



UNIVERSITA' DEGLI STUDI DI  
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base  
Corso di Laurea in Ingegneria Informatica

Elaborato finale in Sistemi Operativi

***Studio della piattaforma ThreatMapper per  
il monitoraggio della sicurezza***

Anno Accademico 2021/2022

Candidato:

**Lorenzo Discolo**

**Matr. N46004190**

*“One single vulnerability is all an  
attacker needs”*

*Window Snyder*

# Indice

---

Indice.....	III
Introduzione.....	4
Capitolo 1: ThreatMapper e piattaforme supportate.....	5
1.1 Architettura.....	6
1.2 Management Console.....	7
1.3 Sensors .....	8
1.4 Docker.....	10
1.5 Kubernetes.....	12
Capitolo 2: Funzionalità di ThreatMapper .....	14
2.1 Mappatura della superficie di attacco .....	14
2.2 Individuazione delle vulnerabilità .....	17
2.3 Classificazione delle vulnerabilità.....	21
Capitolo 3: Utilizzo e analisi di ThreatMapper .....	25
3.1 Setup dell'infrastruttura software test.....	25
3.2 Test ThreatMapper.....	28
Conclusioni.....	34
Ringraziamenti.....	35
Bibliografia.....	36

## Introduzione

---

L'obiettivo di questo elaborato è quello di analizzare il funzionamento della piattaforma di monitoraggio della sicurezza informatica ThreatMapper, la quale ha come scopo quello di individuare le possibili vulnerabilità di un'infrastruttura di produzione software. Nel primo capitolo viene descritto dal punto di vista architetturale la piattaforma di monitoraggio, in modo tale da dare una visione generale della sua struttura e dei servizi supportati, focalizzando la nostra attenzione principalmente sull'analisi del funzionamento dei software Docker e Kubernetes. Nel secondo capitolo approfondiamo le molteplici funzionalità offerte dalla piattaforma ThreatMapper per il monitoraggio delle possibili vulnerabilità nascoste nelle diverse parti di un progetto software. Infine, nel terzo capitolo si analizza un esempio di utilizzo di ThreatMapper all'interno di un ambiente Linux, precisamente sul sistema operativo Ubuntu, dove poter visionare i dati presentati dalla piattaforma dopo un'analisi approfondita della superficie di attacco dell'intera infrastruttura software presa ad esempio.

## Capitolo 1: ThreatMapper e piattaforme supportate

---

È sempre più frequente incontrare sistemi software costruiti con un approccio architetturale basato sui microservizi. Tale approccio porta ad una scomposizione delle applicazioni in un insieme di servizi indipendenti tra loro. Questo consente di semplificare e velocizzare le diverse fasi di sviluppo delle applicazioni, diminuire il time-to-market (il tempo necessario per passare dalla progettazione del software alla sua effettiva distribuzione), migliorare la scalabilità e favorire un basso accoppiamento tra le parti, rispetto ad una classica architettura monolitica. D'altra parte, [1] la natura distribuita di questo approccio mina la sicurezza informatica dell'intera applicazione, poiché complica la struttura del software con la creazione di una rete formata da servizi forniti da aziende di terze parti, ampliando, difatti, la cosiddetta superficie d'attacco. Inoltre, a causa dell'eterogeneità dei servizi è necessario centralizzare i log, così da poter facilitare le analisi di sicurezza e poter collegare eventi che si verificano su diverse piattaforme.

Per porre rimedio a questi problemi di sicurezza è stata sviluppata dalla compagnia Deepfence, un'azienda internazionale conosciuta nel mondo della sicurezza informatica, la piattaforma open-source e multi-cloud ThreatMapper. Essa si occupa in primo luogo di mappare la topologia dell'ambiente di sviluppo, osservando passivamente il traffico di rete. Indentifica ed interroga le istanze cloud, nodi Kubernetes, le risorse serverless e i container in modo tale da risalire all'intera infrastruttura software in tempo reale. In secondo luogo, ThreatMapper monitora costantemente ed esegue scansioni degli host e dei container con lo scopo di scovare le vulnerabilità, derivanti da dipendenze da software commerciali di terze parti o software open-source, confrontandosi con più di 50 banche dati di minacce note

pubblicamente. Infine, ThreatMapper permette di valutare la pericolosità delle vulnerabilità scoperte mediante un'analisi qualitativa, espressa da un punteggio numerico sulla base di standard, come il CVSS, acronimo per Common Vulnerability Scoring System. Con quest'ultima funzionalità è possibile individuare le minacce che rappresentano un rischio maggiore di compromissione del sistema, di conseguenza correggerle con maggiore priorità.

ThreatMapper supporta diverse piattaforme di produzione contemporaneamente, come per esempio Kubernetes e Docker, in modo tale da permettere all'amministratore dell'infrastruttura software di monitorare, in ogni sua singola parte, una più ampia superficie d'attacco.

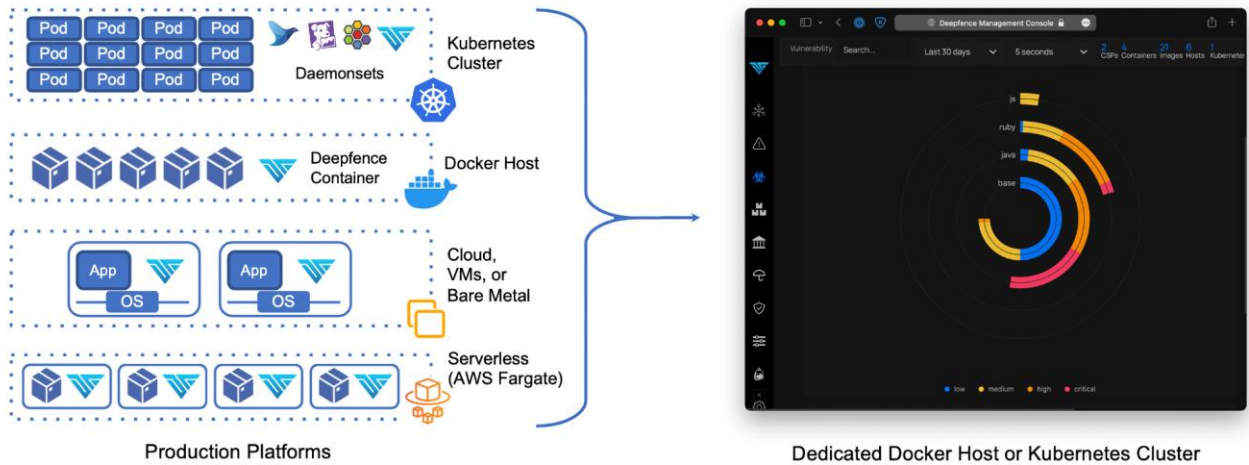


**Figura 1:** logo di Deepfence

## 1.1 Architettura

La struttura della piattaforma ThreatMapper è composta da due elementi fondamentali: la Management Console e un insieme di sensori. La console di gestione ricostruisce in maniera automatica la topologia dell'intera applicazione attraverso gli SBOMs, acronimo dall'inglese di distinta dei materiali software. Lo SBOM è un documento specifico dell'applicazione, in cui è presente una lista di tutte le componenti software open-source e di terze parti, le quali formano l'applicativo. Scansiona l'infrastruttura software alla ricerca delle vulnerabilità e crea graficamente la cosiddetta "Threat Map". I sensori ThreatMapper

sono implementati all'interno delle piattaforme di produzione compatibili con lo scopo di inoltrare, attraverso canali sicuri di comunicazione, le SBOMs e i dati di telemetria alla Management Console [2].



**Figura 2:** Struttura logica dell'architettura di ThreatMapper

## 1.2 Management Console

La Management Console può essere considerata il cuore di ThreatMapper, è un programma autonomo e indipendente realizzato come un insieme di container:

- Deepfence\_agent\_build\_ce
- Deepfence\_agent\_ce
- Deepfence\_api\_ce
- Deepfence\_diagnosis\_ce
- Deepfence\_discovery\_ce
- Deepfence\_elastic\_cek
- Deepfence\_fetcher\_ce
- Deepfence\_init\_ce
- Deepfence\_postgres\_ce
- Deepfence\_redis\_ce
- Deepfence\_router\_ce

- Deepfence\_ui\_ce
- Deepfence\_vulnerability\_mapper\_ce

Ognuno di questi container racchiude un servizio offerto dalla piattaforma ThreatMapper. Per la loro implementazione sono stati utilizzati linguaggi di programmazione diversi : Javascript, Go, Python, HTML e SCSS.

In base alle dimensioni dell'ambiente di sviluppo, è necessario installare e configurare la Management Console su un singolo Docker Host oppure su di un Cluster Kubernetes dedicato. L'amministratore può raggiungere la console attraverso un'interfaccia Web, la quale utilizza il protocollo per la comunicazione sicura HTTPS, acronimo di HyperText Transfer Protocol over Secure Socket Layer.

L'utente finale attraverso l'utilizzo dell'interfaccia grafica della console può usufruire di tutte le funzionalità offerte da ThreatMapper. È possibile gestire i permessi degli utenti in grado di accedere alla piattaforma. Si può visualizzare in tempo reale il grafico dell'infrastruttura di produzione, la quale può essere formata da Cluster Kubernetes, container e immagini, macchine virtuali, processi in esecuzione e connessioni di rete qualsiasi; Individuare le eventuali vulnerabilità presenti negli elementi che compongono l'ambiente di sviluppo software e visionare i risultati di diversi tipi di scansioni; analizzare i registri dei container per ripartire equamente il carico di lavoro sui nodi; ispezionare gli host dell'infrastruttura, prima che questi vengano aggiunti ad un cluster Kubernetes e scansionare container e filesystem host alla ricerca di segreti non protetti come password, chiavi e token di accesso.

### 1.3 Sensors

I ThreatMapper sensors vengono distribuiti all'interno delle piattaforme di produzione. In seguito ad una configurazione basata su di un API key univoca e l'indirizzo IP della Management Console, scambiano informazioni telemetriche attraverso un protocollo Transport Layer Security (TLS) con la suddetta Management Console, la quale invia istruzioni sul come ricostruire l'insieme di componenti software ed eseguire le scansioni.



I sensori ThreatMapper supportano i due più diffusi strumenti per la containerizzazione: Docker e Kubernetes.

