto più una porta all'interno di Docker (compito ora delegato a Traefik) come nella macchina locale, quindi necessitano di una rete virtuale [6] che possa fare da collegamento tra i due servizi, qui chiamata "internal-private" per risolvere le dipendenze.
Un'altra problematica bloccante è stata la gestione dei file all'interno del cluster, con cui si è poi optato per mantenere tutto su un unico nodo. Per arrivare a ciò tuttavia, c'è stato bisogno di uno studio approfondito dei volumi [9]. In un primo luogo sono

stati usati i volumi *unnamed* (senza nome) che Docker mette a disposizione, però è stato evidente che per svolgere attività di backup e di ripristino sui dati sarebbe risultato ostico. Per usare i volumi che hanno un nome sono stati specificati i *path*, gli indirizzi per raggiungere le cartelle dove memorizzare i dati. Non è stato affatto una operazione scontata dato che la configurazione fornita online di TheHive non permette di lavorare immediatamente su cluster. Infatti, l'errore più frequente durante il deployment dei servizi è stato relativo al salvataggio degli *index* (indici ai dati) e dei dati stessi. Nonostante il volume di Docker fosse impostato correttamente nella configurazione di TheHive, il servizio non è stato in grado di ottenere i permessi di lettura al suo interno (i.e nel container dove è eseguito) e pertanto avviarsi. Per questo motivo è stato modificata l'immagine di TheHive per creare le cartelle necessarie al *montaggio* del volume e mantenere i diritti di lettura e scrittura intatti.
Superati tali problemi di deployment, all'utilizzo degli *analyzer* di Cortex si è notato che l'azienda al suo interno opera un *proxy* che non permette di accedere a determinati siti, facendo così ritornare un errore con codice 104 seguito dal messaggio "Connection reset by peer". Per risolvere è stato tentato di aggiungere una macchina apposita per fare uscire il cluster su Internet e ovviare alle limitazioni imposte dal *proxy*, tuttavia non è stato ottenuto il risultato desiderato ed è stata fatta una richiesta formale alla direzione aziendale per chiedere la possibilità di connessione agli *analyzer* bloccati per gli indirizzi IP del cluster.

7.2 Raggiungimento degli Obiettivi

Tutti gli obiettivi obbligatori sono stati raggiunti con successo, mentre solo alcuni degli obiettivi desiderabili sono stato soddisfatti. La tabella seguente illustra in dettaglio il raggiungimento degli obiettivi.

Per raggiungimento dell'obiettivo definiamo:

* **S**: l'obiettivo è stato raggiunto;
* **N**: l'obiettivo non è stato raggiunto.

Gli obiettivi ripresi sono identificati nella sezione 2.6.2.

Tabella 7.1: Raggiungimento degli obiettivi fissati

Obiettivo	Descrizione	Raggiunto
O1	Stesura del documento "Requisiti di strumentazione".	S
O2	Stesura del documento "Requisiti hardware e software per il deployment".	S
O3	Stesura del documento "Specifiche SW".	S
O4	Stesura del documento "README".	S
O5	Installazione e configurazione di una soluzione di *Incident Management* (p.es. TheHive).	S
O6	Installazione e configurazione di una soluzione di *Communication Management* e *Document Sharing* per l'Incident Response (p.es. Nextcloud).	S*

Tabella 7.1 – *continuazione da pagina precedente*

Obiettivo	Descrizione	Raggiunto
O7	Installazione e configurazione di una soluzione di analisi collaborativa per l'*IR* (p.es. Timesketch).	S*
D1	Installazione e configurazione di una soluzione di Threat Intelligence (p.es. OpenCTI)	S*
D2	Installazione e configurazione di una soluzione di collaborazione per Ethical Hacking (p.es. Dradis Framework)	N
D3	Esposizione del lavoro di tesi ad un meeting interno "Lunch&Learn".	N

O6: una soluzione di *Communication Management* e *Document Sharing* per l'Incident Response è stata individuata in MISP piuttosto che nell'esempio fornito (Nextcloud).
O7: una soluzione di analisi collaborativa per l'Incident Response è stata individuata in TheHive piuttosto che nell'esempio fornito (Timesketch).
D1: una soluzione di Threat Intelligence è stata individuata in Cortex e MISP piuttosto che nell'esempio fornito (OpenCTI).
Non è stato possibile raggiungere l'obiettivo D2 perché si è scelto col team di dedicare tutto il lavoro di stage sulle attività di Incident Response e Threat Intelligence.
Non è stato possibile raggiungere l'obiettivo D3 perché l'organizzazione del meeting interno richiedeva un effort maggiore di quello previsto.

7.3 Conoscenze acquisite e valutazione personale

Il risultato è stato soddisfacente: il cluster e i servizi sono in funzione e forniscono un forte strumento agli analisti degli incidenti di sicurezza informatica. Sono stati raggiunti tutti gli obiettivi obbligatori nei tempi previsti e il team è sempre stato disponibile nella risoluzione delle difficoltà incontrate durante il periodo di stage. L'esperienza è stata positiva come prima attività commissionata in ambito Cybersecurity, vi è stata l'occasione di imparare la funzione e l'utilizzo delle attività di Incident Response e Threat Intelligence, e finalmente estendere il gergo tecnico.

7.4 Sviluppi futuri

I processi che compongono i servizi "deployati" possono essere automatizzati con strumenti dedicati: Shuffle [13], a titolo d'esempio, può essere configurato per far avviare gli *analyzer* di Cortex quando si creano o si modificano gli observable da TheHive.
Gli strumenti e le configurazioni inerenti possono essere venduti a terzi interessati nell'analisi degli incidenti di sicurezza informatica.

