POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea

Protocollo MACsec applicato su canali di un cluster Kubernetes

Relatori
Prof. Antonio Lioy
Ing. Ignazio Pedone

Candidato
Chiara Bisceglia

Ottobre 2023

Alle mie due ali,
che mi hanno permesso di volare fino a qui
Mamma e Papá

Sommario

Gli attacchi informatici rappresentano una minaccia costante per le reti e la difesa efficace richiede l'implementazione di misure di sicurezza avanzate. Tali misure possono essere adottate sui differenti livelli del modello ISO/OSI di comunicazione. Protocolli come IPsec e WireGuard garantiscono la protezione del pacchetto a partire dal livello 3, lasciando spazio a eventuali attacchi sulla rete fisica. Per far fronte a questo tipo di problematica è stato definito lo standard MACsec. Il protocollo si pone l'obiettivo di fornire sicurezza sul livello 2, garantendo integritá, autenticitá, confidenzialitá e protezione da attacchi di replay. La tesi è incentrata sull'integrazione di questo protocollo in un contesto cloud dove diversi componenti di un orchestratore comunicano su canali fisici, soggetti a potenziali attacchi. L'aggiunta di MACsec va dunque a completare la protezione fornita sui collegamenti fisici tra una macchina e l'altra, affiancando i protocolli di sicurezza che agiscono sui livelli piú alti dello stack. Utilizzando Kubernetes come orchestratore, la tesi si prefigge lo scopo di automatizzare l'integrazione di MACsec nei canali di comunicazione, in concomitanza dell'installazione del plug-in incaricato della gestione della rete e di verificarne l'impatto sulle performances rispetto alla sicurezza garantita.

Ringraziamenti

In tutto il mio percorso accademico, sin dalle scuole medie, mi è sempre stato rimproverato di non essere in grado di scrivere un testo letterario con i giusti toni o le giuste parole (perché troppo aulici), è un difetto che mi sono portata sino alla maturitá a quanto pare. Adesso invece mi ritrovo a non sapere scrivere testi scientifici. Questo è l'unico spazio che mi rimane per tirare fuori due parole semplici, senza scienza, senza toni altisonanti, vediamo se anche il mondo universitario mi rimprovererá.

Ho compreso a pieno come puó essere dura lavorare su un progetto e vederlo terminare (con tempi lunghi devo dire) e questo mi ha portato a pensare a chi è stato partecipe anche solo in parte di questo lavoro, e chi mi ha portato a terminarlo.

Il primo grazie va a chi mi ha visto salire da quei primi errori di scrittura sino a qui, che mi ha permesso di arrivare qui, di essere qui e a chi ha anteposto i miei desideri ai loro (penso che cittá piú lontana non potevo sceglierla). Non penso proprio che sarei diventata la persona che sono ora senza di loro, sono i responsabili di tutti i miei pregi e dei miei difetti nonché le mie solide spalle. Grazie mamma e papá.

Ovviamente ci sono altre due figure che mi hanno accompagnata in questo percorso, tra liti e gioie se posso dire, ma diciamo che non ci annoiamo mai. Dalla creazione di un gruppo whatsapp per non farmi piangere i primi giorni fino a qui in cui non penso di aver mai smesso di versare lacrime. Grazie Mero e Raffo (ovviamente anche relativi consorti Michele e Ornella e i miei piccoli Celeste e Noah).

Senza voler escludere nessuno, ringrazio solidamente tutta la mia famiglia, perché io penso che una famiglia cosí non la potevo neanche lontanamente desiderare, è stata il mio muro portante che ha retto tutti i miei crolli fino ad ora. Grazie.

Ringrazio le mie amiche da una vita, quelle persone che sono 20 anni che ci conosciamo e di cui mia madre chiede ancora i nomi quando esco. Sono diventata quello che sono grazie a loro, grazie alle loro tirate di orecchie, alle loro attenzioni, alle grida, alle videochiamate mai fatte, ma perché in fondo effettivamente non ce n'era bisogno, grazie ragazze per tutto. Grazie Gio, Lu e Li.

Passiamo alla sede di Torino, dove penso di aver incontrato le persone piú pazze del pianeta che mi hanno capita dall'inizio e che mi hanno mostrato come la vita universitaria possa essere un'avventura che non finisce mai, perché è questo che siamo, un ricordo indelebile. So che sembra un pó vecchiotto da dire, ma nella triennale non avevo la tesi quindi, grazie Pomba 21 (voi sapete chi siete).

Per uno strano caso del destino invece mi sono ritrovata in casa con una persona altrettanto insensata come me, con cui ci siamo capite al volo in un attimo e con cui il mio rientro dal lavoro o dall'universitá ho potuto davvero chiamare casa. Grazie Oana.

Tra i banchi invece ho conosciuto altrettanti soggetti che mi hanno fatto vivere il Politecnico e non, in tutte le sue forme. Ringrazio Mariam per aver reso tutte le lezioni sempre piú leggere e per il coraggio che mi ha dato nell'affrontare tutto come se fosse sempre possibile. Grazie.

Grazie a Guglielmo, Marta, Matteo e Riccardo. Perché quando tutto il mondo è andato giú eravamo ancora lí a sostenerci e a fare il tifo gli uni per gli altri. Siamo andati avanti e siamo arrivati alla fine perché ci siamo dati semplicemente la forza di arrivare alla fine. Grazie di tutto.

E grazie, grazie, grazie, e forse non smetteró di dirlo, alla persona che è entrata nella mia vita a causa o grazie al Politecnico, per un semplice progetto (che semplice non era per fortuna) siamo riusciti ad incontrarci e da lí è come se avessi vissuto tutto con leggerezza e con la consapevolezza di avere una spalla sempre pronta su cui poggiarmi. Grazie per avermi ascoltata, per aver asciugato le mie innumerevoli lacrime, questa tesi è in parte anche tua. Grazie Dani.

Come succedeva un tempo, mi sono dilungata in parole altisonanti e troppo evocative. Peró ormai sono un quasi ingegnere e di parole ne dovrei usare ben poche, quindi grazie a tutti quelli che non ho nominato, il mio percorso inizia da qui.

Indice

Capitolo 1

Introduzione

1.1 Motivazioni

La sicurezza delle reti informatiche è di fondamentale importanza per proteggere i dati sensibili e garantire la privacy degli utenti. Gli attacchi informatici rappresentano una minaccia costante per le reti, e la difesa efficace richiede l'implementazione di misure di sicurezza avanzate. Tali misure possono essere adottate sui differenti livelli del modello ISO/OSI di comunicazione. Protocolli come *IPsec* [1] e *WireGuard* [2], garantiscono la protezione del pacchetto a partire dal livello 3, lasciando spazio a eventuali attacchi sulla rete fisica (ARP Spoofing, MAC flooding, DoS, etc.). Per far fronte a questo tipo di problematica è stato definito lo standard *Media Access Control Security (MACsec)* [3]. Il protocollo si prefigge l'obiettivo di fornire sicurezza sul livello 2, garantendo integrità, autenticità, confidenzialità è protezione da attacchi di replay.

Uno scenario del genere si puó presentare anche in un contesto cloud, in cui diversi componenti di un orchestratore comunicano su canali fisici, soggetti a potenziali attacchi. L'orchestratore è una piattaforma incaricata di controllare e organizzare il ciclo di vita di un container, unità di calcolo principale all'interno del cloud. Nativamente i componenti di un orchestratore non comunicano su canali sicuri, ma attraverso strumenti esterni è possibile fornire ai container una protezione maggiore dalle eventuali vulnerabilità.

Le principali risorse coinvolti negli orchestratori sono le basi dati, gli scheduler (componente che assegna a ogni servizio avviato il nodo worker su cui eseguirlo), i container in sè e la rete. Le vulnerabilità che coinvolgono questi elementi sono:

- Errori derivanti da una mal configurazione; possono essere dovuti a errori umani o automatismi sbagliati;
- *Man in the Middle* (MITM) nel flusso di comunicazione tra i vari componenti dell'orchestratore, o tra i container;
- Compromissione del database di autenticazione che permette a utenti non autorizzati di operare sulla gestione;
- Difficoltà nel rilevare problemi di sicurezza o eventuali intrusioni a causa dell'eccessivo utilizzo del processore per il check continuo dei servizi attivi.

Nel nostro caso di studio, l'orchestratore in esame è *Kubernetes* [4] e presenta le stesse criticità di sicurezza elencante nel paragrafo precedente, in parte potrebbero essere sanate con configurazioni ideali, ma nella maggior parte dei casi di installazione questo non avviene. La vulnerabilità maggiore risiede nel passaggio dei dati tra un container presente in un nodo del cluster e l'altro. Questo collegamento è costituito da una rete fisica che collega direttamente le due macchine e pertanto, per quanto potrebbe essere protetto sui livelli sovrastanti di comunicazione, il livello piú basso rimarrebbe scoperto.

L'intrusione è possibile attraverso un collegamento diretto alla rete fisica (molto sottovalutato) e con un attacco MitM è possibile carpire i dati di passaggio nel flusso di pacchetti tra servizi attivi, semplicemente inserendosi nel mezzo della comunicazione (possono essere anche container del medesimo servizio che si scambiano informazioni sensibili).

La tesi si prefigge lo scopo di integrare il protocollo MACsec nel canale in concomitanza dell'installazione del plug-in di gestione della rete (plug-in esterno di Kubernetes che provvede alla configurazione automatica di tutte le reti e sotto reti di worker e nodo master), di modo da inserirsi correttamente nel contesto della rete preconfigurata e installata. L'integrazione verrà automatizzata attraverso l'utilizzo di script ansible evitando una configurazione manuale che comporterebbe problemi di overhead e di possibili errori umani. I canali instaurati in tal modo saranno sicuri a partire dal settaggio del cluster.

L'elaborato procederà in tal senso secondo quanto indicato di seguito:

- Panoramica generale sul protocollo MACsec, funzionamento;
- Illustrazione della piattaforma Kubernetes con particolare attenzione alla configurazione di rete e ai plug-in determinanti per tale configurazione;
- Analisi delle tecnologie utilizzate e design della soluzione adottato;
- Implementazione eseguita con relativi test e conclusioni

Capitolo 2

MACsec

In questo capitolo analizzeremo i concetti base e le caratteristiche tecniche dello standard *Institute of Electrical and Electronics Engineers (IEEE) 802.1AE-2018* [3] relativo al protocollo MACsec, primo elemento caratterizzante del caso di studio. In tal senso, verrà ripercorso il funzionamento, possibili applicazioni, standard di sicurezza, vantaggi e svantaggi, di modo da poter dare una base di partenza per l'esaustiva comprensione del lavoro realizzato.

2.1 Panoramica

Oggi giorno il mondo dell'IoT è sempre piú al centro dell'attenzione e il suo utilizzo cresce esponenzialmente di giorno in giorno, ma per far fronte alle necessità intrinseche di una minore latenza nel servizio e una maggiore protezione in termini di comunicazione, è necessario un miglioramento della rete sotto i diversi aspetti. Le varie soluzioni adottate riguardano, in termini di sicurezza e di prestazioni, i diversi livelli del modello ISO/OSI, a partire dal livello superiore con il *Transport Layer Security (TLS)* [5] sul quarto, *IPsec* [1] sul terzo e altri, sino a quello inferiore con MACsec sul secondo; quest'ultimo è di nostra particolare attenzione nonchè oggetto del caso di studio per la protezione sul livello 2 in un collegamento peer-to-peer fisico.

Media Access Control Security (MACsec) è uno standard di sicurezza di rete con cifratura hop by hop operante su dispositivi collegati da rete Ethernet, introdotto nel 2006 attraverso lo standard *802.1AE* [3]. MACsec cifra ogni pacchetto entrante a livello della Local Area Network (LAN) e lo verifica una volta giunto sul nodo destinatario. Il funzionamento prevede: uno scambio di chiavi per creare una comunicazione bidirezionale e per garantire un'autenticazione preventiva, integrità e cifratura dei dati per proteggere i pacchetti trasmessi.

Di seguito vengono analizzate tali caratteristiche piú nel dettaglio [6]:

- *Confidenzialità:* Un utente esterno non sarà in grado di leggere il contenuto dei pacchetti attraverso l'utilizzo della cifratura solida offerta da MACsec. Questa funzionalità offerta puó anche essere disabilitata, in quanto le caratteristiche di base sono solo integrità e autenticità del dato.
- *Integrità:* Il pacchetto non puó essere alterato da terzi poiché MACsec opera effettuando un check del frame (modalità descritte in seguito 2.5), in modo da verificare che i dati arrivino come inoltrati dal mittente.
- *Flessibilità:* In qualsiasi comunicazione è facile abilitare o disabilitare lo standard e allo stesso modo aggiungerlo su piú tratti del flusso dei dati.
- *Autenticazione origine dei dati:* Il protocollo prevede uno scambio preventivo di chiavi che determina un'autenticazione degli utenti e garantisce una certezza dell'origine dei dati. Se questo non avvenisse come primo passo non sarebbe possibile comunicare attraverso MACsec, o comunque servirebbe un passaggio ulteriore preventivo allo scambio di pacchetti.

- *Intelligenza della rete:* MACsec è un protocollo che sfrutta una cifratura hop by hop, pertanto i pacchetti in ingresso in ogni dispositivo verranno sempre prima decifrati per poter essere letti e questo viene effettuato automaticamente nello scambio dei frame.

2.2 Applicazioni ad attacchi noti

Come illustrato in precedenza, uno dei livelli ISO/OSI da salvaguardare è il data link layer. Esistono differenti tipi di attacchi suddivisi in base al destinatario dell'attacco e alla modalità di esecuzione. Questi ultimi possono essere attacchi diretti al nodo, attacchi rivolti al singolo software, o attacchi avvenuti attraverso lo sfruttamento di vulnerabilità conosciute. I principali attacchi su una rete fisica sono i seguenti:

Man-in-the-middle (MITM) attacco in cui un utente malintenzionato puó ascoltare e alterare i pacchetti passanti tra due nodi, semplicemente, come dice il termine, inserendosi nel mezzo della comunicazione con la finalità di leggere il traffico o alterarlo.

Denial-of-Service (DOS) quantità di traffico cablato di pacchetti per esaurire le risorse necessarie all'erogazione di un servizio con la finalità di renderlo inaccessibile al client o per dar vita a un attacco di altra natura.

MAC Spoofing impersonificazione di un dispositivo sulla rete fisica attraverso il cambiamento del proprio indirizzo MAC con quello di un altro nodo presente sulla rete. Le finalità di questo attacco sono legate all'accesso autorizzato a una determinata risorsa (con l'utilizzo delle Access Control List che garantiscono l'accesso autorizzato), oppure al furto del traffico destinato al dispositivo sotto attacco.

MAC Flooding attacco che consiste nell'invio di pacchetti continui a uno switch con l'unico fine di riempire la *Content Addressable Memory table (CAM)* [7] (struttura dati che associa all'indirizzo MAC la propria porta sullo switch). In seguito alla saturazione della tabella, in base alla configurazione dello switch, il dispositivo va in fail open e si comporta come un hub inviando tutti i pacchetti ricevuti in broadcast a tutti i dispositivi collegati a esso. In seguito, l'attaccante puó rubare traffico o eseguire attacchi di MitM.

ARP Spoofing in genere una Address Resolution Protcol (ARP) Request ha come obiettivo la risoluzione di un indirizzo MAC nel suo corrispondente IP e contrariamente l'obiettivo dell'attacco è invece quello di avvelenare le tabelle che contengono questa traduzione per poter direzionare il traffico verso un'utenza malevola o per poter procede con un attacco di DoS.

Rouge DHCP il server Dynamic Host Configuration Protocol (DHCP) crea e distribuisce le configurazioni di rete, l'attacco consiste nel causare un malfunzionamento del server attraverso l'invio di pacchetti continui che creano un Denial of Service, di conseguenza l'attaccante impersona il server DHCP stesso e distribuisce configurazioni errate con l'obbiettivo di un attacco di DoS o di MitM.

Gli attacchi appena elencati possono essere scongiurati in parte dall'aggiunta del protocollo MACsec. Com'è possibile evincere dalle specifiche tecniche degli attacchi, una delle problematiche principali riguarda una mancata autenticazione sulla linea di collegamento. MACsec puó garantire questo determinato passaggio attraverso lo scambio preventivo di chiavi, per cui un nodo che non le possiede e non fa parte del collegamento non potrà leggere i pacchetti. In conseguenza diretta gli attacchi di MitM e di seguito quelli di DoS vengono scongiurati. Un'altra protezione che il protocollo puó garantire è quella di *anti-replay*, in quanto puó essere abilitato il numbering dei pacchetti per evitare perdite, ma anche per scongiurare l'utilizzo di pacchetti copia e di conseguenza eventuali attacchi che scatenano un Denial-of-Service.

2.3 Casi d'uso

Il sempre piú elevato utilizzo di Ethernet come principale metodo di comunicazione fra datacenter o router server è dettato dalla necessità di velocità di connessione piú elevate rispetto a quelle utilizzate per connessioni su Livello 3 o 4. L'unico vero ostacolo per l'utilizzo della tecnologia Ethernet era il livello di sicurezza adottato in quanto il canale risultava vulnerabile per tutti i rischi di attacco elencati in precedenza. Con l'utilizzo del protocollo MACsec si è giunti a una soluzione per oltrepassare tali rischi e pertanto attualmente l'utilizzo è sempre piú frequente per i seguenti casi:

- router Wide Area Network (WAN);
- router e switch di data center;
- server e storage;
- switch in una LAN.

Un esempio pratico è la connessione tra due datacenter con l'utilizzo della WAN (grande rete di comunicazione che si estende su una vasta area geografica), di seguito illustrata nella figura 2.1. Il pacchetto entra all'interno di una WAN protetta da tecnologia MACsec passante attraverso router configurati e router centrali di instradamento del traffico. Una volta giunto a destinazione il pacchetto viene decifrato e inoltrato all'utente finale.

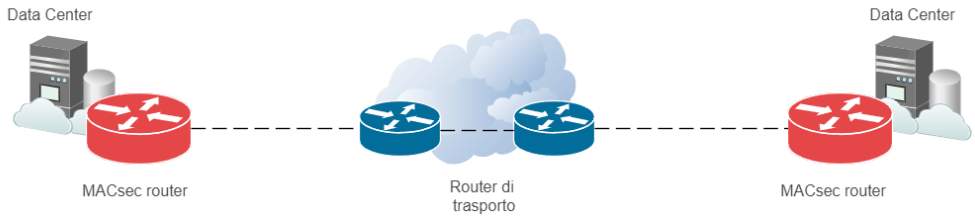


Figura 2.1: Connessione MACsec tra data center.

2.4 Architettura

Per poter comprendere come questi pacchetti viaggino tra un nodo e l'altro è importante prestare attenzione a quali siano i principali canali di comunicazione e come questi vengano configurati e instaurati. Questo paragrafo è volto a presentare le principali architetture di comunicazione a un alto livello in modo da poi poter scendere nel dettaglio con le sezioni successive.

Nel contesto di due nodi connessi fisicamente fra di loro, per poter stabilire una connessione sicura attraverso il protocollo MACsec è necessario che essi facciano parte della medesima Connectivity Association (CA). Nello specifico le CA sono canali sicuri creati tra nodi sulla stessa LAN, abilitati all'utilizzo dello standard, che condividono la stessa chiave e la stessa cipher suite. La chiave in questione è chiamata Connectivity Association Key (CAK) associata a una Connectivity Association Key Name (CKN).

La CAK nello specifico è una chiave statica preconfigurata e condivisa con tutti gli appartenenti alla medesima CA. La diffusione e la generazione di tale chiave vengono eseguite da uno standard differente da MACsec, nello specifico *IEEE 802.1.X-2010* [9] (analizzato in seguito 2.8).

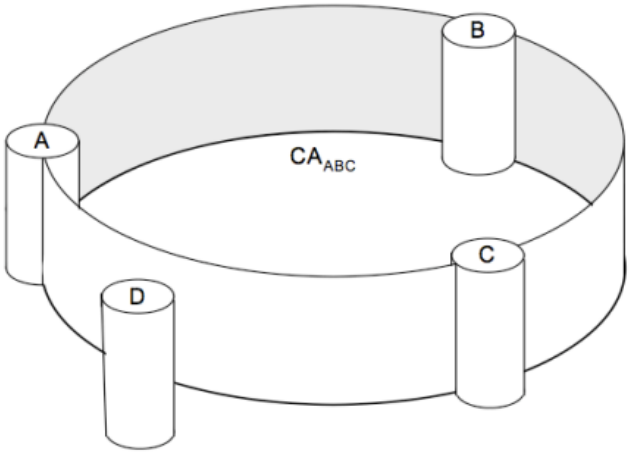


Figura 2.2: connessione CA [8].

La comunicazione tra i vari nodi, appartenenti sempre alla stessa CA, dovrà essere supportata dalla creazione di un Security Channel (SC), tale connessione è unidirezionale e puó essere presente in una comunicazione point-to-point o point-multipoint. Ogni peer potrà avere un solo canale in uscita, ma piú SC in entrata. Ad esempio, in una connessione tra due nodi vi sarà un canale per i pacchetti in uscita e un canale per i pacchetti in entrata. L'identificativo per questi canali è il secure channel identifier (SCI), un valore unico creato tramite il MAC address e la porta di connessione. Parallelamente al SC viene creata un'ulteriore connessione di sessione denominata Secure Association (SA), canale ultimo di comunicazione sicura a cui vengono associati i principali elementi per la cifratura del dato come in figura 2.3. Nello specifico a ogni SA è associata una Security Association Key (SAK), chiave simmetrica utilizzata per la cifratura del pacchetto inoltrato che avviene secondo determinati algoritmi compresi nella cipher suite. Ogni SAK all'interno del medesimo SC dev'essere unica e associata a un Secure Association Identifier (SAI), composta dal SCI e dall'Access Number (AN). All'interno di una connessione non è possibile avere piú di quattro SA nello stesso momento.

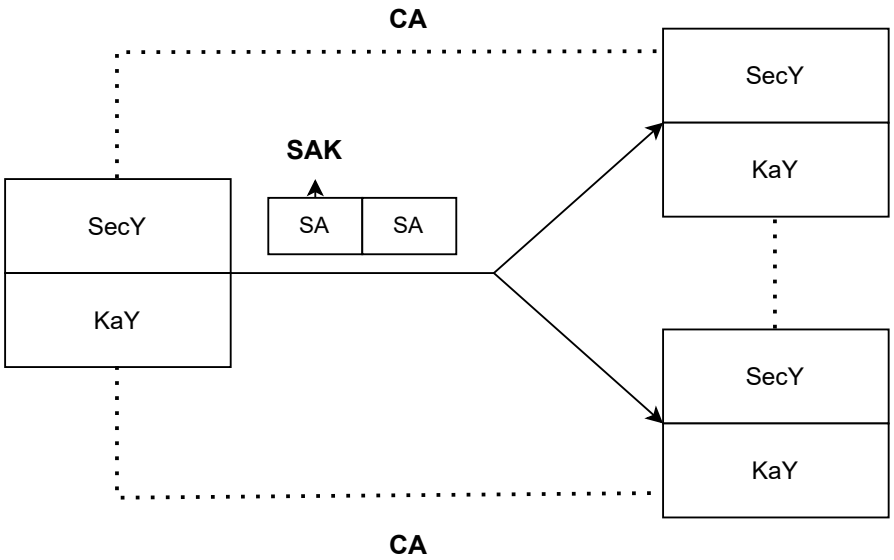


Figura 2.3: Connessioni MACsec.