Su Kubernetes, il sensore ThreatMapper è installato tramite Helm, una collezione di strumenti utili alla gestione di pacchetti software. È distribuito come daemonset per garantire che tutti i nodi Kubernetes possiedano una copia del pod, il quale consente di eseguire il processo del sensore in background. I pod presenti in ogni nodo utilizzano un accesso privilegiato alle API standard per tracciare il traffico di rete e interrogare i container in esecuzione [3].

```
helm repo add deepfence https://deepfence-helm-charts.s3.amazonaws.com/threatmapper
helm show readme deepfence/deepfence-agent
helm show values deepfence/deepfence-agent

helm install deepfence-agent deepfence/deepfence-agent \
  --set managementConsoleUrl=x.x.x.x \
  --set deepfenceKey=C8TtyEtNB0gBo1wGhpeAZICNSAaGkw71BSdS2kLELY0
```

**Figura 3:** Comandi da terminale Linux per la distribuzione del sensore su Kubernetes

In Docker, il sensore ThreatMapper è distribuito come container privilegiato su ogni docker host e viene inizializzato attraverso il comando presente nella **figura 4**.

```
docker run -dit --cpus=".2" --name=deepfence-agent --restart on-failure --pid=host --net=host \
--privileged=true -v /sys/kernel/debug:/sys/kernel/debug:rw -v /var/log/fenced \
-v /var/run/docker.sock:/var/run/docker.sock -v /:/fenced/mnt/host/:ro \
-e USER_DEFINED_TAGS="" -e MGMT_CONSOLE_URL="---CONSOLE-IP---" -e MGMT_CONSOLE_PORT="443" \
-e DEEPFENCE_KEY="---DEEPFENCE-API-KEY---" \
deepfenceio/deepfence_agent_ce:latest
```

**Figura 4:** Comandi da terminale Linux per la distribuzione del sensore su Docker

I sensori ThreatMapper possono essere distribuiti anche su diverse piattaforme basate su Kubernetes e Docker:

- Amazon ECS
- AWS Fargate
- Google Kubernetes Engine
- Servizio Azure Kubernetes
- Macchine bare-metal o virtuali



**Figura 5a:** *Logo di Amazon ECS*



**Figura 5b:** *Logo di AWS Fargate*



**Azure Kubernetes Service (AKS)**

**Figura 5c:** *Logo di Azure Kubernetes service*



**Google Kubernetes Engine**

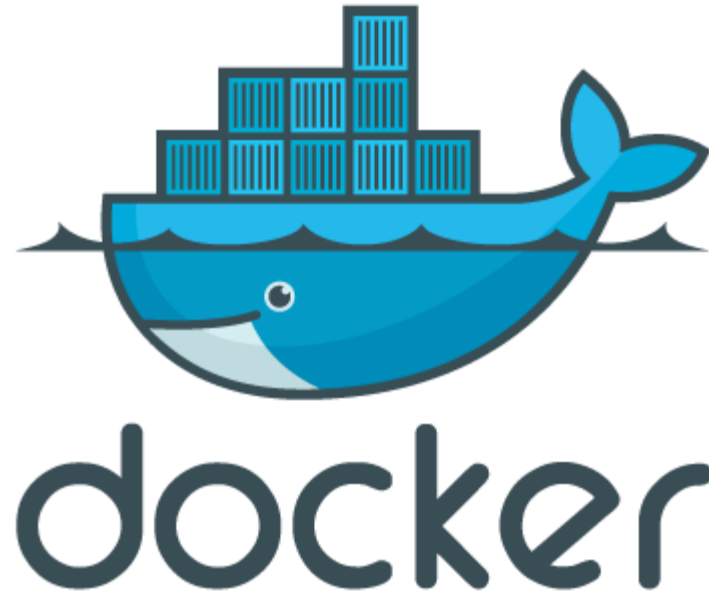
**Figura 5d:** *Logo di Google Kubernetes Engine*

## 1.4 Docker

Docker è un software libero, scritto in linguaggio di programmazione Go, rilasciato da Solomon Hykes e un gruppo di ingegneri nel 2013, il cui codice sorgente è stato reso open-source nel marzo dello stesso anno.

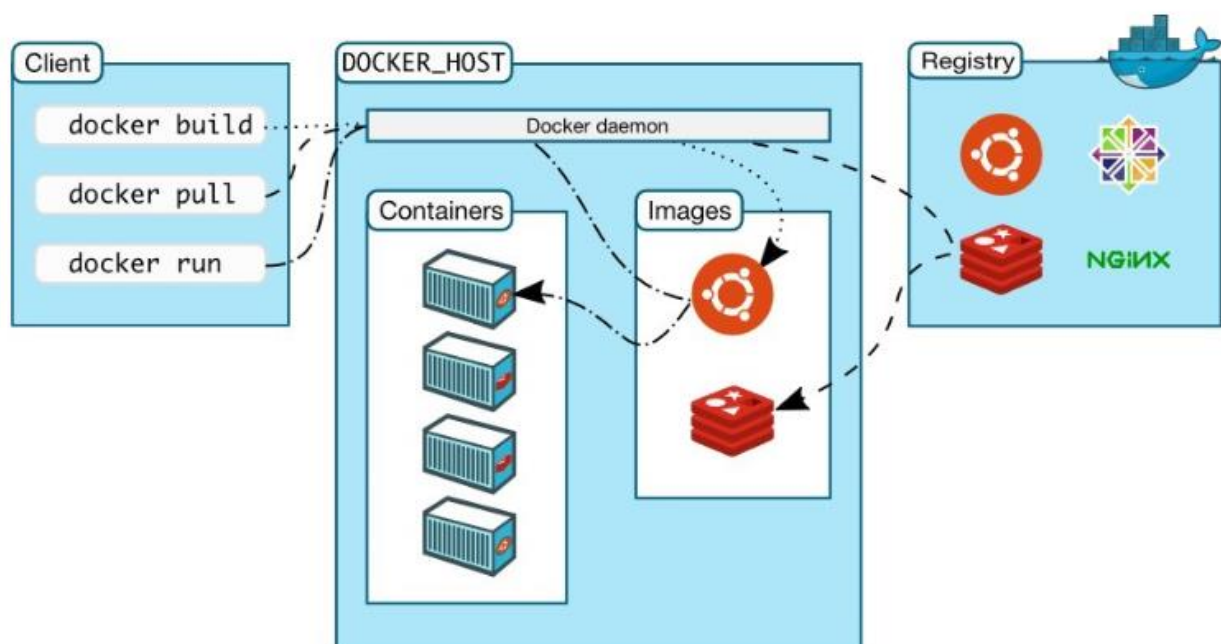
La piattaforma Docker è basata sull'utilizzo di container Linux, i quali permettono di rendere più facile e dinamico lo sviluppo, la distribuzione e l'esecuzione di applicativi. I container permettono di creare un ambiente d'esecuzione al livello applicativo isolato dall'esterno e all'interno sono presenti tutte le componenti e risorse necessarie al funzionamento di processi e applicazioni. A differenza delle macchine virtuali, che creano una copia separata di un intero sistema operativo virtuale, i container utilizzano le funzionalità d'isolamento offerte dal sistema operativo ospitante. Pertanto, l'immagine del container si rivela più leggera e indipendente dall'hardware in cui viene eseguito.

Nel caso di container Linux, Docker si serve degli strumenti di virtualizzazione delle risorse del kernel Linux, come ad esempio namespace e cgroup. [4]



**Figura 6:** *Logo di Docker*

L'architettura di Docker è strutturata con un'organizzazione di tipo Client-Server. Il cuore della piattaforma è il server Docker Daemon, che tramite delle API REST, acronimo di Representational State Transfer, comunica con il client Docker o altri Docker Daemon remoti. Il Daemon si occupa sotto richiesta delle API docker di costruire, distribuire ed eseguire gli oggetti docker come immagini, volumi e container. Il client Docker è un'interfaccia dotata di comandi utili per comunicare con uno o più Docker Daemon. Il client e il server Docker possono essere eseguiti sullo stesso sistema oppure su host differenti. Infine, per poter condividere e memorizzare agevolmente le immagini Docker si fa uso di registri Docker, i quali possono essere pubblici, per esempio il famoso DockerHub, o privati su cui è possibile effettuare operazioni di push o pull delle immagini [5].



**Figura 7:** *Architettura di Docker*

## 1.5 Kubernetes

Kubernetes, indicato spesso con l'abbreviazione K8s, è un software open-source scritto in linguaggio Go, ideato da Google e attualmente mantenuto da Cloud Native Computing Foundation. Ha lo scopo di automatizzare le operazioni manuali coinvolte nella orchestrazione e gestione dei container nei processi di produzione e distribuzione software.



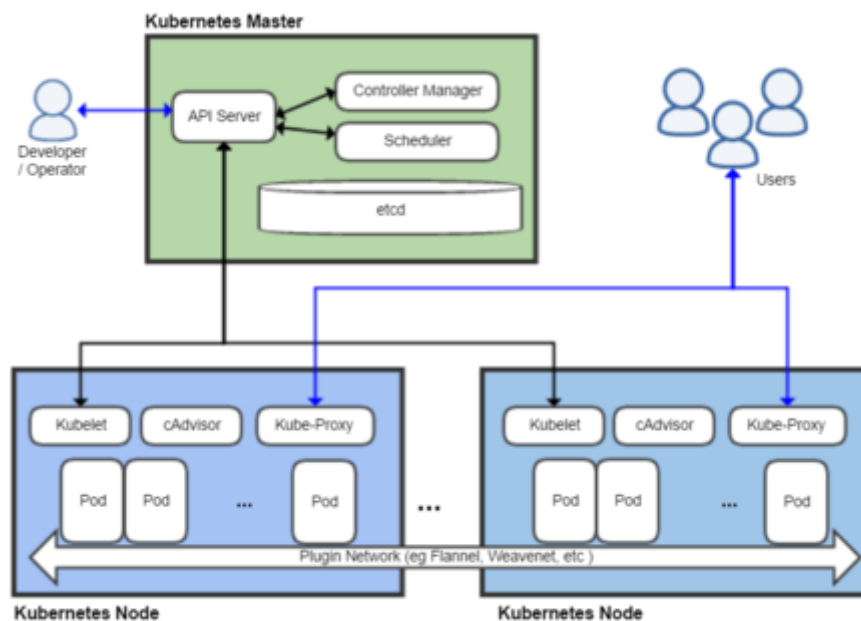
**Figura 8:** *Logo di Kubernetes*

La struttura di Kubernetes si basa su diversi livelli di raggruppamento utilizzati per gestire gli host e i container in essi distribuiti. Il più alto livello di raggruppamento è il cosiddetto cluster, formato a sua volta da nodi, ognuno dei quali rappresenta un singolo host (macchina virtuale o fisica). I nodi si distinguono in master e worker, seguendo un pattern di tipo

orchestrator all'interno del cluster. Il master o Control Plane Node è responsabile di orchestrare i nodi worker. Il nodo worker esegue il carico di lavoro assegnato dal master grazie all'utilizzo di container, gestiti ed eseguiti da uno strumento come ad esempio Docker.