Appendice A

traefik/traefik.yml

In questo appendice è disponibile il contenuto del file di configurazione utilizzato per Traefik.

```
version: '3.8'

services:

  traefik:
    # Use the latest v2.2.x Traefik image available
    image: traefik:v2.5.1
    ports:
      # Listen on port 80, default for HTTP, necessary to
          redirect to HTTPS
      - 80:80
      # Listen on port 443, default for HTTPS
      - 443:443
    deploy:
      placement:
        constraints:
          # Make the traefik service run only on the node with
              this label
          # as the node with it has the volume for the
              certificates
          - node.labels.traefik-public.traefik-public-
              certificates == true
      labels:
        # Enable Traefik for this service, to make it available
            in the public network
        - traefik.enable=true
        # Use the traefik-public network (declared below)
        - traefik.docker.network=traefik-public
        # Use the custom label "traefik.constraint-label=traefik-
            public"
        # This public Traefik will only use services with this
            label
        # That way you can add other internal Traefik instances
            per stack if needed
        - traefik.constraint-label=traefik-public
```

```
      # admin-auth middleware with HTTP Basic auth
      # Using the environment variables USERNAME and HASHED_
        PASSWORD
      - traefik.http.middlewares.admin-auth.basicauth.users=${
        USERNAME?Variable not set}:${HASHED_PASSWORD?Variable
        not set}
      # https-redirect middleware to redirect HTTP to HTTPS
      # It can be re-used by other stacks in other Docker
        Compose files
      - traefik.http.middlewares.https-redirect.redirectscheme.
        scheme=https
      - traefik.http.middlewares.https-redirect.redirectscheme.
        permanent=true
      # traefik-http set up only to use the middleware to
        redirect to https
      # Uses the environment variable DOMAIN
      - traefik.http.routers.traefik-public-http.rule=Host(`${
        DOMAIN?Variable not set}`)
      - traefik.http.routers.traefik-public-http.entrypoints=
        http
      - traefik.http.routers.traefik-public-http.middlewares=
        https-redirect
      # traefik-https the actual router using HTTPS
      # Uses the environment variable DOMAIN
      - traefik.http.routers.traefik-public-https.rule=Host(`${
        DOMAIN?Variable not set}`)
      - traefik.http.routers.traefik-public-https.entrypoints=
        https
      - traefik.http.routers.traefik-public-https.tls=true
      # Use the special Traefik service api@internal with the
        web UI/Dashboard
      - traefik.http.routers.traefik-public-https.service=
        api@internal
      # Use the "le" (Let's Encrypt) resolver created below
      - traefik.http.routers.traefik-public-https.tls.
        certresolver=le
      # Enable HTTP Basic auth, using the middleware created
        above
      - traefik.http.routers.traefik-public-https.middlewares=
        admin-auth
      # Define the port inside of the Docker service to use
      - traefik.http.services.traefik-public.loadbalancer.
        server.port=8080
  volumes:
    # Add Docker as a mounted volume, so that Traefik can read
      the labels of other services
    - /var/run/docker.sock:/var/run/docker.sock:ro
    # Mount the volume to store the certificates
    - traefik-public-certificates:/certificates
  command:
    # Enable Docker in Traefik, so that it reads labels from
      Docker services
    - --providers.docker
    # Add a constraint to only use services with the label "
```

```
        traefik.constraint-label=traefik-public"
      - --providers.docker.constraints=Label(`traefik.constraint-
        label`, `traefik-public`)
      # Do not expose all Docker services, only the ones
        explicitly exposed
      - --providers.docker.exposedbydefault=false
      # Enable Docker Swarm mode
      - --providers.docker.swarmmode
      # Create an entrypoint "http" listening on port 80
      - --entrypoints.http.address=:80
      # Create an entrypoint "https" listening on port 443
      - --entrypoints.https.address=:443
      # Create the certificate resolver "le" for Let's Encrypt,
        uses the environment variable EMAIL
      - --certificatesresolvers.le.acme.email=${EMAIL?Variable
        not set}
      # Store the Let's Encrypt certificates in the mounted
        volume
      - --certificatesresolvers.le.acme.storage=/certificates/
        acme.json
      # Use the TLS Challenge for Let's Encrypt
      - --certificatesresolvers.le.acme.tlschallenge=true
      # Enable the access log, with HTTP requests
      - --accesslog
      # Enable the Traefik log, for configurations and errors
      - --log
      # Enable the Dashboard and API
      - --api
    networks:
      # Use the public network created to be shared between
        Traefik and
      # any other service that needs to be publicly available
        with HTTPS
      - traefik-public

volumes:
  # Create a volume to store the certificates, there is a
    constraint to make sure
  # Traefik is always deployed to the same Docker node with the
    same volume containing
  # the HTTPS certificates
  traefik-public-certificates:

networks:
  # Use the previously created public network "traefik-public",
    shared with other
  # services that need to be publicly available via this Traefik
  traefik-public:
    external: true
```