Su ogni nodo sono presenti due entità, la MAC Security Entity (SecY) e la Key agreement entity (KaY). La prima è responsabile di tutti i processi di cifratura e validazione dei frame in entrata e uscita; principalmente essa detiene la cipher suite da dover utilizzare per mettere in sicurezza il pacchetto dal punto di vista dell'integrità ed eventualmente procedere con la cifratura del Mac Service Data Unit (MSDU). Nello specifico per verificare l'integrità del frame procede con l'utilizzo della chiave segnalata dal SecTag per potere ricavare l'Integrity check value (ICV) e confrontarlo con quello ricevuto nel frame in entrata. Nel paragrafo successivo verrà analizzata la cipher suite possibile da utilizzare attraverso la SecY. Per quanto riguarda la KaY, le sue principali mansioni sono legate allo standard MACsec Key Agreement (MKA) per la gestione delle chiavi e la loro distribuzione e creazione, inoltre è il metodo principale per poter riconoscere gli altri nodi appartenenti alla CA (verificando la possessione della medesima CAK).

2.5 Frame MACsec

Come specificato nei paragrafi precedenti lo standard *IEEE 802.1AE* [3] opera solamente sul livello 2, pertanto i pacchetti su cui applicare il protocollo saranno MAC frame. I nuovi pacchetti, con l'utilizzo di MACsec, manterranno invariato il MAC del destinatario e il MAC sorgente, verranno estesi con campi aggiuntivi e opzionalmente potrà essere applicata la sicurezza ai dati inoltrati dal mittente del messaggio (figura 2.4). In poche parole, l'unica parte alterata dal protocollo sarà quella del MSDU.

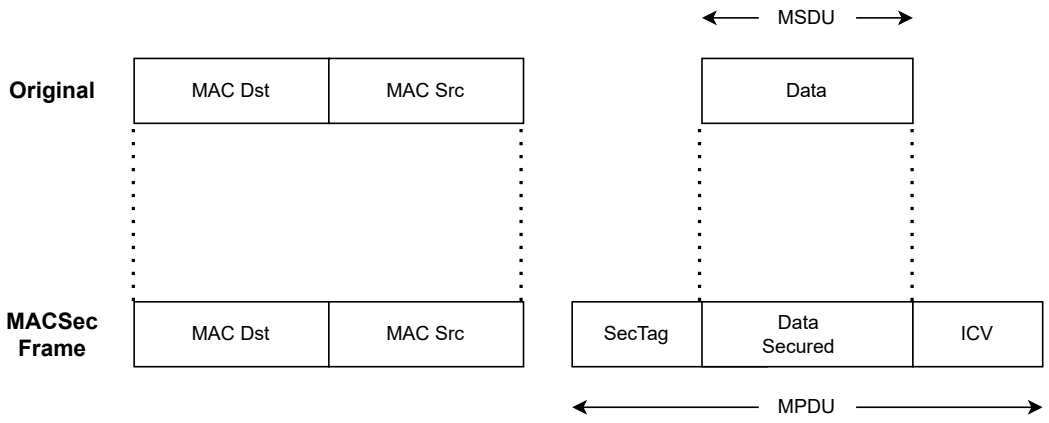


Figura 2.4: MACsec Frame trasformazione.

La nostra attenzione è focalizzata su tale parte, in cui il messaggio verrà cifrato attraverso opportuni algoritmi presenti nella suite crypto dello standard (descritti in seguito 2.7), mentre i due campi aggiuntivi, quali SecTag e ICV, entrambi da 16 byte, verranno analizzati nel dettaglio.

Il SecTag è il principale campo per la descrizione delle configurazioni caratterizzanti per l'applicazione di MACsec sul pacchetto, dai campi per poter ricavare le chiavi di autenticazione e dunque autenticarsi, al numero di pacchetto con cui evitare un attacco di replay, all'identificatore del protocollo stesso.

- Il primo campo presente nel SecTag è il MACsec Ethertype, con un valore pari sempre a `0x88e5`, utilizzato per designare il pacchetto come appartenente allo standard MACsec. In genere oltre alla sua funzione primaria, viene utilizzato anche per identificare sistemi non conosciuti dalla comunicazione.
- Il secondo campo, TAG Control Information (TCI), è suddiviso a sua volta in differenti campi quali:
  - versione del protocollo;

  - un bit per indicare che il MAC sorgente corrisponda al SCI trasmesso, se tale check è affermativo allora il SCI non è encodato;
  - un bit per il SC, questo campo è presente solo se il campo SCI non è encodato;
  - un bit per chiarificare il Single copy broadcast (SCB);
  - un campo per indicare se è avvenuta cifratura o meno;
  - un campo per indicare se i dati trasmessi in chiaro sono uguali a quelli cifrati.
- Il terzo campo AN indica il numero di SA associata al SC.
- Il quarto campo Secure Length (SL) rappresenta il numero di byte del campo SecTag, se va oltre i 48 viene impostato a 0.
- A seguire, il campo PacketNumber (PN) che indica il numero di pacchetto trasmesso, semplicemente per evitare attacchi di replay.
- L'ultimo campo è lo SCI per identificare la SecY o la SA in una connessione CA che è composta da piú nodi. Quando c'è una connessione uno a uno non è necessario.

Il campo ICV viene utilizzato per controllare l'integrità dei dati, in genere viene elaborato attraverso algoritmi di cifratura identificati dalla cipher suite, e una volta arrivato al destinatario viene rielaborato e nel caso di corrispondenza il pacchetto viene accettato, altrimenti scartato.

2.6 Funzionamento

Di seguito si proseguirà con l'illustrazione del flusso di un pacchetto MACsec illustrandone da principio i metodi di validazione e cifratura, proseguendo in seguito con lo studio di due scenari differenti: la comunicazione diretta tra due nodi e lo scambio di pacchetti tra due nodi comunicanti tramite uno switch.

2.6.1 Cifratura

Come definito in precedenza nel paragrafo 2.4 il principale responsabile dei processi di cifratura e validazione è il SecY, il quale è necessario per la creazione di un canale protetto da protocollo MACsec. Il primo passaggio da eseguire è l'associazione del frame a una SA, identificata da un AN. L'AN serve a individuare la SAK da utilizzare. In seguito AN, PN e SCI vengono inseriti all'interno del SecTag. Al termine di queste operazioni il SecY cifrerà la parte di payload e genererà con la cipher suite scelta il campo ICV.

2.6.2 Validazione

All'arrivo del pacchetto sul nodo destinatario la prima operazione eseguita è la derivazione di tutti i campi necessari (quali AN e SCI) dal SecTag per poter ottenere SA e SAK. In seguito viene derivata la cipher suite con cui sarà possibile rielaborare il campo ICV e confrontarlo con quello inoltrato. Se corrispondono il pacchetto viene accettato.

Scenario con due nodi - In tale scenario sono presenti due macchine, facenti parte della medesima CA, che possiedono un SC in entrata e un SC in uscita (poiché è una connessione end-to-end). Il Nodo1, in riferimento allo schema 2.5, stabilisce una connessione SA con il Nodo2 per poter inoltrare il pacchetto e in tal modo genera e stabilisce la SAK (chiave simmetrica di cifratura) per la protezione dei dati inoltrati. Al frame MACsec originale viene eseguita l'aggiunta del campo SecTag (contenente tutte le informazioni utili ai fini della decifratura sul nodo ricevente) e del ICV per il controllo dell'integrità. L'Ethertype viene settato a `0x88E5` per indicare il protocollo usato, mentre il payload viene cifrato secondo l'algoritmo scelto dal Key Server (KS). Nel caso di una connessione uno a uno il campo SCI viene omesso (come illustrato

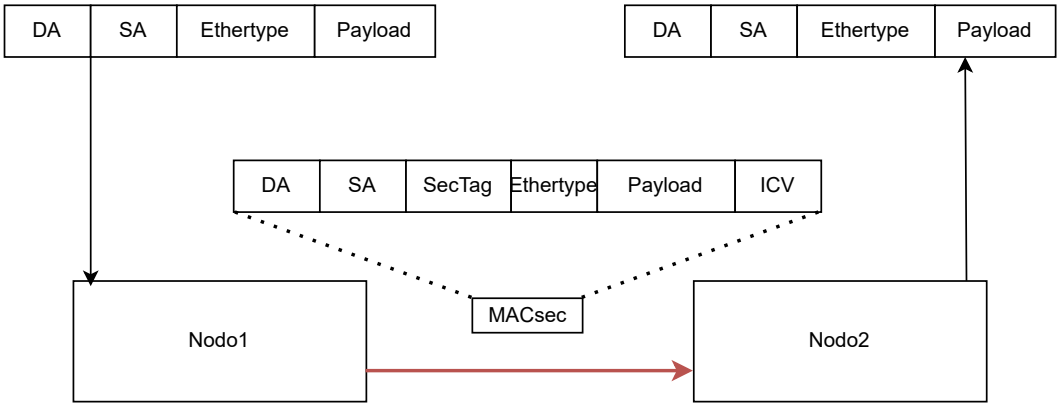


Figura 2.5: Scambio di frame MACsec tra due nodi.

in precedenza). Una volta atterrato sul Nodo2 il pacchetto viene decifrato, validato e consegnato ai livelli piú alti.

Scenario con due nodi e uno switch - Nel secondo scenario sono rappresentate due macchine non facenti parte della stessa CA, ma aventi ognuna delle due una CA condivisa con la macchina operante come switch. In tal modo il pacchetto inviato dal `Nodo1` diretto al `Nodo2` atterrerà preventivamente sul nodo switch. Lo switch puó essere configurato con porte abilitate all'utilizzo di MACsec o privo di quest'ultime. Nel primo caso il flusso del pacchetto non verrà interrotto dalla presenza dello switch. Nel secondo caso il pacchetto verrà decifrato per poter essere letto ed essere direzionato correttamente, e in seguito se il canale di uscita è abilitato all'utilizzo di MACsec verrà cifrato nuovamente, altrimenti verrà reinviato in chiaro. In questo caso il pacchetto avrà un punto in cui sarà in chiaro nel suo flusso di comunicazione, piú precisamente all'interno dello switch.

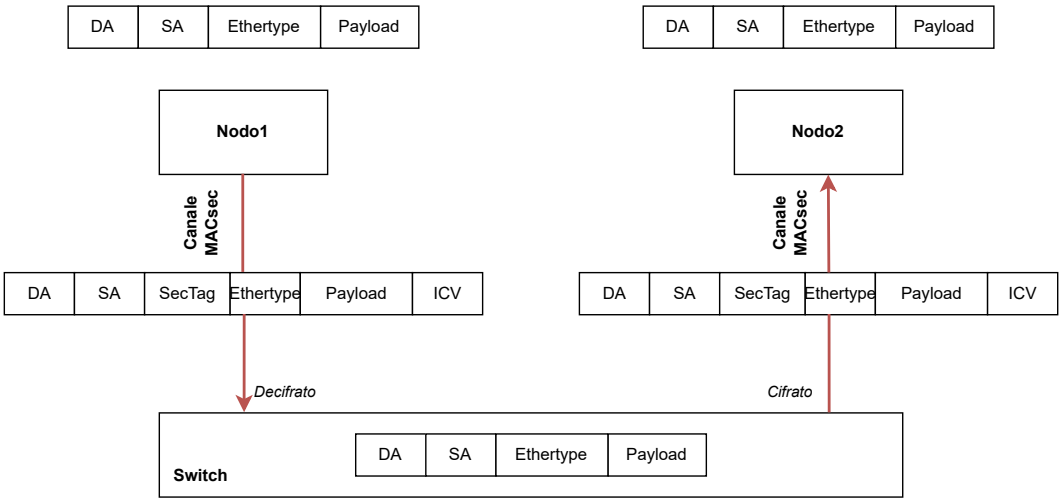


Figura 2.6: Scambio di frame MACsec tra due nodi con uno switch intermedio e porte non abilitate a MACsec.

2.6.3 Vulnerabilità conosciute

Nel paragrafo precedente sono stati analizzati i principali attacchi possibili in una connessione di livello 2, ma per poter eseguire un attacco un utente malintenzionato deve sfruttare una vulnerabilità presente nel sistema. Per classificare le vulnerabilità presenti viene utilizzato il *Common Vulnerability Scoring System (CVSS)* [10] che detta in una scala da 0 a 10 la gravità delle

conseguenze di attacco basate su differenti metriche che valutano la facilità d'implementazione dell'exploit. In base a tale scala è possibile dare priorità ad alcune vulnerabilità piuttosto che ad altre. Le vulnerabilità vengono etichettate attraverso un *Common Vulnerabilities and Exposures (CVE)* [11], un dizionario di tutte le vulnerabilità conosciute che indica l'anno di scoperta e il tag relativo al singolo attacco specifico. Di seguito vengono presentate le vulnerabilità, da cui è affetto MACsec, con un CVSS maggiore di 7.

CVE-2018-0021 (8.8) [12]

Se tutti i 64 caratteri del CKN o i 32 del CAK non vengono popolati, il loro valore verrà settato a 0 e pertanto si potrebbero scoprire, tramite un attacco di brute-force o dictionary-based, le chiavi per autenticarsi all'interno della CA, di conseguenza chiunque potrebbe inserirsi nella comunicazione senza essere considerato come utente malevolo.

CVE-2022-0435 (8.8) [13]

La vulnerabilità riguarda il Berkley Packet Filter (BPF) verifier (ha il compito di prendere i dati dallo stack), in alcune situazioni il bpf verifier non restringe ad alcuni tipi di puntatore e questa è l'apertura adatta a un attacco di DoS. Per sanare tale vulnerabilità la soluzione consiste in un aggiornamento del kernel a una versione migliorata.

CVE-2017-2342 (8.1) [14]

Vulnerabilità che riguarda essenzialmente MACsec in connessione con le reti Juniper, questa viene scatenata quando il collegamento avviene su una porta non abilitata per MACsec e questo fa sí che un utente ritenga il collegamento sicuro quando in realtà non lo è. Il flusso dei frame continua senza sicurezza, pertanto continua una comunicazione in chiaro.

CVE-2018-15372 (8.1) [15]

La vulnerabilità è presente nel protocollo MKA utilizzando *Extensible Authentication Protocol-Transport Layer Security (EAP-TLS)* [16] per l'autenticazione e questo fa sí che un utente malintenzionato possa evitare l'autenticazione e collegarsi al traffico sul livello 3. La vulnerabilità è presente nella logica del software e solo chi utilizza EAP-TLS e ha una connessione in modalità chiusa ne è affetto. La conseguenza è una completa rottura del sistema di accesso dello standard 802.1x e soprattutto un'unione forzata alla CA che permette un'autenticazione malevola e di conseguenza uno sniffing dei frame o un attacco di MitM.

CVE-2017-7477 (7) [17]

Errore di buffer overflusso sull'heap che un utente malintenzionato puó causare per ottenere dei privilegi amministrativi. In realtà l'overflusso viene causato solo se i flag `MAXSKBFRAGS + 1` e `NETIFF FRAGLIST` vengono settati.

2.7 Cipher Suite

MACsec nella maggior parte dei casi predilige come cipher suite la 128-bit *Advanced Encryption Standard-Galois Counter Mode (AES-GCM)* [18] per garantire la confidenzialità e l'integrità dei dati trasmessi, ma sono possibili anche altri algoritmi quali: GCM-AES-256, GCM-AES-XPN-128, GCM-AES-XPN-256, presi piú raramente in considerazione.

AES-GCM è una modalità operativa di cifratura a blocchi simmetrica (medesima chiave di cifratura) che fornisce un'elevata velocità di crittografia autenticata (confidenzialità) e integrità dei dati. Per la chiave in tal caso verrà utilizzata la SAK indicata dall'AN inserito nel frame e associato alla SA di comunicazione.

In modalità GCM, la crittografia a blocchi viene trasformata in crittografia del flusso e pertanto non è necessario alcun riempimento. Gli additional authenticated data (AAD) non saranno cifrati ma utilizzati nel calcolo del tag di autenticazione. Infatti, il frame inviato presenterà la cifratura avvenuta dei dati utente piú il tag ICV citato in precedenza. La cipher suite con *Extended Packet Numbering (XPN)* [19] viene utilizzata per estendere il PN da 32 bit a 64.

Questo è utile soprattutto per raggiungere delle velocità elevate di circa 40Gb/s, semplicemente poiché il PN viene utilizzato insieme al SCI per la generazione della SAK; a 32 bit dopo 75% di 2^{32-1}combinazioni la SAK deve subire rekeying e questo si traduce in una perdita di tempo considerevole, questo accade semplicemente poiché in CounterMode l'Initial Vector (IV) utilizzato dev'essere unico. Invece nel caso di 64 bit impiega piú tempo a raggiungere tutte le combinazioni pari al 75% di 2^{64-1} e il rekeying è piú lontano (questa modalità puó essere eseguita solo in una comunicazione switch-switch, poiché in una switch-host non puó funzionare).

2.8 MACsec Key Agreement

Lo standard IEEE 802.1AE non descrive nel dettaglio le procedure di autenticazione, nè le modalità di generazione e scambio delle chiavi, per tale motivo si avvale di uno standard esterno, per la precisione di IEEE 802.1.X-2010, anche detto MKA.

Lo standard si prefigge come obiettivo primario quello di limitare l'accesso agli access point della LAN solo a dispositivi autenticati e autorizzati, pertanto descrive anche architettura, elementi funzionali e protocolli a supporto di una mutual authentication e una possibile comunicazione sicura tra nodi collegati alle varie porte con l'obiettivo di scoprire e stabilire SA per quanto riguarda il caso di MACsec.

Uno dei compiti principali del protocollo è quello di controllare il possesso delle medesime chiavi CAK, confermando un'autenticazione preventiva, e la produzione nonchè distribuzione delle SAK.

- *CAK*: pre-shared key ottenuta tramite autenticazione EAP. Ad essa è associata una CKN, un identificativo per distinguere le varie CAK.
- *SAK*: chiave simmetrica utilizzata per la cifratura del payload del frame, ottenuta da un server di generazione casuale di valori *(RNG)* [20] che nel processo dell'autenticazione *Extensible Authentication Protocol over LAN (EAPoL)* [21] corrisponde sempre all'authenticator.

Inoltre, MKA provvede al controllo dell'antireplay delle trame e della presenza attiva dei dispositivi collegati alla medesima CA.

2.8.1 Funzionamento

Uno dei principali componenti nel funzionamento dello standard MKA è il KeyServer (KS). Il KeyServer viene eletto tra i nodi disponibili e partecipanti alla CA. Per ogni nodo puó essere deciso un grado di priorità per cui venga eletto come KeyServer. Il KS è il componente che si occupa principalmente della distribuzione e creazione delle chiavi, della selezione della cipher suite, dell'assegnazione della SA e sopratutto dell'utilizzo di MACsec all'interno di un collegamento. Se la CAK dovesse essere ottenuta tramite autenticazione esterna allora l'autenticatore EAP diventerebbe il KS.

2.8.2 Generazione chiavi

La generazione delle chiavi citate nel paragrafo precedente, come già accennato, segue due procedimenti differenti. Per quanto riguarda la CAK essa viene generata da un metodo di autenticazione EAPOL svolto dai concorrenti alla partecipazione della CA. Per quanto riguarda la SAK, in seguito alla generazione tramite server RNG, la sua diffusione segue un procedimento differente, in cui viene utilizzato l'AES key wrap operato dal KS. Come visualizzato nel grafico 2.7 è possibile notare tutto il meccanismo: a partire dalla CAK vengono generate la Key Encrypting key (KEK) e la Integry Check Value (ICV) Key (ICK). La KEK viene utilizzata per la distribuzione effettiva della SAK all'interno della CA, mentre la ICK è una chiave che verrà utilizzata per il

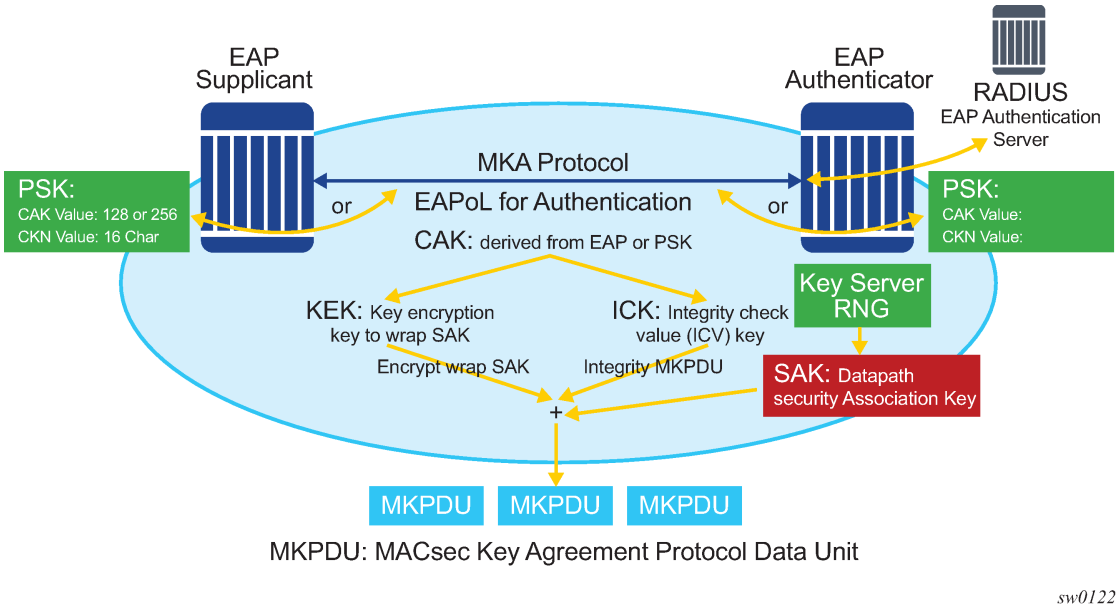


Figura 2.7: Distribuzione chiavi [23].

controllo dell'ICV e dunque dell'integrità del frame attraverso l'algoritmo di *Advanced Encryption Standard-Cipher-based Message Authentication Code (AES-CMAC)* [22].

La parte di body principale della trama EAPOL trasmessa, nel quale viene distribuita la SAK, è MACsec Key Agreement PDU (MKPDU), il quale è a sua volta protetto da integrità tramite ICV (con la stessa chiave ICK). In genere questo pacchetto trasmesso serve, oltre alla distribuzione della chiave, anche per confermare la presenza attiva agli altri partecipanti della CA (mandando un segnale continuo), per la distribuzione della chiave utilizzata per il check dell'integrità e le configurazioni dei vari nodi.

2.9 Tecnologie affini

La protezione considerata fino a ora è sul livello 2, dunque al livello del link layer, mentre sui livelli sovrastanti vi sono altre architetture possibili quali: *IPsec, WireGuard o TLS*. Di seguito verranno analizzate piú nel dettaglio per una comparazione con il protocollo MACsec.

2.9.1 IPsec

Una delle tecnologie piú discusse nel mondo della sicurezza di canale è sicuramente la Virtual Private Network (VPN), una rete privata che garantisce principalmente la protezione del traffico passante e il mascheramento dell'IP. Il funzionamento delle VPN si basa peró su una serie di regole e procedure che lavorano sul livello 3 del modello ISO/OSI, denominate IPsec. I passaggi completi con cui IPsec opera prevedono una prima fase di autenticazione dei dati con lo scopo di evitare attacchi MitM. In seguito sopraggiunge la parte di cifratura dei dati tramite algoritmi e chiavi che sono state concordate nel passaggio precedente. Inoltre, puó garantire un'ulteriore protezione contro attacchi di replay. Le modalità di funzionamento sono due: tunnel per la sicurezza di indirizzo IP e dati, e transport per la sicurezza dei dati singoli. Nello schema 2.8 viene presentato un esempio di utilizzo con il protocollo di Authentication Header (AH). A sua volta IPsec è composto da due protocolli: IPsec Authentication Header (AH) e Encapsulating Security Payload (ESP); il primo garantisce l'integrità dei dati inoltrati nonchè l'autenticità del mittente tramite generazione di un hash del pacchetto, comprendendo solamente i campi non variabili (i campi mutevoli come ad esempio il Time To Live (TTL) vengono impostati a 0),

che verrà verificato una volta giunto al nodo destinatario. AH non garantisce la confidenzialità del messaggio. ESP, invece, elabora autenticità, integrità e confidenzialità, ma rispetto ad AH comporta un pacchetto di grandi dimensioni. Sia ESP che AH supportano entrambe le modalità di IPsec.