Siccome tutti i nodi worker del cluster per coordinarsi nell'esecuzione dei container fanno riferimento al nodo master, è pratica comune replicarlo in più server così da assicurare la continuità del servizio nel caso in cui uno di questi si guastasse. Inoltre, Kubernetes fornisce un livello di raggruppamento logico, chiamato pod, il quale rappresenta l'unità d'esecuzione più piccola di Kubernetes. I pods sono aggregati di uno o più container localizzati in uno o più nodi. I container all'interno di un pod condividono le stesse risorse e dipendenze; ad esempio, hanno lo stesso indirizzo IP in modo tale da poter comunicare tra loro in localhost.

Kubernetes attraverso questa organizzazione fornisce un framework capace di migliorare le prestazioni complessive dell'infrastruttura software. Tra le tante funzionalità offerte da questa piattaforma troviamo il bilanciamento del traffico. Nel caso in cui un container esposto in rete fosse soggetto ad un elevato carico di lavoro, Kubernetes lo distribuisce su più container in modo tale da non causare problemi di servizio. [6/7]



**Figura 9:** Architettura di un cluster Kubernetes

## Capitolo 2: Funzionalità di ThreatMapper

---

Nel primo capitolo abbiamo approfondito l'architettura alla base del funzionamento di ThreatMapper e i due più diffusi strumenti di containerizzazione, Docker per la gestione di un singolo host e Kubernetes per il coordinamento di uno o più host Docker contenuti in un cluster.

In questo capitolo approfondiamo alcune delle problematiche della sicurezza informatica dei suddetti ambienti di produzione, focalizzando con maggiore attenzione sulla possibile presenza di vulnerabilità all'interno dell'infrastruttura software. In seguito, analizzeremo le soluzioni a queste problematiche proposte dalla piattaforma di monitoraggio ThreatMapper, le quali si compongono di tre principali azioni: mappatura della superficie d'attacco, individuazione delle vulnerabilità e classificazione di quest'ultime.

### 2.1 Mappatura della superficie di attacco

Il monitoraggio e la salvaguardia della sicurezza informatica di un'infrastruttura di produzione parte necessariamente dall'individuazione e mappatura degli elementi di rischio nella cosiddetta superficie d'attacco. Bisogna avere una visione completa della topologia esposta in rete prima di attuare una qualsiasi misura di sicurezza. La superficie d'attacco di un sistema è la somma di tutte le risorse IT esposte in una rete, i punti di ingresso e vulnerabilità, che potrebbero permettere ad utenti non autorizzati di accedere all'intera infrastruttura software. Di conseguenza, gli elementi a rischio della superficie d'attacco

sono tutti gli host presenti in rete con segreti sensibili esposti, i container in essi distribuiti con vulnerabilità note, i server esposti a possibili attacchi DDoS, le istanze cloud di terze parti, certificati digitali scaduti, porte aperte di servizi mal configurati e i protocolli di comunicazione utilizzati dalle parti dell'infrastruttura per interagire tra loro non aggiornati all'ultima versione disponibile. La diminuzione della superficie di attacco è diventata una priorità per molte organizzazioni.

ThreatMapper scopre automaticamente la topologia dell'infrastruttura software e la sua superficie di attacco grazie alla distribuzione di sensori nelle diverse piattaforme di produzione. I sensori sono leggeri, facili da configurare e non invasivi. Essi individuano e interrogano le istanze cloud, gli applicativi serverless, i Kubernetes cluster e i container in essi contenuti. Inoltre, identificano i flussi di comunicazione tra ciascun elemento mappando in tempo reale la rete. I sensori ThreatMapper comunicano i dati telemetrici e le istruzioni per ricostruire la SBOM, acquisiti dalle piattaforme di produzione, alla Management Console.

ThreatMapper genera gli SBOMs principalmente in due modi. Il primo fa parte del monitoraggio delle immagini dei container nel CI/CD, una metodologia usata per combinare integrazione continua e consegna continua atti alla distribuzione frequente delle applicazioni. Invece, il secondo in tempo reale genera gli SBOMs dei containers e degli host.

ThreatMapper, dalla versione 1.3.0, offre come funzionalità la creazione in tempo reale degli SBOMs. Uno SBOM in tempo reale è una lista delle componenti e dipendenze del software, utili a monitorare nuove attività, librerie e processi che si verificano all'interno dell'ambiente di produzione, segnalando eventuali comportamenti anomali causati da possibili attacchi informatici. Si può implementare questa funzionalità aggiungendo un contesto in tempo reale ai dati del codice, della compilazione e del cloud, unito ad un monitoraggio costante del traffico di rete. Ciò permette di avere un'immagine olistica degli ambienti di produzione. Sandeep Lahane, fondatore e CEO della compagnia Deepfence, afferma che: "SBOMs have emerged as an essential part of the documentation that supports each software product release or update, which means the added runtime SBOM

capabilities within ThreatMapper are valuable improvements in order to quickly identify potentially vulnerable instances if a security issue is disclosed" [8].

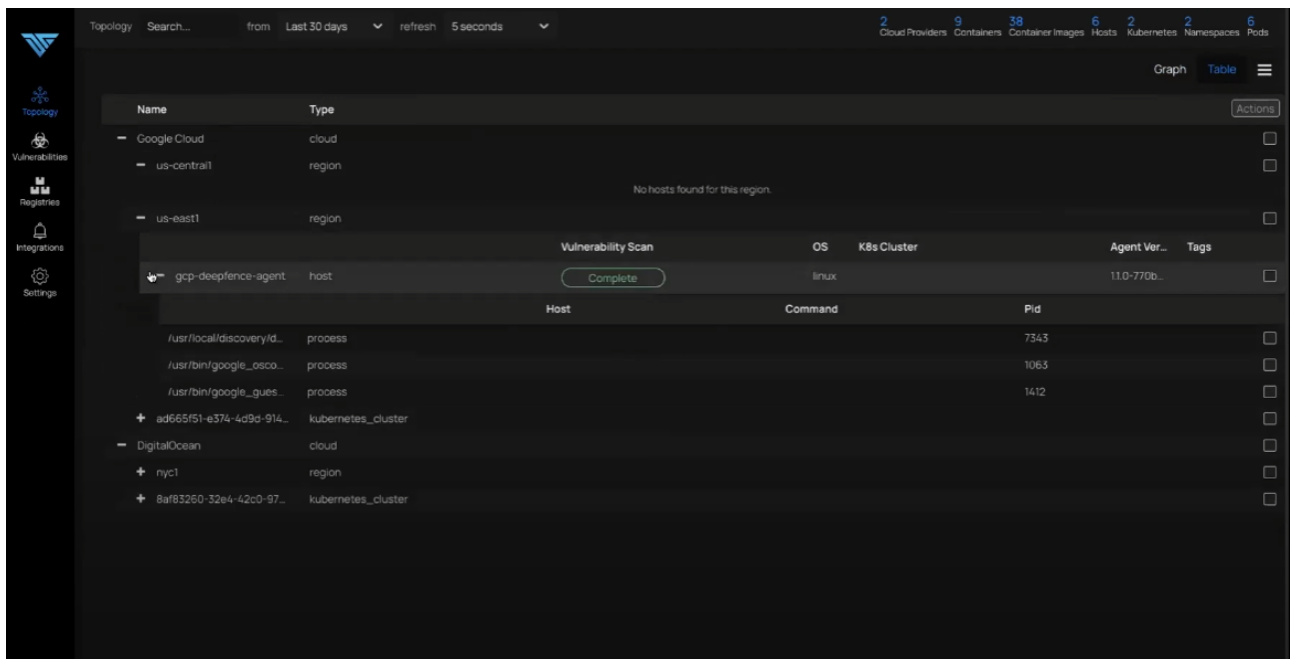


**Figura 10:** Rappresentazione grafica della topologia

La Management Console nella sezione “Topology”, grazie ai dati inviati dai sensori, crea due diverse rappresentazioni dell’ambiente di produzione: grafica e listata. La rappresentazione grafica espone la topologia sottoforma di icone circolari racchiuse tra loro gerarchicamente. Cliccando ognuna di queste icone etichettate, partendo dalle istanze cloud, è possibile visualizzare il sottoinsieme sottostante fino ad arrivare all’atomo dell’infrastruttura, il container. Questo tipo di rappresentazione è in grado di ricreare un’ambiente composto da 25000 parti distinte, distribuite in 12 cluster Kubernetes diversi. La rappresentazione listata è sempre distribuita gerarchicamente e permette di esplorare l’intera infrastruttura di produzione attraverso i soli nomi di ognuna delle parti. ThreatMapper riassume la quantità degli elementi attivi della topologia, specificando il numero esatto di ognuno di essi in alto a destra nella schermata esposta all’utente. Inoltre, dà la possibilità all’amministratore di sistema di personalizzare i tempi di aggiornamento



dei dati raccolti dalla piattaforma di monitoraggio, in base alle dinamiche dell'infrastruttura software monitorata.



**Figura 11:** Rappresentazione listata della topologia

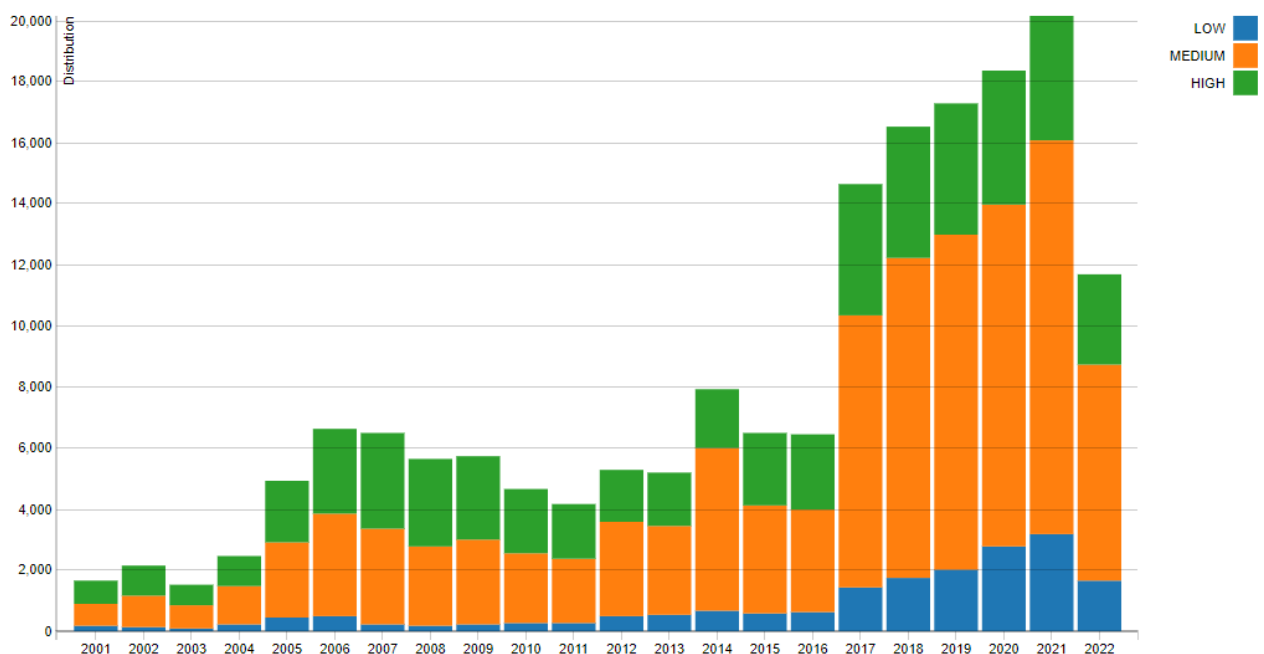
## 2.2 Individuazione delle vulnerabilità

Una vulnerabilità informatica è un difetto o un punto debole nella progettazione, implementazione o funzionamento di un sistema informatico, la quale può portare a un evento inaspettato e indesiderato compromettendo la sicurezza di una risorsa o di un gruppo di risorse. In questo modo viene data la possibilità ad un eventuale fonte di minaccia di ottenere un accesso non autorizzato al sistema informatico. Quindi, la presenza di una vulnerabilità espone un sistema agli attacchi informatici.