Appendice B

cortex/application.conf

In questo appendice è disponibile il file della configurazione interna del servizio di Cortex. Attenzione a non fare confusione con la configurazione del servizio stesso, che è reperibile in docker-compose.yml.

```
# Sample Cortex application.conf file

## SECRET KEY
#
# The secret key is used to secure cryptographic functions.
#
# IMPORTANT: If you deploy your application to several  instances
    ,  make
# sure to use the same key.
play.http.secret.key="msd3232fdn3ofgfbki83ihtzHSD"

## ElasticSearch
search {
  # Name of the index
  index = cortex
  # ElasticSearch instance address.
  # For cluster, join address:port with ',': "http://ip1:9200,ip
     2:9200,ip3:9200"
  uri = "http://elasticsearch:9200"

  ## Advanced configuration
  # Scroll keepalive.
  #keepalive = 1m
  # Scroll page size.
  #pagesize = 50
  # Number of shards
  #nbshards = 5
  # Number of replicas
  #nbreplicas = 1
  # Arbitrary settings
  #settings {
  #  # Maximum number of nested fields
  #  mapping.nested_fields.limit = 100
```

```
  #}

  ## Authentication configuration
  #search.username = ""
  #search.password = ""

  ## SSL configuration
  #search.keyStore {
  #  path = "/path/to/keystore"
  #  type = "JKS" # or PKCS12
  #  password = "keystore-password"
  #}
  #search.trustStore {
  #  path = "/path/to/trustStore"
  #  type = "JKS" # or PKCS12
  #  password = "trustStore-password"
  #}
}

## Cache
#
# If an analyzer is executed against the same observable, the
    previous report can be returned without re-executing the
# analyzer. The cache is used only if the second job occurs
    within cache.job (the default is 10 minutes).
cache.job = 10 minutes

## Authentication
auth {
        # "provider" parameter contains the authentication
            provider(s). It can be multi-valued, which is useful
        # for migration.
        # The available auth types are:
        # - services.LocalAuthSrv : passwords are stored in the
            user entity within ElasticSearch). No
        #   configuration are required.
        # - ad : use ActiveDirectory to authenticate users. The
            associated configuration shall be done in
        #   the "ad" section below.
        # - ldap : use LDAP to authenticate users. The associated
             configuration shall be done in the
        #   "ldap" section below.
        # - oauth2 : use OAuth/OIDC to authenticate users.
            Configuration is under "auth.oauth2" and "auth.sso"
            keys
        provider = [local]

        ad {
                # The Windows domain name in DNS format. This
                    parameter is required if you do not use
                # 'serverNames' below.
                #domainFQDN = "mydomain.local"

                # Optionally you can specify the host names of
```

```
                    the domain controllers instead of using '
                    domainFQDN
                # above. If this parameter is not set, TheHive
                    uses 'domainFQDN'.
                #serverNames = [ad1.mydomain.local, ad2.mydomain.
                    local]

                # The Windows domain name using short format.
                    This parameter is required.
                #domainName = "MYDOMAIN"

                # If 'true', use SSL to connect to the domain
                    controller.
                #useSSL = true
        }

        ldap {
                # The LDAP server name or address. The port can
                    be specified using the 'host:port'
                # syntax. This parameter is required if you don't
                     use 'serverNames' below.
                #serverName = "ldap.mydomain.local:389"

                # If you have multiple LDAP servers, use the
                    multi-valued setting 'serverNames' instead.
                #serverNames = [ldap1.mydomain.local, ldap2.
                    mydomain.local]

                # Account to use to bind to the LDAP server. This
                     parameter is required.
                #bindDN = "cn=thehive,ou=services,dc=mydomain,dc=
                    local"

                # Password of the binding account. This parameter
                     is required.
                #bindPW = "***secret*password***"

                # Base DN to search users. This parameter is
                    required.
                #baseDN = "ou=users,dc=mydomain,dc=local"

                # Filter to search user in the directory server.
                    Please note that {0} is replaced
                # by the actual user name. This parameter is
                    required.
                #filter = "(cn={0})"

                # If 'true', use SSL to connect to the LDAP
                    directory server.
                #useSSL = true
        }

    oauth2 {
        # URL of the authorization server
```

```
        #clientId = "client-id"
        #clientSecret = "client-secret"
        #redirectUri = "https://my-thehive-instance.example/index
            .html#!/login"
        #responseType = "code"
        #grantType = "authorization_code"

        # URL from where to get the access token
        #authorizationUrl = "https://auth-site.com/OAuth/
            Authorize"
        #tokenUrl = "https://auth-site.com/OAuth/Token"

        # The endpoint from which to obtain user details using
            the OAuth token, after successful login
        #userUrl = "https://auth-site.com/api/User"
        #scope = "openid profile"
        # Type of authorization header
        #authorizationHeader = "Bearer" # or token
    }

    # Single-Sign On
    sso {
        # Autocreate user in database?
        #autocreate = false

        # Autoupdate its profile and roles?
        #autoupdate = false

        # Autologin user using SSO?
        #autologin = false

        # Attributes mappings
        #attributes {
        #  login = "login"
        #  name = "name"
        #  groups = "groups"
        #  roles = "roles" # list of roles, separated with comma
        #  organisation = "org"
        #}

        # Name of mapping class from user resource to backend
            user ('simple' or 'group')
        #mapper = group
        # Default roles for users with no groups mapped ("read",
            "analyze", "orgadmin")
        #defaultRoles = []
        # Default organization
        #defaultOrganization = "MyOrga"

        #groups {
        #  # URL to retreive groups (leave empty if you are using
             OIDC)
        #  #url = "https://auth-site.com/api/Groups"
        #  # Group mappings, you can have multiple roles for each
```

```
            group: they are merged
        #  mappings {
        #    admin-profile-name = ["admin"]
        #    editor-profile-name = ["write"]
        #    reader-profile-name = ["read"]
        #  }
        #}
    }
}

job {
  runner = [docker]
}
## ANALYZERS
#
analyzer {
  # analyzer location
  # url can be point to:
  # - directory where analyzers are installed
  # - json file containing the list of analyzer descriptions
  urls = [
    #"https://download.thehive-project.org/analyzers.json"
    "/opt/Cortex-Analyzers/analyzers"
  ]

  # Sane defaults. Do not change unless you know what you are
     doing.
  fork-join-executor {
    # Min number of threads available for analysis.
    parallelism-min = 2
    # Parallelism (threads) ... ceil(available processors *
       factor).
    parallelism-factor = 2.0
    # Max number of threads available for analysis.
    parallelism-max = 4
  }
}

# RESPONDERS
#
responder {
  # responder location (same format as analyzer.urls)
  urls = [
    "https://download.thehive-project.org/responders.json"
    #"/absolute/path/of/responders"
  ]

  # Sane defaults. Do not change unless you know what you are
     doing.
  fork-join-executor {
    # Min number of threads available for analysis.
    parallelism-min = 2
    # Parallelism (threads) ... ceil(available processors *
       factor).
```

```
    parallelism-factor = 2.0
    # Max number of threads available for analysis.
    parallelism-max = 4
  }
}

# It's the end my friend. Happy hunting!
```