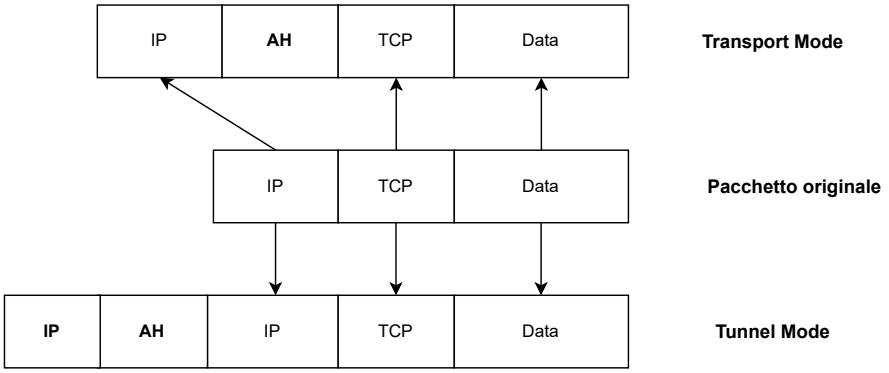


Figura 2.8: Modalità tunnel e trasnport con protocollo AH.

Per la sicurezza completa sarebbe possibile sfruttare IPsec e MACsec insieme, per cui sfruttando MACsec sul livello piú basso si garantisce la protezione che IPsec non puó controllare, vale ugualmente il reciproco. In genere una volta attaccato il livello piú basso è possibile risalire anche su quello piú alto, per questo è buona pratica proteggere anche i primi livelli del modello OSI. L'unico svantaggio di adottare questa soluzione risiede nel periodo di latenza che aumenterebbe vertiginosamente, in quanto per il processo del pacchetto sarebbero da applicare tutte le procedure di IPsec e poi tutte quelle di MACsec. L'installazione dello strumento su sistemi Linux avviene tramite la suite di pacchetti `StrongSwan`. Le principali differenze vengono illustrate nella tabella 2.1.

2.9.2 WireGuard

WireGuard è un software gratuito per la creazione delle VPN per la protezione da utenti non autenticati nel flusso di dati tra due peer. Il protocollo ha come funzione base la creazione dei tunnel cifrati per inoltrare il flusso dei pacchetti tra peer come se fossero collegati direttamente da un cavo di rete. La connessione ha come primo passaggio lo scambio di chiavi pubbliche per il settaggio del canale sicuro, chiunque sia sprovvisto di questa chiave di identificazione non è autorizzato a leggere il traffico dati con pacchetti UDP. In tal modo non utilizza un server per eseguire tutto il settaggio, ma viene eseguito dai peer stessi, dove è possibile installarlo facilmente tramite libreria specifica introdotta su Linux dal solo 2020; le uniche configurazioni manuali riguardano le chiavi e i settaggi di rete per impostare l'interfaccia su cui impostare la VPN. I principali vantaggi riguardano: un codice conciso di modo da poter effettuare un debug piú rapido, una velocità maggiore con minor latenza, e un risparmio di energie dovute alla disconnessione qualora non passi traffico all'interno del canale. Rispetto a IPsec le opzioni di cifratura sono limitate, ad esempio WireGuard non supporta cifratura RSA o le chiavi pre-condivise. Inoltre come detto in precedenza IPsec presenta un codice piú largo e pertanto il costo computazionale è piú elevato. I punti principali di WireGuard sono illustrati nella tabella 2.1.

2.9.3 Transport Layer Security

TLS è un protocollo operante a livello applicativo che garantisce confidenzialità e integrità del pacchetto trasmesso tra due applicazioni. Il protocollo garantisce autenticazione tramite l'utilizzo di certificati volti ad associare una chiave utilizzata per la cifratura dei dati nella comunicazione tra peer, nonchè l'integrità del dato in quanto il primo passo nella comunicazione è dato da un handshake che garantisce l'identità del server e stabilisce le chiavi di sessione, in tal

IPsec	**MACsec**	**WireGuard**
Garantisce sicurezza con tunnel end-to-end	Hop-by-hop: Esposizione del pacchetto all'arrivo su un nuovo switch	Garantisce sicurezza con sistema di vpn protetto da classici tunnel end-to-end di pacchetti UDP
Garantisce massima sicurezza sul livello 3	Lavora livello 2	Garantisce massima sicurezza sul livello 3
Puó lavorare attraverso i Router	Lavora solo su di una LAN / VxLAN	Lavora anche attraverso router
Agendo solo su livello 3 non garantisce sicurezza nei livelli sottostanti	Non garantisce sicurezza sui livelli soprastanti	Agendo solo su livello 3 non garantisce sicurezza nei livelli sottostanti
La cifratura avviene solo alla fine di ogni tunnel	Cifratura e decifratura all'ingresso e all'uscita di ogni switch, comunicazione hop-by-hop	La cifratura avviene tramite impacchettamento su UDP
La complessità del funzionamento	Protocollo facile da configurare	Facilità nel funzionamento e nella configurazione, è tutto standard
Aumenta di gran lunga la lunghezza dell'header Ethernet e porta a inefficienza sulla rete nonchè a costi aggiuntivi	L'espansione dell'header è minima ed è scalabile, ma possiede configurazione manuale	Non mantiene sessioni attive e questo porta a uno svantaggio

Tabella 2.1: Differenza tra le principali tecnologie di protezione del canale

modo non è possibile che un utente malintenzionato possa accedervi. Dati i numerosi passaggi eseguiti, l'installazione del protocollo richiede maggior tempo per la configurazione corretta. Come gli altri protocolli descritti, l'unione tra MACsec e il suddetto protocollo garantirebbero una protezione maggiore alla trasmissione dei pacchetti. L'unica vera pecca di TLS è la protezione sulle applicazioni stesse in quanto l'unica sicurezza garantita è nel collegamento. Le principali caratteristiche del protocollo vengono illustrare nella tabella 2.2.

TLS
Garantisce sicurezza a partire dal livello 4
Il collegamento e la sicurezza vanno garantiti con sistemi aggiuntivi come TCP
Garantisce protezione solo sul collegamento
Facilità nel funzionamento e nella configurazione, ma dev'essere effettuata ad hoc
Difficoltà nella gestione dei certificati e nel mantenimento della cache

Tabella 2.2: caratteristiche principali di TLS

Capitolo 3

Kubernetes

In seguito alle analisi effettuate sul protocollo MACsec analizzeremo il secondo componente principale del nostro caso di studio: Kubernetes. A partire dall'aspetto generale della piattaforma open-source, analizzandone anche il funzionamento e i componenti principali, sino alla composizione della rete e infine ai plug-in aggiuntivi che regolarizzano il flusso di rete del sistema.

3.1 Panoramica

Kubernetes è una piattaforma sviluppata da Google con l'obiettivo di creare applicazioni cloud native in una modalità piú semplificata, rendendo piú autonomo il rilascio sul container nonchè la sua gestione. Rientra infatti nella categoria degli orchestratori, ossia programmi a cui viene delegata la gestione dei container (scheduling, management, gestione dei volumi etc.). In tal modo si lascia piú spazio agli sviluppatori per comprendere e lavorare sulle potenzialità del servizio offerto senza preoccuparsi di un eventuale down-time del sistema o del rilascio stesso. All'utente è lasciata solo la configurazione del prodotto Kubernetes e la definizione dei nodi e dei container customizzati. Il funzionamento di Kubernetes prevede piú elementi fondamentali: un nodo master che gestisca tutte le operazioni, nodi worker che elaborino i dati ed eseguano i comandi, e determinati container racchiusi in pod per far girare le applicazioni virtualizzate. Riassumendo i compiti principali dello strumento sono i seguenti:

- Orchestrazione di container tra piú host;
- Ottimizzazione nell'utilizzo dell'hardware al fine di garantire il giusto rapporto di risorse;
- Gestione automatica delle applicazioni, il loro rilascio cosí come il loro riavvio o la loro replica;
- Gestione dello storage;
- Scalabilità per le applicazioni in quanto è possibile aggiungere facilmente piú applicazioni.

Per poter fornire tutti questi servizi Kubernetes si affida ad altrettanti strumenti open-source che, lavorando in compatibilità con la piattaforma, garantiscono un rapporto complementare per le potenzialità che esso offre. Ad esempio, per la gestione della sicurezza utilizza progetti come *Lightweight directory access protocol (LDAP)* [24], per la gestione delle reti si affida a progetti come *OpenvSwitch* [25] o plug-in esterni per la configurazione automatica, la manutenzione e la gestione del traffico di pacchetti fra i vari nodi e i pod.

3.2 Architettura

Kubernetes, come anticipato nel paragrafo precedente, nella sua struttura è composto principalmente da macchine lavoratrici, anche dette nodi worker, le quali eseguono applicazioni containerizzate. Ogni cluster puó avere piú nodi worker, ma un singolo nodo master, anche detto control-plane, che gestisce i singoli nodi e pod del cluster in modo da garantire disponibilità permanente e alta tolleranza ai guasti.

Il nodo di control-plane è il principale responsabile di tutte le decisioni prese nel cluster, a partire dalle richieste effettuate in entrata e in uscita, gestisce le risorse sui nodi per garantire efficienza massima. Il master è composto da cinque componenti, ognuno ha un obiettivo preciso e lavorano in sinergia per mantenere il cluster in condizioni ottimali:

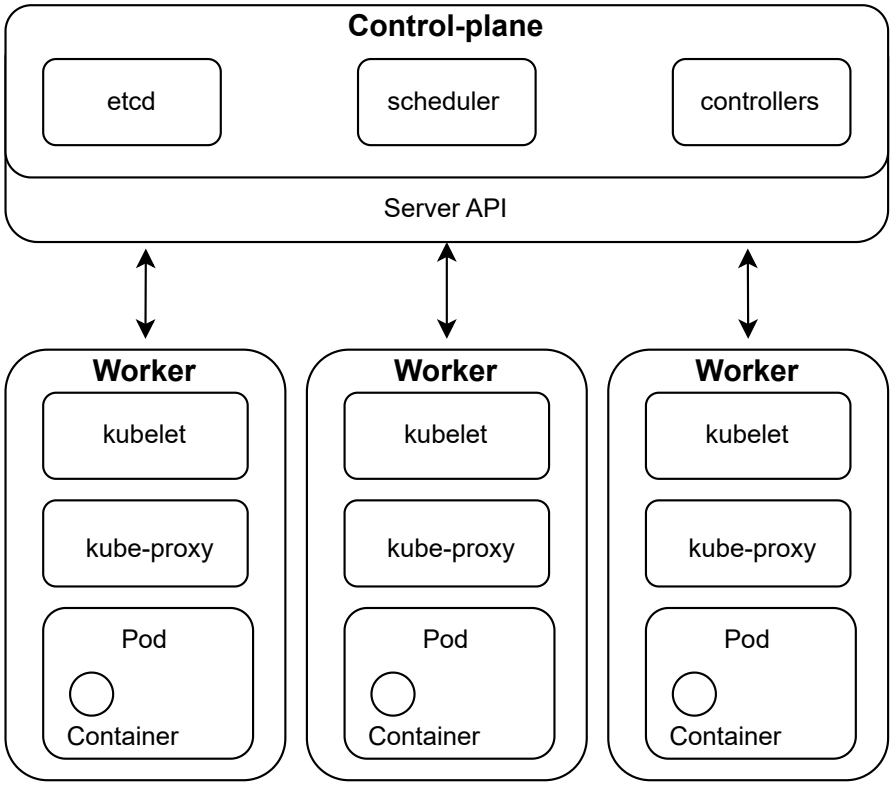


Figura 3.1: Architettura Kubernetes.

kube-apiserver è il principale access point per tutte le richieste provenienti dai client, in base a esse e al traffico in entrata istanzia le risorse necessarie per passare dallo stato attuale a quello desiderato. Il comportamento derivante sarà uno sviluppo orizzontale che garantisce una perfomance massima.

kube-scheduler assegna i vari pod ai nodi slave in base alle risorse accessibili (tra hardware, software, dati e carico di lavoro). è in grado di diffondere il carico di lavoro e memorizzare i dati di utilizzo delle risorse di ciascun nodo. Tiene traccia delle modalità di utilizzo dello stack sul cluster.

kube-controller-manager traccia l'utilizzo delle risorse e collabora con l'apiserver per raggiungere lo stato richiesto in entrata dal client. Vi sono differenti tipi di controller in base alla risorsa da tenere in osservazione. I controller principali sono: il controller dei nodi che verifica l'attività di un nodo e in base a questo istanzia le risorse, il controller dei job che crea il job specifico e vi assegna determinati pod, il controller dei namespace che identifica

il namespace da poter utilizzare, il controller director che esegue i controller per gestire nodi, endpoint, etc.

cloud-controller-manager è simile al kube-controller-manager ma si interfaccia con le risorse in cloud e ha allo stesso modo i controller differenziati in base all'elemento di controllo.

etcd è un database di chiave-valore che regola l'utilizzo delle risorse. Se questo non fosse presente si regolarizzerebbe il lavoro in base alla sessione corrente.

I nodi worker sono un insieme di macchine, denominate nodi, che eseguono applicazioni containerizzate. Ogni cluster ha almeno un nodo di lavoro. I nodi ospitano i pod che sono i componenti responsabili del carico di lavoro dell'applicazione e che raggruppano tutti i container responsabili del singolo servizio. I componenti principali sono quattro:

kubelet è un agent che viene eseguito su ogni nodo di lavoro, si connette alla finestra mobile e si occupa di creare, avviare ed eliminare i contenitori.

kube-proxy instrada il traffico ai contenitori appropriati in base all'indirizzo IP e al numero di porta della richiesta in entrata. In altre parole, il suo principale utilizzo è quello di tradurre gli indirizzi in entrata e in uscita per indirizzarli sulle porte corrette. Periodicamente questo strumento richiama l'api-server per poter aggiornare le eventuali iptables di cui si fa carico, nel caso di un aggiunta di servizio o di rimozione di tale.

pod puó essere definito come un multi livello o un gruppo di container distribuiti su un singolo nodo di lavoro o host *Docker* [26]. Di un'applicazione possono essere creati anche piú container su pod differenti che possono tranquillamente comunicare fra di loro.

Service racchiudono piú pod legandoli a determinate leggi di accesso, di solito distinte per ogni applicazione rilasciata sui container. Sono in generale sistemi di sicurezza e di organizzazione del lavoro.

3.3 Configurazione di rete

La configurazione di rete di Kubernetes pone le basi per comprendere al meglio come pod, container e service comunichino insieme. Il modello di rete che Kubernetes va ad mplementare è basato sulla semplicità e sulla consistenza.

Tale modello rispetta quattro semplici regole: *"ogni pod possiede il proprio indirizzo IP, i container all'interno dello stesso pod possono comunicare tra loro poiché condividono lo stesso indirizzamento, i pod possono comunicare con altri pod all'interno dello stesso cluster senza l'utilizzo del Network Address Translation (NAT), isolamento che consiste nel gestire i dati trasmessi e tra quali nodi attraverso delle policy di networking"* [27]. In tal caso i pod potrebbero essere visti come delle macchine singole su cui girano dei processi istanziati dal control-plane.

Un componente principale di Kubernetes, per quanto riguarda la connettività, è il `kubelet`, un supporto della rete configurato all'avvio della piattaforma, ma le sue funzionalità di base sono legate a una installazione semplice e alla semplice comunicazione fra i nodi e i pod del cluster. Pertanto, un aiuto ulteriore viene dato dai plug-in integrati detti anche Container Network Interface (CNI), di cui parleremo in seguito(3.3.1). Un altro componente, facente parte soprattutto del lato nodo woker, è il `Service` che permette di offrire il servizio all'esterno del cluster senza effettivamente identificare il quantitativo di pod o container che sono stati azionati all'interno. Di solito per accedervi viene fornita una porta o viene configurato un load balancer con un indirizzo accessibile dall'esterno. In ogni cluster è presente anche un servizio di Domain Name System (DNS), questo è rappresentato come un `coreDNS` e viene trattato come ogni altro pod, ma all'interno procede con la traduzione di un namespaces in un indirizzo raggiungibile.

Com'è possibile evincere dalla figura 3.2 una volta che il pacchetto giunge sul nodo worker destinatario, passa attraverso le iptables mediante le quali viene instradato (queste sono collegate

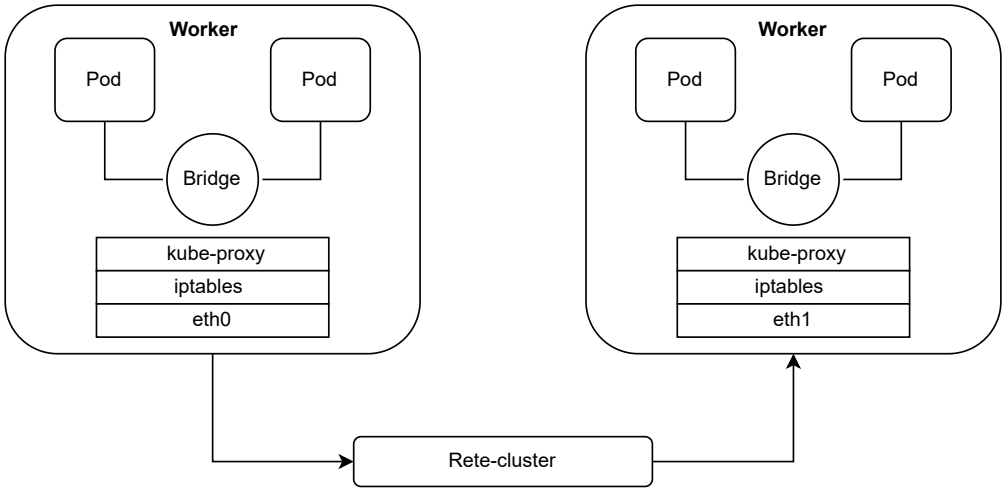


Figura 3.2: Configurazione rete Kubernetes.

al `kube-proxy`) e viene direzionato verso il pod corretto. Per la destinazione effettiva in generale non c'è una vera e propria logica, ma spesso le comunicazioni tra pod dello stesso servizio vengono inviate casualmente a uno dei pod responsabili solo della medesima applicazione, ma non uno nello specifico. Com'è possibile notare il collegamento esterno passa tramite una rete Ethernet che collega i nodi operanti e il nodo master. La maggior parte delle operazioni di rete appena descritte, cosí come la loro configurazione, vengono semplificate attraverso l'utilizzo di un CNI.

3.3.1 Container Network Interface

Uno strumento molto utile per la configurazione automatica della rete in Kubernetes è rappresentato dai plug-in che è possibile installare a configurazione del cluster e che provvedono alla creazione delle varie sottoreti, all'assegnazione degli indirizzi IP, nonchè alla gestione di tutto il flusso del pacchetto di comunicazione all'interno del cluster stesso, tra master e worker, tra worker differenti, oppure la semplice comunicazione con l'esterno, tutto ció attraverso l'utilizzo di librerie scritte in `Go`.

In Kubernetes il CNI è integrato con kubelet per configurare la rete underlay e overlay tra i pod. La rete underlay è una rete fisica utilizzata per la comunicazione costante della propria presenza ai nodi vicini, mentre quella overlay è una rete, di solito VxLAN, utilizzata per il traffico dati relativo al servizio offerto. I principali plug-in di interesse sulla scena attuale sono:

- Flannel;
- Calico;
- Canal.

Flannel

Sviluppato da Coreos, Flannel [28] risulta il CNI piú leggero dal punto di vista computazionale e con l'installazione e la configurazione piú facile rispetto agli altri plug-in, in quanto necessita di un singolo file binario, attraverso il quale è possibile indicare tutte le info aggiuntive o le opzioni da utilizzare in Flannel, come ad esempio la rete su cui passare i dati o il sistema di comunicazione tra una macchina e l'altra. A livello di rete Flannel configura tutta la connessione base tra container e pod attraverso reti overlay assegnate a ogni nodo del cluster di modo da poter associare un IP a ogni elemento interno tra cui pod o daemon.

L'agent principale di flannel è `flannelID`, un daemon con il compito di verificare che tutta l'infrastruttura non cada o non si spenga. Un altro componente è il `flannel.1`, un proxy che prende il posto del kube-proxy naturale e in tal modo direziona il traffico con l'esterno stabilendo anche regole di routing. Il database utilizzato è quello di `etcd` che supporta un controllo constante attraverso chiamate Application programming interface (API). Uno degli elementi fondamentali di Flannel è il bridge `cni0` utilizzato per allocare gli indirizzi IP dei pod interni. In generale funge principalmente da redirezionatore del traffico dall'esterno all'interno, e viceversa, attraverso le iptables configurate in automatico. Per la comunicazione tra nodi worker (come visualizzato in precedenza basata su LAN) Flannel utilizza una VxLAN per l'incapsulamento e l'invio dei pacchetti, ma sono possibili anche alternative per una comunicazione sicura tra cui WireGuard e IPsec, nessuna delle quali garantisce protezione sotto il livello 3. Uno dei vantaggi di Flannel è che una volta installato sul nodo master automaticamente la configurazione viene passata ai nodi worker.

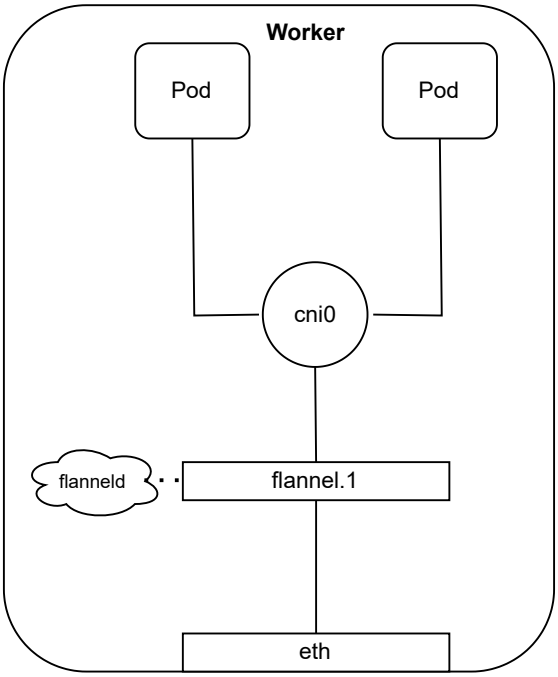


Figura 3.3: Configurazione rete Flannel.

Nella versione *1.21* di Kubernetes, solo per il plug-in Flannel, era presente un sistema di sicurezza dei pod chiamato *PodSecurityPolicy*. Il PodSecurityPolicy è un controller che definisce i requisiti di sicurezza che un pod, appena inserito nel cluster, deve avere. Vengono settate delle regole Role-Based Access Control (RBAC) e a inserimento del pod vengono consultate tali regole decidendone cosí l'ammissione o no di tale pod al cluster. Dalla versione `1.25` questa policy di sicurezza è stata disabilitata [29], per motivi di usabilità (non era facilmente comprensibile l'utilizzo) e di performance, dato l'alto costo computazionale. Attualmente, per rimpiazzarlo, è stato creato il *Pod Security Admission Controller* che svolge funzioni simili a quelle del PodSecurityPolicy, l'unica differenza è che i pod devono essere rilasciati con un namespace privileged, mentre di default vanno in quello `kube-flannel`.