Le vulnerabilità per la loro generica definizione possono essere localizzate a qualsiasi livello di un'infrastruttura software; tuttavia, si possono individuare due principali categorie: vulnerabilità software e vulnerabilità dei protocolli. Le vulnerabilità software, detti anche bug, sono mal funzionamenti causati da difetti di progettazione, scrittura del codice sorgente, distribuzione e configurazione del software. Invece, le vulnerabilità dei protocolli si presentano quando i protocolli di comunicazione non sono forniti di protezione

idonee a mantenere sicure le interazioni fra le componenti presenti in un sistema informatico.

Ogni anno le vulnerabilità segnalate pubblicamente aumentano in numero, basti pensare che secondo il report del 2021 pubblicato dal National Institute of Standards and Technology degli Stati Uniti (NIST), una delle più affidabili fonti di informazioni per i responsabili della sicurezza informatica di tutto il mondo, ha messo in evidenza che per il quinto anno consecutivo è stato rotto il record di vulnerabilità di sicurezza scoperte (21957), con una media di circa 60 al giorno [9].



**Figura 12:** Grafico delle vulnerabilità scoperte suddivise per anno

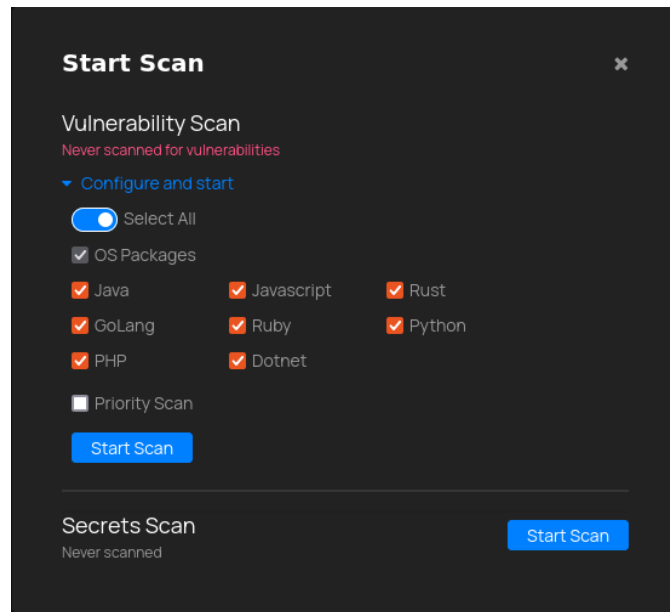
Una delle cause del continuo crescere delle vulnerabilità di anno in anno è l'utilizzo sempre più frequente di componenti open-source di terze parti all'interno dei prodotti software. La maggior parte dei moderni codebase, ovvero l'intera collezione di codice sorgente usata per costruire un applicativo o una parte di esso, sono basati in media dal 97% di codice proveniente da progetti open-source di terze parti. Perciò, gli sviluppatori di una azienda quando integrano un nuovo componente nell'infrastruttura software hanno testato e controllato solo il 3% di quest'ultima, prendendosi però la responsabilità per la sicurezza e l'integrità dell'intero componente. Dai dati riportati dal rapporto dell'OSSRA del 2022 è

stato dimostrato che 81% di questi applicativi contiene almeno una vulnerabilità open-source [10].

Questo forte aumento di vulnerabilità su scala mondiale ha portato numerose aziende a maturare un sempre più consapevole bisogno di migliorare le tecniche di prevenzione, necessarie ad un aumento della sicurezza dei prodotti software. Un chiaro esempio è l'approccio "Shift-Left" adottato all'interno di organizzazioni basate sulla metodologia di sviluppo software DevOps, in cui è necessario sviluppare e distribuire il più rapidamente possibile il prodotto software. Uno studio condotto da IBM, infatti, ha dimostrato che spostare verso il gruppo di sviluppo (sinistra) il controllo delle vulnerabilità, invece di rimandarla alla fase finale del suo ciclo di vita (destra), porta ad una diminuzione sia dei costi sia dei tempi necessari all'eliminare delle vulnerabilità stesse [11]. Questo però non basta a mitigare ogni pericolo poiché non tutti i difetti possono essere risolti prima della produzione come, ad esempio, vulnerabilità causate da tecnologie di rete, ingress controller o plug-in di terze parti. Bisogna anche pensare alla manutenzione degli applicativi, ad ogni aggiornamento potrebbe essere introdotta involontariamente una vulnerabilità non presente nella versione precedente.

ThreatMapper migliora e automatizza le iniziative Shift-Left di un'organizzazione ed esegue scansioni dell'infrastruttura software in ambiente di produzione, individuando tutte le vulnerabilità presenti sia in prodotti proprietari dell'azienda, sia in componenti open-source forniti da terze parti.

La Management Console nella sua inizializzazione fa il bootstrap dei feed di vulnerabilità note da più di cinquanta fonti diverse. La più ricca delle banche dati a cui fa riferimento ThreatMapper è il National Vulnerability Database (NVD), l'archivio del governo degli Stati Uniti che racchiudere un vasto elenco CVE. Un elenco CVE, acronimo di Common Vulnerabilities and Exposures, è una lista di vulnerabilità nella sicurezza informatica distinte da un numero identificativo CVE. Inoltre, ThreatMapper preleva anche feed specifici di linguaggi di programmazione come Java, Ruby, NodeJS, Python e altri.



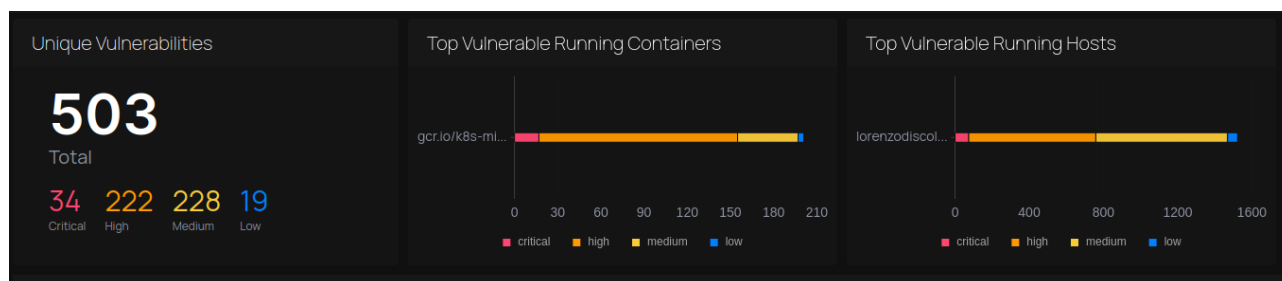
**Figura 13:** Schermata di inizio scansione

A questo punto la Management Console è in grado di effettuare due diverse scansioni del sistema. La prima esegue una scansione dell'intera catena di fornitura software formata da ambienti di istanze cloud, cluster Kubernetes, host, container e API di terze parti, alla ricerca di dipendenze vulnerabili note. In questo tipo scansione è possibile specificare i tipi di linguaggi di programmazione su cui fare particolare attenzione nella ricerca di vulnerabilità specifiche all'interno del codice.

La seconda scansione, chiamata SecretScanner, utilizza le informazioni acquisite e rese disponibili dal progetto open-source "SecretScanner" di Deepfence, unite ad un'analisi accurata del traffico di rete. Confronta con una serie di raccolte di minacce conosciute, aggiornate continuamente da feed, come ad esempio "Emerging Threats", sviluppato e offerto da Proofpoint. Questa scansione permette all'amministratore di sistema di cercare e scovare segreti aziendali sensibili, lasciati privi di protezione nei carichi di lavoro e nelle immagini dei contenitori nei registri. Merce inestimabile per i malintenzionati, i quali grazie a queste informazioni potrebbero ottenere chiavi aziendali, capaci di sbloccare l'accesso alle banche dati dell'organizzazione o a parti critiche dell'infrastruttura.

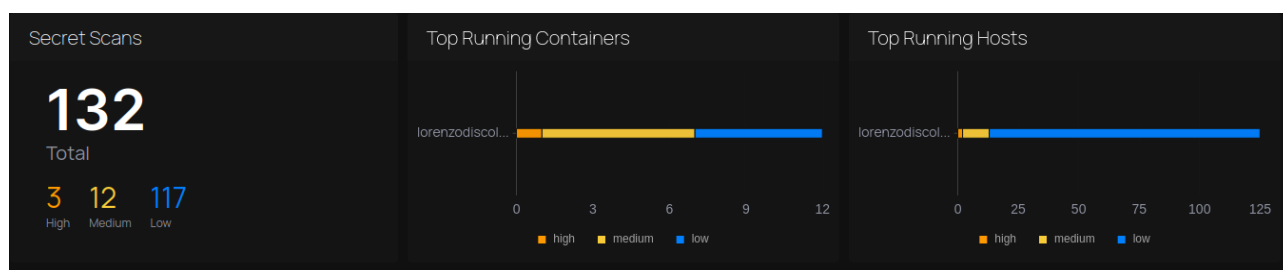
Tutte vulnerabilità individuate in seguito alla prima scansione sono raggruppate ed elencate all'interno della sezione "Vulnerabilities" dell'interfaccia Web, esposta dalla Management Console. In questa sezione sono ulteriormente divise in due sottocategorie: le vulnerabilità

in esecuzione negli host e le vulnerabilità in esecuzione nei container. Dove è possibile distingue in base al nome, ogni singolo host e container presente all'interno del sistema preso in esame.