Appendice C

cortex/Dockerfile

In questo appendice è disponibile il contenuto della personalizzazione attuata all'immagine di Cortex.

```
FROM thehiveproject/cortex:latest

USER root

RUN apt-get update && apt-get install -y sudo

RUN sudo apt-get install -y --no-install-recommends python2
    python3 python3-pip ssdeep libfuzzy-dev libfuzzy2 libimage-
    exiftool-perl libmagic1 build-essential git libssl-dev

RUN sudo pip install -U pip setuptools requests && sudo pip3
    install -U pip setuptools

COPY /Cortex-Analyzers/analyzers /dependencies

RUN for I in $(find /dependencies -name 'requirements.txt'); do
    sudo -H pip install -r $I; done && \
for I in $(find /dependencies -name 'requirements.txt'); do sudo
    -H pip3 install -r $I || true; done
```

Appendice D

thehive/application.conf

In questo appendice è disponibile il file della configurazione interna del servizio di TheHive. Attenzione a non fare confusione con la configurazione del servizio stesso, che è reperibile in docker-compose.yml.

```
play.http.secret.key="t5EeDXh2dEtJxohh"

# JanusGraph
db {
  provider: janusgraph
  janusgraph {
    storage {
      backend: cql
      hostname: ["cassandra"]
      cql {
        cluster-name: thp        # cluster name
        keyspace: thehive            # name of the keyspace
        #read-consistency-level: ONE
        #write-consistency-level: ONE
      }
    }

    ## Index configuration
    index {
      search {
        backend: lucene
        directory: /opt/thp/thehive/index
      }
    }
  }
}

storage {
   provider: localfs
   localfs.location: /opt/thp/thehive/data
}

play.http.parser.maxDiskBuffer: 50MB
```

```
# Cortex configuration
play.modules.enabled += org.thp.thehive.connector.cortex.
    CortexModule
cortex {
  servers = [
    {
      name = local
      url = "http://cortex:9001"
      auth {
        type = "bearer"
        key = "KB9tEVnphcvaf5Gep7k3DyE/Qw66zYlz" # Cortex API key
      }
      # HTTP client configuration (SSL and proxy)
      #  wsConfig {}
     # List TheHive organisation which can use this Cortex server
         . All ("*") by default
     # includedTheHiveOrganisations = ["*"]
     # List TheHive organisation which cannot use this Cortex
         server. None by default
     # excludedTheHiveOrganisations = []
    }
  ]
  # Check job update time intervalcortex
  refreshDelay = 5 seconds
  # Maximum number of successive errors before give up
  maxRetryOnError = 3
  # Check remote Cortex status time interval
  statusCheckInterval = 1 minute
}

# MISP configuration
play.modules.enabled += org.thp.thehive.connector.misp.MispModule
misp {
  interval: 5 min
  servers: [
    {
      name = "MISP THP"            # MISP name
      url = "https://misp/" # URL or MISP
      auth {
        type = key
        key = "cSVNhnkWXTLz10Hjb5KcosrISNJgqS0yxXjq8Vxo" # MISP
            API key
      }
      wsConfig { ssl { loose { acceptAnyCertificate: true } } }
    }
  ]
}

notification.webhook.endpoints = [
  {
    name: local
    url: "http://thehive:5000/"
    version: 0
```

```
    wsConfig: {}
    auth: {type:"none"}
    includedTheHiveOrganisations: []
    excludedTheHiveOrganisations: []
  }
]
```

Appendice E

thehive/Dockerfile

In questo appendice è disponibile il contenuto della personalizzazione attuata all'immagine di TheHive.

```
FROM thehiveproject/thehive4:latest

USER root

RUN mkdir -p /opt/thp/thehive/index
RUN mkdir -p /opt/thp/thehive/data
```