Il PodSecurityPolicy citato in precedenza è differente dal PodSecurityContext, altro sistema di sicurezza, che definisce semplicemente i campi di sicurezza da affidare a ciascun pod (ad es. se puó essere avviato come utente normale o come root). Questo protocollo definisce semplicemente i campi, ma non li applica.

Calico

Sviluppato da Tigera, Calico [30] è un CNI piú incentrato sulla performance, la flessibilità e la potenza. Il sistema di amministrazione di Calico è migliore rispetto a quello di Flannel e inoltre l'installazione prevede un file manifest.

Calico è rappresentato da un DaemonSet installato su ogni host, e presenta 3 componenti principali: `Felix, BIRD e confd`, dove Felix è l'agent principale, BIRD il controller dello stack nonchè gestore delle tabelle di routing per aggiornarle periodicamente e confd il gestore e manutentore di file in locale. Calico principalmente sfrutta il protocollo *Border Gateway Protocol (BGP)* [31], per il settaggio delle tabelle di routing, invece di una configurazione statica. In generale questo riduce i tempi di gestione cosí come l'efficienza, infatti non si perde tempo nell'incapsulamento del pacchetto per poter essere passato al livello successivo. E' possibile utilizzare anche reti overlay tramite VxLAN e IP-in-IP. Attraverso calico fare debugging e tracing è molto piú semplice poiché ogni cosa è controllata e ci sono numerosi strumenti che posso approfondire le azioni eseguite. Rispetto a Flannel l'installazione di Calico deve avvenire su ogni nodo worker, non viene passata in automatico dal nodo master.

Calico puó sfruttare regole di rete per il traffico in ingresso e in uscita, e questo garantisce una maggiore sicurezza in quanto puó bloccare, consentire e gestire i dati del traffico che passa, inoltre puó definire alcuni campi di default come porte, protocolli, versione IP, etc. Le policy di sicurezza vengono applicate per gli accessi agli endpoint singoli (gli stessi pod). Nella figura 3.1 viene presentato un esempio di configurazione del file yaml per l'installazione di Calico.

```
apiVersion: projectcalico.org/v3
kind: NetworkPolicy
metadata:
    name: allow-tcp-6379
    namespace: production
```

Listing 3.1: File yaml di configurazione per Calico.

Questa configurazione non viene dettata da kubectl ma dal `calicoctl`, simile in alcune cose a kubectl ma con funzionalità aggiuntive e specifiche per Calico.

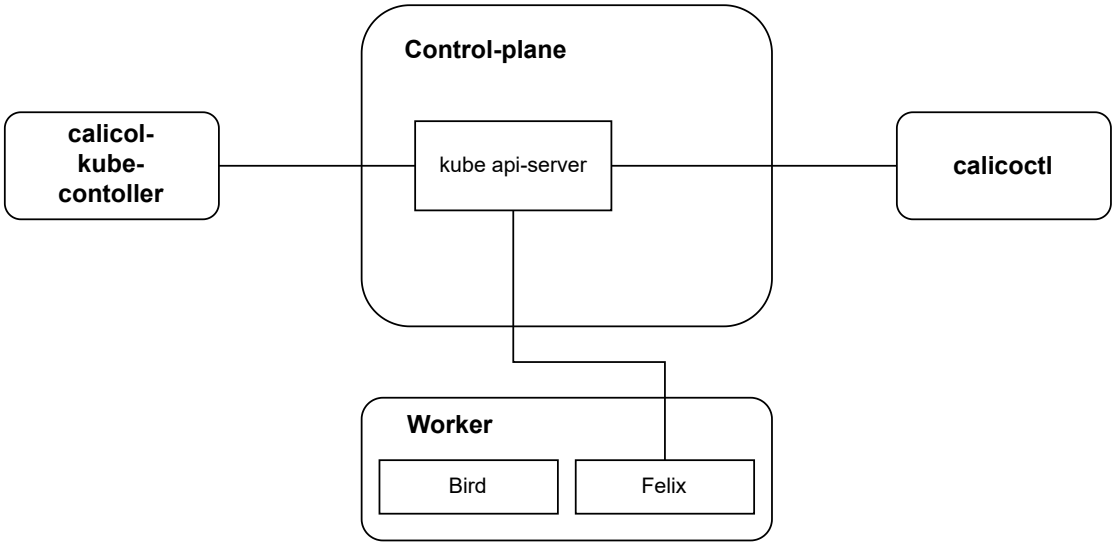


Figura 3.4: Configurazione rete Calico.

Canal

Canal [32] è il mix di configurazione tra Flannel e Calico. Nel dettaglio integra la parte di overlay di Flannel con le regole di rete e i componenti di Calico. L'unico problema è che l'integrazione

tra i due CNI ancora non è sviluppata fino ai minimi dettagli. Canal puó essere applicato tramite due file manifest.

3.4 Security

In termini di sicurezza, Kubernetes, nonostante sia uno degli strumenti principali utilizzati per l'orchestrazione dei container, presenta ancora numerose criticità. Essendo una piattaforma composta da piú servizi e gestita da piú componenti, il rischio di attacchi dovuti a una scarsa protezione è alto. A partire dalla versione utilizzata di Kubernetes che sia on-premise o su cloud le cause di attacco sono molteplici. Per la versione on-premise ovviamente tutto, dagli aggiornamenti alle configurazioni, dovrà essere eseguito manualmente, e cosí il controllo della sicurezza, mentre dal punto di vista del cloud la sicurezza risulta piú articolata, ma gli automatismi rendono le operazioni piú veloci e piú facili da gestire. Di seguito un'analisi piú dettagliata delle principali cause di rischio alla sicurezza in questo ambiente.

Le versioni della piattaforma rappresentano una delle potenziali fonti di rischio, quest'ultime vengono aggiornate molto spesso, pertanto un mancato aggiornamento sulla macchina utilizzata potrebbe portare a un bug del sistema non ancora sanato e che garantirebbe un ottimo appoggio per l'intrusione. Stesso ragionamento per quanto riguarda le macchine su cui girano tutti i nodi di Kubernetes, per cui bisognerebbe seguire l'evoluzione del sistema operativo, cosí come di tutti i servizi che lo compongono per una giusta accortezza.

Il controllo degli accessi al cluster è un'altra problematica rilevante all'interno del cluster. Uno dei sistemi preventivi per limitare l'accesso di un utente è il RBAC, con il quale è possibile controllare le autorizzazioni in base al ruolo. Gli accessi potrebbero essere legati all'intero cluster, cosí come a un singolo nodo o a un singolo servizio, ed essendo attualmente privo di un qualsiasi controllo è facile che un malintenzionato possa accedere e modificare configurazioni o semplicemente rubare del traffico di comunicazione tra le macchine, o tra i servizi e l'utente finale.

Come detto in precedenza, i componenti di Kubernetes da proteggere sono molteplici e uno di questi è lo strato piú interno composto dai container. Le immagini utilizzate dai container tramite virtualizzazione Docker sono altamente vulnerabili poiché la versione scaricata è sempre presa dal tag *latest* pertanto viene aggiornata costantemente, ma la presenza di bug all'interno di questa potrebbe condurre ad accessi indesiderati o a malfunzionamenti, pertanto una soluzione sarebbe quella di selezionare sempre una versione specifica ritenuta sicura.

3.4.1 Network Security

Dal punto di vista della rete invece la situazione è piú complessa poiché, come detto in precedenza, esiste una problematica di autenticazione non consolidata che rimane la causa principale degli attacchi alla rete. Una tra le tante soluzioni apportate è la protezione dei vari collegamenti tramite tecnologie come IPsec, WireGuard e altre, sui collegamenti master-worker, in collegamenti container-container o pod con pod di altri container, ma la protezione sussiste su livello 3 di rete del modello ISO/OSI e non sul livello 2 dei link, e questa casistica è di interesse per il nostro caso di studio. La sicurezza del canale viene gestita in due modalità: o tramite l'utilizzo dei CNI, i quali ottengono in input la configurazione da dover utilizzare nella comunicazione, o manualmente cambiando i settaggi di rete di base delle connessioni tra nodi.

Ad esempio, come visualizzato nella figura 3.5, una delle possibilità risiede nell'utilizzo di WireGuard. Attraverso tale strumento è possibile configurare connessioni VPN in uscita dai nodi worker, per collegamenti diretti, oppure tramite router. Come esemplificato nel paragrafo 2.9.2, la protezione in tal modo avviene solo sul livello 3. Attraverso CNI come Flannel o Calico l'installazione di Wireguard si limita al semplice inserimento del protocollo nella configurazione del backend, ma allo stesso modo è possibile anche configurarlo manualmente inserendolo come DaemonSet di Kubernetes e lasciando all'utente la responsabilità di proteggere e instaurare tutte le connessioni. Un altro strumento utilizzato è quello di IPsec per cui peró è piú complicata

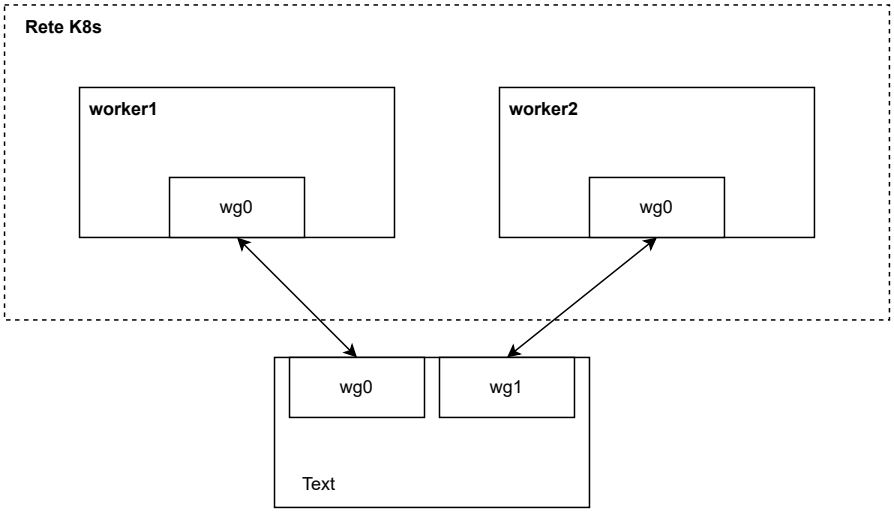


Figura 3.5: Canale di comunicazione WireGuard.

l'installazione. Non è ancora possibile impostarlo tramite il CNI Flannel in quanto attualmente è in via di sperimentazione [33]. Per quanto riguarda Calico la configurazione avviene tramite il suo file manifest. Anche per IPsec è disponibile l'installazione tramite DaemonSet.

Capitolo 4

Analisi delle tecnologie e design della soluzione

Al fine di comprendere al meglio il procedimento e le funzionalità dell'attuazione del protocollo MACsec sul collegamento fisico presente fra i nodi di un cluster Kubernetes, è necessario principalmente tenere conto di determinate informazioni:

- Possibili strumenti attualmente disponibili per l'installazione e la configurazione di MACsec e motivazioni che hanno portato alla scelta di tale protocollo;
- Eventuali distribuzioni della piattaforma Kubernetes, nonchè sistema di configurazione di rete specifico con particolare attenzione sui CNI disponibili e quelli presi in considerazione;
- Architettura del cluster utilizzata ai fini dell'implementazione.

Per l'installazione del cluster e la configurazione del canale protetto è stato predisposto l'utilizzo dello strumento Ansible. Attraverso di questo è possibile con un solo comando instaurare l'intera infrastruttura. Nel primo punto analizzeremo due librerie Linux che forniscono il protocollo: *wpa_supplicant* e *iproute2*. Nella parte di implementazione verranno utilizzate entrambe per poter valutare eventuali differenze di prestazioni e di utilizzabilità. In seguito, verranno analizzate piú nel dettaglio varie versioni di Kubernetes quali *K3s* [34] e *K8s* [35] valutandone pro e contro ai fini del caso di studio. Infine, verrà presentata l'architettura cluster, oggetto dei test, per considerare tutti gli aspetti, dal newtorking all'implementazione delle macchine singole, dalla configurazione della distribuzione linux sino ai collegamenti effettuati.

4.1 MACsec

L'implementazione dello strumento attraverso l'utilizzo di sistemi Linux puó essere eseguita tramite due librerie, nello specifico *wpa_supplicant* [36] e *iproute2* [37]. Di seguito analizzeremo le modalità di utilizzo e le funzionalità intrinseche.

4.1.1 wpa_supplicant

Il protocollo di MACsec non fornisce come servizio la gestione dello scambio di chiavi per la sicurezza dei dati, nè di conseguenza l'autenticazione degli utenti nella rete LAN. Pertanto, avvalendosi di un ulteriore protocollo esterno IEEE 802.1x riesce a soddisfare i requisiti mancanti. Tale protocollo, insieme con le funzionalità di base di MACsec, puó essere implementato attraverso un daemon nominato wpa_supplicant, eseguito completamente in background e diveniente parte

dei componenti wireless. Lo strumento dispone anche di un'interfaccia grafica `wpa_cli` che funge da front-end in text del programma.

Come si evince dal nome del programma stesso, wpa_supplicant è l'implementazione di un componente del Wi-Fi protected access (WPA)(protocollo di sicurezza nelle reti wireless), precisamente quello che viene eseguito da parte del client, wpa authenticator, e che gestisce la negoziazione delle chiavi. Allo stesso tempo gestisce su reti wired l'autenticazione EAP con l'utilizzo di un server di autenticazione fungendo da EAP Supplicant.

Solo dalla versione `2.6` wpa_supplicant supporta il driver MACsec per Linux che è possibile configurare e installare tramite apposito file di configurazione passato tramite linea di comando.

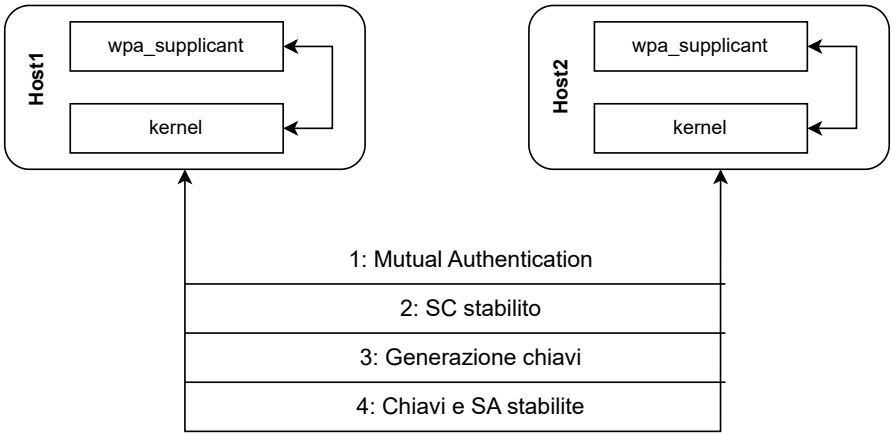


Figura 4.1: Flow di dati wpa_supplicant-kernel.

Per installare MACsec tramite il daemon sono disponibili due opzioni:

- Installazione tramite file di configurazione con CAK/CKN già condivise. In tal modo verranno rilevati i partecipanti alla medesima CA con cui poter scambiare la SAK e creare un canale SC. Tutto ció verrà poi eseguito dalla KayY.
- La soluzione alternativa riguarda l'ottenimento del CAK/CKN tramite un'autenticazione EAPOL ottenuta attraverso l'inserimento di un server RADIUS esterno, anche in questo caso dovrà essere passato un file di configurazione contente i dati di autenticazione di ogni peer.

Come è possibile evincere dalla figura 4.1, i passi eseguiti per lo scambio delle chiavi e dunque per settare un canale protetto su livello 2 riguardano due attori, wpa_supplicant e il kernel. Lo strumento permette la mutua autenticazione tramite lo scambio di chiavi. MKA utilizza i pacchetti EAPOL per trasmettere MKPDU con sicurezza. In questo modo vengono configurate e settate le SA tra i peer. Nel frattempo, wpa_supplicant traduce tutte le impostazioni dell'MKA al kernel per l'implementazione di MACsec. Il kernel in seguito configura un'interfaccia di rete apposita `masecX` che si sovrappone all'interfaccia di rete preesistente per il solo scambio di pacchetti cifrati. Questo processo si ripete continuativamente, sino a quando le chiavi non risultano scadute e pertanto verranno nuovamente generate e reintrodotte all'interno del canale di comunicazione ripetendo il processo da capo. Nella figura 4.1 viene illustrato un esempio di file di configurazione per una connessione MACsec con coppia di valori CAK/CKN già condivisa. Nella figura 4.2 viene illustrata la configurazione con settaggio delle chiavi delegato a un server di autenticazione esterno, come *FreeRADIUS* [38].

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
ap_scan=0
eapol_version=3
```

```
network={
    key_mgmt=NONE
    eapol_flags=0
    macsec_policy=1
    macsec_integ_only=0
    mka_cak=<32 bit> #stringa da 32 bit
    mka_ckn=<64 bit> #stringa da 64 bit
}
```

Listing 4.1: file di configurazione per MACsec con wpa_supplicant.

```
ap_scan=0
fast_reauth=0
eapol_version=3
network={
    key_mgmt=IEEE8021X
    eapol_flags=0
    eap=TLS
    ca_cert="/path/to/ca/pem"
    client_cert="/path/to/client/pem"
    private_key="/path/to/client/key"
    private_key_passwd="my_super_secret_password"
    macsec_policy=1
}
```

Listing 4.2: file di configurazione con server di autenticazione esterno.

Da notare che lo strumento permette un solo canale per volta sull'interfaccia di rete selezionata. Ad esempio, con tre macchine collegate alla medesima rete sarà possibile configurare solo due di loro per l'utilizzo di MACsec. Nel caso in cui si volessero creare canali di comunicazione protetti per ogni connessione fra le tre macchine, dovrà essere predisposto uno switch nel mezzo di modo che le connessioni sicure saranno quelle tra la singola macchina e lo switch, ma facendo diventare lo switch stesso un possibile punto di attacco e un collo di bottiglia. Un attacco di DoS alla macchina centrale creerebbe un disservizio per cui le altre macchine non avrebbero piú accesso alla rete esterna nè avrebbero alcun contatto con gli altri nodi del cluster; anche un'intrusione all'interno della medesima macchina permetterebbe a un utente malintenzionato di leggere il traffico che sulla macchina arriva in quanto tutto il traffico cifrato, proveniente da una singola rete, verrebbe prima decifrato, poi letto e in seguito direzionato verso la macchina corretta nuovamente cifrato.

4.1.2 iproute2

`iproute2` è una collezione di strumenti per linea di comando utilizzati per la gestione del traffico sia IPv4 che IPv6. Gli obiettivi principali di `iproute2` sono: la configurazione di interfacce, la gestione delle tabelle di routing e *ARP* [39] e la gestione del flusso di dati. In origine le azioni elencate in precedenza erano gestite dalla libreria `net-tools` ora sostituita per mancanza di aggiornamenti, anche se attualmente sempre in uso per la gestione delle configurazioni di rete.

Oltre ai vari servizi offerti dalla libreria `net-tools` e riproposti in `iproute2` in modalità aggiornata, sono state aggiunte nuove configurazioni e nuovi elementi che hanno integrato soluzioni di comunicazione preesistenti, tra cui il nuovo strumento di `ip macsec`. L'operazione da eseguire preventivamente a tutta la configurazione dovrà essere l'attivazione del modulo aggiuntivo MACsec dal kernel. In seguito, attraverso il comando `ip macsec` è possibile effettuare la configurazione dei canali SC e le loro SA. E' possibile in tal modo configurarli manualmente sfruttando un'azione preesistente sulla libreria `ip link add` e utilizzando come type della connessione MACsec.