**Figura 14:** Visualizzazione delle vulnerabilità rilevate dopo scansione

Nella sezione “Secrets” sono disponibili i risultati elaborati dal secondo tipo di scansione, organizzati in modo analogo rispetto ai dati ottenuti dalla prima scansione.



**Figura 15:** Visualizzazione dei segreti rilevati dopo scansione

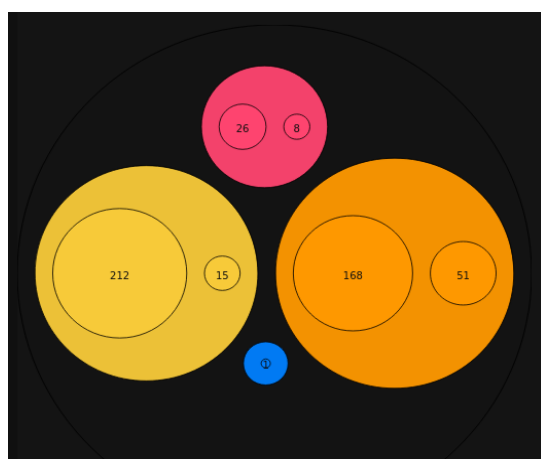
Infine, ThreatMapper mette a disposizione un ulteriore strumento di scansione capace di analizzare manualmente o a cadenza periodica, con procedure automatizzate, i registri delle immagini caricate dagli host nelle più famose librerie di container pubbliche. Possiamo trovare la disponibilità delle seguenti repository: ECR, Azure, GCR, DockerHub, Docker (self hosted), Quay, Harbor, Gitlab e Jfrog.

## 2.3 Classificazione delle vulnerabilità

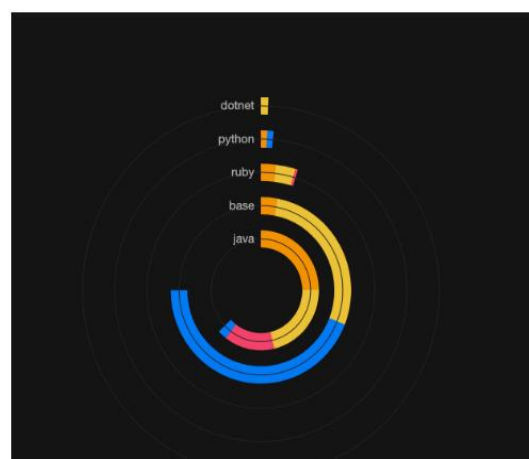
L'attività di identificazione delle vulnerabilità informatiche nella sicurezza dell'infrastruttura software può portare alla scoperta di centinaia se non di migliaia di risultati. Quindi è buona norma accompagnarla da una classificazione, capace di distinguere

le vulnerabilità in base alla loro potenzialità di danno. I proprietari e gli amministratori di un sistema spesso si trovano ad affrontare la decisione su quale vulnerabilità, tra quelle rilevate, bisogna concentrare le risorse aziendali ai fini di risolverla. Per far fronte a questa scelta è necessario una corretta analisi dei risultati ottenuti dalla scansione, in modo tale da scegliere una coerente strategia di gestione del rischio, rispetto alle necessità e la politica sulla sicurezza dell'organizzazione stessa.

ThreatMapper offre un potente strumento di classificazione delle vulnerabilità in grado di rispondere a questa esigenza aziendale. ThreatMapper, in seguito all'individuazione delle vulnerabilità, espone i risultati con l'ausilio di più rappresentazioni grafiche così da rendere più agevole l'analisi all'utente finale. Nelle diverse rappresentazioni, i dati delle vulnerabilità vengono suddivisi in differenti categorie attraverso l'utilizzo di quattro diverse colorazioni, le quali segnalano la pericolosità intrinseca di quest'ultime. Il rosso avvisa la presenza di una vulnerabilità critica indicando una priorità elevata per la sua risoluzione a causa della sua pericolosità, l'arancione annuncia una vulnerabilità con pericolosità alta, il giallo è un livello intermedio e infine il blu rappresenta un rischio basso per il sistema software.



**Figura 15a:** *Classificazione delle vulnerabilità in base alla gravità*



**Figura 15b:** *Suddivisione delle vulnerabilità in base al linguaggio e alla gravità*

La classificazione delle vulnerabilità fatta da ThreatMapper si configura dall'utilizzo dello standard CVSS e altri punteggi di gravità, con l'aggiunta delle informazioni relative al metodo di sfruttamento e alla loro vicinanza alla superficie d'attacco.

Il Common Vulnerability Scoring System (CVSS) è un framework aperto sviluppato dalla FIRST, un'organizzazione statunitense senza scopo di lucro, nata con l'obiettivo di rendere disponibile globalmente un metodo standardizzato e gratuito per segnalare le caratteristiche e la pericolosità delle vulnerabilità di sicurezza software. CVSS utilizza tre gruppi di metriche: Base, Temporale e Ambientale.

Le metriche di base indicano le caratteristiche intrinseche di una vulnerabilità, indipendenti dal tempo e dall'ambiente. Il punteggio di base è un valore compreso tra zero e dieci. Tale punteggio è influenzato da due gruppi di metriche: sfruttabilità e impatto. La sfruttabilità include tutti gli aspetti che denotano la complessità nell'attaccare il singolo componente vulnerabile. Invece, le metriche d'impatto determinano le conseguenze causate dallo sfruttamento di tale vulnerabilità da parte di un aggressore. Per la loro natura quasi immutabile, sono le uniche informazioni sulla sicurezza fornite dalla società produttrice del software vulnerabile.

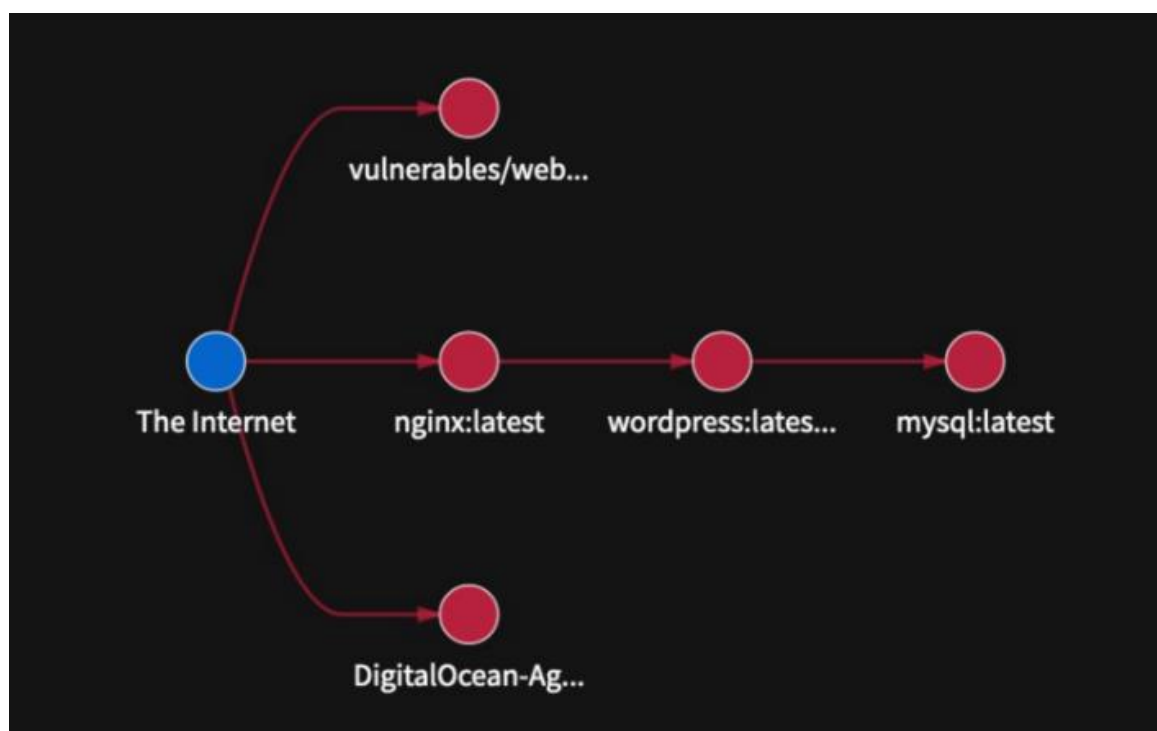
Le metriche temporali indicano tutte le caratteristiche che variano nel tempo di una vulnerabilità, come la disponibilità delle tecniche utili a correggerla e la facilità di trovare il codice necessario a sfruttarla, al momento della valutazione di questo punteggio.

Infine, le metriche ambientali servono a poter dare un contesto alla componente vulnerabile all'interno dell'infrastruttura software, in modo tale da permettere alle aziende di generare un punteggio complessivo più accurato [12/13].

Il risultato dell'analisi CVSS è quantizzato da un punteggio numerico indicativo, generato dalle metriche di base e modificato dalle altre due metriche. Grazie a questo punteggio è possibile definire la pericolosità di una vulnerabilità confrontandolo con i seguenti intervalli di valori:



ThreatMapper migliora il calcolo del punteggio CVSS attribuendo maggiore priorità alle vulnerabilità individuate sulle parti dell'ambiente di produzione interfacciate ad una o più reti connesse ad internet. Rende disponibile la visualizzazione di un “percorso d'attacco”, dove illustra il possibile percorso che un criminale informatico, attraverso la rete internet, potrebbe seguire per individuare e sfruttare una o più vulnerabilità. Tale strumento rintraccia le debolezze nascoste del sistema, ad esempio dietro proxy, le quali ricevono indirettamente traffico di rete potenzialmente dannoso. In questo modo aiuta il gruppo di sicurezza di un'organizzazione a installare un Web Application Firewall per filtrare e monitorare il traffico di rete così da mitigare temporaneamente il problema, in attesa dello sviluppo dell'aggiornamento necessario ad eliminare la vulnerabilità [14].



**Figura 16:** Visualizzazione del percorso di attacco



## Capitolo 3: Utilizzo e analisi di ThreatMapper

---

Nel seguente capitolo esaminiamo un esempio di monitoraggio, ad opera della piattaforma ThreatMapper, di un'infrastruttura di produzione costruita in ambiente locale. Nella prima parte sono descritte le componenti e i protocolli di comunicazione utilizzati all'interno dell'infrastruttura considerata. Inoltre, sono elencati i passaggi da osservare per una corretta configurazione di ThreatMapper. Nella seconda parte osserviamo l'effettivo funzionamento di ThreatMapper, analizzando i risultati prodotti dall'utilizzo dei diversi strumenti offerti dalla piattaforma di monitoraggio.