Appendice F

docker-compose.yml

In questo appendice è disponibile il contenuto del file di configurazione utilizzato per i servizi di TheHive, Cortex e MISP.

```
version: "3.8"

services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.16.2
    environment:
      - http.host=0.0.0.0
      - discovery.type=single-node
      - cluster.name=thp
      - script.allowed_types=inline
      - thread_pool.search.queue_size=100000
      - thread_pool.write.queue_size=10000
      - gateway.recover_after_nodes=1
      - xpack.security.enabled=false
      - bootstrap.memory_lock=true
      - 'ES_JAVA_OPTS=-Xms1024m -Xmx1024m'
      - TAKE_FILE_OWNERSHIP=1
    ulimits:
      nofile:
        soft: 65536
        hard: 65536
    volumes:
      - ./vol/elasticsearch_data:/usr/share/elasticsearch/data
      - ./vol/elasticsearch_logs:/usr/share/elasticsearch/logs
    networks:
      - internal-private
    deploy:
      placement:
        constraints:
          - node.labels.server.name == bushi1

  cortex:
    image: localhost:5000/cortex
    build: ./cortex
```

```
    depends_on:
      - elasticsearch
    volumes:
      - ./cortex/application.conf:/etc/cortex/application.conf
      - ./cortex/Cortex-Analyzers/analyzers:/opt/Cortex-Analyzers
        /analyzers
      - /var/run/docker.sock:/var/run/docker.sock
      - /tmp:/tmp
    environment:
      - http_proxy=${http_proxy}
      - https_proxy=${https_proxy}
    networks:
      - traefik-public
      - internal-private
    deploy:
      placement:
        constraints:
          - node.labels.server.name == bushi1
      labels:
      - traefik.enable=true
      - traefik.docker.network=traefik-public
      - traefik.constraint-label=traefik-public
      - traefik.http.routers.cortex-http.rule=Host(`cortex.
        localhost`)
      - traefik.http.routers.cortex-http.entrypoints=http
      - traefik.http.routers.cortex-http.middlewares=https-
        redirect
      - traefik.http.routers.cortex-https.rule=Host(`cortex.
        localhost`)
      - traefik.http.routers.cortex-https.entrypoints=https
      - traefik.http.routers.cortex-https.tls=true
      - traefik.http.routers.cortex-https.tls.certresolver=le
      - traefik.http.services.cortex.loadbalancer.server.port
        =9001

  cassandra:
    image: cassandra:3.11
    environment:
      - MAX_HEAP_SIZE=1G
      - HEAP_NEWSIZE=1G
      - CASSANDRA_CLUSTER_NAME=thp
    volumes:
      - ./vol/cassandra-data:/var/lib/cassandra/data
    networks:
      - internal-private
    deploy:
      placement:
        constraints:
          - node.labels.server.name == bushi1

  thehive:
    image: localhost:5000/thehive
    build: ./thehive
    depends_on:
```

```
      - cassandra
    command: '--no-config --no-config-secret'
    volumes:
      - ./thehive/application.conf:/etc/thehive/application.conf
      - ./vol/thehive/data:/opt/thp/thehive/data
      - ./vol/thehive/index:/opt/thp/thehive/index
    networks:
      - traefik-public
      - internal-private
    deploy:
      placement:
        constraints:
          - node.labels.server.name == bushi1
      labels:
        - traefik.enable=true
        - traefik.docker.network=traefik-public
        - traefik.constraint-label=traefik-public
        - traefik.http.routers.thehive-http.rule=Host(`thehiveprj
            .localhost`)
        - traefik.http.routers.thehive-http.entrypoints=http
        - traefik.http.routers.thehive-http.middlewares=https-
            redirect
        - traefik.http.routers.thehive-https.rule=Host(`
            thehiveprj.localhost`)
        - traefik.http.routers.thehive-https.entrypoints=https
        - traefik.http.routers.thehive-https.tls=true
        - traefik.http.routers.thehive-https.tls.certresolver=le
        - traefik.http.services.thehive.loadbalancer.server.port
            =9000

  redis:
    image: redis:latest
    networks:
      - internal-private

  db:
    image: mysql:latest
    command: --default-authentication-plugin=mysql_native_
        password
    environment:
      - "MYSQL_USER=misp"
      - "MYSQL_PASSWORD=example"
      - "MYSQL_ROOT_PASSWORD=password"
      - "MYSQL_DATABASE=misp"
    volumes:
      - ./vol/mysql-data:/var/lib/mysql
    networks:
      - internal-private
    deploy:
      placement:
        constraints:
          - node.labels.server.name == bushi1

  misp:
```

```
    image: coolacid/misp-docker:core-latest
    depends_on:
      - redis
      - db
    environment:
      - "HOSTNAME=http://misp.localhost"
      - "REDIS_FQDN=redis"
      - "INIT=true" # Initialize MISP, things includes,
          attempting to import SQL and the Files DIR
      - "CRON_USER_ID=1" # The MISP user ID to run cron jobs as
      - "DISIPV6=true" # Disable IPV6 in nginx
      - "NOREDIR=true"
    networks:
      - traefik-public
      - internal-private
    deploy:
     labels:
      - traefik.enable=true
      - traefik.docker.network=traefik-public
      - traefik.constraint-label=traefik-public
      - traefik.http.routers.misp-http.rule=Host(`misp.localhost
          `)
      - traefik.http.routers.misp-http.entrypoints=http
      - traefik.http.routers.misp-http.middlewares=https-redirect
      - traefik.http.routers.misp-https.rule=Host(`misp.localhost
          `)
      - traefik.http.routers.misp-https.entrypoints=https
      - traefik.http.routers.misp-https.tls=true
      - traefik.http.routers.misp-https.tls.certresolver=le
      - traefik.http.services.misp.loadbalancer.server.port=80

  misp-modules:
    image: coolacid/misp-docker:modules-latest
    environment:
      - "REDIS_BACKEND=redis"
    depends_on:
      - redis
      - db
    networks:
      - internal-private

networks:
  traefik-public:
    external: true
  internal-private:
```

Appendice G

start.sh

In questo appendice è disponibile il contenuto dello script di avvio dei servizi.

```
#!/usr/bin/env bash
if [ $# -eq 0 ]
  then
    tag='latest'
  else
    tag=$1
fi

docker service create --name registry --constraint node.labels.
   server.name==bushi1 --publish published=5000,target=5000
   registry:2

docker build ./cortex -t localhost:5000/cortex:$tag

docker build ./thehive -t localhost:5000/thehive:$tag

docker-compose push

docker stack deploy demo -c docker-compose.yml
```

Appendice H

stop.sh

In questo appendice è disponibile il contenuto dello script di spegnimento dei servizi.

```
docker stack rm demo

docker service rm registry
```

Glossario

API in informatica con il termine Application Programming Interface (ing. interfaccia di programmazione di un'applicazione) si indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione, di solito tra l'hardware e il programmatore o tra software a basso e quello ad alto livello semplificando così il lavoro di programmazione. 3, 10, 12, 14, 70