```
sudo ip link add link eth8 macsec0 type macsec
```

La configurazione dovrà avvenire su entrambi i peer, andando a configurare destinatari e mittenti del canale di comunicazione. In tal modo è possibile solamente configurare una SAK già condivisa senza l'utilizzo di un rekeying (che potrebbe essere effettuato manualmente rompendo la connessione e ristabilendola). Le opzioni da poter inserire sono molteplici, come la presenza di cifratura dei dati, o la scelta dell'algoritmo da utilizzare per tal scopo. Cosí come la scelta della numerazione del pacchetto, o gli id di identificazione. Attraverso le chiavi i peer sono autenticati ed esse rappresentano le SAK utilizzate successivamente per la connessione, queste vengono già condivise manualmente tra i peer partecipanti alla comunicazione.

4.2 Kubernetes

Il secondo strumento analizzato in questo capitolo, nonchè parte centrale del nostro caso di studio, è Kubernetes. La sua installazione e configurazione puó essere eseguita secondo differenti distribuzioni in base alle necessità operative. Tra le distribuzioni piú note ritroviamo:

- K3s;
- K8s (tool kubeadm).

4.2.1 K8s

La prima distribuzione K8s, è la piú complicata e completa per la creazione di un cluster Kubernetes. La sua installazione è facilitata tramite l'utilizzo di un tool chiamato kubeadm [40]. Il suo obiettivo primario è quello di ottenere massime capacità con un cluster completo di tutti i componenti e di raggiungere una velocità di esecuzione massima. Di default kubeadm si occupa dell'inizializzazione del cluster, semplificando notevolmente la configurazione delle macchine worker. Con tali specifiche è adatto soprattutto per l'installazione in locale, nonostante richieda comunque macchine a prestazioni elevate quali:

- RAM: 2 GB
- CPU: 2 (solo per il nodo master)

L'installazione dovrà essere ripetuta su ogni singolo nodo per istanziare ogni componente, dal kubelet al kubectl, sino al container utilizzato per il rilascio delle applicazioni (cri-o, Docker o altri).
Per un eventuale aggiornamento il procedimento dovrà essere ripetuto da capo selezionando la versione piú recente. kubeadm presenta una command-line interface (CLI) piú intuitiva che permette un approccio piú immediato alle procedure di installazione.

Per quanto riguarda la sicurezza, essendo la distribuzione piú simile alla versione ufficiale di Kubernetes, le superfici di attacco, già illustrate nel paragrafo 3.4, sono ridotte al minimo ai fini di permettere un'esecuzione controllata e sicura.

Attraverso questo strumento è possibile testare ogni aspetto del cluster, data la possibilità di modificare i suoi componenti adattandoli al nostro test case. Per tale motivo e ai fini di simulare quanto piú possibile un cluster Kubernetes è stato scelto K8s con l'utilizzo del tool kubeadm.

4.2.2 K3s

Una tra le altre distribuzioni presa in considerazione è stata k3s. Tale distribuzione risulta leggera e viene distribuita tramite un singolo file binario, pesante soli 50 MB. L'installazione è veloce ed estensibile a tutti i nodi senza una vera e propria configurazione preventiva. Inoltre, un possibile aggiornamento puó essere eseguito semplicemente rilanciando il programma. I requisiti minimi per ogni macchina sono:

Pro	Contro
• Architettura completa • usa docker come container	• un nodo alla volta puó essere il master • difficile da installare ed eseguire

Tabella 4.1: pro e contro di kubeadm

- Linux 3.10+
- RAM: 512 MB
- Memoria: 200 MB

Le esigenze sono differenti rispetto alla distribuzione K8s con kubeadm, ma i componenti installati sono quelli standard. Non presenta add-ons e non è possibile apportare modifiche ai suoi componenti. Un'altra problematica di tale strumento è legata al database di cui non dispone internamente, ma presenta una compatibilità con *SQLlite3* [41] che potrebbe essere aggiunto all'esterno del cluster. Infine, K3s è utilizzato maggiormente per cluster piccoli con un singolo nodo worker, tale strumento infatti è utilizzato ai fini di test pratici e non per un vero e proprio utilizzo delle applicazioni rilasciate in quanto queste ultime richiedono prestazioni piú elevate.

In termini di sicurezza, essendo tutto su un singolo file binario con il minimo essenziale e con poche dipendenze da altri componenti, presenta una superficie di attacco minimo e per tale ragione piú controllata.

Pro	Contro
• Facile da rilasciare • versione piú leggera	• non ha molte opzioni in piú • non puó usare docker come container

Tabella 4.2: pro e contro di K3s

4.2.3 Container network interface

Il nostro interesse volge ora alla configurazione di rete del cluster. Questa viene eseguita in automatico dai CNI, pertanto il nostro obiettivo sarà quello di studiarli ai fini di integrare al meglio il protocollo di sicurezza con la rete esistente. Il CNI scelto è stato quello di Flannel, innanzitutto per l'installazione piú semplice e basilare attraverso un singolo file binario. Inoltre, la sua configurazione, una volta attuata sul nodo master viene estesa ai nodi worker che si collegano al cluster senza dunque bisogno di reimpostarla. La motivazione principale riguarda comunque la connessione tra i nodi worker, tra i quali Flannel instaura una comunicazione tramite VxLAN di default (o una scelta dall'utente), la quale ha il vantaggio di essere piú configurabile ai fini del nostro operato rispetto a quella di Calico il quale sfrutta maggiormente il protocollo BGP per sfruttare al massimo i servizi che offre. Tutte le opzioni aggiuntive di Calico e dunque di Canal non sono necessarie per il nostro test case, inoltre quest'ultimi necessitano dell'installazione su ogni singolo nodo, non solo sulla macchina master e prevedono una configurazione tramite un file manifest per Calico e due file manifest per Canal. L'installazione di Flannel puó avvenire in due modalità: o manualmente prima dell'installazione del cluster, o tramite un singolo comando che recupera l'ultima versione del CNI da una repository git (disponibile per versioni kubernetes dalla 1.17+). Nello specifico viene recuperato il file `kube-flannel.yml` dove è possibile modificare, in base alle necessità, alcune delle opzioni presenti, quali connessione del backend, interfaccia di rete

selezionata, range di indirizzi IP da assegnare all'interno delle sottoreti worker etc. All'interno del file è presente anche il nome delle immagini da cui andare a prendere tutti i pacchetti dello strumento, nonchè i relativi componenti come bridge e proxy, operanti tutti su container Docker. Una volta avviato, Flannel risulterà come uno dei pod presenti sul cluster Kubernetes in cui è stato installato.

La connessione di nostro interesse con architettura Flannel sarà quella evidenziata in rosso nella figura 4.2, predisposto dal plug-in come VxLAN (a seconda delle opzioni riportare sul file di configurazione). Nella connessione tra nodi worker questo canale prevede il passaggio di dati fra i pod di uno stesso servizio o di dati a fini di presenza attiva all'interno della CA.

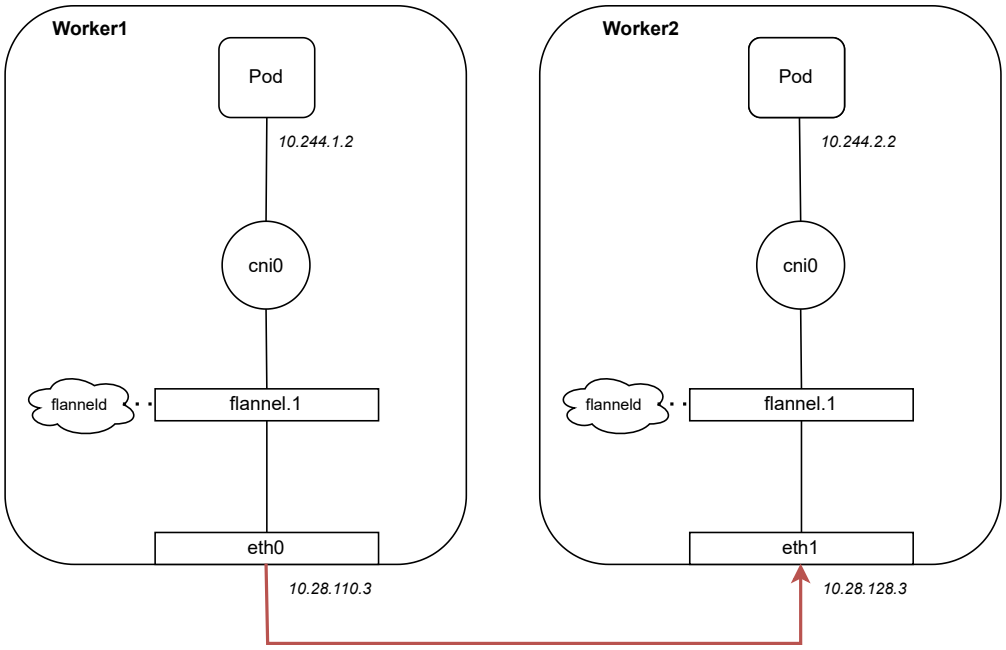


Figura 4.2: Connessione worker-worker con Flannel.

4.3 Architettura del cluster

L'obiettivo del nostro studio è quello di ottenere una connessione sicura sul livello 2 nel collegamento fra i nodi del cluster Kubernetes. Per ottenere tale configurazione è necessario costruire un cluster su cui far girare i nodi worker e il nodo master della piattaforma. Le configurazioni instaurate sono riferite a due scenari differenti:

- cluster configurato con `wpa_supplicant`;
- cluster configurato con `iproute2`.

Nel primo scenario 4.3 sono state predisposte quattro macchine virtualizzate attraverso Oracle VirtualBox con distribuzioni Linux Ubuntu 20.04.5 (la versione selezionata è necessaria ai fini di compatibilità con l'installazione della versione di Kubernetes attraverso kubeadm, spiegato in seguito).

Nella configurazione di rete verranno utilizzate per ogni macchina una rete host che le collega singolarmente alla macchina centrale, su cui invece verranno predisposte 3 reti host collegate alle singole macchine, piú un'unica rete uscente sul web. Dalla figura 4.3 è possibile notare che sulle macchine operatrici non è disponibile una rete internet, ció è dovuto dal fatto che la configurazione permette l'uscita sulla rete internet esterna tramite il nodo centrale. Solo sulle reti host verrà

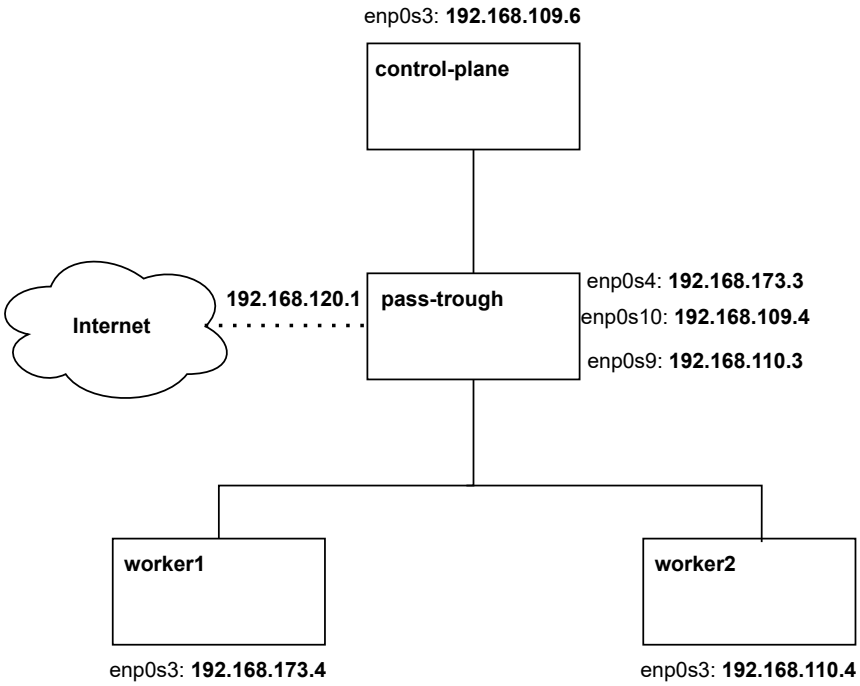


Figura 4.3: architettura con wpa_supplicant.

applicato il protocollo di sicurezza MACsec. I canali protetti saranno in tal modo sia quelli worker-worker sia quelli master-worker. I principali pacchetti inoltrati nel collegamento master-worker sono quelli derivanti dall'api-server il quale invia comunicazioni continue di aggiornamento sui servizi rilasciati o sulle basi dati utilizzate, nonchè i pacchetti che comunicano la conferma di appartenenza alla CA (per tener traccia dei nodi presenti nell'architettura). Mentre nel collegamento worker-worker i principali pacchetti riguardano i dati trasmessi tra un servizio rilasciato su un pod di un nodo worker e il medesimo servizio rilasciato sul secondo worker. Tale architettura è scalabile in quanto è possibile aggiungere nodi fino a quando si necessita, l'unico vero punto di attenzione è l'aggiunta e la configurazione di una rete host aggiuntiva sulla macchina di pass-trough e sulla nuova macchina aggiunta. La macchina pass-through simulerà uno switch con porte non abilitate a MACsec poiché decifrerà e ricifrerà il pacchetto. Pertanto, per tale architettura verrà utilizza la distribuzione di Kubernetes `K8s` con tool `kubeadm` e la libreria `wpa_supplicant` per il protocollo MACsec. La macchina centrale verrà configurata per effettuare correttamente tutto il forwarding dei pacchetti presso i corretti destinatari, nonchè il redirezionamento di tutti i dati destinati al web e relativa risposta. La sua seconda funzione sarà quella di stabilire correttamente un collegamento protetto con MACsec con ognuna delle macchine singolarmente. Il frame in tal modo verrà cifrato una volta uscito dal nodo del cluster e decifrato nella macchina centrale, per poi ripetere la medesima operazione per essere inoltrato al nodo corretto. La macchina centrale puó anche inoltrare il traffico verso un nodo privo di tecnologie MACsec. Un esempio illustrato riguarda la figura 2.6 in cui il processo è il medesimo della macchina pass-trough.

Nel secondo scenario, in riferimento allo schema 4.4, le macchine utilizzate sono tre: due nodi worker e un nodo master, rilasciati sempre su macchine Linux Ubuntu 20.04.5. In tal caso la libreria utilizzata per la configurazione di MACsec è `iproute2` pertanto il collegamento puó essere eseguito fra piú macchine sulla medesima rete andando a configurare solo piú SA differenti. Com'è possibile evincere dalla figura 4.4, le interfacce di rete utilizzate sono due e rappresentano una il collegamento a internet e una il collegamento con tutti gli altri host e su cui andremo ad instaurare il protocollo. L'utilizzo di internet non è mandatorio, in quanto è possibile scaricare tutti i pacchetti necessari alla configurazione e utilizzarli offline inserendoli nelle macchine, solo che per il nostro caso di studio è stato implementato in tal modo per ottenere un'installazione

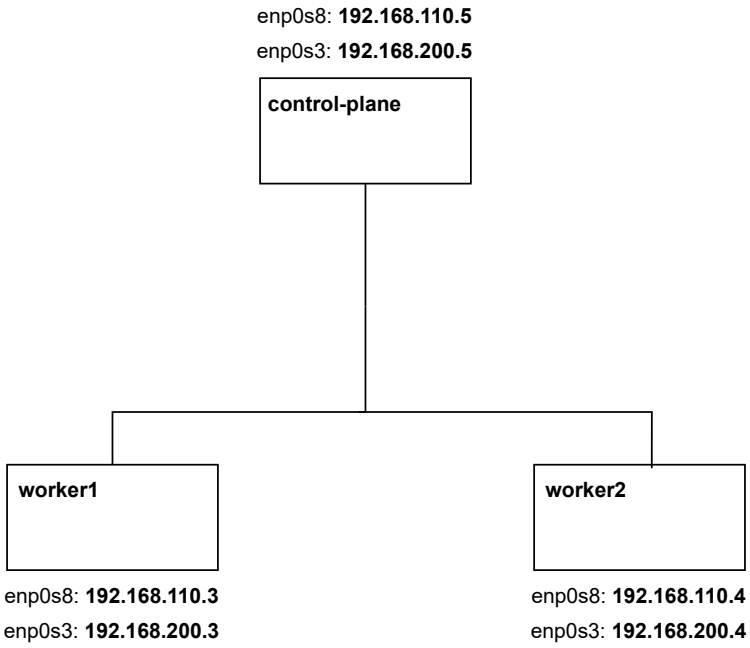


Figura 4.4: architettura con iproute2.

piú veloce. Attraverso questa architettura non è possibile connettere al cluster una macchina che non disponga del protocollo MACsec in quanto il collegamento che avviene su una singola interfaccia di rete, una volta che viene protetta per le altre, di una macchina aggiuntiva priva di protezione non riuscirà a comunicare in quanto non sarà in grado di decifrare i pacchetti. Un esempio specifico di trasmissione del pacchetto tra un nodo e l'altro è visionabile alla figura 2.5.

In entrambi i casi la situazione a livello di pacchetto sarà quella illustrata nella figura 4.5. Com'è possibile evincere, al di sopra della rete utilizzata per il trasporto dei dati viene creata un'interfaccia di rete supplementare denominata come in figura *macsecX* dove avverrà tutto il passaggio dei pacchetti, in tal senso è possibile notare all'interno della rete macsec pacchetti di connessione a piú alto livello, mentre sull'interfaccia originaria sarà possibile vedere quelli di livello 2 per cui comparirà la tipologia `MACSEC`.

Figura 4.5: struttura flow pacchetti.

Capitolo 5

Implementazione

Per l'implementazione e la riproduzione del laboratorio creato, ai fini di testare l'inserimento del protocollo MACsec nel canale di comunicazione fra nodi worker di un cluster Kubernetes, di seguito verranno illustrate le configurazioni e le installazioni necessarie. Per approfondire i comandi relativi ai passaggi eseguiti riferirsi all'appendice B. Inoltre verrá presentato il tool Ansible utilizzato ai fini di automatizzare le configurazioni e installazioni all'interno delle macchine descritte in precedenza.

5.1 Ansible

Ansible è uno strumento in grado di automatizzare tutti i processi di configurazione IT. Le risorse, il codice, ciclo di vita e modifiche vengono gestite da una singola piattaforma tramite inventario, playbook e ruoli. L'obiettivo di Ansible è quello di raggiungere per un determinato server, storage o client destinatario lo stato desiderato per il sistema semplicemente portanto il dispositivo attraverso stati determinati da opportuni task di lavoro.

Il primo vantaggio di Ansible deriva da un'automazione semplificata, per cui la medesima configurazione puó essere riprodotta su piú macchine o semplicemente numerosi passaggi di installazione o impostazioni varie da effettuare con un solo comando vengono eseguite sulla macchina destinataria. Le istruzioni impartite ad ansible vengono elaborato in maniera semplificata tramite file yaml, facili da leggere e interpretare nonché da scrivere e impostare. Attraverso questi ultimi è possibile fornire al tool tutte le opzioni varie per gestire una determinata azione da compiere. Tali file prendono il nome di *playbook*. Le azioni all'interno del playbook vengono espresse sotto il tag task ed elaborate secondo determinate opzioni aggiuntive inserite: è possibile attendere il completamento dell'operazione o eseguire controlli aggiuntivi o stampare messaggi di debug e altri tag sono ancora disponibili.

All'interno dei playbook è anche possibile specificare i gruppi su cui eseguire tali task attraverso il tag hosts che seleziona le macchine destinatarie. Quest'ultimo verrá popolato tramite inventario di Ansible; l'inventario corrisponde ad una lista di host organizzati secondo gruppi, in moda da poter eseguire la medesima azione su piú host contemporaneamente. Un esempio è visualizzabile in figura 5.1. Gli host possono essere anche disposti singolarmente o ripetuti su piú gruppi. L'inventario è giá presente all'interno del path `/etc/ansible/hosts`, ma è possibile passarne uno nuovo tramite linea di comando con l'opzione `-i`.

```
#[] per indicare i gruppi
[master]
    master01
[worker]
    worker01
    worker02
```

Listing 5.1: esempio di configurazione file hosts su ansible

Nel nostro caso di studio i file playbook creati sono quattro: ansible_kube.yml, createmaster.yml, createworker.yml e createwa.yml. Il loro utilizzo e funzionamento verrà illustrato nella sezione A.

5.2 Configurazione cluster

La versione Linux scelta per installazione del cluster è `Ubuntu 20.04.5 LTS` reperibile direttamente dal sito della distribuzione. La scelta è il risultato di una valutazione di compatibilità con la versione di Kubernetes adottata. Per un corretto funzionamento della piattaforma nella distribuzione K8s con tool kubeadm, risultano piú appropriate le versioni in long term support inferiori alla 21.04, mentre sulle altre il kube-proxy non viene eseguito correttamente in quanto la comunicazione tra quest'ultimo e il servizio di containerizzazione (cri-o, Docker o altri) si ferma in errori frequenti di *CrashLoopbackOff*. Lo stesso vale per versioni kernel `5.12.2.arch1-1` e `5.10.35-1-lts` dove, allo stesso modo, lo strumento non funziona. Nel laboratorio invece, verrà utilizzata la versione `5.4.0-148-generic`. L'installazione in tal caso è stata eseguita su macchine virtuali elaborate da *OracleVM Virtualbox*, secondo quanto specificato nel paragrafo 4.2.

Per quanto riguarda la configurazione di rete, invece, sono stati creati due scenari differenti, a seconda che si voglia usare `wpa_supplicant` o `iproute2` per instaurare il protocollo MACsec sul canale.

Nel caso di `wpa_supplicant` come visualizzato nello schema 4.3, per ogni macchina worker o master è stata impostata una rete Host, creata in precedenza per ospitare due singoli nodi: la macchina stessa e il nodo di pass-trough.

VirtualBox Host-Only Ethernet Adapter #4	192.168.173.1/30	Abilita
VirtualBox Host-Only Ethernet Adapter #5	192.168.110.1/30	Abilita
VirtualBox Host-Only Ethernet Adapter #6	192.168.109.1/30	Abilita

Figura 5.1: reti create su VirtualBox.

La macchina centrale su cui verranno indirizzati tutti i flussi di dati ha tre interfacce di rete, ognuna delle quali connessa a una delle reti sopraelencate, piú una connessione a internet da cui attingeranno tutti gli altri nodi. Infatti, la macchina centrale svolgerà il ruolo di access point per l'esterno.

Nel caso di `iproute2`, come visualizzato nella figura 4.4, per ogni macchina è stata configurata una duplice rete uscente su internet, in quanto la destinataria del protocollo non avrà piú disponibilità di accesso e per tale motivo l'uscita a internet dovrà essere sempre disponibile attraverso la rete aggiuntiva. Le configurazioni necessarie pre-installazione del cluster non sono necessarie.

Di seguito vengono riportati i passaggi fondamentali eseguiti ai fini di testare l'ambiente analizzato nei capitoli precedenti:

- Configurazione sistemistica macchina worker
  - Configurazione rotte
  - Impostazione DNS
- Configurazione sistemistica macchine pass-trough
  - Configurazione forwarding
  - Impostazione DNS
- Installazione kubeadm
  - Installazione pacchetti necessari per kubeadm;

  - Disabilitazione swap su macchine Linux;
  - Installazione container runtime;
  - Installazione plug-in Flannel;
  - Comandi eseguiti su nodo master;
  - Comandi eseguiti su nodo worker per il join.
- Installazione MACsec
  - kubeadm
  - iproute2

5.2.1 Configurazione macchine worker

Configurazione rotte - Sui nodi worker ci concentreremo sulle impostazioni di rete, in cui verranno definite la rotta di default e le rotte verso i nodi worker. In tal modo, qualsiasi pacchetto la cui destinazione non combacia con alcuna entry delle Routing table verrà reindirizzato verso la macchina posta al centro del cluster. Tale settaggio è stato effettuato ai fini di consentire ai nodi di raggiungere internet. In seguito, verranno impostate le rotte di uscita verso tutte le altre macchine all'interno del cluster di modo da direzionare correttamente tutto il traffico diretto agli altri nodi singoli.

Impostazione DNS - Occorre poi configurare il server DNS, di modo che rimanga stabile per tutta la durata dell'installazione. La medesima operazione verrà eseguita anche sulla macchina centrale. All'interno della macchina Linux è presente un *resolver* che garantisce l'accesso ai servizi del DNS. Il suo file di configurazione viene creato in automatico al primo tentativo di connessione a internet. Al suo interno è presente un campo chiamato *nameserver* che rappresenta l'indirizzo IP del server DNS che si vuole utilizzare. La configurazione manuale del file *resolv.conf* viene resettata automaticamente a ogni richiesta verso l'esterno, pertanto per poter rendere la configurazione persistente si necessita del pacchetto `resolvconf`. Questo passaggio stabilizza la connessione rendendo permanente la configurazione dell'IP del DNS.

5.2.2 Configurazione macchina pass-trough

Configurazione forwarding - Il nodo pass-trough dev'essere configurato poiché possa smistare il traffico verso le tre macchine collegate e verso internet. Innanzitutto dovrà essere impostata la traduzione del *Network Address translation (NAT)* [42], di modo che i pacchetti vengano letti, convertiti e direzionati, agendo sulle iptables che vengono rese persistenti all'accensione. In questo modo è stato eseguito il NAT correttamente. Il passo successivo riguarda la configurazione del forwarding del pacchetto per rendere possibile il redirezionamento corretto verso i nodi destinatari e verso internet.

Impostazione DNS - Una problematica riscontrata è la connessione a internet, come già illustrato in precedenza per il nodo worker. Vi è un bug che non risolve correttamente il DNS server, pertanto è necessario impostare il *resolv.conf* automaticamente poiché effettuandolo manualmente questo cambierebbe ogni volta che viene contattato il DNS o al riavvio della macchina. Di conseguenza bisogna riccorrere al supporto del pacchetto *resolvconf* come illustrato in precedenza 44. Al termine di queste operazioni, le macchine sono pronte a ospitare l'installazione richiesta. E' possibile effettuare un test pingando reciprocamente le macchine e internet da ognuna di esse.

5.2.3 Kubeadm

Installazione pacchetti - Una volta impostate le macchine per supportate la piattaforma procediamo con l'installazione dello strumento. Il nostro test è stato eseguito su una macchina `Ubuntu 20.04.5` e in base alla versione selezionata l'installazione richiede strumenti aggiuntivi

per il corretto funzionamento. Il primo passaggio da eseguire su ciascun host è l'installazione dei pacchetti di kubeadm quali `kubeadm`, `kubectl` e `kubelet`. Inizialmente si procede con l'aggiunta delle repository necessarie al download, partendo dall'inserimento delle chiavi necessarie a identificare il pacchetto da installare, e terminando con la creazione della lista in cui vengono identificati i path da cui ottenere le risorse desiderate.

Disabilitazione swap - Un altro passaggio fondamentale è la disabilitazione dello swap, ció avviene puramente per motivi di performance. Kubernetes ha il solo scopo di utilizzare tutte le risorse senza limiti di spazio o CPU, in quanto se queste dovessero terminare su un nodo, passerebbe all'altro e cosí via. L'utilizzo dello swap rallenterebbe il processo di assegnazione di un pod a un nodo, per tale motivo lo disabilitiamo. Nel kernel vengono abilitati due moduli:

```
sudo modprobe overlay
sudo modprobe br_netfilter
```

il primo modulo viene utilizzato per la gestione del filesystem e il secondo per facilitare la comunicazione attraverso una VxLAN tra i nodi del cluster Kubernetes. Segue la configurazione del file *kubernetes.conf* per indicargli quale tipo di pacchetti dovranno riferirsi alle iptables per poter essere processati. Si attiva questa funzionalità poiché tutti i componenti di kubeadm si rifanno alle iptables per la gestione del traffico.

Installazione container runtime - Il terzo passaggio richiede l'installazione di un container runtime, utilizzato dai container per essere eseguiti all'interno dei pod. In genere, se questo non venisse installato preventivamente per poi essere selezionato, verrebbe scelto il runtime di default previsto dal pacchetto kubeadm, in tal caso Docker. Nel nostro laboratorio è stato scelto `cri-o`, il quale meglio supporta la distribuzione scelta di Kubernetes. E' stato testato l'utilizzo di Docker, ma piú volte era necessario cambiare i file di configurazione `kubernetes.conf` per attenersi alle configurazioni di rete, altrimenti il kube-proxy sarebbe andato in errore in quanto dalla versione 1.24 in poi di Kubernetes Docker non è piú supportato (il nostro laboratorio utilizza la versione 1.25). Per avviare cri-o dev'essere impostata la subnet da affidare al bridge centrale per l'assegnazione degli IP ai pod di ciascun nodo.

Configurazione nodo master - I comandi dettati in precedenza vengono eseguiti su ogni host del cluster. Una volta selezionato il nodo master, su quest'ultimo verrà abilitato `kubelet` e verranno avviati tutti i componenti, tra cui il database *etcd* e l'api-server.

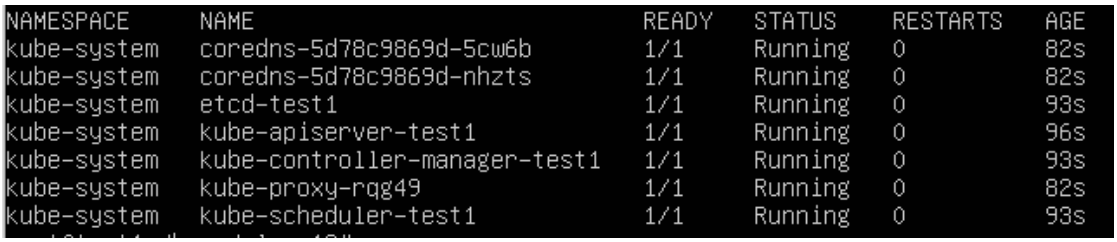
Il comando principale da eseguire esclusivamente sul nodo control-plane è il `kubeadm init`, che puó essere seguito da diverse opzioni:

`--control-plane-endpoint`: in caso di master multipli viene definito il singolo endpoint a cui contattare il control-plane. `--pod-network-cidr`: utilizzato per settare la rete da affidare ai pod in caso di utilizzo di un add-on. `--cri-socket`: se sono stati installati piú di un container runtime, per indicare quale utilizzare, dev'essere definito il suo path attraverso questo flag. `--apiserver-advertise-address`: viene definite l'indirizzo principale sulla macchina master a cui riferirsi. Ai fini del laboratorio utilizzeremo solamente il `-pod-network-cidr` in quanto per l'utilizzo dell'add-on Flannel sarà necessario definire uno specifico range di IP per le subnet affidate ai pods all'interno dei worker node.