### 3.1 Setup dell'infrastruttura software test

L'infrastruttura software presa come esempio è costruita da tre Docker host. Ognuno dei quali è un'istanza di una macchina virtuale creata grazie all'ausilio del software, gratuito e open-source, Oracle VM VirtualBox. Infatti, VirtualBox attraverso un'operazione di virtualizzazione crea uno o più ambienti isolati dal calcolatore ospitante(host), dette macchine virtuali, nei quali è possibile configurare e utilizzare un sistema operativo "ospite". Le macchine virtuali emulano il comportamento di un sistema fisico, il loro avvio e spegnimento è regolato in maniera indipendente. Inoltre, l'utente ha la possibilità di salvare in un file lo stato del sistema virtuale al momento dello spegnimento, in modo tale da preservare l'integrità dei dati e dei processi in esecuzione. Ad ognuna delle macchine virtuali vengono riservate delle risorse della macchina ospitante, la quale non ha avrà più accesso a quest'ultime. È possibile scegliere il numero di core del processore da riservare ad ogni macchina virtuale, la quantità di memoria RAM e quali file condividere tra sistema operativo host e quello ospite [15].

Le tre macchine virtuali utilizzate in quest' esempio sono inizializzate con l'immagine del sistema operativo Ubuntu versione 20.04.4 LTS, acronimo di long-term support, versione scelta per la sua stabilità e affidabilità. Inoltre, in ognuna di esse è stata installata la versione 20.10.17 del software Docker. Le suddette macchine virtuali hanno le seguenti caratteristiche:

- Docker-01 con 8 core CPU, 8192 MB di RAM e 160 GB di spazio su disco.
- Docker-02 con 2 core CPU, 2048 MB di RAM e 40 GB di spazio su disco.
- Docker-03 con 2 core CPU, 2048 MB di RAM e 40 GB di spazio su disco.

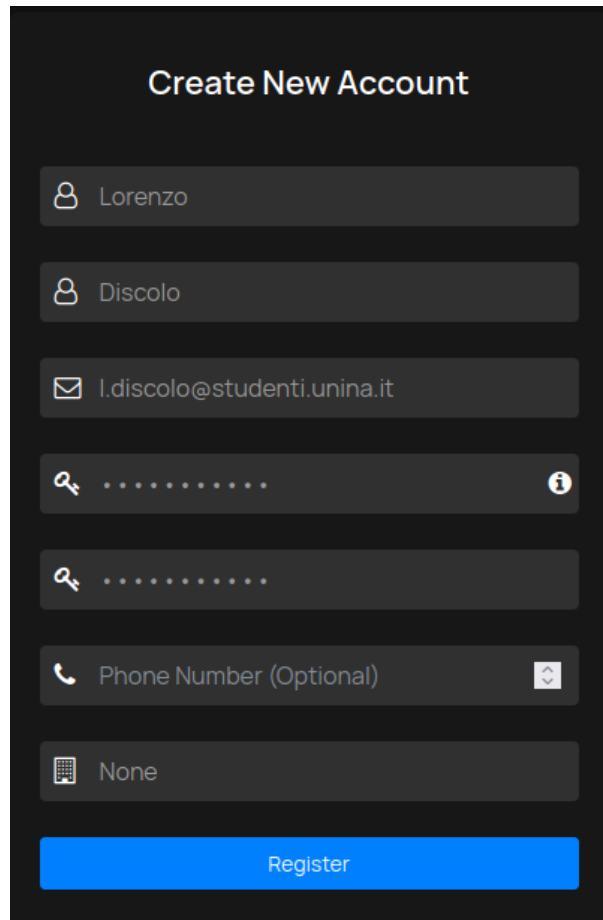
VirtualBox tra le diverse modalità di gestione della rete all'interno degli ambienti virtualizzati, offre la possibilità di creare una rete NAT personalizzata. L'insieme delle macchine virtuali all'interno dell'infrastruttura di produzione presa in esempio, sono collegate tra di loro mediante una rete NAT chiamata NatNetwork. Questa rete funziona in modo simile a un router domestico, assegna ad ogni macchina virtuale un IP diverso, scelto all'interno di un range definito di indirizzi privati, creando così una rete locale. Questa rete impedisce ai sistemi esterni di accedere ai dispositivi presenti in questa rete ma lascia la possibilità a quest'ultimi di comunicare tra loro.

Per il corretto funzionamento di ThreatMapper è necessario installare la Management Console su di un Docker host e distribuire i sensori sui rimanenti host della rete.

La Management Console è installata sulla macchina virtuale chiamata Docker-1, alla quale sono state riservate più risorse rispetto agli altri due host. Dopo aver eseguito i comandi presenti nella documentazione di ThreatMapper per l'installazione su singolo Docker Host [16], è possibile accedere all'interfaccia Web della piattaforma di monitoraggio andando a digitare nella barra di ricerca di un qualsiasi browser, in questo caso è stato utilizzato Mozilla Firefox, l'indirizzo IP esposto nella rete NAT dell'host Docker-1. È possibile ricavare questo identificativo andando a digitare sul terminale il seguente comando:

```
docker-1@docker1-VirtualBox:~$ hostname -I  
10.0.2.15 172.17.0.1 172.18.0.1 192.168.49.1
```

A questo punto è necessario creare un nuovo account. Le informazioni richieste sono indicate nella **figura 17**.

The image shows a mobile application interface for creating a new account. The title 'Create New Account' is at the top. Below it are several input fields: a name field with 'Lorenzo', a company field with 'Discolo', an email field with 'l.discolo@studenti.unina.it', two password fields (both masked with dots), an optional phone number field, and a company dropdown menu currently set to 'None'. A blue 'Register' button is at the bottom.

**Figura 17:** Registrazione al servizio ThreatMapper

Nel campo “company” è stato posto il valore None per poter concretizzare la registrazione essendo un campo obbligatorio.

Nelle macchine virtuali di nome Docker-2 e Docker-3 sono distribuiti i sensori ThreatMapper necessari all’integrazione degli host all’interno dell’infrastruttura di produzione. I sensori vengono installati con l’utilizzo dei comandi riportati nella **figura 4** del *paragrafo 1.3* del primo capitolo. Tuttavia, nel comando da terminale Linux occorre specificare l’IP della Management Console, il quale corrisponde in questo caso a 10.0.2.15, e l’API key. Questa chiave è un codice alfanumerico univoco della console di gestione fondamentale per instaurare un canale di comunicazione con i sensori. L’API key è specificata nella sezione “User Management” delle impostazioni dell’interfaccia Web.

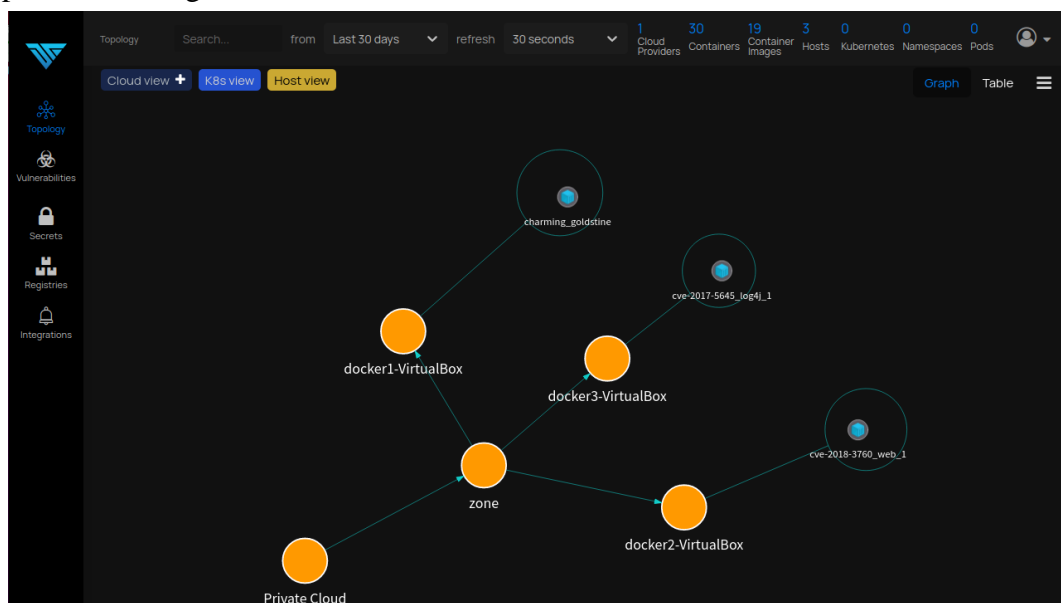
A questo punto la piattaforma ThreatMapper è completamente configurata e pronta all’utilizzo.

### 3.2 Test ThreatMapper

In questo test utilizziamo il registro VulHub allo scopo di verificare le prestazioni di ThreatMapper nell'individuazione e classificazione di specifiche vulnerabilità all'interno dell'infrastruttura di produzione. VulHub è un progetto nato ad opera di Owen Gong nel 2017 e ha come obiettivo quello di rendere disponibile, per motivi di ricerca, una collezione open-source di Docker container vulnerabili pronti ad essere eseguiti [17].