CERT sono organizzazioni incaricate di raccogliere e gestire le segnalazioni di incidenti informatici e potenziali vulnerabilità nei *software* che provengono dalla comunità degli utenti. I CERT si pongono come un punto di riferimento per gli utenti della rete, in grado di aiutarli a risolvere qualunque problema legato alla sicurezza informatica. Di norma un CERT è composto da persone specializzate in diversi ambiti, per esempio amministratori di rete, amministratori di sistema ed esperti in sicurezza informatica. Essi sono a conoscenza di come dovrebbero apparire, in condizioni normali, i settori di loro competenza, e dunque sono in grado di accorgersi immediatamente dell'eventuale verificarsi di un'anomalia. I compiti fondamentali di un CERT consistono nel rispondere alle segnalazioni degli utenti vittime di incidenti informatici e nell'analizzare i sistemi hardware e software per individuarvi eventuali vulnerabilità. 8, 65, 70

CIDR conosciuto come Classless Inter-Domain Routing, è un metodo per l'assegnazione di indirizzi IP e per il routing IP. Il CIDR si basa sull'idea delle maschere di sottorete. Una maschera viene applicata a un indirizzo IP, creando così una sottorete: si tratta di una rete subordinata a Internet. La maschera di sottorete segnala al router quale parte dell'indirizzo IP è subordinato all'host (i singoli partecipanti della rete) e quale definisce la rete. 11, 65, 70

Cluster è un insieme di computer connessi tra loro tramite una rete. Lo scopo del cluster è distribuire un'elaborazione molto complessa tra i vari computer, aumentando la potenza di calcolo del sistema e/o garantendo una maggiore disponibilità di servizio, a prezzo di un maggior costo e complessità di gestione dell'infrastruttura: per essere risolto il problema che richiede molte elaborazioni viene infatti scomposto in sottoproblemi separati i quali vengono risolti ciascuno in parallelo. 15–17, 20–22, 25–28, 39–44, 65

CSIRT sono gruppi predisposti a rispondere in caso di incidenti informatici, sono meglio conosciuti come CERT. 8, 70

Deployment è il rilascio al cliente di un sistema software o di un'applicazione, in genere nel contesto di un sistema informatico aziendale, con la relativa installazione

e la sua messa in esercizio. Per questo motivo, si può parlare del deployment come di una parte del ciclo di vita del *software*, che termina lo sviluppo e il *testing* necessario per avviare la manutenzione. 4–6, 21, 40–43, 65, 66

DevSecOps sta per sviluppo, sicurezza e operazioni. È un approccio alla cultura, all'automazione e alla progettazione delle piattaforme che integra la sicurezza come responsabilità condivisa lungo l'intero ciclo di vita IT. 5, 66

Ethical Hacking è una particolare variante delle operazioni di *Penetration Testing* che consiste in una vera e propria simulazione di attacco informatico volto ad impossessarsi del sistema *target* sfruttando le vulnerabilità, informatiche e sociali, riscontrate; l'attività può essere svolta sia dalla rete interna dell'azienda sia dalla rete esterna, riproducendo il modus operandi di un attaccante. 4, 5, 7, 44, 66

gRPC conosciuto come Google Remote Procedure Call, è un *framework open source* ad alte prestazioni che può essere eseguito in qualsiasi ambiente. Può connettere in modo efficiente i servizi tra i *data center* con un supporto per il bilanciamento del carico, il tracciamento, il controllo della salute e l'autenticazione dei sistemi. È anche applicabile nell'ultimo *step* del calcolo distribuito per collegare dispositivi, applicazioni mobili e *browser* ai servizi di *back-end*. 14, 70

HTTPS conosciuto come HyperText Transfer Protocol over Secure Socket Layer, è un protocollo per la comunicazione sicura attraverso una rete di computer utilizzato su Internet. Consiste nella comunicazione tramite il protocollo *HTTP* (*Hypertext Transfer Protocol*) all'interno di una connessione criptata, tramite crittografia asimmetrica, dal *TLS* (*Transport Layer Security*) o dal suo predecessore, *SSL* (*Secure Sockets Layer*) fornendo come requisiti chiave:

- un'autenticazione del sito web visitato;
- protezione della privacy (riservatezza o confidenzialità);
- integrità dei dati scambiati tra le parti comunicanti.

La porta utilizzata tipicamente è la 443. 14, 24, 70

ICT conosciuto come Information and Communication Technologies, è un termine estensivo per la tecnologia dell'informazione (IT) che sottolinea il ruolo delle comunicazioni unificate e l'integrazione delle telecomunicazioni (p.es. linee telefoniche e segnali *wireless*) e dei computer, così come il *software* aziendale, di archiviazione e audiovisivo, che permettono agli utenti di accedere, memorizzare, trasmettere, comprendere e manipolare le informazioni. 11, 70

Incident Response è la metodologia che un'organizzazione utilizza per gestire un attacco informatico. Un attacco o una violazione dei dati può portare scompiglio, colpendo potenzialmente i clienti, la proprietà intellettuale, il tempo e le risorse dell'azienda e il valore del marchio. Una risposta agli incidenti mira a ridurre questi danni e a recuperare lo stato dei sistemi il più rapidamente possibile. Poiché molte aziende oggi sperimentano una violazione prima o poi, è fortemente suggerito un piano di risposta agli incidenti ben sviluppato e ripetibile per proteggersi. 1, 4, 7, 8, 43, 44, 67, 70