```
kubeadm init --pod-network-cidr=10.244.0.0/16
```

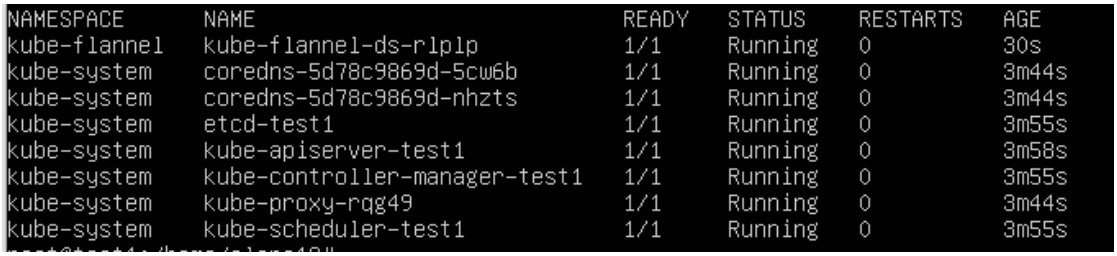
Una volta azionati tutti i pod, sarà possibile visualizzare tutti i componenti come in figura 5.2.

Installazione Flannel - Come quarto passaggio vi è l'aggiunta di Flannel, ripreso direttamente dalla repository git. Una volta installato sul nodo master, come detto in precedenza, passerà in automatico ai nodi worker che effettueranno il join. L'installazione del plug-in avviene tramite lettura del file yaml, che è possibile scaricare dalla repository, contentente tutte le informazioni utili a definire le configurazioni di rete da applicare, come il canale di backend utilizzato o le immagini selezionate per essere eseguite nel container. Una volta avviata l'istanza di Flannel risulterà come un pod runnato 5.3.



NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	coredns-5d78c9869d-5cw6b	1/1	Running	0	82s
kube-system	coredns-5d78c9869d-nhzts	1/1	Running	0	82s
kube-system	etcd-test1	1/1	Running	0	93s
kube-system	kube-apiserver-test1	1/1	Running	0	96s
kube-system	kube-controller-manager-test1	1/1	Running	0	93s
kube-system	kube-proxy-rqg49	1/1	Running	0	82s
kube-system	kube-scheduler-test1	1/1	Running	0	93s

Figura 5.2: pod attivi alla creazione del nodo.



NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-flannel	kube-flannel-ds-rlplp	1/1	Running	0	30s
kube-system	coredns-5d78c9869d-5cw6b	1/1	Running	0	3m44s
kube-system	coredns-5d78c9869d-nhzts	1/1	Running	0	3m44s
kube-system	etcd-test1	1/1	Running	0	3m55s
kube-system	kube-apiserver-test1	1/1	Running	0	3m58s
kube-system	kube-controller-manager-test1	1/1	Running	0	3m55s
kube-system	kube-proxy-rqg49	1/1	Running	0	3m44s
kube-system	kube-scheduler-test1	1/1	Running	0	3m55s

Figura 5.3: pod a seguito dell'attivazione di Flannel.

Configurazione nodo worker - I worker node potranno eseguire il join lanciando un singolo comando:

```
kubeadm join <indirizzo IP su cui \‘e in esecuzione il nodo master>:
6443 --token <ottenuto dalla kubeadm token list>  --discovery-token
-ca-cert-hash sha256:<certificate generato>
```

5.2.4 MACsec

wpa_supplicant - Per la prima configurazione la libreria utilizzata è wpa_supplicant in quanto è stato creato un cluster apposito con una macchina centrale per il pass-trough dei dati. wpa_supplicant permette una connessione sicura solamente tra due nodi su una rete. Dal momento che Kubernetes utilizzerà una sola interfaccia di rete per tutte le comunicazioni, è stata creata la macchina centrale che crea una connessione end-to-end con ogni macchina e tale connessione sarà quella protetta. Ogni host ha una sola interfaccia di rete. Per l'installazione di wpa_supplicant su tale interfaccia è necessario il download del pacchetto. Per tale installazione è necessario un file di configurazione che indichi tutte le opzioni per la connessione.

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
ap_scan=0
eapol_version=3
network={
    key_mgmt=NONE
    eapol_flags=0
    macsec_policy=1
    macsec_integ_only=0
    mka_cak=<32 bit> #stringa da 32 bit
    mka_ckn=<64 bit> #stringa da 64 bit
}
```

Listing 5.2: file di configurazione per wpa_supplicant.

Di seguito i flag analizzati piú nel dettaglio:

ctrl_interface indica il percorso in cui è presente lo strumento.

ctrl_interfacegroup utilizzato per specificare che anche utenti non root possono interagire con l'interfaccia.

apscan utilizzato per settare gli access point ma per MACsec non è necessario quindi viene impostato a 0.

eapol_version viene settato a 3 poiché è la versione di EAPOL utilizzato da MKA.

key_mgmt definisce il protocollo di autenticazione scelto, puó essere WPA-PSK WPA-EAP, IEEEE8021X o NONE. Nel nostro caso sarà NONE e definisce una cifratura secondo protocollo Wired Equivalent Privacy (WEP) e quindi senza definire alcuna password e con delle chiavi già condivise, mentre le altre necessitano un server esterno per l'autenticazione.

eapol_flags puó avere valore di 1 o 0 e definisce la distribuzione delle chiavi, ma nel caso di MACsec dev'essere settato a 0 il che preclude una distribuzione unicast delle chiavi WEP.

macsec_policy definisce se MACsec è abilitato o meno, inserendo 1 o 0.

macsec_integ_only definisce se cifrare il traffico oppure no, impostando 0 o 1, di default è a 0.

mka_cak è la chiave CAK ed è composta da 32 caratteri.

mka_ckn è la CKN associata alla CAK composta da 64 caratteri.

Vi sono altre opzioni attive per MACsec quali:

macsec_replayprotect che definisce l'attivazione della protezione da attacchi di replay settando 1 o 0.

macsec_replaywindow definisce la finestra in cui è tollerato il replay, o è 0 in caso di replay inattivo o il numero di pacchetti accettati.

macsec_port definisce la porta dello SCI da affidare, è sempre 1.

mka_priority definisce la priorità con cui un host è scelto come key server.

L'attivazione su ogni host è eseguita attraverso il lancio di un singolo comando

```
wpa_supplicant -K -t -i <interfaccia di rete> -D macsec_linux -c <file di
    configurazione> -dd
```

Quest'ultimo viene lanciato sia su un host che sull'altro e verranno inviati sulla rete pacchetti EAPOL per identificare il peer con la medesima chiave CAK e facente parte dunque della stessa CA. Una volta completato verrà eseguita la distribuzione della chiave SAK necessaria alla comunicazione. Alla scadenza della chiave utilizzata, verrà rigenerata la chiave e ridistribuita tra i peer. Nel frattempo i peer continuano a mandare segnali di liveness per comunicare che sono ancora partecipi della CA. Ai fini del nostro cluster Kubernetes a ogni interfaccia MACsec verrà affidato l'indirizzo IP della rete su cui è stato impostato il protocollo semplicemente poiché le comunicazioni avvengono tramite collegamenti di IP e se non affidassimo il medesimo non riuscirebbero a parlarsi. Al termine avremo uno schema simile 5.4.

iproute2 - Per questa topologia si è ricorsi all'utilizzo della libreria `iproute2`. In questo caso non viene effettuato il rekeying e l'applicazione puó essere effettuata sulla stessa interfaccia di rete a piú connessioni end-to-end, di conseguenza è stata scelta questa topologia 5.5. A differenza di wpa_supplicant, la configurazione deve avvenire manualmente per ciascun host. In tal senso, verrà configurato su ogni macchina un canale tx di trasmissione e piú canali rx di ricezione. A ogni canale sarà affidata la propria SAK distribuita manualmente. Dal momento che l'instaurazione dei collegamenti MACsec avviene su una singola interfaccia di rete, il dato da dover cambiare è il numero della SA creata. La nuova topologia sarà quella illustrata in 5.5.

Le configurazioni illustrate, sia per quanto riguarda la libreria `iproute2` sia per `wpa_supplicant` possono essere riadattate anche a architetture complesse della piattaforma Kubernetes, ad esempio

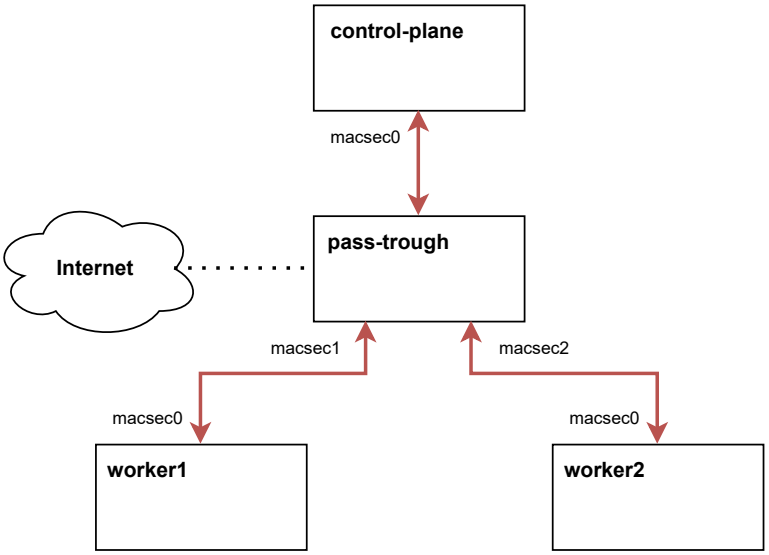


Figura 5.4: schema di rete con protocollo MACsec.

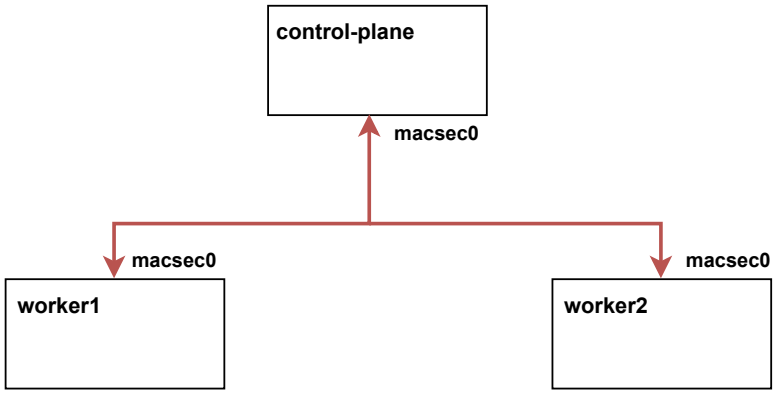


Figura 5.5: schema di rete con protocollo MACsec.

una multi-cluster illustrata nella figura 5.6. Com'è possibile evincere i collegamenti in rosso sono protetti con l'utilizzo del protocollo MACsec, nello specifico worker-worker e worker-master. Nel caso di wpa_supplicant la configurazione necessita di una macchina aggiuntiva fra i collegamenti protetti che funga da pass-through.

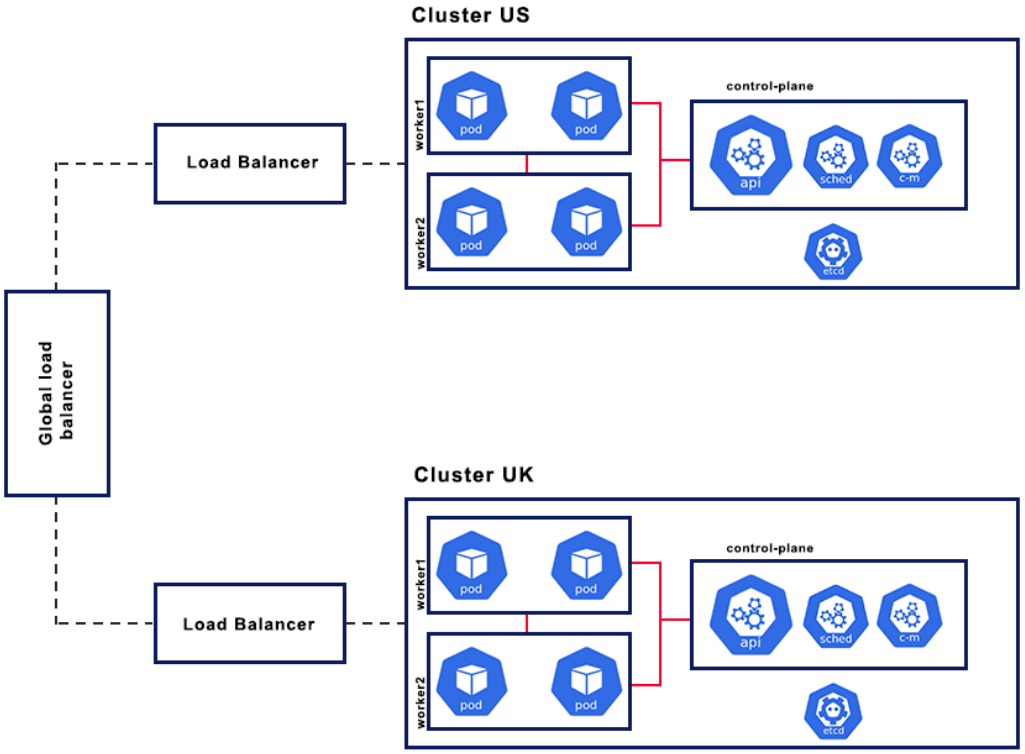


Figura 5.6: struttura multi-cluster di un data-center implementato con Kubernetes

Capitolo 6

Test e conclusioni

Attraverso lo studio di questa tesi abbiamo analizzato l'inserimento del protocollo MACsec all'interno dei canali di comunicazione fra i nodi di un cluster Kubernetes. A configurazione completa del laboratorio è stato possibile visualizzare come il traffico proveniente da un pod rilasciato su di un nodo worker e destinato al pod presente su di un worker differente presentasse tutti frame etichettati con il type MACSEC. Lo studio effettuato è stato condotto su un'architettura composta da soli due nodi worker e un nodo master con rilascio del servizio nginx come test; è possibile effettuare uno studio piú ampio su cluster complessi, ma il funzionamento è simile e pertanto non era necessario riprodurlo nel nostro laboratorio. L'utilizzo della quarta macchina in funzione di pass-through era legato all'esigenza di un rilascio del cluster Kubernetes su una singola interfaccia di rete su cui comunicassero tutti i componenti, e ció era limitante per l'utilizzo della libreria wpa_supplicant la quale gestisce una sola connessione sicura fra due nodi alla volta sulla medesima interfaccia. Con la libreria iproute2 ció era possibile attraverso l'utilizzo di SA differenti, ma non era possibile sfruttare tale libreria a causa del rekeying non fornito. Di seguito verranno illustrati i test eseguiti con relativi risultati:

- tempistiche di esecuzione script di installazione;
- bitrate nel caso di connessione con canale sicuro;
- bitrate nel caso di connessione con canale in chiaro;
- bitrate con canale sicuro ma privo di confidenzialità;
- test di sovraccarico delle risorse sulle macchine.

6.1 Test

I test effettuati riguardano il cluster con wpa_supplicant su 3 macchine, di cui una è il nodo master e due sono i nodi worker. L'installazione eseguita è avvenuta con script ansible, citati nei capitoli precedenti e nel manuale utente A. L'architettura utilizzata è illustrata nel grafico A.1. Non è stato analizzato il caso di inserimento di due master, mentre l'utilizzo di soli due worker è limitato dal virtualizzatore Oracle VM Virtualbox poiché per la macchina pass-through è possibile inserire fino a quattro interfacce di rete, e in tal caso teniamo conto delle tre riferite alle macchine del cluster e la singola in bridge riferita alla connessione in uscita su internet. Una volta avviati i playbook ansible come illustrato in precedenza per la creazione del cluster otterremo il cluster completo di tutti i componenti, il passaggio successivo è l'instaurazione del canale sicuro. In tal caso nel momento in cui viene avviato il tool di wpa_supplicant su di un nodo, verrano inoltrate varie richieste EaPoL come illustrato in precedenza, in cui dovrá essere riconosciuto il nodo destinatario appartenente alla madesima CA e per decretare il KS. Nella figura 6.1 è possibile visualizzare il traffico iniziale di pacchetti per decretare il KS all'interno della CA.

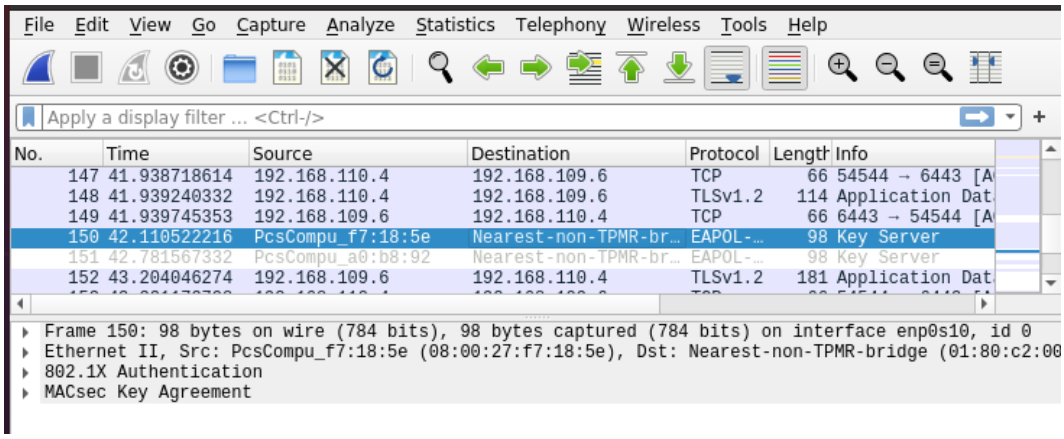


Figura 6.1: Frame iniziali di instaurazione del canale sicuro.

In seguito vengono inoltrati frame di richiesta dei nodi ancora attivi e per l'instaurazione del canale viene inoltrata la SAK dal KS e distribuita fra i nodi. I pacchetti visualizzati nella figura 6.2 riguardano i pacchetti inviati e ricevuti dal singolo nodo appartenente al cluster in comunicazione con la macchina centrale. Tutti i dettagli riguardanti le chiavi e il KS è possibile notare come siano affidati al protocollo MKA. Poi in seguito il primo MACsec frame scambiato.

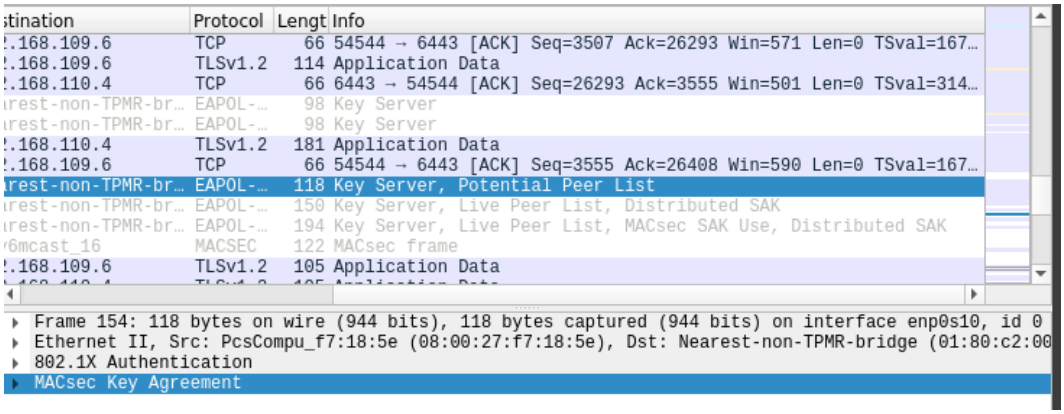


Figura 6.2: Frame di instaurazione del canale sicuro con MACsec.

Come definito in precedenza il primo test effettuato riguarda l'esecuzione degli script riferiti in precedenza. Nel grafico 6.3 è presente la differenziazione tra le tempistiche di completamento dei differenti playbook. In tal senso il primo eseguito riguarda l'installazione del cluster Kubernetes, mentre il secondo è riferito all'instaurazione della connessione sicura.

Com'è possibile evincere dalla figura il tempo piú elevato analizzato è quello riferito allo script *ansible_kube.yml* rappresentato dal campo *kube configuration* che comprende *ansible_master*, *ansible_worker* e *createwa*. Si avvicina a circa i 10 minuti e ció è dovuto al fatto che venga settato tutto il cluster da principio con il download di tutti i componenti necessari al fine del mantenimento di tutti i nodi, il tempo piú elevato viene riscontrato nello scaricamento delle immagini da parte del nodo master. Nel grafico non è stato analizzato il tempo necessario al settaggio di tutte le impostazioni ssh necessarie al funzionamento di ansible stesso poiché vengono date per assodate. Nel grafico 6.4 è invece presente l'elaborazione delle configurazioni riguardante le macchine. Il *setup machine* e il *setup node* sono script bash utilizzati per la configurazione delle rotte corrette e i loro valori sono minimi. Il valore piú elevato riguarda l'aggiunta di un nodo poiché viene riconfigurata l'installazione dei componenti necessari per il funzionamento di Kubernetes. Da tenere presente in ogni caso che la macchina su cui sono stati eseguiti i test non presenta prestazione elevate.