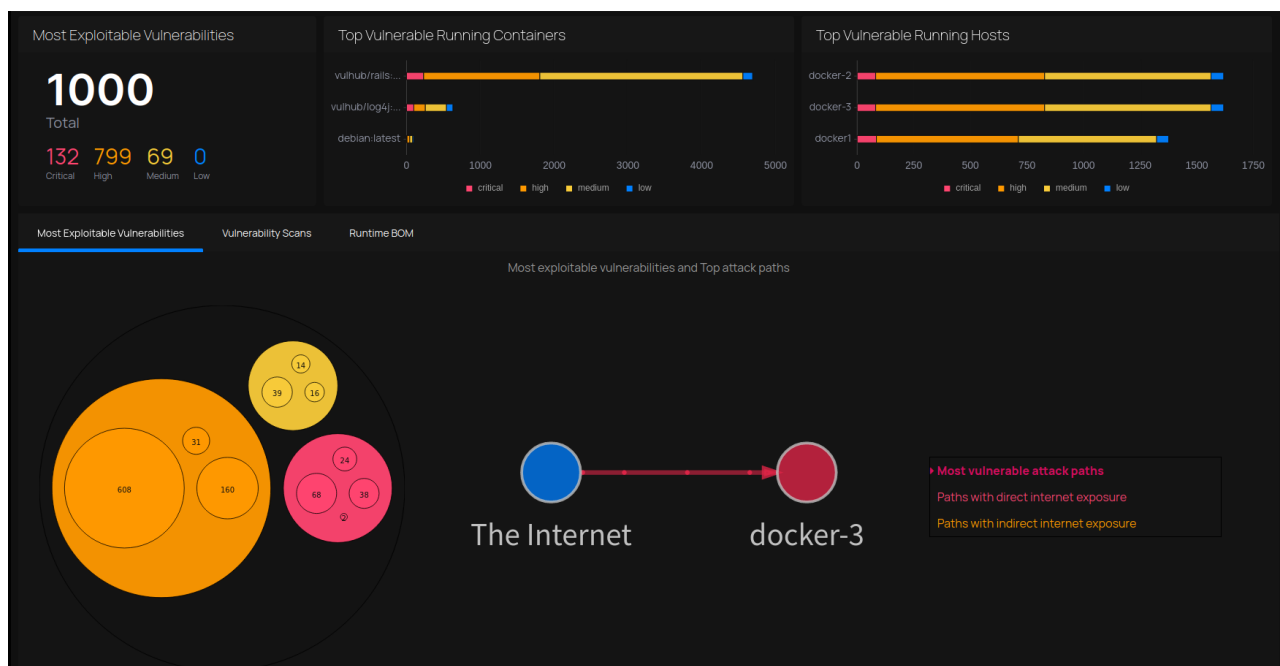
La raccolta messa a disposizione da VulHub è organizzata come un insieme di cartelle etichettate con il nome dell'applicativo esaminato. In ognuna di esse sono presenti delle sottocartelle denominate secondo l'identificativo CVE della vulnerabilità riscontrata in una delle versioni dell'applicativo preso in esame.

Per questo test distribuiamo due Docker container vulnerabili: **log4j** con la vulnerabilità **CVE-2017-5645** [18] e l'applicativo **Ruby on Rails** che utilizza la versione 3.7.2 della libreria del linguaggio di programmazione Ruby chiamata **Sprockets**, la quale presenta la vulnerabilità **CVE-2018-3760** [19]. Il primo container vulnerabile è stato scaricato ed eseguito sull'host Docker-2 e il secondo sulla macchina virtuale Docker-3. Nella sezione "Topology" dell'interfaccia Web, esposta dalla Management Console, possiamo visualizzare la topologia dell'infrastruttura di produzione ricostruita automaticamente da ThreatMapper, grazie ai dati inviati dai sensori. Nella **figura 18** è riportata la sua rappresentazione grafica.



**Figura 18:** Topologia dell'infrastruttura di produzione

In questa visualizzazione grafica della topologia, è possibile consultare i dettagli di ogni singolo componente attraverso una finestra laterale che compare quando andiamo a cliccare sulla sua icona circolare. A questo punto effettuiamo una scansione delle vulnerabilità sull'intera topologia, selezionando nelle opzioni di scansione tutti i linguaggi di programmazione disponibili. La prima scansione risulta più lenta delle successive a causa della necessità di scaricare l'insieme di dati a cui fa riferimento ThreatMapper per l'individuazione e la classificazione delle vulnerabilità. Al termine della scansione, nella sezione “Vulnerabilities” sono disponibili diverse rappresentazioni grafiche di tutte le vulnerabilità rilevate da ThreatMapper con la loro rispettiva classificazione, basata sui diversi criteri di gravità esposti nel *paragrafo 2.3* del capitolo precedente. Nella **figura 19** è possibile osservare il risultato di questa scansione.

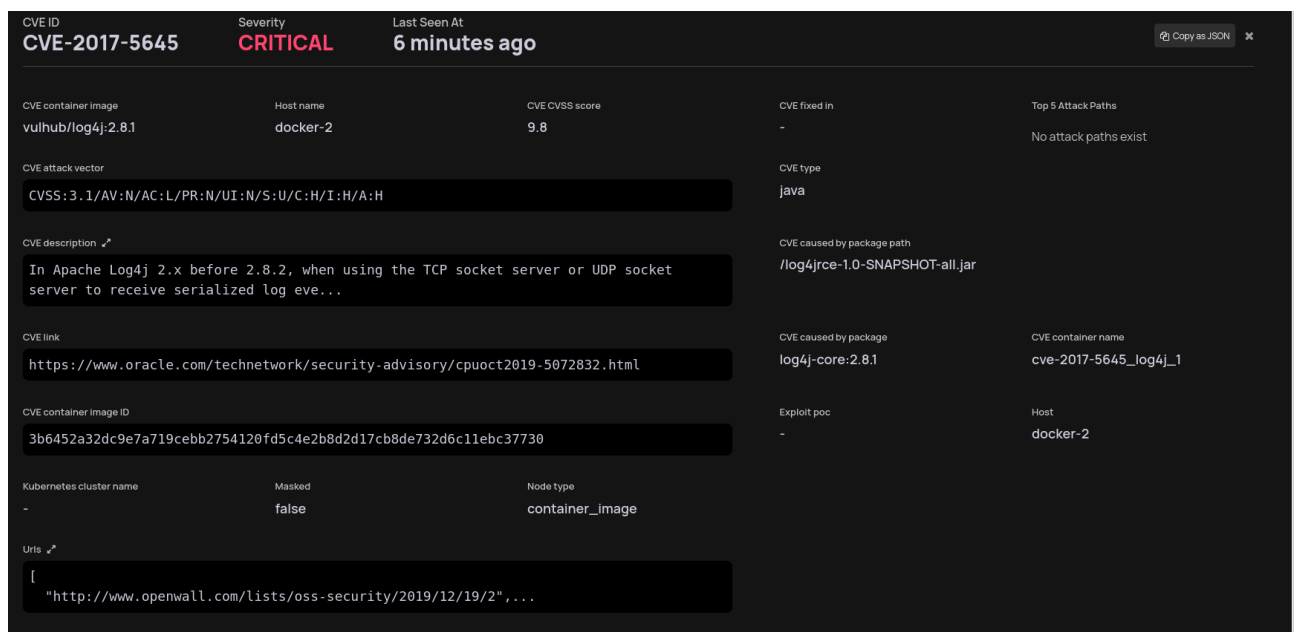


**Figura 19:** Risultato della scansione delle vulnerabilità

In questa sezione troviamo tre diverse schermate: Most Exploitable Vulnerabilities, Vulnerability Scans e Runtime BOM.

La prima schermata integra un contesto “Runtime” all’analisi dei risultati della scansione. Raggruppa le vulnerabilità individuate nelle parti del sistema collegate alla rete internet, ovvero esposte ad un traffico di rete esterno al sistema. Inoltre, è possibile osservare i

percorsi di attacco più vulnerabili, i quali legano uno o più vulnerabilità per il loro possibile utilizzo concatenato. Nella seconda schermata sono presenti i risultati statici della scansione, organizzati rispecchiando la struttura del sistema analizzato. Infine, nel Runtime BOM sono riportati tutte le SBOMs, costruiti in tempo reale, di ogni componente dell'infrastruttura di produzione. Come possiamo verificare nei risultati di questa scansione, ThreatMapper ha individuato le due vulnerabilità presenti nei Docker container vulnerabili. Possiamo vedere nella **figura 20** i dati raccolti da ThreatMapper e il punteggio assegnato in base alla gravità della vulnerabilità **CVE-2017-5645**, presente all'interno del Docker container in esecuzione su Docker-2.



**Figura 20:** Caratteristiche della vulnerabilità CVE-2017-5645

La **figura 20** riporta le seguenti informazioni:

- CVE container images: file statico con il codice eseguibile che ha creato il Docker container vulnerabile.
- Host name: identificativo del dispositivo affetto dalla vulnerabilità.
- CVE CVSS score: punteggio di gravità della vulnerabilità.
- CVE attack vector: stringa vettoriale che rappresenta i valori utilizzati per derivare il punteggio.

- CVE description: breve descrizione della vulnerabilità.
- CVE link: collegamento ad una pagina esterna in cui sono descritte le caratteristiche della vulnerabilità.
- CVE container image ID: file contenente un hash SHA256 dell'oggetto di configurazione JSON dell'immagine.
- CVE fixed in: versione dell'applicativo in cui è stata rilasciata la patch necessaria a mitigare la vulnerabilità.
- CVE type: linguaggio di programmazione del codice vulnerabile.
- CVE caused by package: risorsa software causa della vulnerabilità.
- CVE container name: identificativo del Docker container.
- Exploit PoC: collegamento ad una pagina esterna in cui è indicata la procedura teorica capace di sfruttare la vulnerabilità.

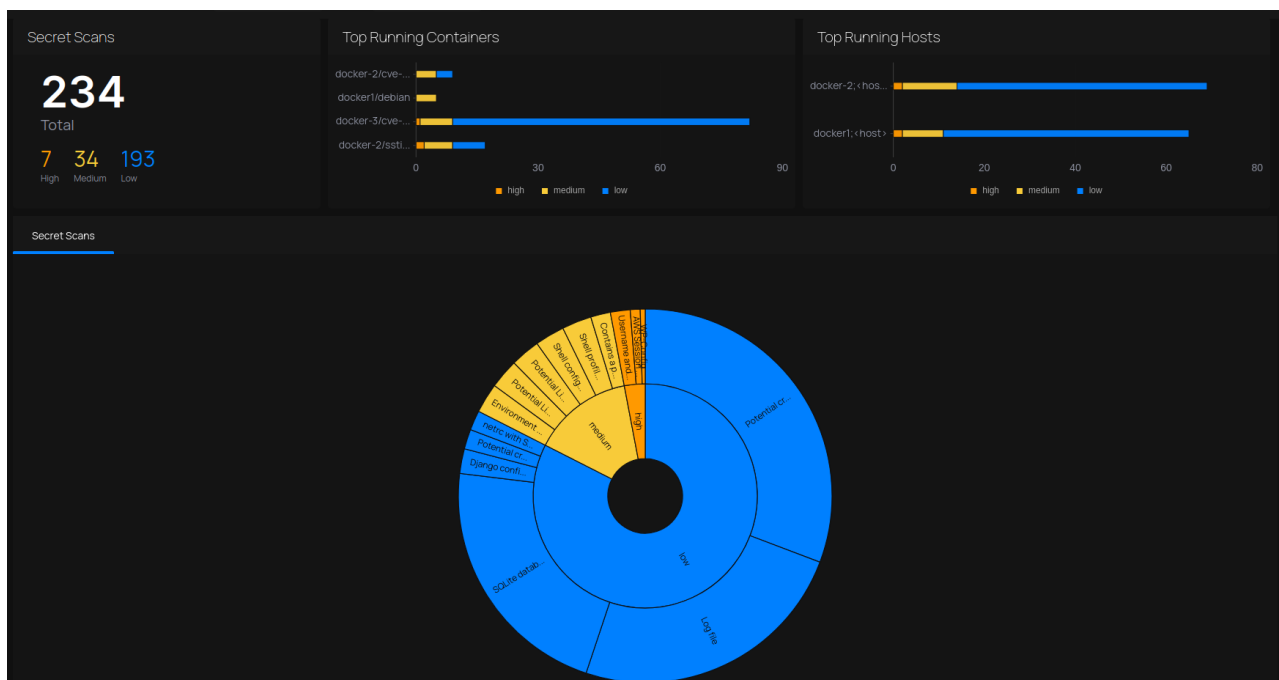
Invece, nella **figura 21** vengono rappresentati i dati raccolti da ThreatMapper della vulnerabilità **CVE-2018-3760**, individuata nel Docker container in esecuzione su Docker-3.