IoC conosciuto come Indicator of Compromise, costituisce la prova del *data breach*, ovvero la fuoriuscita dei dati. Queste tracce digitali rivelano non soltanto che è avvenuto l'incidente, ma spesso permettono anche di scoprire quali strumenti sono stati usati per sferrare l'attacco e da chi. Gli IoC possono anche essere utilizzati per determinare in quale grado l'incidente informatico abbia colpito l'organizzazione, e per mettere in sicurezza la rete da possibili attacchi futuri. Gli IoC vengono tipicamente raccolti da appositi software, come gli *antivirus* e gli *anti malware*, ma anche nuovi strumenti basati sull'intelligenza artificiale vengono utilizzati sempre più spesso per aggregare e organizzare gli IoC durante le fasi di Incident Response. 9, 11, 67, 70

LDAP conosciuto come Lightweight Directory Access Protocol, è un protocollo standard per l'interrogazione e la modifica dei servizi di *directory*, come p.es. un elenco aziendale di email o una rubrica telefonica, o più in generale qualsiasi raggruppamento di informazioni che può essere espresso come *record* di dati e organizzato in modo gerarchico. 10, 70

LIDS conosciuto come Learning Intrusion Detection System, è un sistema con capacità di apprendimento del modello comportamentale negli attacchi al fine di rilevare le intrusioni in modo più efficace. 11, 70

Multi-tenant si riferisce ad una architettura *software* in cui una sua singola istanza è eseguita da un server ed è fruita da diverse organizzazioni che, ciascuna con le sue peculiarità ambientali che costituiscono concettualmente uno specifico *tenant* (come in un immobile le cui unità o vani sono affittati a locatari diversi), vedono il *software* (cioè il sistema o l'applicazione) come a loro utilizzo esclusivo e, per gli aspetti eventualmente finanziari, ad ognuna di esse fatturato come servizio. 9, 67

NIDS conosciuto come Network Intrusion Detection System, è un insieme di strumenti informatici, *software* o *hardware*, dediti ad analizzare il traffico di uno o più segmenti di una rete locale al fine di individuare anomalie nei flussi o probabili intrusioni informatiche. 11, 12, 70

Observable è un evento (benigno o maligno) su una rete o un sistema. 8–11, 18, 19, 21, 40, 44, 67

OpenIoC è un *framework* aperto, destinato a condividere le informazioni di *intelligence* sulle minacce in un formato leggibile dalla macchina. È scritto in *XML* (*eXtensible Markup Language*) e può essere facilmente personalizzato per l'*intelligence* in modo che chi risponde agli incidenti possa tradurre le proprie conoscenze in un formato standard. Le organizzazioni possono sfruttare questo formato per condividere gli ultimi IoC relativi alle minacce con altre organizzazioni, consentendo una protezione in tempo reale contro le ultime minacce. 12, 67

OPSEC conosciuto come Operations Security NATO AAP-6, è il processo di protezione dei dati che potrebbero essere aggregati. OPSEC è la protezione di informazioni critiche ritenute essenziali per la missione da parte di comandanti militari, dirigenti o altri organi decisionali. 11, 67, 70

Orchestratore è una figura preposta alla soluzione di problemi non banali e lo fa coordinando in modo automatico la gestione delle risorse di un sistema informatico complesso, il che comporta l'intervento di diverse componenti, soprattutto *software* ma non necessariamente, come servizi software disponibili e accessibili via *Cloud*, o come apparati resi disponibili in forma di servizi. Con un comando *software* assimilabile al richiamo di altro *software* si attivano risorse che grazie alla *Cloud* possono essere remote, e vengono presentate come se fossero servizi calcolo richiamabili in modo semplice anche quando si tratta in realtà di attivare interi sistemi di calcolo distribuito, *database*, ed apparati misti *software*/*hardware*. 13, 14, 68

RAT conosciuto come Remote Access Trojan, è un *malware* che si può scaricare da Internet o che viene installato sul computer della vittima a sua insaputa. Il RAT permette all'*hacker* di avere il controllo amministrativo del computer, quindi può effettuare qualsiasi tipo di azione. Infatti, il *payload* legato a questo *malware* (ovvero la porzione di codice eseguibile del *malware*, le azioni che il virus esegue dopo aver infettato il sistema) consente di attuare azioni identiche se non maggiori a quelle che può effettuare l'utente proprietario del PC stesso. 12, 68, 70

RBAC conosciuto come Role-based access control, è una tecnica di accesso ristretto per utenti autorizzati. Si tratta di un meccanismo di accesso definito basandosi sui concetti di ruolo e privilegio. I componenti del RBAC, come i permessi dei ruoli, il ruolo utente e le relazioni ruolo-ruolo, fanno in modo di semplificare l'assegnamento dei ruoli agli utenti. Il RBAC può essere usato per facilitare la gestione della sicurezza nelle organizzazioni composte da centinaia di utenti e migliaia di permessi diversi. 9, 68, 70

REST conosciuto come Representational state transfer, è uno stile architetturale per sistemi distribuiti. L'architettura REST si basa su *HTTP*. Il funzionamento prevede una struttura degli URL ben definita che identifica univocamente una risorsa o un insieme di risorse e l'utilizzo dei metodi *HTTP* specifici per il recupero di informazioni, per la modifica e per altri scopi. 10, 14, 68, 70

SIEM conosciuto come Security Information and Event Management, è un sistema di gestione delle informazioni e degli eventi di sicurezza. Integra le funzioni di *Incident Management* con i *workflow* automatizzati per gestire da un'unica posizione l'intero processo di sicurezza, consentendo agli analisti di ridurre i tempi necessari alla risoluzione e alle indagini su allarmi e incidenti di sicurezza. 2, 9, 11, 70

SOC conosciuto come Security Operation Center, è un centro da cui vengono forniti servizi finalizzati alla sicurezza dei sistemi informativi dell'azienda stessa (il cosiddetto SOC interno) o di clienti esterni. Un SOC fornisce tre tipologie di servizi:

* Servizi di gestione: tutte le attività di gestione delle funzionalità di sicurezza legate all'infrastruttura IT (rete, sistemi ed applicazioni) sono centralizzate dal SOC.
* Servizi di monitoraggio: l'infrastruttura IT e di Sicurezza vengono monitorate in tempo reale al fine di individuare tempestivamente tentativi di intrusione, di attacco o di *misuse* dei sistemi.