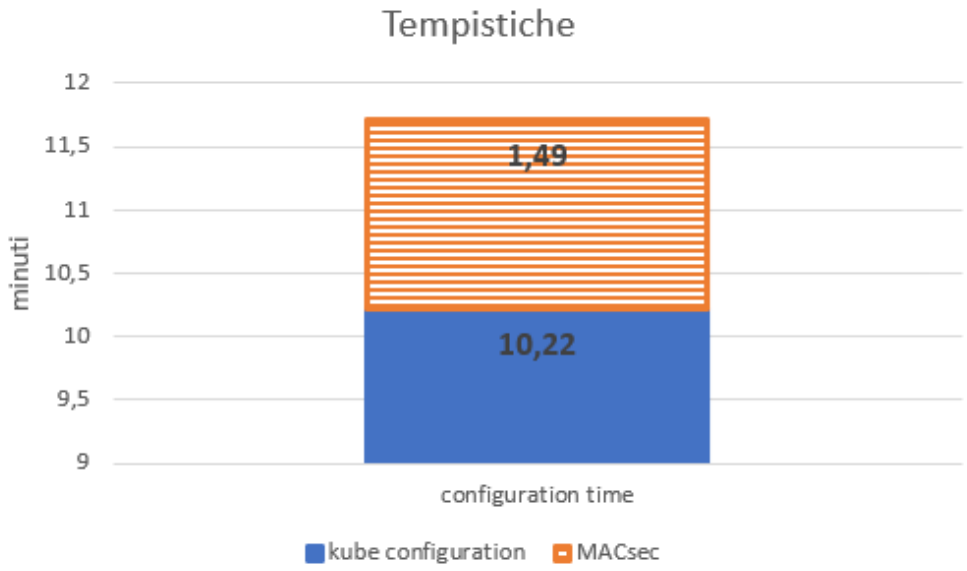


Figura 6.3: Tempistiche di riferimento per l'esecuzione degli script.

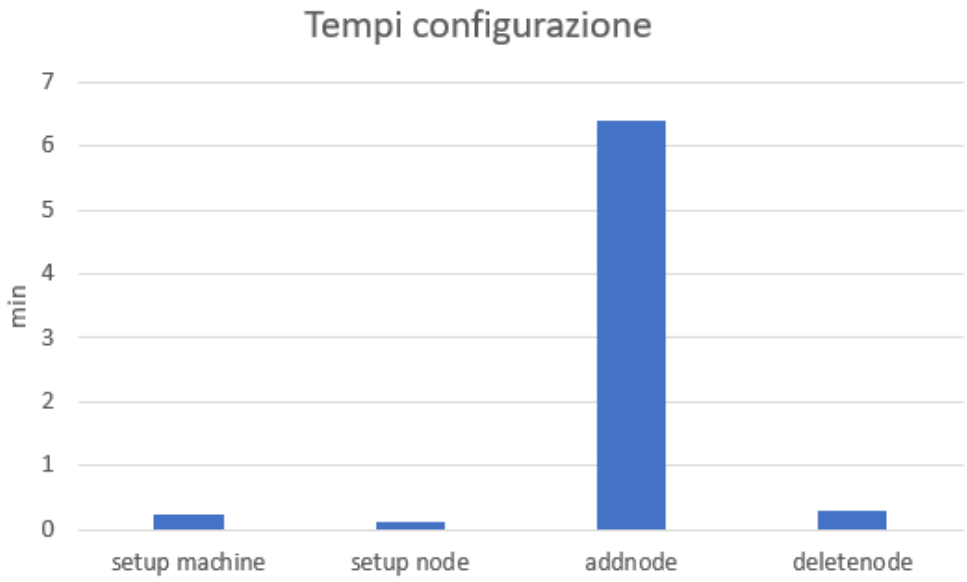


Figura 6.4: Tempistiche di riferimento per la configurazione delle macchine.

Il secondo test è riferito al tentativo di invio di informazioni tra i nodi con l'avvio di un servizio *nginx* su entrambi i worker, e collegandosi in remoto da un pod della macchina worker1 abbiamo inviato pacchetti continui al worker2 nell'arco di 60 secondi attraverso il tool *iperf*. Dalla figura 6.5 è possibile notare nell'arco di 60 secondi come i pacchetti abbiano superato i 200 Mbits/s nel caso di un canale di comunicazione tra i worker privo di protezione.

Il terzo test è riferito al medesimo ambiente proposto in precedenza, ma con la configurazione del protocollo MACsec tra i nodi partecipanti al cluster, nel medesimo grafico proposto in precedenza i dati risultano leggermente differenti rispetto alla situazione proposta in chiaro, i valori sfiorano i 140 Mbits/s, pertanto c'è stato un carico maggiore sulla rete che ha portato alla diminuzione della velocità su cluster. Da confronto fra i due andamenti del grafico 6.5 è possibile notare questa differenza, ció è dovuto dal fatto che una maggior quantità di informazioni risulta in una diminuzione del Bitrate. Pertanto tra la versione senza MACsec e quella protetta

la differenza è di circa 70 Mbits/s, ma considerata tutta la fase di cifratura eseguita i dati sono accettabili.

Nello specifico nella creazione del canale sicuro è possibile fare una selezione, se utilizzare solo l'integrità del dato (funzionalità di base definita di MACsec) oppure garantire anche la cifratura dei dati trasmessi. Tale modifica è resa possibile da un singolo flag impostato nel file yaml di configurazione della connessione con wpa_supplicant portando il campo *macsec_integ_only* da 0 a 1. Nel medesimo grafico è possibile notare la differenza evidente tra una connessione priva di confidenzialità, ma con sola autenticità e integrità dei dati. Il divario di circa 40 Mbits/sec è dovuto all'assenza delle operazioni di cifratura dei dati che pertanto richiedono maggior tempo per poter essere eseguite.

Nella figura 6.5 è possibile notare la differenza di velocità tra una condizione priva di confidenzialità, in cui vengono eseguite solamente le procedure di autenticazione e pertanto il riconoscimento dell'appartenenza alla medesima CA, e una condizione a canale in chiaro. Il divario ovviamente è diminuito poiché non sono presenti tutte le operazioni di cifratura e validazione del pacchetto.

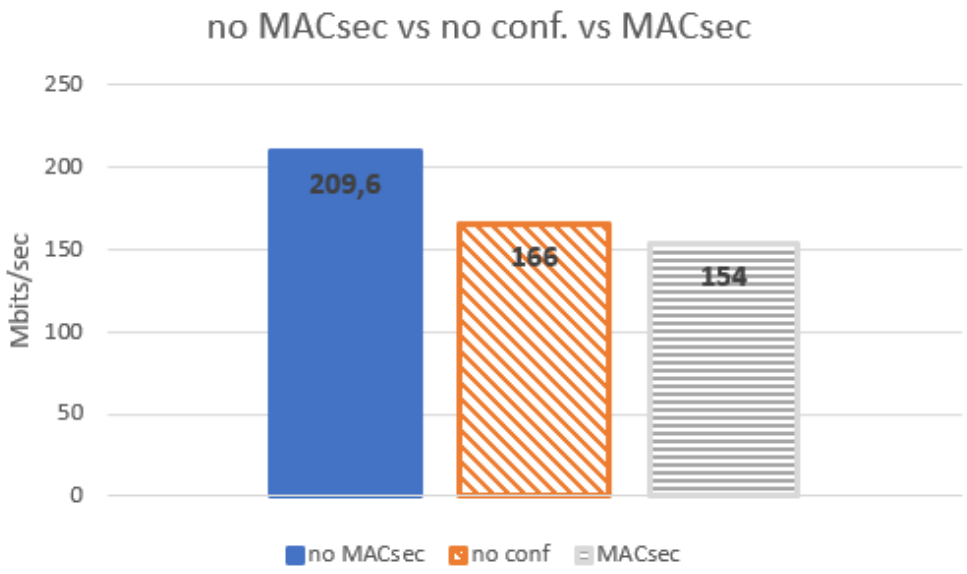


Figura 6.5: Confronto tra canale protetto e non e confidenzialità disabilitata.

L'ultimo test effettuato riguarda l'utilizzo di un servizio che sovraccarichi la rete e soprattutto il canale di comunicazione protetto con MACsec tentando di esaurire le risorse disponibili. E' possibile evincere queste deduzioni dal grafico 6.6 dove il bitrate, per una condizione di carico maggiore, aumenta. La differenza con una situazione scarica è di circa 20 o 30 Mbits/s e ció è plausibile data la mole di informazioni che la connessione sta inoltrando.

6.2 Conclusioni

Configurazione - La problematica piú rilevante riguarda la configurazione del protocollo tramite la libreria wpa_supplicant, cosí come sono presenti dei difetti nella configurazione tramite la libreria iproute2. Per quanto riguarda la prima l'esecuzione del protocollo solo fra due nodi è limitante nel caso dell'installazione di un cluster Kubernetes poiché in tal caso sarà sempre necessario l'utilizzo di uno switch intermedio o di una macchina centrale che faccia da pass-through per tutto il traffico, in tal modo la problematica principale riguarda il forwarding dei pacchetti che nel caso degli host potranno leggere pacchetti con destinazione differente dalla loro e un attaccante potrebbe anche collegarsi alla macchina centrale e fare sniffing di tutto il traffico. Inoltre la presenza di uno switch è un danno alla sicurezza poiché è una delle vulnerabilità

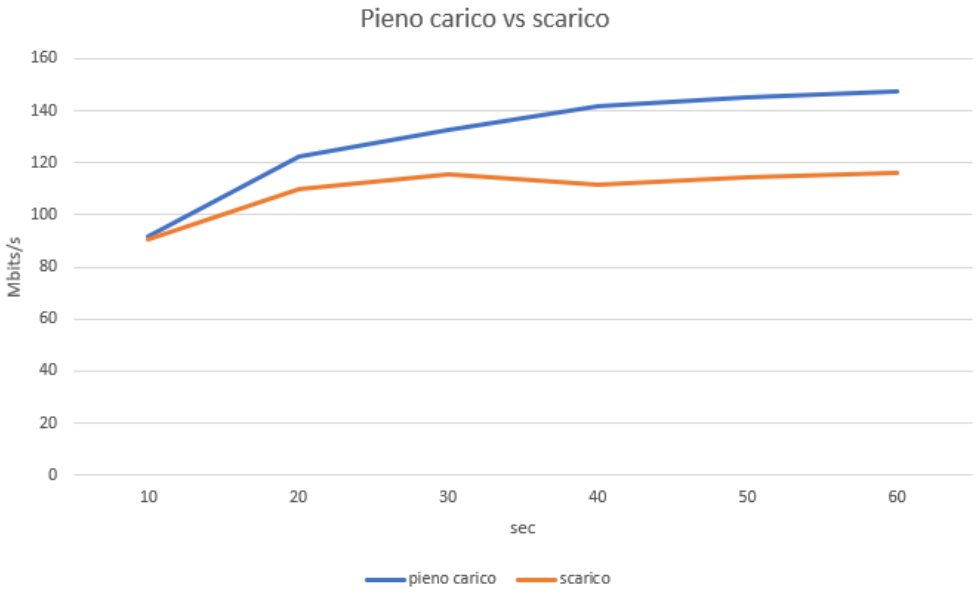


Figura 6.6: Confronto fra connessione con carico e senza.

presentate per il protocollo MACsec in quanto il suo è un passaggio hop-by-hop che lascia aperta la possibilità di una lettura in chiaro all'interno nel caso di porte non abilitate all'utilizzo di MACsec. La risoluzione fisica potrebbe essere legata all'utilizzo di uno switch con porte MACsec abilitate, ma comunque rimarrebbe l'unico punto di fallimento del cluster. Inoltre l'utilizzo del virtualizzatore Oracle VM Virtualbox in tal caso risulta limitante poiché puó ospitare fino a quattro interfacce di rete contemporaneamente, pertanto l'aggiunta di macchine ulteriore non è possibile. Nel caso della libreria iproute2 invece la problematica era legata al rekeying per cui, in seguito all'utilizzo di tutte le possibili chiavi necessarie alla criptazione del pacchetto (questo è definito dal quantitativo di bit espresso dalla suite scelta), sarà necessario ristabilire manualmente tutte le connessioni per reimpostare le chiavi, altrimenti queste ultime potrebbero essere scoperte anche attraverso attacchi di BruteForce e il flusso diverebbe in chiaro.

Per quanto riguarda le tempistiche di configurazione dell'intero test tramite script ansible, i dati risultano elevati per quanto riguarda il settaggio di tutto il cluster (nodi master e worker piú installazione di tutti i componenti della libreria kubeadm), ma è da tener conto l'utilizzo di macchine a basse prestazione nonchè la connessione utilizzata per lo scaricamento delle immagini utili al nodo master. In confronto peró alle tempistiche di installazione, la configurazione del canale è irrisoria. L'unica sarà, per ogni nuovo nodo aggiunto, la configurazione delle rotte e dunque l'utilizzo dello script *setupnode* che dovrà essere eseguito direttamente sulle macchine, altrimenti sono irrangiungibili.

Connessione - Il peso del protocollo sull'elaborazione dei dati all'interno del flusso di dati è minimale, in quanto dai test eseguiti è emerso che il bitrate aumentasse di un valore irrisorio ai fini del nostro caso di studio e comunque all'interno dei limiti considerando le operazioni eseguite per stabilire la connessione e per cifrare i dati e inviarli al nodo destinatario, nonchè il tempo speso per validare il pacchetto una volta giunto al nodo.

Carico - Per quanto riguarda il test di carico, il risultato è coerente con quanto specificato e non sembra riportare risultati significativi.

6.3 Lavori futuri

Il test eseguito per tale tesi era finalizzato al corretto e possibile inserimento del protocollo MACsec all'interno di un canale di comunicazione fra i componenti di cluster Kubernetes. Fra

le varie configurazioni visualizzate un possibile sviluppo futuro potrebbe essere collegato allo studio della libreria *wpa_supplicant* di modo che diventi possibile la configurazione su un'unica interfaccia di rete di piú connessioni sicure attraverso l'utilizzo di SA differenti. Un'altra possibile applicazione potrebbe essere l'utilizzo di uno switch con porte MACsec abilitate in modo da poter verificare una condizione di passaggio dei dati senza dati in chiaro. L'applicazione di tale configurazione su un virtualizzatore, ma su di una macchina fisica effettiva puó presentare lo studio di macchine aggiuntive al cluster. L'ultima possibile applicazione è la configurazione di MACsec sul collegamento VxLAN già presente in altri studi, ma che sarebbe possibile adattare al contesto Kubernetes.

Appendice A

Manuale utente

Di seguito verranno riportati i passaggi che un utente finale deve eseguire per proseguire con l'installazione del cluster Kubernetes con protezione di canale abilitata a MACsec illustrata all'interno dei capitoli precedenti di tesi. L'installazione viene eseguita tramite script ansible, per automatizzare tutti i passaggi necessari alla configurazione. I file elencati nei paragrafi successivi sono presenti sulla repository git `https://github.com/Skyron13/macsec-kubeadm.git`.

A.1 Requisiti

L'architettura utilizzata è composta da 2 macchine worker, 1 macchina master e 1 macchina di pass-through A.1. A tale configurazione è possibile inserire ulteriori macchine. Nella configurazione di un cluster Kubernetes è possibile l'utilizzo di due master, ma tale configurazione non è stata presa in considerazione per il nostro caso di studio.

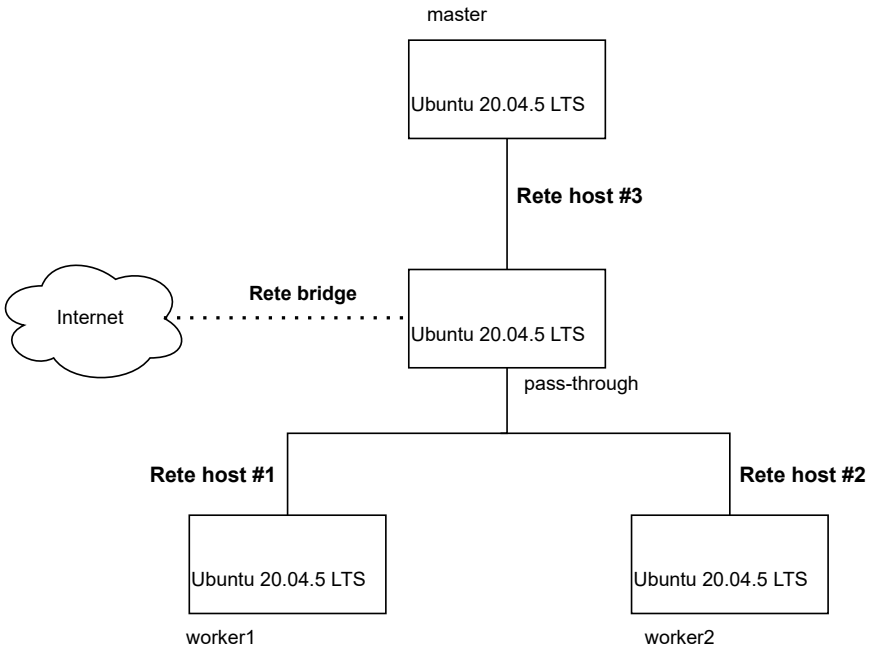


Figura A.1: architettura cluster.

Come virtualizzatore è stato selezionato Oracle VM Virtualbox nell'ultima versione, ma è possibile utilizzarne anche uno alternativo oppure riprodurre l'implementazione su macchine

fisiche. Il sistema su cui è stata testata l'architettura è Ubuntu Linux 20.04.5 LTS per ragioni di compatibilitá illustrate in precedenza. Per quanto riguarda le macchine partecipanti al cluster questa scelta è obbligatoria, mentre per quanto riguarda la macchina centrale è mandatorio, peró per continuitá è stata scelta la medesima distribuzione. Le specifiche hardware per ogni macchina sono elencate nella tabella A.1.

Hardware	**Requisiti**
RAM	2 GB
Memoria	25 GB
CPU	2 core (necessario per l'installazione di kubernetes)

Tabella A.1: Requisiti hardware

Per poter procedere con l'installazione di MACsec tramite libreria wpa_supplicant è necessaria una configurazione di rete all'interno di Oracle VM Virtualbox con tre reti host (riferita a ciascuna macchina collegata alla macchine centrale) come riportato nella figura A.2. La loro creazione viene effettuata direttamente sul tool di VirtualBox sotto il tab `File` e nella sezione `Strumenti` alla voce `Gestore di rete` in cui è possibile configurare reti host, con nat o cloud.

Figura A.2: reti create su VirtualBox.

All'interno della rete viene data netmask pari a 30 in quanto dovrà ospitare solamente il nodo della macchina centrale e la macchina operatrice stessa, ció non è strettamente necessario, ma in tal modo escludiamo l'insesrimento di ulteriori macchine sul collegamento end-to-end della macchina del cluster e la macchina pass-through. Una volta create le macchine sará sufficiente assegnare alle macchine del cluster la rete host designata all'interno della sezione `Network`, mentre alla macchina pass-through verranno assegnati tali reti host piú una rete bridge (la quale sfrutta la rete presente sul computer locale dove viene eseguito il tool) per permettere il collegamento ad internet. La limitazione derivata dall'utilizzo di Oracle VM Virtualbox è il numero di reti possibili da inserire nella macchina centrale, il quale ha un limite di esattamente 4 schede di rete da inserire.

Per configurare e installare i vari componenti del cluster si è ricorsi all'utilizzo di un sistema di automatismo di tutte le operazioni, anche perché molti di questi passaggi dovranno essere ripetuti su tutte le macchine. Per poter utilizzare ansible dovranno essere scambiate le chiavi fra le varie macchine per abilitare la connessione in ssh, ma questo passaggio in tal caso lo diamo per scontato.

A.2 Installazione

Gli script ansible disponibili per l'installazione sono quattro, piú due aggiuntivi per la gestione dei nodi:

ansible_kube viene utilizzato per impostare e installare le configurazioni iniziali del clsuter Kubernetes, quali installazioni dei pacchetti del tool scelto, installazione della tecnologia di containerizzazione, impostazione e configurazione per le varie subnet affidate all'interno delle macchine operatrici. Inoltre tale playbook richiama ed esegue i playbook di installazione del master e dei nodi worker.

ansible_master è un playbook richiamato dal file ansible_kube.yml, viene utilizzato per configurare solamente il nodo master riferendosi come host al gruppo master. Procede in tal

senso con lo scaricamento di tutte le immagini necessarie ai componenti e oltre all'avvio del nodo stesso all'installazione del plug-in flannel. Al termine viene salvato in un file esterno il comando che verrá eseguito per effettuare il join dei nodi worker.

ansible_worker viene utilizzato esclusivamente per eseguire il join del nodo, l'unico comando che esegue è quello riportato all'interno del file salvato in precedenza dal master.

createwa è il playbook utilizzato per la creazione del canale sicuro, in tal senso elabora il file di configurazione da passare al comando lanciato per wpa_supplicant e verrá riprodotto lo stesso file sul nodo centrale oltre alla macchina del cluster designata.

addnode è il playbook utilizzato per aggiungere un ulteriore nodo al cluster (worker). Le funzioni svolte sono simili a quelle illustrate nel file ansible_kube e ansible_worker.

deletenode elabora l'eliminazione di ogni servizio attivo per non farlo riprodurre sul nodo da eliminare, e infine elimina il nodo stesso.

I passaggi da eseguire sono i seguenti:

- configurazione sistemistica macchina worker;
- configurazione sistemistica macchina pass-through;
- installazione cluster Kubernetes;
- installazione MACsec.

Il primo passaggio riguarda le configurazioni necessarie per il settaggio delle macchine per quanto riguarda la gestione del flow dei pacchetti nonchè l'abilitazione alla connessione internet (da effettuare una volta avviate le macchine) sono presenti all'interno di due script presenti sulla repository git: *setupnode.sh* e *setupdmachine.sh*, il primo viene utilizzato per la configurazione delle macchina del cluster (per abilitazione a internet e flow dei pacchetti verso il nodo corretto) e il secondo per la macchina centrale per abilitare il forwarding del traffico.

Il secondo passaggio richiede l'avvio del playbook globale che effettua la configurazione di tutto il cluster Kubernetes con master e worker indicati nell'inventario (file .yml) che dev'essere impostato in precedenza contenente tutti gli host corretti sotto i gruppi determinati come in figura A.1. Tale inventory sará passato all'avvio del playbook in modo che tutte le macchine siano giá settate.

```
central:
  hosts:
    *inserire nodo centrale*:
master:
 hosts:
   *inserire nodo master*:
worker:
 hosts:
   *inserire nodi worker*:
worker1:
 hosts:
   *inserire nodo worker 1*:
worker2:
 hosts:
   *inserire nodo worker 2*:
```

Listing A.1: esempio inventory.

Un altro elemento fondamentale è il file contenente tutte le variabili utilizzate all'interno dei playbook. Tale file soprannominato *vars.yml* ha una struttura presentata in figura A.2 completata di esempi, in cui è intuitivo predisporre i dati come valorizzati.

```
---
OS: xUbuntu_20.04
VERSION: 1.27
conf: 0
cak: <valore a 32 bit>
ckn: <valore a 64 bit>
interfacemaster: enp0s10
ip: 192.168.109.6
interface: macsec0
ip2: 192.168.109.4
conf_worker1: 0
cak_worker1: <valore a 32 bit>
ckn_worker1: <valore a 64 bit>
interfacemaster_worker1: enp0s9
ip_worker1: 192.168.110.4
interface_worker1: macsec1
ip_for_worker1: 192.168.110.3
conf_worker2: 0
cak_worker2: <valore a 32 bit>
ckn_worker2: <valore a 64 bit>
interfacemaster_worker2: enp0s8
ip_worker2: 192.168.173.4
interface_worker2: macsec2
ip_for_worker2: 192.168.173.3
```

Listing A.2: esempio file vars.yml.