CVE ID	Severity	Last Seen At		
CVE-2018-3760	HIGH	4 minutes ago		
CVE container image	Host name	CVE CVSS score	CVE fixed in	Top 5 Attack Paths
vulhub/rails:5.0.7	docker-3	7.5	-	No attack paths exist
CVE attack vector		CVE type		
CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N		ruby		
CVE description		CVE caused by package		
There is an information leak vulnerability in Sprockets. Versions Affected: 4.0.0.beta7 and lower, 3.7.1 and lower, 2...		sprockets:3.7.2		
CVE link		CVE container name		Exploit poc
https://www.debian.org/security/2018/dsa-4242		cve-2018-3760_web_1		-
CVE caused by package path		Host		Kubernetes cluster name
/usr/local/bundle/specifications/sprockets-3.7.2.gemspec		docker-3		-
CVE container image ID		Masked		Node type
3d3bd96af97f815c5a71d68fc7ad1daa16c7c9674daa8b975169d203c3d827f1		false		container_image
Urls				
[ "http://www.openwall.com/lists/oss-security/2018/06/19/2", ... ]				

**Figura 21:** Caratteristiche della vulnerabilità CVE-2018-3760

ThreatMapper ha la capacità di eseguire un secondo tipo di scansione chiamata “SecretScanner” come abbiamo visto nel *paragrafo 2.2* del secondo capitolo. Quest’ultima permette di scovare le

informazioni sensibili presenti nell’infrastruttura di produzione considerata. I risultati di questa scansione li possiamo osservare nella **figura 22**.



**Figura 22:** Risultato della scansione SecretScanner

Dalla **figura 22** si evince l’individuazione di 234 possibili “segreti” vulnerabili rilevati all’interno dell’ambiente di produzione. Molte di queste informazioni sensibili sono necessarie per il corretto funzionamento dei server e dei carichi di lavoro in esecuzione. Tuttavia, una piccola parte di queste informazioni sensibili potrebbe essere stata lasciata accidentalmente dall’amministratore di sistema all’interno di un carico di lavoro. Infatti, l’obiettivo di questa scansione è quello di controllare i risultati ottenuti e rimuovere i “segreti” lasciati per errore nei carichi di lavoro in esecuzione.

Infine, riportiamo il risultato della schermata Runtime BOM, prendendo a scopo esplicativo il Docker container contenente la vulnerabilità **CVE-2018-3760**. Nella **figura 23** notiamo la dipendenza del Docker container da 512 risorse software diverse. Tali informazioni sono ricavate da ThreatMapper attraverso le modalità approfondite nel *paragrafo 2.1* del secondo capitolo. Ognuna di queste dipendenze può essere caratterizzata da una vulnerabilità.



Package Name	File Path	Top CVE ID	Vulnerability Status
six:1.10.0	/usr/lib/python2.7/dist-packages/six-1.10.0.egg-info/...		Not Vulnerable
adduser:3.115			Not Vulnerable
apt:1.4.8		CVE-2019-3462	High
base-files:9.9+deb9u5			Not Vulnerable
base-passwd:3.5.43			Not Vulnerable
bash:4.4-5		CVE-2019-18276	High
bsdutils:1:2.29.2-1+deb9u1		CVE-2016-2779	High
coreutils:8.26-3		CVE-2016-2781	Medium
dash:0.5.8-2.4			Not Vulnerable
debconf:1.5.61			Not Vulnerable

Previous 1 2 3 ... 51 52 Next

**Figura 23:** SBOM del Docker container *cve-2018-3760\_web\_1*

## Conclusioni

---

In conclusione, i moderni ambienti di produzione, come abbiamo constatato nell'elaborato, diventano sempre più complessi e dinamici, aumentando difatti il perimetro della superficie d'attacco. Inoltre, il costante aumento delle vulnerabilità conosciute pubblicamente espone continuamente le organizzazioni a possibili attacchi informatici. Di conseguenza, risulta fondamentale l'utilizzo di un sistema di monitoraggio della sicurezza, continuo e automatico, capace di rilevare e rispondere rapidamente alle minacce. La piattaforma di monitoraggio della sicurezza ThreatMapper, dalle analisi svolte nell'elaborato, risulta un efficace strumento per far fronte a questa necessità.

Per semplicità di trattazione e di analisi in questo elaborato, abbiamo verificato le funzionalità di ThreatMapper all'interno di un ambiente di produzione formato da soli tre Docker Host; tuttavia, ci ha permesso di osservare concretamente l'efficacia degli strumenti messi a disposizione da ThreatMapper. Infatti, ThreatMapper, dopo aver ricostruito rapidamente la topologia dell'infrastruttura di produzione, è riuscito ad individuare e a classificare in tempo reale le vulnerabilità poste volontariamente all'interno dei due Docker container distribuiti in due degli host scansionati.

ThreatMapper è un progetto in rapida evoluzione, numerose aziende come Amyris, Flexport e Harness lo utilizzano attualmente. Infatti, come abbiamo potuto verificare con lo studio in quest'elaborato delle funzionalità rilasciate del recente aggiornamento 1.3.0 del software, è costantemente aggiornato con lo scopo di implementare ulteriori capacità di osservabilità della sicurezza.

## Ringraziamenti

---

Innanzitutto, ringrazio il mio relatore, Roberto Natella, per la sua disponibilità e tempestività nel rispondere ad ogni mia domanda. Ha saputo guidarmi con suggerimenti pratici nella stesura di questo elaborato.

Ringrazio infinitamente i miei genitori per il continuo supporto e affetto riservatomi durante tutto il mio percorso di studi. Ringrazio mio padre per avermi dato la forza di superare ogni limite con i suoi incoraggiamenti. Ringrazio inoltre mia madre per avermi ascoltato e confortato quando ne avevo bisogno. Non dimenticherò mai le sere prima di ogni esame, nelle quali in compagnia di Bismarck e Django mi ascoltavi con pazienza ripetere ad alta voce gli argomenti più disparati.

Ringrazio mio fratello Gennaro, capace di farmi ridere nei momenti più duri con una delle sue storie o uno dei tanti video trovati online. Grazie anche a mia sorella Alessandra fonte di ispirazione e coraggio, mi hai saputo dare i giusti consigli quando ne avevo bisogno.

Ringrazio in particolare mio nonno Gennaro, la prima persona che chiamavo al termine di ogni mio esame. Grazie per aver tifato sempre per me con entusiasmo.

Desidero ringraziare il mio migliore amico, Gigi, senza la quale non sarei riuscito ad affrontare quest'ultimo anno di università. Le nostre interminabili chiamate su Discord sono un ricordo prezioso che porterò sempre con me.

Infine, grazie di cuore a Lisa; sei stata un punto fisso nella mia vita quest'ultimo anno. Mi hai donato momenti di spensieratezza nei periodi più difficili della mia carriera universitaria. Mi hai sempre sopportato, calmato, spronato e incoraggiato. Sei sempre stata con me, non mi hai mai lasciato da solo e mi hai sempre fatto sentire quanto tu credessi in me, dopo ogni vittoria come dopo ogni sconfitta. Grazie amore mio, te ne sarò sempre grato

## Bibliografia

---

- [1] Laura Zanotti, Sicurezza dei microservizi: linee guida per gli sviluppatori, TechTarget, 2020.
- [2] Owen Garrett, wiki GitHub, ThreatMapper.
- [3] Visualize Attack Paths in Production Environments with ThreatMapper, <https://deepfence.io/visualize-attack-paths-in-production-environments-with-threatmapper/>
- [4] Wikipedia (Docker), <https://it.wikipedia.org/wiki/Docker>
- [5] Docker docs, <https://docs.docker.com/get-started/overview/>
- [6] Wikipedia (Kubernetes), <https://it.wikipedia.org/wiki/Kubernetes>
- [7] IMB Cloud Education, <https://www.ibm.com/it-it/cloud/learn/kubernetes>
- [8] Deepfence revamps ThreatMapper with new scanner, runtime SBOMs, Shweta Sharma, <https://www.csoononline.com/article/3654470/deepfence-revamps-threatmapper-with-new-scanner-runtime-sboms.html>
- [9] National Vulnerability Database (NVD), <https://nvd.nist.gov/>
- [10] Synopsys, rapport OSSRA 2022, <https://www.synopsys.com/blogs/software-security/open-source-trends-ossra-report/>
- [11] IMB Cloud Education, <https://www.ibm.com/cloud/learn/devsecops>
- [12] Sistema di valutazione delle vulnerabilità comuni (CVSS), Bernhard Zwickler, <https://www.cybersecurity360.it/soluzioni-aziendali/sistema-di-valutazione-delle-vulnerabilita-comuni-cvss-cose-come-funziona-gli-sviluppi-futuri/>
- [13] NIST, Metriche di vulnerabilità, <https://nvd.nist.gov/vuln-metrics/cvss>
- [14] Deepfence brings ‘attack path’ visualizations to ThreatMapper vulnerability

- platform, Paul Sawers, <https://venturebeat.com/2022/01/24/deepfence-brings-attack-path-visualizations-to-threatmapper-vulnerability-platform/>
- [15] Wikipedia (VirtualBox), <https://it.wikipedia.org/wiki/VirtualBox>
  - [16] Owen Garrett, wiki GitHub, Installing the Management Console, <https://github.com/deepfence/ThreatMapper/wiki/Installing-the-Management-Console>
  - [17] Registro VulHub, Owen Gong, <https://github.com/vulhub/vulhub>
  - [18] CVE-2017-5645 Detail, NVD, <https://nvd.nist.gov/vuln/detail/CVE-2017-5645>
  - [19] CVE-2018-3760 Detail, NVD, <https://nvd.nist.gov/vuln/detail/cve-2018-3760>