- Servizi proattivi: sono servizi finalizzati a migliorare il livello di protezione dell'organizzazione (*Security Assessment*, *Vulnerability Assessment*, ecc.).

In generale il SOC è un servizio che si collega, a livello più generale, ai processi di governo e gestione dell'infrastruttura IT aziendale. 8, 68–70

STIX conosciuto come Structured Threat Information Expression, è un linguaggio e un formato di serializzazione usato per scambiare informazioni sulle minacce informatiche. STIX è *open source* e gratuito. 12, 69, 70

Threat Intelligence è l'insieme della conoscenza, delle competenze e delle informazioni basate sull'esperienza maturata al verificarsi delle minacce e dello sviluppo dei gruppi criminali, al fine di aiutare a mitigare i potenziali attacchi ed eventi dannosi. Le fonti di *intelligence* sulle minacce informatiche includono più tipi: *open source*, *social media*, umana, tecnica, i file di log dei dispositivi, i dati acquisiti forensicamente o l'*intelligence* dal traffico Internet e i dati derivati dal *deep* e *dark web*. 1, 4, 7, 44, 70

TLP conosciuto come Traffic Light Protocol, è un sistema per classificare le informazioni sensibili basato sui colori del semaforo (rosso, giallo, verde, bianco). Il concetto fondamentale è che chi genera le informazioni segnala quanto ampiamente vuole che esse siano fatte circolare oltre il destinatario immediato. È progettato per migliorare il flusso di informazioni in modo controllato e fidato. È importante che chiunque gestisca comunicazioni etichettate TLP capisca e obbedisca alle regole del protocollo. 9, 69, 70

TTP conosciuto come Tactics, Techniques, and Procedures, sono i modelli di attività o metodi associati a uno specifico attore o gruppo di attori della minaccia. L'analisi delle TTP aiuta il controspionaggio e le operazioni di sicurezza descrivendo come i gruppi criminali eseguono gli attacchi. 4, 19, 21, 40, 69, 70

Acronimi

API Application Program Interface. 65

CERT Computer Emergency Response Team. 65

CIDR Classless Inter-Domain Routing. 65

CSIRT Computer Security Incident Response Team. 65

gRPC Google Remote Procedure Call. 66

HTTPS HyperText Transfer Protocol over Secure Socket Layer. 66

ICT Information and Communication Technologies. 66

IoC Indicator of Compromise. 67

IR Incident Response. 66

LDAP Lightweight Directory Access Protocol. 67

LIDSP Learning Intrusion Detection System. 67

NIDS Network Intrusion Detection System. 67

OPSEC Operations Security NATO AAP-6. 67

RAT Remote Access Trojan. 68

RBAC Role-based access control. 68

REST Representational state transfer. 68

SIEM Security Information and Event Management. 68

SOC Security Operation Center. 68

STIX Structured Threat Information eXpression. 69

TI Threat Intelligence. 69

TLP Traffic Light Protocol. 69

TTP Tactics, Techniques, and Procedures. 69

Bibliografia

Siti web consultati

[1] *Cortex - GitHub*. URL: https://github.com/TheHive-Project/Cortex (cit. alle pp. 4, 9, 10).

[2] *Cortex - Hardware Pre-requisites*. URL: https://github.com/TheHive-Project/CortexDocs#:~:text=Hardware%20Pre-requisites (cit. a p. 22).

[3] *Docker*. URL: https://www.docker.com/ (cit. a p. 13).

[4] *Docker - Compose file versions and upgrading*. URL: https://docs.docker.com/compose/compose-file/compose-versioning/ (cit. a p. 24).

[5] *Docker - Install Docker Engine on Ubuntu*. URL: https://docs.docker.com/engine/install/ubuntu/ (cit. a p. 24).

[6] *Docker - Networking Overview*. URL: https://docs.docker.com/network/ (cit. a p. 42).

[7] *Docker - Post-installation steps for Linux*. URL: https://docs.docker.com/engine/install/linux-postinstall/ (cit. a p. 24).

[8] *Docker - Swarm mode overview*. URL: https://docs.docker.com/engine/swarm/ (cit. a p. 13).

[9] *Docker - Use volumes*. URL: https://docs.docker.com/storage/volumes/ (cit. a p. 42).

[10] *Gluster*. URL: https://www.gluster.org/ (cit. a p. 22).

[11] *MISP - Basic information on MISP ressource usage*. URL: https://www.misp-project.org/MISP-sizer/README.txt#:~:text=%23%20Basic%20information%20on%20MISP%20ressource%20usage (cit. a p. 22).

[12] *MISP - GitHub*. URL: https://github.com/MISP/MISP (cit. alle pp. 4, 8, 11).

[13] *Shuffle - GitHub*. URL: https://github.com/frikky/Shuffle (cit. a p. 44).

[14] *TheHive - GitHub*. URL: https://github.com/TheHive-Project/TheHive (cit. alle pp. 3, 4, 8).

[15] *TheHive - Hardware Pre-requisites*. URL: http://docs.thehive-project.org/thehive/installation-and-configuration/#:~:text=Hardware%20Pre-requisites (cit. a p. 22).

[16] *Traefik - GitHub*. URL: https://github.com/traefik/traefik (cit. a p. 13).