Il playbook puó essere avviato tramite comando A.3. Viene utilizzata l'opzione `-u` per indicare l'utenza presente sulla macchina destinataria e `--ask-become-pass` per forzare la richiesta della password per tale utenza. L'utilizzo del flag `-i` indica invece l'utilizzo dell'inventory creato precedentemente da noi. Tale playbook coinvolgerá inizilamente solo le macchine del cluster non la macchina centrale poiché le prime configurazioni riguardano l'installazione del cluster stesso, in seguito verrá effettuato il collegamento protetto.

```
ansible-playbook ansible_kube.yml -u <utente da utilizzare> -i <inventario
    da passare> --ask-become-pass
```

Listing A.3: comando unico da lanciare per la configurazione.

In tal modo il cluster sarà configurato e funzionante, ma privo di un canale sicuro.

L'ultimo passaggio, presente sempre all'interno del medesimo playbook, esegue la configurazione del canale sicuro con MACsec. Recuperando le informaizoni elencate all'interno del file contenente tutte le variabili sará possibile impostare le chiavi CAK per ciascuna connessione e relative interfacce da configurare.

La prima operazione eseguita sará relativa alla creazione del file *wpa.conf* che verrá utilizzato dalla macchina centrale e dal nodo del cluster per stabilire la connessione sicura, infatti tale file sará il medesimo su di una e l'altra maccchina (chiavi identiche). Una volta termianta tale operazione verrá lanciato il comando *wpa_supplicant* impostato con il flag -B per farlo eseguire in background sulle macchine destinatarie, di modo che l'esecuzione del playbook non venga interrotta. Una volta stabilito il canale all'interfaccia macsecX creata verrá assegnato l'IP relativo all'interfaccia di comunicazione con il cluster.

Al termine della configurazione sará possibile visualizzare un traffico cifrato proveniente dai nodi e diretto verso gli altri nodi del cluster.

Infine i due playbook di *addnode* e *deletenode* vengono predisposti con un avvio semplificato tramite comando ansible-playbook come illustrato in precedenza, semplicemente dovrá essere passato una variabile aggiunta *node* riferito al nodo da eliminare o aggiungere.

Appendice B

Manuale sviluppatore

Di seguito un elenco dettagliato delle configurazioni e dei comandi da eseguire per replicare correttamente il test implementato ai fini del nostro caso di studio. Gli scenari prodotti sono due: uno con libreria per l'implementazione di MACsec `wpa_supplicant` e l'altro con libreria `iproute2`. Tutti i comandi illustrati di seguito sono presenti all'interno dei playbook ansible per l'automazione dell'installazione. Gli script indicati sono i seguenti:

- ansible_kube;
- ansible_master;
- ansible_worker;
- createwa;
- addnode;
- deletenode.

piú due script bash di setup iniziali quali *setupnode* e *setupmachine*. Il loro funzionamento puntuale è illustrato nel capitolo A. Come illustrato in precedenza le macchine possiedono distribuzioni Ubuntu 20.04.5 LTS per ragioni di compatibilitá con il tool kubeadm, e vengono eseguite su un virtualizzatore quale Oracle VM Virtualbox nell'ultima versione. Dal punto di vista del sistema i requisiti da soddisfare per le macchine sono i seguenti

Hardware	**Requisiti**
RAM	2 GB
Memoria	25 GB
CPU	2 core (necessario per l'installazione di kubernetes)

Tabella B.1: Requisiti hardware

I passaggi da eseguire sono i seguenti:

- Configurazione sistemistica macchina worker
  - Configurazione rotte
  - Impostazione DNS
- Configurazione sistemistica macchine pass-trough
  - Configurazione forwarding

  - Impostazione DNS

- Installazione cluster Kubernetes
  - Installazione pacchetti necessari per Kubeadm;
  - Disabilitazione swap su macchine Linux;
  - Installazione container runtime;
  - Installazione plug-in Flannel;
  - Comandi eseguiti su nodo master;
  - Comandi eseguiti su nodo worker per il join.
- Installazione MACsec

Per una corretta comprensione è possibile visualizzare lo schema del cluster nella figura B.1.

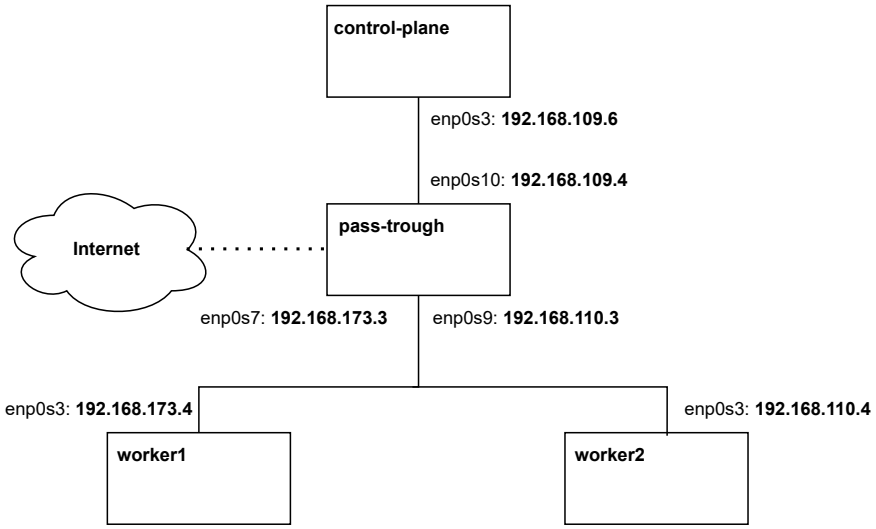


Figura B.1: topologia wpasupplicant

B.1 Configurazione worker

All'interno dello script *setupnode* è possibile eseguire tutta la configurazione del networking relativo ad un nodo del cluster, che sia master o worker. Di seguito illustrato piú nel dettaglio. Per la configurazione delle macchine worker, aggiunte in seguito o facenti parte del cluster a priori dovranno essere configurate le rotte per il redirezionamento corretto del traffico verso il nodo della macchina centrale. sudo ip route add default via ¡nodo macchine pass-trough¿ In seguito dovrà essere configurato il DNS per rendere perpetuo il suo indirizzo IP di riferimento, in quanto a ogni connessione tramite la macchina centrale l'indirizzo DNS viene annullato pertanto è necessario configurare il file della libreria `resolvconf`.

```
sudo nano /etc/resolvconf/resolv.conf.d/head
```

aggiungendo le linee *nameserver indirizzo_IP_DNS*. Nel nostro caso sono stati aggiunti l'indirizzi di google 8.8.8.8 e 8.8.4.4. Poiché il file *head* viene richiamato prima del resolver questo avrà effetto sulla configurazione del file *resolv.conf* rendendo la modifica permanente una volta riavviato il servizio.

```
sudo systemctl restart resolvconf.service
sudo systemctl restart systemd-resolved.service
```

B.2 Configurazione macchina pass-trough

Tale passaggio è presente all'interno dello script *setupmachine*. ll nodo pass-trough va configurato affinchè possa smistare il traffico verso le tre macchine collegate e verso internet. Innanzitutto dovrà essere impostata la traduzione del NAT, di modo che i pacchetti vengano letti, convertiti e redirezionati.

```
sudo iptables -A FORWARD -o <interfaccia di uscita a internet> -i
    <interfaccia di entrata del pacchetto> -s 192.168.0.0/24 -m conntrack
    --ctstate NEW -j ACCEPT
```

Questo commando serve a redirezionare il traffico proveniente da ogni interfaccia sulla rete internet senza che venga scartato dal nodo di pass-trough. Verrà eseguito per tutte le tre interfacce di rete. L'estensione conntrack attivata con il flag -m viene usata per tracciare tutti i pacchetti passanti e analizzarli piú nello specifico. Il flag -s indica la rete di provenienza da cui redirezionare il pacchetto. Il flag –ctstate è associato all'opzione conntrack e in questo modo viene visualizzato il traffico con stato NEW in quanto sono pacchetti che non sono passati da tale macchina. Il flag finale -j indica quale azione compiere per tale pacchetto.

```
sudo iptables -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

Attraverso questo comando è possibile eseguire il routing dei pacchetti per gli stati indicati nel comando stesso, nel caso di connessioni giá stabilite.

```
sudo iptables -t nat -F POSTROUTING
sudo iptables -t nat -A POSTROUTING -o enp0s10 -j MASQUERADE
```

In questo modo le regole vengono eseguite anche per le tabelle di NAT (indicate dal flag -t nat). Con l'opzione -F viene effettuato il flush delle tabelle di postrouting, mentre con la seconda opzione -A viene inserita la regola alla fine di quelle presenti e in tal caso è riferito al mascheramento dell'indirizzo che viene eseguito. Con entrambe le opzioni vengono cosí gestite le tabelle di NAT. Per far sí che la configurazione delle iptables sia persistente, procediamo con il suo salvataggio.

```
sudo iptables-save | sudo tee /etc/iptables.sav
```

Inseriamo nel file */etc/rc.local* la riga:

```
iptables-restore < /etc/iptables.sav
```

Dopodichè verrà abilitato il forwarding modificando l'opzione di forwarding e impostandola a 1 per attivarla.

```
sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

In seguito verranno aggiunte due righe a /etc/sysctl.conf per il forwarding di indirizzi ipv4

```
net.ipv4.conf.default.forwarding=1
net.ipv4.conf.all.forwarding=1
```

e dal medesimo file verrà disattivato il commento da *net.ipv4.ipforward=1*. La configurazione è cosí terminata per il forwarding del pacchetto.

In seguito l'impostazione del DNS segue i medesimi step illustrati per il nodo worker.

B.3 Installazione cluster Kubernetes

La configurazione di K3s con il tool kubeadm avviene all'interno del playbook *ansible_kube*, in cui viene eseguita tutta l'installazione del cluster, dal download dei pacchetti necessari sino all'aggiunta di nodi master e worker. Tale playbook richiama al suo interno anche *ansible_master*,

ansible_worker e *createwa*. Inizialmente si procede con l'aggiunta dei repository necessari al download, partendo dall'inserimento delle chiavi necessarie a identificare il pacchetto da installare, e terminando con la creazione della lista in cui vengono identificati i path da cui ottenere le risorse desiderate.

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg |
sudo apt-key add -
echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" |
sudo tee /etc/apt/sources.list.d/kubernetes.list
sudo apt -y install vim git curl wget kubelet kubeadm kubect
```

In seguito disabilitiamo lo swap per motivi di performance illustrati nel capitolo 5.2.3.

```
sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab #(aggiunge un # davanti
    all'opzione di swap)
sudo swapoff -a
sudo mount -a
free -h
```

Segue l'attivazione dei moduli kernel per gestione del filesystem e abilitazione al flusso su VxLAN

```
sudo modprobe overlay
sudo modprobe br_netfilter
```

L'ultima funzionalità viene configurata per poter indicare ai componenti del cluster la presenza delle iptables da cui attingere sia per ipv4 che per ipv6.

```
sudo tee /etc/sysctl.d/kubernetes.conf<<EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
```

L' installazione di cri-o segue i passaggi di impostazione delle chiavi associate alla libreria e il download.

```
echo "deb https://download.opensuse.org/repositories/devel:/kubic
    :/libcontainers:/stable/$OS/ /" >
    /etc/apt/sources.list.d/devel:kubic:libcontainers:stable.list
echo "deb http://download.opensuse.org/repositories/devel:/kubic
    :/libcontainers:/stable:/cri-o:/$VERSION/$OS/ /" >
    /etc/apt/sources.list.d/devel:kubic:libcontainers:stable:cri-o:
    $VERSION.list
curl -L https://download.opensuse.org/repositories/devel:kubic:libcontainers
    :stable:cri-o:$VERSION/$OS/Release.key | apt-key add -
curl -L
    https://download.opensuse.org/repositories/devel:/kubic:/libcontainers
    :/stable/$OS/Release.key | apt-key add -
sudo apt update
sudo apt install cri-o cri-o-runc
```

Impostata la rete da affidare al bridge che configurerá le sottoreti all'interno di ogni nodo worker, e in questo caso per i pod che verranno instanziati.

```
sudo sed -i 's/10.85.0.0/172.24.0.0/g' /etc/cni/net.d/100-crio-bridge.conf
sudo sed -i 's/10.85.0.0/172.24.0.0/g' /etc/cni/net.d/100-crio-bridge.conflis
```

Infine riavviamo cri-o per abilitarlo.

```
sudo systemctl daemon-reload
sudo systemctl restart crio
sudo systemctl enable crio
sudo systemctl status crio
```

Per il nodo master, per cui viene richiamato il playbook *ansible_master* i primi comandi da eseguire sono i seguenti

```
sudo systemctl enable kubelet
sudo kubeadm config images pull
```

I quali abilitano il servizio di kubelet e caricano tutte le immagini per i componenti principali di Kubernetes. Il comando principale è

```
kubeadm init --pod-network-cidr=10.244.0.0/16
```

dove passiamo solo la subnet da affidare ai pod per la compatibilità con il plug-in di configurazione di rete, da qui verranno lanciati i comandi visualizzati a schermo

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

e in seguito installato Flannel tramite download dal file yaml presente sulla git repository.

```
kubectl apply -f
    https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml
```

Per il join del nodo, eseguito su *ansible_worker*, l'unico comando presente è il seguente

```
kubeadm join <indirizzo IP su cui \e in esecuzione il nodo master>:6443
   --token <kubeadm token list> --discovery-token-ca-cert-hash
    sha256:<certificategenerator>
```

il quale viene inserito dal nodo master, ottenendolo tramite la schermata di indicazione per il join corretto, e tale file con il comando all'interno viene inviato ai nodi designati come worker. In seguito lanciando `kubectl get nodes` sulla macchina master sarà possibile visualizzare la lista dei nodi facente parte del cluster.

B.4 MACsec

Per l'installazione del canale sicuro è stato predisposto il playbook *createwa* sempre richiamato da *ansible_kube*. Al suo interno viene creato il file di configurazione illustrato in precedenza 5.2.4 per creare il canale MACsec, questo viene ripetuto sia per la macchina interessata che per la macchina centrale con cui dovrá avvenire il collegamento. Nel caso della libreria `wpa_supplicant` è possibile lanciare il comando seguente su entrambe le macchine.

```
wpa_supplicant -B -K -t -i <interfaccia di rete> -D macsec_linux -c <file di
    configurazione> -dd
```

In tal caso il flag -K serve ad avere nell'output piú informazioni anche su chiavi e altro. Il flag -t viene usato per indicare le tempistiche di avvio delle operazioni. Con il flag -D viene specificato il driver utilizzato che nel nostro caso puó essere solo MACsec. Il flag -B indica l'esecuzione in backgroud poiché venendo avviato tramite un playbook, per poter eseguire le operazioni succesive, tale esecuzione dev'essere passata in background. Mentre il flag finale predispone un output piú parlante. Al termine di tutta questa configurazione dev'essere impostato l'indirizzo IP inerente alla rete presa in considerazione sull'interfaccia di rete appena creata che prenderá come nominativo macsecx dove x è un numero progressivo. Nel caso delle macchine del cluster questo sará sempre 0 mentre per la macchina centrale saranno presenti in numeri progressivi fino al 2.

Nomenclature

AH Authentication Header

AN Access Number

CA Connectivity Association

CAK Connectivity Association Key

CKI Connectivity Association Key Identifier

$CVSS$ Common Vulnerability Scoring System

DOS Denial of Service

EAP Extensible Authentication Protocol

$EAPOL$ Extensible Authentication Protocol over LAN

ESP Encapsulating Security Payload

ICK Integrity Check Value Key

ICV Integrity check value

IV Initial Vector

KEK Key Encrypting Key

KI Key Identifier

KN Message Numbers

KS Key Server

LAN Local Area Network

$MACSEC$ Media Access Control Security

MI Message identifier

$MITM$ Man-in-the-middle

MKA MACsec Key Agreement

$MSDU$ Mac Service Data Unit

PN Packet Number

RNG Random Number Generation

SA Secure Association

SAK Secure Association Key

SC	Secure Channel
SCI	Secure Channel identifier
$SecY$	Security Entity
SKI	Secure Association Key Identifier
SL	Secure Length
TCI	TAG Control information
TLS	Transport Layer Security
$VxLAN$	Virtual Extensible LAN

Bibliografia

[1] S. Frankel and S. Krishnan, "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap", RFC 6071, February 2011, DOI 10.17487/RFC6071

[2] WireGuard project, https://www.wireguard.com/

[3] "IEEE Standard for Local and metropolitan area networks-Media Access Control (MAC) Security", IEEE Std 802.1AE-2018 (Revision of IEEE Std 802.1AE-2006), 2018, pp. 1–239, DOI 10.1109/IEEESTD.2018.8585421

[4] Kubernetes project, Jun, 2020, https://kubernetes.io/it/docs/concepts/overview/what-is-kubernetes/

[5] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, August 2018, DOI 10.17487/RFC8446

[6] R. Press, Dec, 2020, https://www.rambus.com/blogs/macsec/#advantages

[7] CAM table configuration, https://www.techtarget.com/whatis/definition/content-addressed-memory-CAM

[8] B. Weis, "Overview of IEEE 802.1X-REV Dynamic Session Key Agreement", https://www.ietf.org/proceedings/76/slides/msec-5.pdf

[9] "IEEE Standard for Local and metropolitan area networks–Port-Based Network Access Control", IEEE Std 802.1X-2010 (Revision of IEEE Std 802.1X-2004), 2010, pp. 1–205, DOI 10.1109/IEEESTD.2010.5409813

[10] CVSS project, https://www.first.org/cvss/

[11] CVE project, https://cve.mitre.org/

[12] "CVE-2018-0021.", 2017

[13] "CVE-2022-0435.", 2022

[14] "CVE-2017-2342.", 2016

[15] "CVE-2018-15372.", 2018

[16] D. Simon, R. Hurst and Dr. B. Aboba, "The EAP-TLS Authentication Protocol", RFC 5216, March 2008, DOI 10.17487/RFC5216

[17] "CVE-2017-7477.", 2017

[18] D. McGrew and K. Igoe, "AES-GCM Authenticated Encryption in the Secure Real-time Transport Protocol (SRTP)", RFC 7714, December 2015, DOI 10.17487/RFC7714

[19] M. Sieman, "Gcm cipher suites with extended packet numbering", Jul, 2011

[20] C. Cremers, L. Garratt, S. Smyshlyaev, N. Sullivan and C. Wood, "Randomness Improvements for Security Protocols", RFC 8937, October 2020, DOI 10.17487/RFC8937

[21] Autenticazione EAPOL, https://vocal.com/secure-communication/eapol-extensible-authentication-protocol-over-lan/

[22] T. Iwata, J. Song, J. Lee and R. Poovendran, "The AES-CMAC Algorithm", RFC 4493, June 2006, DOI 10.17487/RFC4493

[23] CAK configuration, https://infocenter.nokia.com/public/7750SR217R1A/index.jsp?topic=%2Fcom.nokia.Interface_Configuration_Guide_21.7.R1%2Fmacsec_static_c-ai9emdynxp.html

[24] LDAP project, https://ldap.com/

[25] Open vSwitch project, http://www.openvswitch.org

[26] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment", Linux journal, vol. 2014, no. 239, 2014, p. 2

[27] Tigera project, https://www.tigera.io/learn/guides/kubernetes-networking/

[28] Flannel project, https://github.com/flannel-io/flannela
[29] Sicurezza nei pod, https://kubernetes.io/blog/2021/04/06/podsecuritypolicy-deprecation-past-present-and-future/
[30] Calico project, https://docs.tigera.io/calico/latest/about/
[31] Protocollo BGP, https://www.cloudflare.com/it-it/learning/security/glossary/what-is-bgp/
[32] Canal project, https://github.com/projectcalico/canal
[33] Flannel backend work, Mar, 2017, https://github.com/flannel-io/flannel/blob/master/Documentation/backends.md
[34] K3s project, https://k3s.io/
[35] K8s project, https://kubernetes.io/
[36] wpa supplicant project, https://linux.die.net/man/8/wpa_supplicant
[37] iproute2 project, https://wiki.linuxfoundation.org/networking/iproute2
[38] RADIUS project, https://www.cisco.com/c/it_it/support/docs/security-vpn/remote-authentication-dial-user-service-radius/12433-32.html
[39] H. Shah, G. Heron, V. Kompella and E. Rosen, "Address Resolution Protocol (ARP) Mediation for IP Interworking of Layer 2 VPNs", RFC 6575, June 2012, DOI 10.17487/RFC6575
[40] kubeadm project, https://kubernetes.io/docs/reference/setup-tools/kubeadm/
[41] R. D. Hipp, "SQLite", 2020
[42] A. Bates, J. Manton, S. Jagannathan, M. Costa, P. Schlegel, T. Rohlfing and G. Jefferis, "The natverse, a versatile toolbox for combining and analysing neuroanatomical data", Elife, vol. 9, Apr 2020, DOI 10.7554/eLife.53350
[43] Howard, Aug, 2022, https://community.fs.com/it/blog/macsec-a-cool-security-option-for-your-enterprise-switch.html
[44] AWS project, https://aws.amazon.com/it/
[45] Apache project, https://mesos.apache.org/documentation/latest/containerizers/
[46] S. Singh, Dec, 2021, https://zindagitech.com/what-is-ieee-802-1ae-macsec-how-does-it-work/
[47] JavaInterviewPoint, Jun, 2019, https://www.javainterviewpoint.com/java-aes-256-gcm-encryption-and-decryption/
[48] F. Hauser, M. Schmidt, M. Haberle, and M. Menth, "P4-macsec: Dynamic topology monitoring and data layer protection with macsec in p4-based sdn", IEEE Access, vol. 8, 2020, pp. 58845–58858, DOI 10.1109/ACCESS.2020.2982859
[49] S. Dubroca Oct, 2016. https://developers.redhat.com/blog/2016/10/14/macsec-a-different-solution-to-encrypt-network-traffic#
[50] B. Brad Hedlund, Aug, 2021, https://www.linkedin.com/pulse/securing-your-network-connection-cloud-macsec-vs-ipsec-brad-hedlund/
[51] N. Velayudhan, Jun, 2022, https://opensource.com/article/22/6/kubernetes-networking-fundamentals
[52] M. T. Vallim, Mar, 2022, https://mvallim.github.io/kubernetes-under-the-hood/documentation/kube-flannel.html
[53] S. Jain, Jul, 2018, https://medium.com/@jain.sm/flannel-vs-calico-a-battle-of-l2-vs-l3-based-networking-5a30cd0a3ebd
[54] M. Nuss, Mar, 2015, https://www.electronicdesign.com/technologies/communications/article/21800649/the-skinny-on-ipsec-vs-macsec
[55] H. Mehndiratta, Apr, 2021, https://kubevious.io/blog/post/comparing-kubernetes-container-network-interface-cni-providers
[56] MACsec vulnerability, https://vulmon.com/searchpage?q=macsec
[57] Zafar, Jun, 2020, https://urclouds.com/2020/06/15/what-is-the-kubernetes-cluster-and-how-its-work/
[58] Mr. Anand, Mrs. Sridevi and Dr. J. Mungara, "Linux Based Implementation of MACSec Key Agreement (MKA)", International Journal of Engineering Research and Development, vol. 3, Settembre 2012, pp. 29–34
[59] S. Dubroca, "Encryption for the wired LAN", 2016

[60] Z. U. Abdin, “Protecting Integrity and Confidentiality of Network Traffic with Media Access Control Security (MACsec)”, 2015